

TRẦN ĐÌNH QUẾ

GIÁO TRÌNH

**PHÂN TÍCH VÀ THIẾT KẾ
HỆ THỐNG THÔNG TIN**

Tài liệu dùng cho sinh viên D15-CNTT

Năm học 2018-2019

Ghi chú: Tài liệu này có một số lỗi sẽ được hiệu chỉnh khi trình bày trên lớp

HÀ NỘI - 2018

MỤC LỤC

LỜI NÓI ĐẦU	vi
CHƯƠNG 1: CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG....	1
1.1 GIỚI THIỆU	1
1.2 CÁC KIỂU HỆ THỐNG THÔNG TIN	2
1.3 CÁC CÁCH TIẾP CẬN PHÁT TRIỂN HỆ THỐNG THÔNG TIN	3
1.3.1 Các thành phần chủ yếu của phát triển hệ thống thông tin.....	3
1.3.2 Một số khái niệm cơ bản cho nguyên lý phát triển phần mềm	4
1.3.3 Phân tích và thiết kế hệ thống thông tin	5
1.3.4 Từ phát triển hướng cấu trúc đến phát triển hướng đối tượng.....	5
1.4 MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA HỆ HƯỚNG ĐỐI TƯỢNG	6
1.4.1 Lớp và đối tượng.....	6
1.4.2 Phương thức và thông điệp.....	8
1.4.3 Đóng gói và ẩn dấu thông tin	9
1.4.4 Đa hình và ràng buộc động	12
1.4.5 Quan hệ giữa các lớp.....	12
1.5 SỬ DỤNG LẠI	19
1.6 PHƯƠNG PHÁP LUẬN PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG ...	21
1.6.1 Khái niệm phương pháp luận	21
1.6.2 Các pha phát triển truyền thống	23
1.6.3 Phương pháp luận hướng đối tượng.....	24
1.6.4 UP	25
1.6.5 Các pha phát triển phần mềm.....	31
1.7 KẾT LUẬN	32
BÀI TẬP	32
CHƯƠNG 2: MÔ HÌNH HÓA HỆ PHẦN MỀM HƯỚNG ĐỐI TƯỢNG	34
2.1 GIỚI THIỆU VỀ UML	34
2.1.1 Vài nét lịch sử phát triển của UML	34
2.1.2 Một số đặc điểm của ngôn ngữ mô hình hóa UML	34
2.1.3 Các khung nhìn trong UML.....	35
2.1.3 Các khái niệm cơ bản trong UML	36
2.2 CÁC BIỂU ĐỒ TRONG UML	37
2.2.1 Biểu đồ ca sử dụng.....	39
2.2.2 Biểu đồ lớp.....	41
2.2.3 Biểu đồ trạng thái	46
2.2.4 Biểu đồ tuần tự.....	48
2.2.5 Biểu đồ giao tiếp.....	50
2.2.6 Biểu đồ hoạt động	51
2.2.7 Biểu đồ thành phần	53
2.2.8 Biểu đồ triển khai.....	54

2.3 CÔNG CỤ PHÁT TRIỂN PHẦN MỀM	54
2.4 KẾT LUẬN	57
BÀI TẬP	58
CHƯƠNG 3: XÁC ĐỊNH YÊU CẦU	59
3.1 GIỚI THIỆU	59
3.2 CÁC BƯỚC TRONG PHA XÁC ĐỊNH YÊU CẦU	59
3.2.1 <i>Yêu cầu là gì?</i>	59
3.2.2 <i>Xác định yêu cầu</i>	61
3.3 XÁC ĐỊNH YÊU CẦU NGHIỆP VỤ	61
3.3.1 <i>Xác định và mô tả các tác nhân</i>	62
3.3.2 <i>Xây dựng Bảng Thuật ngữ</i>	63
3.3.3 <i>Xác định và mô tả các ca sử dụng nghiệp vụ</i>	64
3.3.4 <i>Mô tả chi tiết ca sử dụng</i>	65
3.3.5 <i>Xây dựng biểu đồ giao tiếp</i>	66
3.3.6 <i>Xây dựng biểu đồ hoạt động</i>	67
3.4 XÁC ĐỊNH YÊU CẦU HỆ THỐNG	68
3.4.1 <i>Xác định và mô tả các tác nhân</i>	68
3.4.2 <i>Xác định và mô tả các ca sử dụng</i>	69
3.4.3 <i>Xây dựng biểu đồ ca sử dụng</i>	70
3.4.4 <i>Xây dựng kịch bản</i>	73
3.4.5 <i>Xếp ưu tiên các ca sử dụng</i>	76
3.4.6 <i>Phác họa giao diện người dùng</i>	77
3.5 CASE STUDY: HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ	78
3.6 KẾT LUẬN	88
BÀI TẬP	89
CHƯƠNG 4: PHÂN TÍCH YÊU CẦU	90
4.1 GIỚI THIỆU	90
4.2 CÁC BƯỚC TRONG PHA PHÂN TÍCH	90
4.3 PHÂN TÍCH TĨNH	92
4.3.1. <i>Xác định các lớp</i>	92
4.3.2 <i>Xác định quan hệ giữa các lớp</i>	92
4.3.3 <i>Xây dựng biểu đồ lớp</i>	93
4.3.4 <i>Xác định thuộc tính lớp</i>	96
4.4 PHÂN TÍCH ĐỘNG	100
4.4.1 <i>Xây dựng biểu đồ giao tiếp</i>	101
4.4.2 <i>Gán phương thức cho các lớp</i>	103
4.5 CASE STUDY: HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ	106
4.5.1 <i>Phân tích tĩnh</i>	107
4.5.2 <i>Phân tích động</i>	113
4.6 KẾT LUẬN	115
BÀI TẬP	115
CHƯƠNG 5: THIẾT KẾ KIẾN TRÚC HỆ THỐNG	116

5.1 GIỚI THIỆU	116
5.2 CÁC BƯỚC TRONG PHA THIẾT KẾ.....	117
5.3 LỰA CHỌN CÔNG NGHỆ MẠNG CHO HỆ THỐNG.....	118
5.3.1 Kiến trúc mạng đơn tầng và hai tầng	118
5.3.2 Kiến trúc ba tầng	120
5.3.3 Kiến trúc Client – Server và kiến trúc phân tán	122
5.3.4 Biểu diễn hình trạng mạng với UML	123
5.4 THIẾT KẾ TƯƠNG TRANH VÀ AN TOÀN-BẢO MẬT.....	124
5.4.1 Thiết kế tương tranh	124
5.4.2 Thiết kế an toàn-bảo mật.....	125
5.5 PHÂN RÃ HỆ THỐNG THÀNH CÁC HỆ THỐNG CON	126
5.5.1 Hệ thống và hệ thống con	127
5.5.2 Các cụm	127
5.5.3 Ví dụ : Java Layers - Applet plus RMI.....	129
5.6 XÂY DỰNG BIỂU ĐỒ GÓI	130
5.7 KẾT LUẬN	131
BÀI TẬP	131
CHƯƠNG 6: THIẾT KẾ CHI TIẾT.....	133
6.1 GIỚI THIỆU	133
6.2 XÂY DỰNG MÔ HÌNH LỚP THIẾT KẾ	133
6.2.1 Ánh xạ các phương thức	134
6.2.2 Các kiểu biến	134
6.2.3 Phạm vi của các trường.....	134
6.2.4 Các toán tử truy nhập	135
6.2.5 Ánh xạ các lớp, thuộc tính và kiểu quan hệ hợp thành.....	135
6.2.6 Ánh xạ các kiểu quan hệ khác.....	136
6.3 XÂY DỰNG LUỢC ĐỒ CƠ SỞ DỮ LIỆU.....	140
6.3.1 Các hệ quản trị cơ sở dữ liệu.....	140
6.3.2 Mô hình quan hệ	141
6.3.3 Ánh xạ các lớp thực thể	142
6.3.4 Ánh xạ các liên kết	142
6.3.5 Ánh xạ kế thừa	145
6.4 THIẾT KẾ GIAO DIỆN NGƯỜI SỬ DỤNG	148
6.5 SỬ DỤNG FRAMEWORK, MẪU VÀ THƯ VIỆN	152
6.6 KẾT LUẬN	153
BÀI TẬP	153
PHỤ LỤC A: LỰA CHỌN CÔNG NGHỆ	154
A.1 GIỚI THIỆU	154
A.2 CÔNG NGHỆ TẦNG CLIENT	154
A.3 GIAO THỨC GIỮA TẦNG CLIENT VÀ TẦNG GIỮA	155
A.4 CÔNG NGHỆ TẦNG GIỮA	156

A.5 CÔNG NGHỆ TÀNG GIỮA ĐẾN TÀNG DỮ LIỆU	156
A.6 CÁC CÔNG NGHỆ KHÁC	157
PHỤ LỤC B: HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ	161
B.1 XÁC ĐỊNH YÊU CẦU	161
B.2 PHÂN TÍCH TĨNH	188
B.3. PHÂN TÍCH ĐỘNG	194
TÀI LIỆU THAM KHẢO.....	214

LỜI NÓI ĐẦU

Phân tích và thiết kế các hệ thống thông tin là một môn học bắt buộc thuộc chương trình Đại học dành cho sinh viên ngành Công nghệ thông tin. Có nhiều cách tiếp cận phát triển, nhưng cách tiếp cận hướng đối tượng đã trở thành phổ biến trong công nghiệp phát triển phần mềm ngày nay do tính hiệu quả về mặt phát triển cũng như sự hỗ trợ mạnh mẽ của nhiều công nghệ. Cách tiếp cận này xem hệ phần mềm như một tập các lớp với các thuộc tính và thao tác hay hành vi tương ứng cùng với các tương tác giữa các đối tượng trong các lớp. Hơn nữa, sự phát triển mạnh mẽ về kỹ thuật, công nghệ, công cụ hỗ trợ và đặc biệt ngôn ngữ mô hình hóa UML (Unified Modeling Language) đã làm thay đổi căn bản quan niệm và cách phát triển các hệ phần mềm.

Giáo trình này được xây dựng dựa vào chương trình đào tạo theo tín chỉ ngành Công nghệ Thông tin tại Học viện Công nghệ Bưu chính Viễn thông. Nội dung tập trung trình bày một số vấn đề cơ bản của phân tích và thiết kế theo hướng đối tượng trong các pha xác định yêu cầu, phân tích yêu cầu và thiết kế. Giáo trình được biên soạn dựa vào những tài liệu có sẵn được liệt kê trong phần tài liệu tham khảo và kinh nghiệm giảng dạy nhiều năm của tác giả. Ngoài những ví dụ minh họa riêng rẽ, Case study Hệ quản lý học tập theo tín chỉ được sử dụng xuyên suốt trong nhiều chương nhằm giúp cho bạn đọc dễ dàng theo dõi các bước của các pha phát triển.

Mục đích của tài liệu này là nhằm phục vụ sinh viên ngành công nghệ thông tin khi học môn Phân tích và Thiết kế Hệ thống Thông tin. Để có thể đọc được tài liệu này, bạn đọc cần có những kiến thức cơ bản về lập trình Java. Những kiến thức về Nhập môn Công nghệ phần mềm là cần thiết nhưng không bắt buộc. Nội dung tài liệu bao gồm:

Chương 1: Cơ sở phát triển phần mềm hướng đối tượng

Giới thiệu các kiểu hệ thống thông tin và mô hình hệ thống dựa vào cách tiếp cận hướng đối tượng cùng các khái niệm cơ bản liên quan đến cách tiếp cận này. Một số phương pháp luận phát triển phần mềm hướng đối tượng hiện nay cũng sẽ được điểm qua và đặc biệt phương pháp luận UP sẽ được khảo sát chi tiết hơn. Cuối chương sẽ trình bày các pha và các bước thực hiện trong mỗi pha để phục vụ cho các chương sau này.

Chương 2. Mô hình hóa phần mềm hướng đối tượng

Nội dung bao gồm giới thiệu ngôn ngữ mô hình hóa thông nhất UML cho mô hình hệ hướng đối tượng. Nội dung tập trung trình bày một số biểu đồ UML thông dụng và sử dụng các biểu đồ UML trong các pha xác định yêu cầu, phân tích yêu cầu và thiết kế. Trong mỗi biểu đồ, tài liệu đều giới thiệu ý nghĩa của biểu đồ và ví dụ minh họa.

Chương 3. Xác định yêu cầu

Nội dung chương này tập trung trình bày pha xác định yêu cầu dựa trên quan điểm nghiệp vụ và quan điểm người phát triển. Xác định yêu cầu nghiệp vụ nhằm mô hình thực tế hiện thời của hệ thống trong quá trình thu thập yêu cầu. Trong khi đó, xác định yêu cầu phát triển nhằm mô hình hệ thống mà chúng ta định phát triển trong tương lai. Một case study về Quản lý đăng ký học theo tín chỉ sẽ được xem xét.

Chương 4. Phân tích yêu cầu

Nội dung trình bày tổng quan quá trình phân tích bao gồm phân tích tĩnh cũng như phân tích động. Phần phân tích tĩnh đề cập đến việc xác định các lớp và quan hệ giữa các lớp, các thuộc tính. Phần phân tích động bàn về việc thực thi các lớp, các lớp biên, điều khiển và thực thể, các biểu đồ giao tiếp, phương thức và cách gán phương thức cho lớp dựa trên gán trách nhiệm cho lớp.

Chương 5. Thiết kế kiến trúc hệ thống

Trình bày các bước trong thiết kế hệ thống, lựa chọn cấu hình mạng cho thiết kế, một số chủ đề về công nghệ, thiết kế đồng thời và an toàn hệ thống. Nội dung bao gồm: Các công nghệ tầng client; Các công nghệ tầng trung gian; Các công nghệ tầng trung gian đến tầng dữ liệu; Các kiểu cấu hình; Các gói theo UML.

Chương 6. Thiết kế chi tiết

Chương này trình bày ánh xạ mô hình lớp phân tích thành mô hình lớp thiết kế và mô hình lớp thành lược đồ cơ sở dữ liệu. Một số vấn đề liên quan đến giao diện người sử dụng, mẫu thiết kế, framework và thư viện sử dụng lại cũng được trình bày một cách khái quát.

Tác giả vô cùng biết ơn các đồng nghiệp thuộc Khoa Công nghệ Thông tin, Học viện Công nghệ Bưu chính Viễn thông đã tạo động lực để tác giả hoàn thành tài liệu này. Trong quá trình soạn thảo giáo trình, sinh viên Khoa Công nghệ Thông tin qua nhiều thế hệ đã có nhiều đóng góp và đặc biệt thể hiện case study Hệ Quản lý Học tập theo tín chỉ. Tác giả dành món quà này cho các bạn sinh viên qua nhiều năm đã nuôi dưỡng niềm say mê giảng dạy và là động lực để tác giả viết giáo trình này. Mặc dù tác giả đã có nhiều nỗ lực để giáo trình này ra đời nhưng chắc chắn không thể tránh khỏi những thiếu sót. Tác giả rất mong nhận được nhiều ý kiến đóng góp của các đồng nghiệp cùng các bạn sinh viên để tài liệu ngày được hoàn thiện hơn.

Tác giả

CHƯƠNG 1

CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Chương này tập trung trình bày một số chủ đề sau đây:

- Khái niệm hệ thống thông tin
- Các kiểu kệ thống hệ thống thông tin
- Các cách tiếp cận phát triển hệ thống thông tin
- Một số khái niệm cơ bản của hệ hướng đối tượng
- Vấn đề sử dụng lại
- Phương pháp luận phát triển phần mềm

1.1 GIỚI THIỆU

Hiểu được khái niệm hệ thống và cách thức hoạt động của nó là yêu cầu cơ bản để có thể hiểu được thế nào là phân tích và thiết kế hệ thống. Hệ thống được hiểu là một tập hợp các thành phần có quan hệ ràng buộc với nhau và cùng nhau hoạt động để đạt được mục đích chung đặt ra cho hệ thống. Các hệ thống tổ chức xã hội hay các hệ thống liên quan đến máy tính đều có đặc trưng chung như vậy.

Thuật ngữ *hệ thống thông tin* (information system) thường được sử dụng để chỉ một phạm trù liên quan đến máy tính bao gồm các phần tử như hệ phần mềm, các hệ phần cứng, các giao thức mạng, cơ sở hạ tầng, người sử dụng...và các quan hệ ràng buộc lẫn nhau giữa các phần tử này. *Hệ phần mềm* (software system) được xem là phần tử chủ yếu trong một hệ thống thông tin vì chỉ bản thân nó mới có khả năng kết nối và điều khiển các phần tử khác cùng hoạt động với nhau. Như vậy, hệ thống thông tin là khái niệm bao trùm khái niệm hệ phần mềm. Đặc quan trọng của một hệ phần mềm là nó chỉ có thể hoạt động khi có sự tương thích với các thiết bị phần cứng, cơ sở hạ tầng, môi trường...

Mục đích của tài liệu này là cung cấp cho sinh viên các kiến thức, công cụ và kỹ thuật cơ bản để phát triển các hệ phần mềm hướng đối tượng. Do đó, khi nói đến hệ thống thông tin ta hiểu là hệ phần mềm và tập trung xem xét các khía cạnh liên quan đến việc tạo dựng các hệ phần mềm theo hướng đối tượng.

Ví dụ Hệ thống đăng ký học theo tín chỉ liên quan đến môn học, nhân viên, sinh viên, giảng viên. Nó là hệ phần mềm nhằm cung cấp các môn học sẽ giảng dạy trong một học kỳ để sinh viên đăng ký học. Hay Hệ thống đăng ký vé máy bay là hệ phần mềm liên quan đến việc cung cấp thông tin các chuyến bay như loại máy bay, thời gian khởi hành/thời gian đến đích, giá vé...để hành khách đăng ký mua. Các hệ phần mềm này liên quan đến các cơ sở hạ tầng mạng, các loại máy tính khác nhau, các thiết bị thông minh di động...nhưng những chủ đề này không phải là mục tiêu của tài liệu này.

Để hiểu rõ hơn khái niệm hệ thống thông tin, ngoài một số khái niệm quen thuộc như giao diện người dùng, đầu vào, đầu ra chúng ta sẽ xem xét chi tiết hơn một số yếu tố đặc trưng liên quan đến hệ thống [6] như thành phần, thành phần liên quan, biên, môi trường.

- *Thành phần* (component) cũng còn gọi là *hệ thống con* (subsystem) được hiểu là một bộ phận đơn lẻ hay là hợp của một số bộ phận khác. Việc hiểu khái niệm thành phần

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

có ý nghĩa rất quan trọng như hiểu một bộ phận nào đó của xe máy nghĩa là khi ta sửa chữa hay nâng cấp xa máy ta chỉ thực hiện trên một bộ phận nào đó mà không phải toàn bộ hệ thống.

- *Thành phần liên quan* (interrelated components) hay còn gọi là các thành phần kết nối mà hoạt động của nó có liên quan mật thiết với hoạt động của các thành phần khác. Ví dụ, nhân viên trong hệ quản lý đăng ký học theo tín chỉ đóng vai trò kết nối vì sinh viên chỉ có thể đăng ký một môn học nào đó sau khi nhân viên hoàn thành danh sách môn học giảng viên đã đăng ký.
- *Biên* (boundary) của hệ thống nhằm chỉ ra các thành phần bên trong mà chúng ta cần phải phát triển và thành phần bên ngoài không phải phát triển nhưng có liên quan đến hoạt động của hệ thống. Ví dụ, hệ thống quản lý đăng ký học tín chỉ mà chúng ta cần phát triển tách biệt với thành phần bên ngoài là hệ thống thanh toán điện tử. Như vậy, việc thay đổi như cập nhật, nâng cấp hệ thống đăng ký học không liên quan đến hệ thống thanh toán điện tử.
- *Môi trường* (environment) nhằm chỉ những gì bên ngoài biên và có ảnh hưởng đến hệ thống. Hệ thống tương tác với môi trường thông qua *giao diện* (interface) để nhận dữ liệu hay thông tin (dữ liệu đã được xử lý) và hiển thị kết quả. Giao diện cũng xảy ra giữa các hệ thống con hay giữa các hệ thống khác nhau. Ví dụ giao diện giữa hệ quản lý đăng ký học theo tín chỉ và hệ thống thanh toán điện tử.

1.2 CÁC KIỂU HỆ THỐNG THÔNG TIN

Trong thực tế nhiều người khác nhau có thể tham gia sử dụng các hệ phần mềm khác nhau. Tính khác biệt của các kiểu hệ thống thông tin là do có sự khác nhau về chức năng hay công nghệ sử dụng để xây dựng nó. Hơn nữa, các hệ thống khác nhau đòi hỏi những phương pháp luận, các kỹ thuật và các công nghệ phát triển khác nhau. Do đó, ngoài những hiểu biết về lĩnh vực dự định ứng dụng, chúng ta cần phải nắm được các đặc trưng của các kiểu hệ thống và các công nghệ, kỹ thuật thích hợp cho các hệ thống đó. Điều này đòi hỏi chúng ta không những cần phải có các kiến thức sâu rộng mà còn phải tích lũy kinh nghiệm qua nhiều dự án phần mềm. Sau đây là một số hệ thống thông tin thường gặp:

Hệ xử lý giao dịch: Dành cho các nhà quản lý vận hành hay quản lý hệ thống để thu thập được những câu trả lời cho các câu hỏi thường ngày. Ví dụ, hệ thống lưu vết trình tự các hoạt động và giao dịch hàng ngày như các hệ xử lý giao dịch mua bán hàng hay giao dịch ngân hàng, điều khiển thiết bị hay dây chuyền sản xuất, lập lịch, xử lý đơn đặt hàng...

Hệ cơ sở tri thức: Dành cho các nhân viên tri thức và xây dựng dữ liệu nhằm tổ chức, khám phá và tích hợp tri thức mới từ tri thức hiện thời vào nghiệp vụ của họ hay điều khiển luồng công việc. Ví dụ, các hệ thống hoạt động dựa trên tri thức, tự động hóa văn phòng, hệ xử lý ngôn ngữ, hệ thống tư vấn trong thương mại điện tử...

Hệ thông tin quản lý: Dành cho các nhà quản lý trung gian để phục vụ việc giám sát, điều khiển, ra quyết định và các hoạt động quản trị hay hỗ trợ ra các quyết định quan trọng (ít có cấu trúc) với các yêu cầu về thông tin không rõ ràng. Ví dụ các hệ thông tin quản lý, hệ

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

hỗ trợ quyết định, hệ quản lý bán hàng cần biết hàng tồn kho, ngân sách hàng năm để lập kế hoạch sản xuất, phân tích chi phí, phân tích giá cả/lợi nhuận, hệ thống tính cước và chăm sóc khách hàng, hệ thống quản lý thư viện, hệ thống đào tạo trực tuyến....

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. Ví dụ các trang báo điện tử, trang dịch vụ bán vé máy bay...

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. *Hệ thời gian thực* và *hệ thống nhúng* cũng có thể xem là các hệ thống thuộc lớp này.

Hệ thống chiến lược: Dành cho các nhà quản lý chính để giúp giải quyết và vạch ra các chiến lược và các xu hướng lâu dài; so sánh khả năng của tổ chức với những thay đổi của môi trường bên ngoài và cơ hội xảy ra trong khoảng thời gian dài. Ví dụ, hệ hỗ trợ thực thi cho dự đoán ngân sách, xu hướng bán hàng trong năm nay, kế hoạch hoạt động trong năm nay, kế hoạch về lợi nhuận, nhân lực.

Tương ứng với các kiểu hệ thống thông tin, các quá trình xử lý và lưu trữ cũng sẽ được thực hiện khác nhau. Do đó, các nhà phát triển cần chú ý nắm vững những đặc trưng của kiểu hệ thống mà mình cần phải phát triển. Hơn nữa, đa phần các hệ thống thông tin trên môi trường Internet ngày nay càng trở nên phức tạp khi nó cần phải tích hợp nhiều kiểu hệ thống thông tin.

Ví dụ, các hệ thương mại điện tử như eBay, Amazon là những hệ phần tán không chỉ có chức năng quản lý giao dịch mua, đặt hàng, bán hàng, mà còn có chức năng quản lý giao dịch thanh toán và tư vấn khách hàng... Để có thể thực hiện các chức năng này, các nhà phát triển hệ thống cần phải biết sử dụng các công nghệ, kỹ thuật lưu trữ và xử lý phân tán cũng như nắm vững các kỹ thuật trong biểu diễn và xử lý tri thức của các lĩnh vực liên quan như trí tuệ nhân tạo, khai phá dữ liệu, web ngữ nghĩa...

1.3 CÁC CÁCH TIẾP CẬN PHÁT TRIỂN HỆ THỐNG THÔNG TIN

1.3.1 Các thành phần chủ yếu của phát triển hệ thống thông tin

Dù hệ thống thông tin có thuộc kiểu nào thì quá trình phát triển một hệ thống cũng thường dựa vào các thành phần chủ yếu sau đây:

- *Phương pháp luận* (methodology): là một loạt các bước có thể thực hiện lặp đi lặp lại mà chúng ta cần phải tuân theo để xây dựng sản phẩm phần mềm. Chi tiết các phương pháp luận phát triển phần mềm sẽ được trình bày trong Mục 1.4.

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- *Mô hình hệ thống* (modeling system): để xây dựng hay hiểu được sự hoạt động của bất kỳ hệ thống nào chúng ta cũng cần đến ngôn ngữ sử dụng để mô hình hóa hệ thống đó. Ngôn ngữ mô hình hóa cho các hệ kế hướng đối tượng sẽ được trình bày trong Chương 2.
- *Kỹ thuật* (technique): Các kỹ thuật liên quan đến các công việc như phỏng vấn, điều tra...để thu thập yêu cầu khách hàng; kỹ thuật lập kế hoạch và quản lý dự án; kỹ thuật xây dựng các biểu đồ để mô hình chức năng hệ thống...Một số kỹ thuật sẽ được đề cập tương ứng với các pha phát triển phần mềm và nội dung chi tiết sẽ trình bày trong các chương sau.
- *Công nghệ* (technology): Các công nghệ nền tảng hỗ trợ phát triển phần mềm như J2EE, .NET...hay các Framework như Spring....
- *Công cụ* (tool): Các công cụ là các phần mềm hỗ trợ cho phát triển như công cụ thiết kế phần mềm VP, môi trường lập trình Netbean, Eclipse...

1.3.2 Một số khái niệm cơ bản cho nguyên lý phát triển phần mềm

Một số khái niệm sau đây được cho là quan trọng để hiểu được hệ thống và là nguyên lý cho phát triển các hệ phần mềm.

- *Phân rã* (decomposition) là quá trình phân chia hệ thống thành các thành phần nhỏ hơn mà đôi khi còn gọi là *hệ thống con* (subsystem) và các hệ thống con này cũng có thể phân chia thành các hệ thống con nhỏ hơn nữa. Việc phân rã hệ thống giúp chúng ta tập trung vào xem xét một phần của hệ thống và vì vậy dễ dàng để suy nghĩ, phát triển, chỉnh sửa một bộ phận mà không quan tâm nhiều đến toàn bộ hệ thống.
- *Môđun hóa* (modularity) là hệ quả trực tiếp của việc phân rã. Nó liên quan đến việc phân chia hệ thống thành các módun có kích thước tương đối đồng đều nhau. Các módun có thể giúp chúng ta hiểu hệ thống dễ dàng và hơn nữa giúp dễ dàng thiết kế và xây dựng lại.
- *Kết nối* (coupling) nhằm chỉ sự phụ thuộc giữa các hệ thống con. Nguyên lý thiết kế cho rằng các hệ thống con nên phụ thuộc vào nhau một cách lỏng lẻo để sự đình trệ của một thành phần không ảnh hưởng đến hoạt động của các thành phần khác. Ví dụ, phần cập nhật thông tin môn học của nhân viên có bị lỗi cũng không ảnh hưởng đến việc đăng ký học của sinh viên.
- *Kết dính* (cohesion) liên quan đến mức độ kết nối của các thành phần bên trong một hệ thống con. Nguyên lý thiết kế cho rằng các yếu tố bên trong một thành phần nên kết dính chặt với nhau. Ví dụ, việc thực thi lấy thông tin đăng ký của sinh viên, lưu vào cơ sở dữ liệu và thống kê danh sách sinh viên đăng ký môn học nên gán cho các thành phần kác nhau nhưng có kết dính chặt với nhau.

1.3.3 Phân tích và thiết kế hệ thống thông tin

Phân tích và thiết kế hệ thống thông tin (information system analysis and design) là một tiến trình phức tạp, có tổ chức mà các thành viên của nhóm xây dựng và tổ chức khách hàng cần phải tuân theo để phát triển và bảo trì hệ thống thông tin máy tính.

Mặc dù phụ thuộc nhiều vào các yếu tố như công nghệ, kỹ thuật hỗ trợ cho phát triển nghiệp vụ như môi trường Internet, thiết bị cầm tay...nhưng việc phân tích và thiết kế các hệ thống thông tin vẫn phải được định hình từ chính yêu cầu của tổ chức khách hàng. Nghĩa là, hệ thống đã được đưa vào sử dụng lại phải nâng cấp cũng chỉ nhằm để tăng cường và cải tiến các hoạt động của tổ chức khách hàng. Vì vậy, có thể nói rằng phân tích và thiết kế hệ thống thông tin là một tiến trình cải tiến tổ chức khách hàng. Hệ thống thông tin [6] liên quan đến nhiều khía cạnh như các hệ phần cứng, các hệ phần mềm, cơ sở hạ tầng mạng, tài liệu huấn luyện sử dụng... Tuy nhiên, sản phẩm quan trọng nhất của kết quả quá trình phân tích và thiết kế hệ thống là *phần mềm ứng dụng*.

Mục đích của thiết kế các phần mềm ứng dụng là nhằm tăng cường hiệu quả các hoạt động của các tổ chức hay doanh nghiệp như quản lý lương, quản lý kho, phân tích thị trường, đặt hàng qua mạng, phân phối sản phẩm... Do hệ phần mềm được xem là phần quan trọng nhất của hệ thống thông tin nên trong tài liệu này, các thuật ngữ phát triển phần mềm hay phát triển hệ thống được xem là như nhau. Thuật ngữ *phát triển phần mềm* (software development) nhằm chỉ một quá trình hay các pha mà chúng ta phải trải qua trong việc tạo ra phần mềm thực thi được.

1.3.4 Từ phát triển hướng cấu trúc đến phát triển hướng đối tượng

Phân tích và thiết kế các hệ thống thông tin máy tính đã có từ những năm 50 của thế kỷ trước và được xem như là một nghệ thuật. Từ đó đến nay, phương pháp phát triển phần mềm đã được tiến hóa qua nhiều giai đoạn với những cách tiếp cận khác nhau. Sau đây là một số phương pháp phổ biến:

- Phương pháp phát triển hướng cấu trúc
- Phương pháp phát triển hướng đối tượng
- Phương pháp phát triển hướng thành phần
- Phương pháp phát triển hướng dịch vụ

Cách tiếp cận phát triển phần mềm hướng thành phần và dịch vụ có thể xem là sự tiến hóa của cách tiếp cận hướng đối tượng. Để làm cơ sở cho việc trình bày các nội dung tài liệu sau này, chúng ta cần điểm lại một số nét chính cùng những hạn chế của phương pháp hướng cấu trúc để làm sáng tỏ cách tiếp cận hướng đối tượng mà chúng ta hướng đến.

Phát triển hướng cấu trúc

Trong những năm 70–80 của thế kỷ trước, phương pháp hướng cấu trúc được coi là cách tiếp cận chuẩn mực để phát triển phần mềm. 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

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

thực hiện một công việc xác định. Trong phương pháp hướng cấu trúc, phần mềm được thiết kế dựa trên một trong hai hướng: hoặc theo hướng dữ liệu hoặc theo hướng hành động. Cách tiếp cận hướng dữ liệu dựa vào việc phân rã phần mềm theo các chức năng và dữ liệu tương ứng chức năng đó. Điều này giúp cho những người phát triển hệ thống dễ dàng xây dựng các cơ sở dữ liệu. Ngược lại, 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.

Như vậy, 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 hệ thống thành các hệ thống nhỏ hơn, rồi tiếp tục phân rã các hệ thống con cho đến khi có được các thành phần có thể cài đặt được bằng cách sử dụng các hàm của ngôn ngữ lập trình hướng cấu trúc như ngôn ngữ C. Phương pháp này có ưu điểm là tư duy phân tích thiết kế rõ ràng, chương trình sáng sửa dễ hiểu. 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 việc sử dụng lại như yêu cầu trong công nghiệp phần mềm ngày nay.

Phát triển hướng đối tượng

Trong khi phương pháp hướng cấu trúc chỉ tập trung 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 kiểu tư duy theo cách ánh xạ các thành phần trong miền nghiệp vụ của bài toán vào các đối tượng trong cuộc sống 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 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.

Ngày nay, khả năng lưu trữ và xử lý mạnh mẽ của máy tính, tính đa dạng của môi trường phát triển ứng dụng đặc biệt trên mạng Internet và thiết bị thông minh đã làm thay đổi căn bản quan niệm về phân tích và thiết kế các hệ phần mềm. Tuy nhiên, cách tiếp cận hướng đối tượng đã trở thành chuẩn mực của công nghiệp phần mềm và đã được sử dụng rộng rãi cho phát triển các hệ thống phần mềm trong quản lý các doanh nghiệp, thương mại điện tử, các hệ thời gian thực, các hệ thống thông minh...

1.4 MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA HỆ HƯỚNG ĐỐI TƯỢNG

Phần này nhằm trình bày một số khái niệm cơ bản của các hệ hướng đối tượng như *lớp, đối tượng, phương thức, thông điệp, đóng gói, ẩn dấu thông tin, kế thừa, đa hình, ràng buộc động*. Hiểu rõ những khái niệm này sẽ giúp các nhà phát triển có cái nhìn khi phân rã một hệ thống phức tạp thành những môđun nhỏ hơn nhằm dễ dàng phát triển, quản lý, thực thi và đồng thời dễ dàng sử dụng lại sau này.

1.4.1 Lớp và đối tượng

Lớp (class) là một thuật ngữ để xác định một tập hợp các thể hiện hay các đối tượng có các đặc trưng chung nào đó. Lớp đóng gói các đặc điểm chung của một nhóm các đối tượng. Khái

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

niệm lớp đã được các nhà phát triển phần mềm hướng đối tượng sử dụng để mô tả các đặc trưng mà các dạng đối tượng cụ thể có thể có. *Đối tượng* (object) là một thể hiện của một lớp. Đối tượng có thể là một sự vật, một thực thể, một danh từ hoặc bất cứ cái gì mà bạn có thể nhặt lên hoặc ném đi, hay những gì mà bạn có thể tưởng tượng ra với một số đặc tính nào đó của nó. Mỗi đối tượng có các *thuộc tính* (attribute) mô tả thông tin về đối tượng đó.

- *Trạng thái* (state) của một đối tượng được xác định bởi bộ giá trị của những thuộc tính và quan hệ với những đối tượng khác tại một thời điểm cụ thể.
- *Hành vi* (behavior) nhằm đặc tả những gì đối tượng này có thể thực hiện được. Trong các mô hình hướng đối tượng, các thuật ngữ *hành vi* (behavior), *hành động* (action), *thao tác* (operation), *phương thức* (method) đều có nghĩa như nhau nhưng được dùng cho các ngữ cảnh khác nhau. Ví dụ, khi nói về các đối tượng thực tế thì ta thường hay nói hành vi hay hành động của đối tượng đó; khi đề cập đến đối tượng trong lập trình người ta thường dùng phương thức hay thao tác nhưng phương thức được dùng phổ biến hơn, nó được xem là thể hiện qua cài đặt của hành vi.

Ví dụ, lớp Sinh viên *Sinhvien* trong Hệ quản lý học tín chỉ có các thuộc tính là mã sinh viên, họ và tên, địa chỉ.... Các đối tượng như sinh viên Nguyễn Ngọc Lan có mã sinh viên *maSinhvien* CNTT12345, địa chỉ *diachi* 168 Nguyễn Trãi, Hà nội... và có thể có các hành vi như đăng ký học *dangkyhoc*, hủy đăng ký *huyDangky*, xem lịch học *xemLichhoc*...

Lớp Sách *Sach* trong Hệ quản lý thư viện có các thuộc tính là tên sách *ten*, tên tác giả *tacgia*, nhà xuất bản *nhaXuatban*, năm xuất bản *namXuatban*... Các đối tượng như cuốn sách “Phân tích và thiết kế hướng đối tượng” của tác giả Đặng Văn Đức, nhà xuất bản Giáo dục, năm 2002 có thể có các hành vi như xóa sách *xoaSach*, thêm sách *themSach*, cập nhật thông tin sách *capnhatSach*...

Biểu diễn đối tượng và lớp

Lớp và đối tượng sẽ được biểu diễn theo ngôn ngữ mô hình hóa thông nhất UML (Ngôn ngữ mô hình UML sẽ được trình bày chi tiết trong Chương 2).

Lớp Sách *Sach* được mô tả trong UML bao gồm ba thành phần là tên lớp, các thuộc tính và các phương thức kèm theo dấu ngoặc đơn như Hình 1.1. Các tên lớp thường là một danh từ bắt đầu bằng chữ in hoa và kết hợp với các cụm danh từ khác sao cho thể hiện được ngữ nghĩa rõ ràng cho người đọc. Ví dụ *Sinhvien*, *SinhvienTaichuc*, *SinhvienChinhquy*, *Giangvien*, *Sach*, *SachThamkhoa*, *SachThieunhi*. Các thuộc tính, phương thức thường bắt đầu bằng chữ thường. Ví dụ tên sách *ten*, tác giả *tacgia*, và các phương thức như *xoaSach()*, *themSach()*, *capnhatThongtinSach()*:

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Sach
ten
tacgia
nhaXuatban
namXuatban
xoaSach()
themSach()
capnhatThongtinSach()

Hình 1.1: Biểu diễn lớp sách

và đối tượng sách được biểu diễn bởi Hình 1.2

Sach1 : Sach
ten = PhantichvaThietkeHuongdoituong
tacgia = DangVanDuc
nhaXuatban = Giaoduc
namXuatban = 2002
xoaSach()
themSach()
capnhatThongtinSach()

Hình 1.2: Biểu diễn đối tượng sách

Mỗi đối tượng được mô tả bởi ba thành phần tương ứng với lớp của nó:

- Tên của đối tượng có gạch chân và dấu $a:A$ thể hiện a thuộc lớp A
- Các giá trị của thuộc tính
- Các phương thức để trong dấu ngoặc thể hiện thao tác hay hành vi của đối tượng

1.4.2 Phương thức và thông điệp

Phương thức (method) cài đặt hành vi (hay còn gọi hành động, thao tác) của đối tượng. Phương thức chính là hành động mà một đối tượng có thể thực hiện và nó tương tự như một hàm hay thủ tục trong ngôn ngữ truyền thống C, Pascal. Tuy nhiên, sự khác biệt có thể dễ thấy là phương thức phải cài đặt trong một lớp nào đó, còn hàm có thể viết bất kỳ ở đâu.

Thông điệp (message) là thông tin được gửi cho đối tượng để kích hoạt các phương thức. Một thông điệp chính là lời gọi hàm hay thủ tục truyền thống nhưng lại từ một đối tượng này đến đối tượng khác.

Ví dụ, trong Hệ quản lý thư viện, nếu nhân viên thư viện cần chèn thêm một cuốn sách mới, cô ta sẽ nhập thông tin qua giao diện chương trình và sẽ được hệ thống gửi đi dưới dạng một thông điệp “yêu cầu chèn cuốn sách mới”. Khi nhận thông điệp, đối tượng sách sẽ thực hiện các công việc cần thiết gọi là “thực hiện phương thức” để chèn cuốn sách mới vào cơ sở dữ liệu của hệ thống.

1.4.3 Đóng gói và ẩn dấu thông tin

Các ý tưởng *đóng gói* (encapsulation) và *ẩn dấu thông tin* (information hiding) có liên quan mật thiết nhau trong các hệ hướng đối tượng. Trong khi các cách tiếp cận truyền thống chỉ chú trọng đến hoặc hành động hoặc dữ liệu, cách tiếp cận hướng đối tượng thể hiện tính đóng gói bằng cách kết hợp cả hai hành động và dữ liệu vào trong đối tượng.

Ẩn dấu thông tin thực ra đã được thể hiện trong phương pháp phát triển các hệ phần mềm theo hướng cấu trúc. Nguyên lý của ẩn dấu thông tin cho rằng chỉ thông tin được đòi hỏi để sử dụng môđun phần mềm là được công khai cho sử dụng môđun đó. Nghĩa là, chỉ thông tin được yêu cầu chuyển đến môđun này và thông tin trả về từ môđun đó là được công khai mà không quan tâm đến cách thức đối tượng thực hiện chức năng của nó thế nào.

Trong các hệ hướng đối tượng, việc kết hợp đóng gói thông tin và nguyên lý ẩn dấu thông tin có nghĩa rằng nguyên lý này được áp dụng vào các đối tượng thay vì chỉ áp dụng vào hàm hay tiến trình. Khi đó, các đối tượng được xem như là các hộp đen.

Ví dụ trong Hệ quản lý thư viện, chúng ta chỉ quan tâm đến thông điệp noi gửi yêu cầu chèn một cuốn sách mới nhưng thuật toán bên trong đối tượng nhận cần đáp ứng với thông điệp là ẩn dấu với nơi gửi. Thông tin duy nhất mà đối tượng gửi cần biết là tập các phép toán hay phương thức mà các đối tượng nhận có thể tiến hành và các thông điệp nào cần phải gửi để kích hoạt đối tượng đó. Các nghiên cứu trong cách tiếp cận hướng đối tượng đã chỉ ra rằng đóng gói có một số lợi ích sau đây:

- Đóng gói là an toàn vì một đối tượng không thể can thiệp để thay đổi trạng thái tức là các giá trị thuộc tính của các đối tượng khác.
- Đóng gói làm đơn giản hóa việc chuyển đổi một lớp đang tồn tại thành một lớp được dùng để tạo các đối tượng phân tán (từ xa). Đối tượng phân tán thường nằm ở máy chủ, các phương thức của nó có thể được gọi bởi các ứng dụng nằm trên các máy khác (nói cách khác là có thể được gọi thông qua mạng). Do đó, các thuộc tính thuộc đối tượng đó không thể được gọi trực tiếp như với đối tượng cục bộ. Tuy nhiên, nếu đưa vào các phương thức *set* và *get*, ta có thể làm được điều này dễ dàng.
- Ngoài ra, việc sử dụng các phương thức *set* và *get* sẽ cài đặt chi tiết thêm một tính chất. Ví dụ, có thể đổi thuộc tính từ kiểu *int* sang kiểu *String* mà không ảnh hưởng tới các lớp khác, miễn là chúng ta thực hiện việc chuyển đổi thích hợp trong các phương thức *set* và *get*.

Ví dụ 1.1: Lớp nhân viên *Employee* chứa các thông tin mà ứng dụng cần khi mô tả một nhân viên gồm họ tên và mã nhân viên như sau:

```
public class Employee{  
    public int employeeID;  
    public String firstName;  
    public String midName;  
    public String lastName;
```

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

}

Các thuộc tính đều ghi là public nghĩa là chúng có thể được truy cập từ mọi lớp bằng các câu lệnh sau:

```
Employee emp = new Employee();
emp.employeeID = 123456;
emp.firstName = "Que";
emp.midName = "Dinh";
emp.lastName = "Tran";
```

Mặc dù Java cho phép đọc và sửa đổi các trường theo cách như thế, nhưng thông thường ta không nên làm như vậy. Thay vào đó, ta cần thay đổi phạm vi truy nhập tới các trường để giới hạn khả năng truy nhập của chúng bằng cách sử dụng các phương thức set, get cho mỗi trường để có thể truy cập tới thuộc tính.

```
public class Employee{
    protected int employeeID;
    protected String firstName;
    protected String midName;
    protected String lastName;
    public int getEmployeeID() {
        return employeeID;
    }
    public void setEmployeeID(int id) {
        employeeID = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String name) {
        firstName = name;
    }
    public String getMidName() {
        return midName;
    }
    public void setMidName(String name) {
        midName = name;
    }
    public String getLastname() {
        return lastName;
    }
    public void setLastName(String name) {
        lastName = name;
    }
}
```

Ví dụ 1.2: Tiếp tục Ví dụ 1.1, chúng ta sẽ chỉ ra ý nghĩa của đóng gói như thế nào khi ẩn giấu việc vài đặt chi tiết

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

```
public class Employee{  
    protected String employeeID;  
    protected String firstName;  
    protected String midName;  
    protected String lastName;  
    public int getEmployeeID() {  
        return Integer.parseInt(employeeID);  
    }  
    public void setEmployeeID(int id) {  
        employeeID = Integer.toString(id);  
    }  
    public String getFirstName() {  
        return firstName;  
    }  
    public void setFirstName(String name) {  
        firstName = name;  
    }  
    public String getMidName() {  
        return midName;  
    }  
    public void setMidName(String name) {  
        midName = name;  
    }  
    public String getLastname() {  
        return lastName;  
    }  
    public void setLastName(String name) {  
        lastName = name;  
    }  
}
```

Mặc dù, việc cài đặt thuộc tính *employeeID* bị thay đổi, nhưng các lớp khác khi đọc thuộc tính sửa đổi này sẽ không nhìn thấy bất kỳ thay đổi nào trong hành vi của nó, vì việc thay đổi trong cài đặt được che giấu bởi các phương thức *set* và *get*. Đóng gói các thuộc tính của lớp cho phép định nghĩa các giá trị dẫn xuất có khả năng truy cập.

Ví dụ, bạn có thể định nghĩa phương thức *getFullName()* trong lớp *Employee* để trả về họ tên đầy đủ dưới dạng một chuỗi đơn:

```
public String getFullName() {  
    return firstName + " " + midName+ " " + lastName;  
}
```

Tất nhiên, có thể lấy giá trị dẫn xuất mà không cần tạo phương thức *get*, nhưng như vậy thường sẽ tạo ra các đoạn mã trùng nhau khi sử dụng. Ví dụ, để dẫn xuất tên đầy đủ ở một số vị trí trong ứng dụng, bạn phải copy đoạn mã (*firstName + “ “ + midName +” ” + lastName*) vào các vị trí đó và nếu thay đổi cài đặt, bạn phải thay đổi từng vị trí. Nhưng khi sử

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

dụng phương thức `getFullName()` thì bạn chỉ cần thay đổi ở một vị trí trong mã nguồn mà thôi.

1.4.4 Đa hình và ràng buộc động

Đa hình (polymorphism) có nghĩa là cùng một thông điệp hay một phương thức có thể được thể hiện khác nhau qua các lớp đối tượng khác nhau.

Ví dụ, trong hệ quản lý thư viện, cùng một phương thức chèn `chenthem()` nhưng chèn thêm một cuốn sách *Sach* khác với chèn thêm một bạn đọc *Bandoc* hay chèn một nhân viên *Nhanvien* vì thông tin các đối tượng này được đưa vào và truyền đi khác nhau và sau đó cũng lưu trữ khác nhau.

May mắn là các ngôn ngữ lập trình hiện nay cho phép chúng ta xử lý tính đa hình này thông qua *ràng buộc động* (dynamic binding). Nó là một kỹ thuật cho phép hoãn định kiểu một đối tượng cho đến thời gian chạy. Nghĩa là một phương thức thực sự được gọi khi chương trình chạy. Ngược lại, trong ràng buộc tĩnh kiểu của đối tượng được xác định tại thời điểm biên dịch và do đó người phát triển phải tự chọn phương thức nào được gọi thay vì để hệ thống tự thực hiện.

Ví dụ trong ngôn ngữ lập trình C, chúng ta phải viết một logic quyết định bằng cách sử dụng các toán tử `if` hay `case` để xác định đối tượng nào cần chèn thêm và phải gọi tên hàm tương ứng (thêm sách, thêm bạn đọc hay thêm nhân viên). Tuy nhiên, trong lập trình hướng đối tượng, chúng ta có thể thiết kế chương trình để cho hệ thống tự lựa chọn hàm thực thi phù hợp vào thời gian chạy.

1.4.5 Quan hệ giữa các lớp

1.4.5.1 Quan hệ liên kết, kết hợp và hợp thành

Không có một đối tượng nào có thể tồn tại và hoạt động riêng lẻ. Tất cả các hành động của đối tượng đều liên quan tới các đối tượng khác một cách trực tiếp hoặc gián tiếp, mạnh hoặc yếu. Các kiểu liên quan giúp ta khám phá ra nhiều thông tin và hành vi phụ thuộc nhau giữa các đối tượng.

Ví dụ, nếu ta đang xử lý một đối tượng khách hàng *Customer* có tên Lan và ta muốn gửi cho Lan một thông báo thì ta cần phải biết Lan đang sống ở số nhà 168 Nguyễn Trãi, Hà nội. Như vậy, ta muốn có thông tin về địa chỉ được lưu trữ trong đối tượng *Address*, vì vậy cần có kết nối giữa *Customer* và *Address* để biết thư được gửi đến đâu.

Trong mô hình hóa đối tượng với UML, quan hệ giữa các đối tượng được thể hiện theo ba cách chính sau đây: liên kết, kết hợp và hợp thành. Không phải dễ dàng để xác định chính xác ngay từ đầu sự khác biệt giữa ba quan hệ này, sau đây là một số gợi ý:

- *Liên kết* (association): là một dạng quan hệ giữa hai đối tượng hay lớp đối tượng có liên quan và phụ thuộc vào nhau. Có hai dạng liên kết một chiều và hai chiều tùy theo đối tượng phụ thuộc nhau một chiều hay cả hai chiều. Ví dụ, xét các đối tượng khách

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

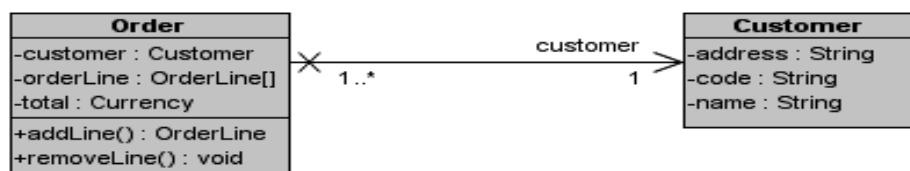
hàng và sách. Chỉ khi khách hàng mua một cuốn sách thì khi đó đối tượng khách hàng và đối tượng sách mới được liên kết với nhau qua hóa đơn bán hàng. Đối tượng hóa đơn phụ thuộc vào sách và khách hàng vì sẽ lấy thông tin từ hai đối tượng này để thể hiện trong hóa đơn.

- *Kết hợp* (aggregation): nghĩa là một đối tượng là một bộ phận của đối tượng kia. Nghiã là các đối tượng kết hợp với nhau để tạo thành một đối tượng lớn hơn. Ví dụ, máy tính được tạo bởi các bộ phận như màn hình, ổ cứng, bàn phím... Kết hợp thường có dạng phân cấp *bộ phận-toàn thể* nghĩa là ám chỉ sự phụ thuộc giữa bộ phận và toàn thể. Ví dụ, màn hình vẫn là màn hình nếu lấy nó ra khỏi máy tính, nhưng máy tính sẽ mất tác dụng nếu thiếu màn hình.
- *Hợp thành* (Composition): là dạng mạnh hơn của kết hợp trong đó bộ phận phụ thuộc vào toàn thể một cách duy nhất và đối tượng toàn thể có trách nhiệm tạo lập và hủy bỏ đối tượng bộ phận. Như vậy, khi đối tượng toàn thể bị huỷ thì đối tượng bộ phận cũng huỷ theo.

Như đã trình bày, không có một ranh giới rõ ràng để phân biệt giữa các quan hệ này đặc biệt quan hệ kết hợp và hợp thành. Nhưng không phải việc phân biệt này lúc nào cũng có thể giải quyết được vấn đề mà cần phải suy nghĩ thường xuyên và cần có kinh nghiệm. Việc lựa chọn này ảnh hưởng rất nhiều đến cách ta thiết kế phần mềm sau này. Các quan hệ này sẽ được trình bày chi tiết hơn trong Chương 2.

Ví dụ 1.3: Vẫn đề quản lý hóa đơn bán hàng liên quan đến đơn đặt hàng Order, các mặt hàng khách hàng đặt mua OrderLine, danh sách đơn hàng OrderList và khách hàng Customer. Có nhiều cách cài đặt cho các quan hệ này, ở đây chúng ta sẽ trình bày một cách cài đặt để minh họa rõ hơn sự khác biệt của quan hệ này.

Liên kết: Quan hệ liên kết thường được miêu tả giống như quan hệ tham chiếu (reference), trong đó một đối tượng có một tham chiếu đến đối tượng khác. Trong Hình 1.3, Order liên kết 1 chiều đến Customer để lấy thông tin khách hàng



Hình 1.3: Quan hệ liên kết

Có thể cài đặt với Java như sau:

```
package relation;
public class Customer{
    private String address;
    private String code;
    private String name;
}
```

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

```
package relation;
public class Order{
    private Customer customer;
    private OrderLine[] orderLine;
    private Currency total;
    public OrderLine addLine() {
    }
    public void removeLine() {
    }
}
```

Đây là quan hệ dễ cài đặt nhất, trong UML nó được biểu diễn bởi một đường thẳng nối giữa hai lớp. Chiều mũi tên nói lên rằng ta gọi đối tượng *Customer* từ đối tượng *Order* nhưng không gọi *Order* từ *Customer*.

Kết hợp: Quan hệ giữa lớp *OrderList* và lớp *Order* là thuộc kiểu kết hợp. Nghĩa là một danh sách *OrderList* bao gồm nhiều *Order* nhưng các *Order* có đời sống riêng của nó và không cần phải là một bộ phận của danh sách *OrderList* cụ thể.



Hình 1.4: Quan hệ kết hợp

Cài đặt:

```
package relation;
public class Order {
    private Customer customer;
    private OrderLine[] orderLine;
    private Currency total;
    public OrderLine addLine() {
    }
    public void removeLine() {
    }
}
package relation;
import java.util.Vector;
import aggregation.Order;
public class OrderList{
    Vector<Order> order = new Vector<Order>();
    public void add() {
    }
    public int getCount() {
    }
    public OrderIterator getIterator() {
    }
    public void remove() {
    }
}
```

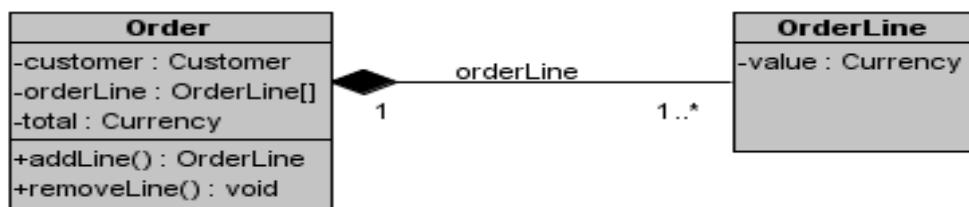
```

    }
}
}
```

Quan hệ kết hợp được biểu diễn trong UML bởi một đường thẳng có hình quả trám rỗng ở một đầu. Điều này xác định không có quan hệ sở hữu trong quan hệ này và các thẻ hiện của lớp được kết hợp sẽ được quản lý bên ngoài lớp kết hợp. Gian chứa *Order* chỉ ra một giới hạn, trong hàm khởi tạo của lớp *OrderList* có một tham số là *Order* để giới hạn số lượng *Order* tương ứng với một *Customer* trong Danh sách *OrderList*.

Hợp thành

Quan hệ giữa *Order* và *OrderLine* là thuộc kiểu hợp thành. Các mặt hàng mà khách hàng đặt *OrderLine* trong một đơn hàng *Order* đều thuộc về *Order* và không có ý nghĩa bên ngoài *Order* đó. *Order* có trách nhiệm hoàn toàn trong việc tạo, quản lý và xóa bất kỳ *OrderLine* nào trong *Order* đó. Trong UML, quan hệ này được biểu diễn bởi đường thẳng với một đầu có hình quả trám màu đen.



Hình 1.5: Quan hệ hợp thành

Cài đặt:

```

package relation;
public class OrderLine {
    private Currency value;
}

package relation;
public class Order {
    private Customer customer;
    private OrderLine[] orderLine;
    private Currency total;
    OrderList unnamedOrderList;
    public OrderLine addLine() {
    }
    public void removeLine() {
    }
}
```

1.4.5.2 Kế thừa

Kế thừa (inheritance) là khái niệm đã được đề xuất và sử dụng rộng rãi để xây dựng mô hình dữ liệu trong những năm 70, 80 của thế kỷ trước. Nó cho phép ta tiến hành tư duy *đặc biệt*

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

hóa (specialization) khi xây dựng một lớp mới bằng cách lấy lại những thuộc tính từ lớp cha và sau đó thêm vào một số thuộc tính riêng của nó. Kế thừa cũng cho phép ta tiến hành tinh tổng quát hóa (generalization) bằng cách nhóm một số lớp thành một lớp tổng quát hơn với những thuộc tính chung để có thể hình thành các đối tượng rộng lớn hơn.

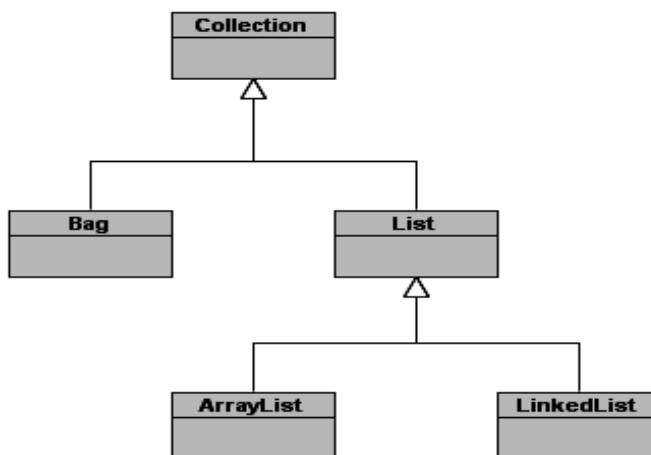
Để hiểu một cách đầy đủ hơn khái niệm kế thừa, chúng ta hãy xem xét một ví dụ thiết kế bộ sưu tập để lưu trữ các đối tượng mà ta thường gặp trong thiết kế các ứng dụng.

Thiết kế cấu trúc phân cấp lớp

Thông thường có bốn kiểu bộ sưu tập như sau:

- *List* (*Danh sách*): lưu các đối tượng theo thứ tự được đưa vào.
- *Bag* (*Túi*): lưu các đối tượng nhưng không theo thứ tự.
- *LinkedList* (*Danh sách liên kết*): lưu các đối tượng như một chuỗi sao cho mỗi đối tượng có một tham chiếu tới một đối tượng khác trong chuỗi. Danh sách liên kết cho phép cập nhật dễ dàng nhưng chậm khi truy cập vì phải duyệt toàn bộ danh sách.
- *ArrayList* (*Danh sách mảng*): lưu các đối tượng theo thứ tự sử dụng như là một mảng, nghĩa là một chuỗi các ô nhớ liên tiếp. Mảng cho phép truy cập nhanh nhưng cập nhật chậm vì ta có thể phải dịch các phần tử hoặc tạo một mảng mới mỗi khi cập nhật.

Vấn đề đặt ra là làm thế nào để thiết kế kiến trúc phân cấp các lớp trên theo kiểu kế thừa? Điểm mấu chốt là cần phải xem xét sự tương đồng giữa các khái niệm với nhau. Rõ ràng, chúng đều là các bộ sưu tập, vì vậy ta sẽ tạo một lớp *Collection* chứa các đặc trưng của các lớp trên và sẽ đặt ở vị trí đầu. Trong bốn bộ sưu tập trên, chỉ có *Bag* là không lưu các đối tượng theo thứ tự, còn lại đều lưu đối tượng theo thứ tự. Do đó, ta đặt *Bag* trực tiếp ngay bên dưới *Collection* trong một nhánh riêng. *List* không có ràng buộc gì về cài đặt bên trong, trong khi *LinkedList* và *ArrayList* thì có. Vì vậy, *List* sẽ là lớp cha của *LinkedList* và *ArrayList*. Như vậy, ta có cây phân cấp sau đây (Hình 1.6):



Hình 1.6: Thiết kế cấu trúc phân cấp

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Trong thực tế, ta lại thường hay làm ngược lại. Trước hết, ta mô tả các lớp ở mức lá (*Bag*, *ArrayList* và *LinkedList*) và sau đó, tìm các khái niệm tổng quát hơn. Trong khi xây dựng mô hình kế thừa, ta tìm các *thông điệp* để có thể chia sẻ, đưa chúng vào mô hình kế thừa càng ở các lớp trên càng tốt. Việc tìm các thông điệp diễn ra trước khi tìm các thành phần lớp khác vì các thông điệp biểu diễn giao tiếp giữa các đối tượng với thế giới bên ngoài. Xét các thông điệp trong mô hình phân cấp *Collection*:

contains(:Object): boolean Tìm các đối tượng trong bộ sưu tập và trả về true nếu bộ sưu tập chứa đối tượng đó, ngược lại trả về false.

elementAt(.int): Object trả về đối tượng ở vị trí được xác định bởi tham số truyền vào.

numberOfElements(): int trả về số nguyên là số đối tượng trong bộ sưu tập.

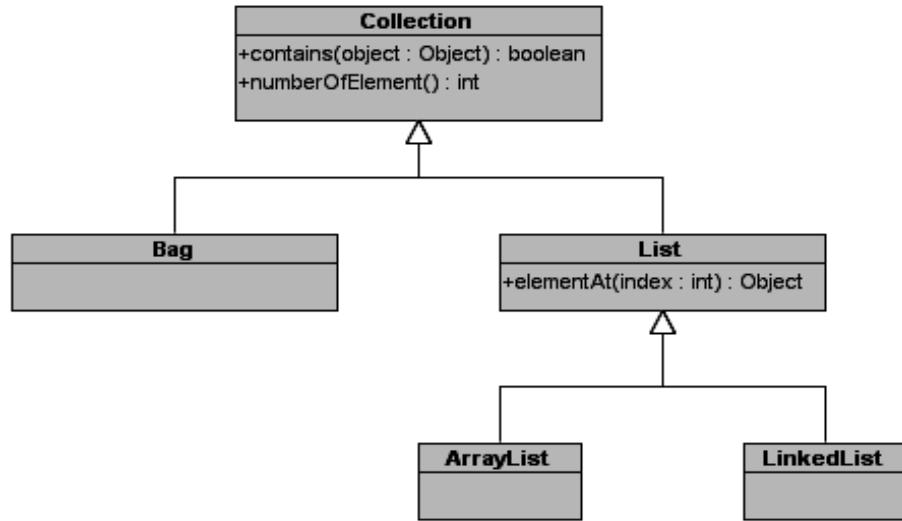
Đặt các thông điệp vào các lớp *contains()* có thể dùng đối với mọi bộ sưu tập, vì vậy đặt nó trong *Collection*. *elementAt(: int)* lấy đối tượng ở vị trí xác định nên phải đặt trong *List*, để tránh sự lặp lại nếu để trong *ArrayList* và *LinkedList*. *numberOfElements()* có thể dùng với mọi bộ sưu tập nên để nó trong *Collection*. Ta có cây phân cấp trình bày trong Hình 1.6.

Cài đặt các phương thức trong phân cấp lớp

Vì thuật toán tìm kiếm sẽ được xử lý khác nhau đối với bộ sưu tập theo thứ tự hay không nên *contains()* không thể cài đặt trong *Collection*. Trong *Bag* và *List* ta sẽ cài đặt hàm *contains()* theo các cách khác nhau.

```
//Cài đặt hàm contains trong lớp List
boolean contains(Object obj) {
    for (int i= 0; i< numberOfElements(); ++i) {
        if (elementAt(i) == obj) {
            return true;
        }
    }
    return false;
}
```

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

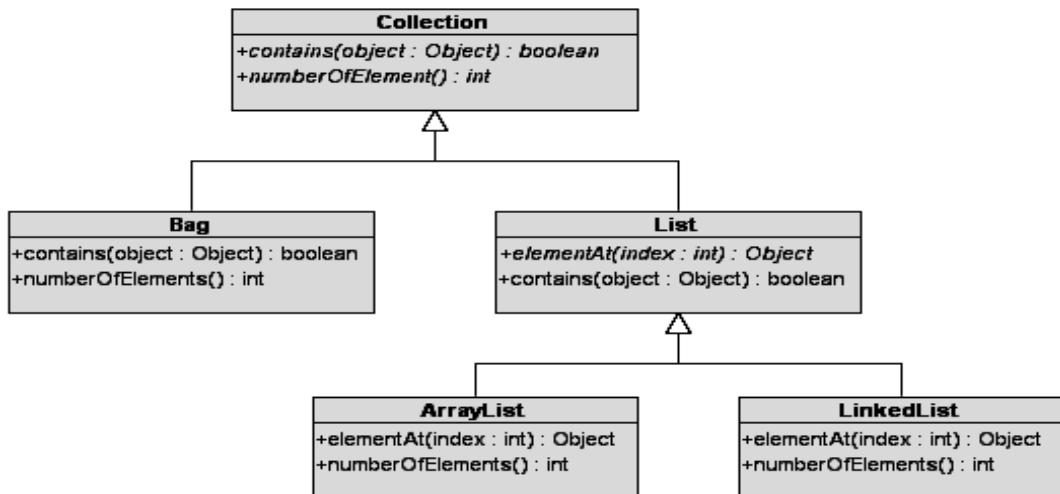


Hình 1.7: Đưa các thông điệp vào các lớp

Phương thức `elementAt()` được cài đặt khác nhau trong hai lớp `ArrayList` và `LinkedList`. Vì vậy, ta phải có hai phương thức `elementAt()` riêng, một cho lớp `ArrayList` – truy cập các phần tử một cách trực tiếp, một cho lớp `LinkedList` – duyệt toàn bộ danh sách. Cài đặt phương thức `numberOfElements()` phụ thuộc vào việc ta lưu số phần tử trong một trường hay tính số phần tử khi cần.

- *Lưu số phần tử trong một trường*: trường này sẽ tăng khi thêm phần tử và giảm khi xóa phần tử. Cách này cho phép ghi số phần tử một cách nhanh chóng tùy thuộc vào bộ nhớ nhưng chậm trong việc thêm và xóa đối tượng.
- *Tính toán số phần tử khi cần*: đối với `LinkedList` thì chậm vì phải duyệt toàn bộ số phần tử. Đối với `ArrayList` và `Bag`, các đối tượng bên trong có thể lưu trữ số phần tử, do đó sẽ nhanh hơn. Cách này sẽ không tốn bộ nhớ và không bị chậm trong việc thêm và xóa đối tượng.

Ta có thiết kế phân cấp lớp như trong Hình 8:



Hình 1.8: Đưa các thông điệp vào các lớp

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Các phương thức in nghiêng là các phương thức ảo, còn lại là các phương thức thực. Phương thức ảo được hiểu là chỉ có tên mà không có các dòng mã cài đặt, còn phương thức thực thì ngược lại.

Các lớp ảo

Lớp ảo là lớp có ít nhất một phương thức ảo – nó có thể nằm trong lớp đó hoặc được kế thừa từ lớp cha. Khi thiết kế phân cấp lớp, ta nên hình dung trong đầu rằng lớp cha cao nhất là ảo.

Định nghĩa lại các phương thức

Lập trình hướng đối tượng cho phép ta “định nghĩa lại” các phần tử dựa trên kế thừa. Dạng đơn giản nhất là định nghĩa lại cho phép lớp con thay đổi việc cài đặt phương thức được kế thừa. Tên phương thức vẫn như cũ nhưng các dòng mã trong thân sẽ được thay thế hoặc chuyển thông điệp từ private sang public hoặc đổi tên hoặc kiểu của một thuộc tính... Sau đây, ta sẽ tập trung bàn về định nghĩa lại nội dung của phương thức, vì đó là lý do quan trọng nhất cho việc định nghĩa lại. Có ba lý do chính giải thích tại sao ta phải định nghĩa lại:

- Phương thức được kế thừa là ảo và ta muốn biến nó thành hiện thực bằng cách thêm vào một số dòng mã. Ví dụ, *contains* là ảo trong *Collection* nhưng cần là thực trong *Bag* và *List*.
- Phương thức cần phải thực hiện thêm một số công việc khi nằm ở lớp con.
- Ta có thể cung cấp một cài đặt tốt hơn cho lớp con. Ví dụ, nếu thêm một chỉ số vào lớp *LinkedList*, ta có thể định nghĩa lại *contains* để làm việc nhanh hơn thuật toán tuần tự được dùng với *List*.

Khi ta thêm công việc, phải chắc chắn rằng định nghĩa lớp cha vẫn làm mọi thứ bình thường – để tăng việc chia sẻ mã nguồn và đơn giản hóa việc bảo trì (ví dụ, nếu sửa đổi định nghĩa của lớp cha, lớp con sẽ tự động có hành vi mới). Mỗi ngôn ngữ hướng đối tượng cho phép phương thức được định nghĩa lại có thể gọi phương thức trong lớp cha. Ví dụ trong Java:

```
void initialize() {  
    //invoke the inherited initialize method  
    super.initialize();  
    ...  
}
```

Đa kế thừa Mỗi lớp con có nhiều lớp cha. Java có một dạng đa kế thừa bằng cách sử dụng khái niệm interface và lớp abstract để khai báo các phương thức ảo (không cài đặt cụ thể). Chủ đề này nằm ngoài phạm vi của tài liệu, bạn đọc quan tâm có thể tham khảo thêm tài liệu [11].

1.5 SỬ DỤNG LẠI

Sử dụng lại (reuse) là khái niệm đã được bàn cãi rất nhiều trong công nghiệp phần mềm. Nhiều nghiên cứu và thực tế phát triển phần mềm đã chỉ ra rằng sử dụng lại sẽ dẫn đến phát

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

triển nhanh hơn, hiệu quả hơn, đáng tin cậy hơn và bảo trì cũng sẽ dễ dàng hơn vì mã đã được kiểm thử nhiều lần. Chúng ta có thể liệt kê ra đây một số cách để sử dụng lại mã nguồn:

- *Sử dụng lại các chức năng trong một hệ thống:* Dạng đơn giản nhất là dùng lại mã nguồn đã được sử dụng rộng rãi trong phát triển các hệ thống truyền thống như với C. Điều này liên quan đến việc viết các hàm tiện ích được gọi từ nhiều nơi. Ví dụ, một số module trong hệ thống cần sử dụng chức năng tìm kiếm thông qua một danh sách tên khách hàng, do đó có thể viết một hàm tìm kiếm chung để có thể gọi từ các tình huống khác nhau.
- *Sử dụng lại các phương thức trong một đối tượng:* Các phương thức được đóng gói trong một đối tượng có thể được gọi từ các phương thức khác. Ví dụ, trong Java phương thức không public có thể sử dụng trong lớp nào thuộc cùng gói với nó. Bạn nên nghĩ đến việc sử dụng lại các phương thức trong một đối tượng bất cứ khi nào cần.
- *Sử dụng lại các lớp trong một hệ thống:* Nhiều lớp đã định nghĩa có thể được dùng trong các phần khác nhau của hệ thống. Ví dụ, nếu bạn xây dựng lớp khách hàng Customer trong hệ thống marketing, bạn cũng muốn các đối tượng Customer tương tự xuất hiện trong nhiều phần khác nhau của mã nguồn hệ thống. Kiểu dùng lại này là cơ sở của cách tiếp cận hướng đối tượng.
- *Sử dụng lại các chức năng trong một số hệ thống:* Các chức năng chung có thể được sử dụng lại giữa các nhóm phát triển trong phát triển hệ hướng cấu trúc cũng như hướng đối tượng. Với một chức năng để đồng nghiệp có thể dùng lại, ta phải giải thích để họ hiểu được. Giải thích có thể đặt vào nơi chứa tài nguyên dùng lại như một cơ sở dữ liệu các chức năng mà các lập trình viên có thể xem xét khi viết mã nguồn mới.
- *Dùng lại các lớp trong một số hệ thống:* Chúng ta cũng có thể sử dụng lại toàn bộ lớp (thuộc tính và phương thức) hơn chỉ là một phương thức. Ví dụ, ta có thể viết một lớp nhân viên Employee với một số thuộc tính cùng với các phương thức để sử dụng cho một số dự án của cùng một công ty.
- *Dùng lại các lớp trong tất cả các hệ thống:* Thành phần phần mềm cũng tương tự như các thành phần phần cứng. Nó có thể tạo ra và sử dụng lại toàn bộ cho nhiều dự án phần mềm. Các thành phần trong phần mềm được thiết kế để có thể dùng lại trong nhiều ngữ cảnh. Nó được đóng gói chặt chẽ (bên yêu cầu không biết công việc bên trong là gì) cùng với tiêu chuẩn của interface và được cung cấp sẵn từ bên thứ ba, thường là phải trả chi phí. Ví dụ, trong Java J2EE, các kiểu JavaBeans có thể sử dụng lại cho nhiều hệ phần mềm khác nhau.
- *Các thư viện hàm:* Các hàm liên quan, có chất lượng tốt có thể được nhóm lại thành một thư viện để sẵn sang sử dụng lại. Ví dụ trong ngôn ngữ C, thư viện stdio, bắt nguồn từ các hệ điều hành UNIX, đã cung cấp khả năng I/O cho các lập trình viên C. Các thư viện hàm được dùng cả trong việc phát triển phần mềm truyền thống cũng như phát triển phần mềm hướng đối tượng. Các thư viện được thiết kế tốt khi được

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

chuẩn hóa bởi các tổ chức như ISO hay ANSI. Các thư viện hàm có thể xuất phát từ bên trong công ty, sử dụng miễn phí hoặc thương mại.

- *Các thư viện lớp*: Là sự phát triển của các thư viện chức năng, thường là các lớp hoàn chỉnh chứ không đơn thuần là các hàm. Viết một thư viện lớp đòi hỏi có nhiều kinh nghiệm. Ví dụ thư viện J2EE, cung cấp mã nguồn cho tất cả các kiểu dùng lại được liệt kê ở trên.
- *Mẫu thiết kế (design pattern)*: Mẫu thiết kế là một sự mô tả cách tạo ra các phần của hệ thống hướng đối tượng một cách hợp lý và hiệu quả. Các mẫu cũng được áp dụng trong các lĩnh vực khác như kiến trúc hệ thống. Mỗi mẫu là một miêu tả ngắn, chi tiết, cho biết khi nào dùng nó và mã nguồn minh họa. Thiết kế các mẫu đòi hỏi phải có nhiều kinh nghiệm, nhưng không bằng việc tạo ra thư viện lớp.
- *Khuôn dạng (Framework)*: Là một cấu trúc đã có sẵn để bạn gắn mã nguồn của mình vào. Trong hướng đối tượng, một framework bao gồm một số lớp đã được viết trước, cùng với tài liệu hướng dẫn lập trình viên các quy tắc phải tuân theo. Ví dụ, EJB framework – bao gồm thư viện J2EE và tài liệu đặc tả, dài hàng trăm trang – chỉ ra cách để lập trình viên viết được các thành phần có khả năng dùng lại, và cách để các bên thứ ba cài đặt máy chủ ứng dụng Java. Phần lớn các framework được thiết kế bởi các chuyên gia cao cấp (guru).

1.6 PHƯƠNG PHÁP LUẬN PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

1.6.1 Khái niệm phương pháp luận

Sản xuất một phần mềm dù có kích cỡ lớn hay nhỏ, dù phát triển dày dặn trọn vẹn cả một sản phẩm hay chỉ là một phần như một módun thì sản phẩm đó cũng thường được tạo ra bởi nhiều người và như vậy cần phải có một *phương pháp luận* (methodology) để sản xuất phần mềm đó.

Phương pháp luận được hiểu là cách thức làm việc một cách có hệ thống. Bản chất của nó là một *tiến trình* (process) có thể lặp đi lặp lại từ trạng thái đầu tiên trong việc phát triển phần mềm (một ý tưởng được hình thành hoặc một cơ hội kinh doanh mới) và xuyên suốt đến phần bảo trì một hệ phần mềm đã được cài đặt. Cũng giống như tiến trình, một phương pháp luận cần phải chỉ rõ sản phẩm mà chúng ta mong được sản xuất khi ta thực hiện theo tiến trình đó. Một phương pháp luận cũng có thể bao gồm cả những lời khuyên hay kỹ thuật để quản lý tài nguyên, kế hoạch, lập lịch và các nhiệm vụ quản lý khác. Một phương pháp luận đầy đủ thường đề cập đến những nội dung sau đây:

- *Lập kế hoạch (Planning)*: Quyết định những gì cần thực hiện.
- *Lập lịch (Scheduling)*: Xác định thời gian thực hiện các công việc
- *Tài nguyên (Resourcing)*: Ước lượng và thu thập các nguồn nhân lực, phần mềm, phần cứng và các tài nguyên cần thiết khác.

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- Luồng công việc (Workflow): các tiến trình con trong tiến trình lớn như thiết kế kiến trúc hệ thống, mô hình dữ liệu...
- Hoạt động (Activites): Các công việc riêng lẻ bên trong luồng công việc như kiểm thử một thành phần, vẽ một biểu đồ lớp...
- Vai trò (Roles): Vai trò của cá nhân như người phát triển, kiểm thử...trong phương pháp luận
- Sản phẩm (Artifacts): Các sản phẩm có liên quan đến quá trình phát triển phần mềm như tài liệu thiết kế, kế hoạch huấn luyện và viết tài liệu hướng dẫn sử dụng.
- Huấn luyện (Training): Xác định cách huấn luyện cho khách hàng, người dùng cuối, người bán hàng...sử dụng hệ thống.

Lợi ích của phương pháp luận cho dự án phần mềm nhỏ

- Phương pháp luận giúp đưa ra những quy định trong viết mã nguồn.
- Phương pháp luận giúp chúng ta hiểu vấn đề hơn, từ đó cải tiến được cách giải quyết vấn đề.
- Viết chương trình chỉ là một trong nhiều hoạt động của phát triển phần mềm. Việc tiến hành các hoạt động khác sẽ giúp chúng ta hiểu được các lỗi về khái niệm và thực tế trước khi tiến hành viết mã nguồn.
- Phương pháp luận giúp chúng ta dễ thay đổi mã nguồn, dễ sử dụng lại hơn cho lớp bài toán khác và cũng giúp dễ dàng soát lỗi hơn.

Lợi ích của phương pháp luận cho những dự án phần mềm lớn

- *Tài liệu*: Tất cả các phương pháp luận đều chú trọng đến khâu viết tài liệu trong mọi giai đoạn phát triển nên tiện lợi cho bảo trì sau này.
- *Giảm được độ trễ*: Vì các luồng công việc, các hoạt động, các vai trò và những mối liên quan bên trong đã được hiểu một cách thấu đáo hơn nên ít có thời gian nhàn rỗi cho nhân viên.
- *Cải thiện may rủi*: Giảm được sự may rủi trong phân phối sản phẩm đúng thời hạn và chi vượt kinh phí.
- *Cải tiến giao tiếp*: Một phương pháp luận tốt sẽ giúp việc giao tiếp tốt hơn giữa những người sử dụng, những người bán sản phẩm, những người quản lý và những nhà phát triển.
- *Tính lặp lại*: Vì những hoạt động được xác định rõ ràng nên những dự án giống nhau có thể được phân theo những giai đoạn tương tự nhau và chi phí như nhau. Nếu chúng ta sản xuất ra những hệ thống tương tự nhau lặp đi lặp lại cho những khách hàng khác nhau (ví dụ như những hệ thương mại điện tử), chúng ta có thể tổ chức được phương pháp luận tốt hơn để chỉ tập trung vào những mặt quan trọng mà sản phẩm mới này sinh.

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- *Tính toán chi phí chính xác hơn:* Dựa trên kinh nghiệm đã có với các dự án trước nên có thể đưa ra ước lượng chi phí chính xác hơn.

1.6.2 Các pha phát triển truyền thống

Các phương pháp luận nào đều có những pha giống nhau. Việc kết hợp các pha này theo một cách nào đó sẽ dẫn đến các phương pháp luận khác nhau như mô hình thác nước, xoắn ốc, lặp và tăng dần phổ biến trong phát triển các hệ hướng cấu trúc.

Xác định yêu cầu

Đây là pha đầu tiên của quá trình phát triển phần mềm. Nó bao gồm hai giai đoạn:

Mô hình hóa nghiệp vụ: liên quan đến việc hiểu nghiệp vụ hiện thời với ngữ cảnh cụ thể của nó mà dự án cần phát triển. Nó có thể là mô hình quản lý đang hoạt động hiện thời hay phần mềm đang sử dụng hiện thời...

Mô hình hóa yêu cầu hệ thống: xác định được yêu cầu thực sự của khách hàng cho hệ thống định xây dựng, hiểu được khách hàng cần gì để biết ta sẽ phải làm gì và không cần làm gì. Điều này cần xác định càng chi tiết và càng rõ ràng càng tốt.

Phân tích yêu cầu

Phân tích có nghĩa là ta phải xét xem ta đang phải đối mặt với vấn đề gì. Trước khi tiến hành thiết kế một giải pháp, ta cần phải hiểu rõ ràng những thực thể liên quan, những tính chất và những mối quan hệ bên trong của chúng. Từ đó mới biết được ta đã thực sự hiểu về sản phẩm mà khách hàng yêu cầu hay chưa vì điều này liên quan đến khách hàng và người dùng cuối.

Thiết kế

Thiết kế là đưa ra cách thức giải quyết vấn đề. Nó bao gồm hai giai đoạn:

Thiết kế hệ thống (system design) hay *thiết kế kiến trúc* (architecture design): Thiết kế hệ thống là phân rã một hệ thống thành các hệ thống con. Nó bao gồm thiết kế hệ thống con liên quan nhau về mặt logic (tiến trình) và hệ thống con liên quan nhau về vật lý (máy tính, mạng) dựa trên các công nghệ, thiết bị máy móc đã lựa chọn.

Thiết kế các hệ thống con (subsystem design) hay *thiết kế chi tiết* (detailed design). Thiết kế hệ thống con là cách quyết định chuyền hệ thống con liên quan nhau về mặt logic thành mã thế nào.

Cài đặt

Pha này sẽ viết mã cho các hệ thống con, viết phần thân các phương thức trong lớp thiết kế. Tích hợp thành hệ thống lớn theo yêu cầu.

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Kiểm thử

Các pha đều có yêu cầu kiểm thử riêng. Kiểm thử khi phần mềm đã hoàn thành là xem sản phẩm đã đáp ứng các yêu cầu của khách hàng chưa. Cách tốt nhất thường được tiến hành làm là thực hiện các kiểm thử trong suốt quá trình phát triển phần mềm để đảm bảo chất lượng.

Triển khai

Pha này liên quan đến việc đưa phần cứng và phần mềm vào nơi người dùng cuối cùng với tài liệu hướng dẫn sử dụng và huấn luyện.

Bảo trì

Sau khi hệ thống được triển khai, có thể sẽ có nhiều lỗi hệ thống hay cũng có thể khách hàng yêu cầu sửa đổi, nâng cấp....vì vậy cần có quá trình bảo trì hệ thống.

1.6.3 Phương pháp luận hướng đối tượng

Theo những nhà sáng lập UML, Grady Booch, Ivar Jacobson và James Rumbaugh, mọi cách tiếp cận phát triển hệ thống hướng đối tượng phải tuân theo ba nguyên tắc (i) dựa vào ca sử dụng, (ii) dựa vào hướng kiến trúc; (iii) dựa vào lặp và gia tăng.

Ca sử dụng

Dùng các ca sử dụng để xác định hành vi ban đầu của hệ thống. *Ca sử dụng* (use case) mô tả cách người dùng hay một hệ thống khác tương tác với hệ thống muốn phát triển để tiến hành một hoạt động nào đó như tạo hóa đơn, đặt hàng hay tìm kiếm thông tin. Khái niệm ca sử dụng và quan hệ của chúng sẽ được trình bày chi tiết trong Chương 2.

Các ca sử dụng được dùng để xác định và chuyển tải các yêu cầu hệ thống cho những người lập trình sau này. Các ca sử dụng trong phương pháp hướng đối tượng được xem là khá đơn giản vì chúng tập trung vào chỉ một chức năng tại mỗi thời điểm. Trong khi đó, phương pháp luận hướng cấu trúc sử dụng các biểu đồ mô hình tiến trình để mô hình các chức năng. Tuy nhiên, cách này phức tạp vì nó đòi hỏi các nhà phân tích phải xây dựng các mô hình cho cả hệ thống.

Hướng kiến trúc

Hướng kiến trúc có nghĩa là kiến trúc phần mềm của hệ thống sẽ định hướng cách đặc tả, cách xây dựng và cách viết tài liệu hệ thống. Mọi cách tiếp cận phân tích và thiết kế các hệ thống hướng đối tượng phải hỗ trợ ít nhất ba quan điểm theo kiến trúc của hệ thống: Quan điểm chức năng, quan điểm tĩnh, quan điểm động. Những quan điểm này thể hiện bằng mô hình với các biểu đồ UML và chi tiết các biểu đồ này sẽ được trình bày trong Chương 2.

- *Quan điểm chức năng* (functional view): Mô tả hành vi của hệ thống theo cách nhìn của người sử dụng. Các ca sử dụng và các biểu đồ ca sử dụng là cách tiếp cận ban đầu được sử dụng để minh họa quan điểm chức năng. Trong một số trường hợp, các biểu đồ hoạt động cũng được sử dụng để hỗ trợ các ca sử dụng.

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- *Quan điểm tĩnh* (static view): Mô tả cấu trúc của hệ thống theo các lớp, thuộc tính, phương thức và quan hệ giữa các lớp. Các biểu đồ cấu trúc phác họa cách nhìn tĩnh của hệ hướng đối tượng đang tiến hóa và được biểu diễn bởi các biểu đồ cấu trúc trong UML.
- *Quan điểm động* (dynamic view) mô tả hành vi bên trong của hệ thống theo các thông điệp được truyền giữa các đối tượng và sự thay đổi trạng thái bên trong một đối tượng. Quan điểm động được biểu diễn bởi các biểu đồ hành vi trong UML.

Lắp và gia tăng

Các cách tiếp cận phân tích và thiết kế hệ hướng đối tượng hiện nay nhấn mạnh phát triển lắp và tăng dần bằng cách tiến hành kiểm thử và mịn hóa liên tục suốt trong vòng đời của dự án. Mỗi quá trình lắp trong phát triển sẽ làm cho hệ thống tiến gần hơn với yêu cầu thực sự của người sử dụng.

Cho đến nay có khá nhiều phương pháp luận hướng đối tượng đã được đề nghị. Mỗi phương pháp đều bàn đến triết lý đằng sau mỗi pha, luồng công việc và những hoạt động, sản phẩm của mỗi pha và cách sử dụng các biểu đồ UML tương ứng cho mỗi pha. Các phương pháp luận được đưa ra đều là những phát triển dựa trên mô hình tiến trình hợp nhất (Unified Software Development Process) hay viết tắt là UP (Unified Process) do Booch, Jacobson, Rumbaugh đề xuất ([7][10]). Chúng ta có thể kể ra đây một số phương pháp luận phổ biến tiến hóa từ UP:

- Agile Unified Process (AUP)
- Basic Unified Process (BUP)
- Enterprise Unified Process (EUP)
- Essential Unified Process (EssUP)
- Open Unified Process (OUP)
- Rational Unified Process (RUP)
- Rational Unified Process – System Engineering (RUP-SE)

1.6.4 UP

Để giúp hiểu rõ phương pháp luận hướng đối tượng sau này, phần này sẽ dành trình bày chi tiết phương pháp luận UP. Là một phương pháp phát triển phần mềm theo hướng ca sử dụng, kiến trúc và lắp và tăng dần, UP (http://en.wikipedia.org/wiki/Unified_Process) đã khắc phục được các nhược điểm của các mô hình truyền thống thác nước và xoắn ốc.

UP có thể thay đổi tùy theo các công ty, tổ chức và còn phụ thuộc vào loại dự án. Nguyên lý lắp và tăng dần trong mô hình UP cho phép chia nhỏ các pha trong vòng đời dự án thành các giai đoạn nhỏ. Trong đó, mỗi giai đoạn có thể được lắp đi lắp lại và tăng thêm chức năng

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

theo thời gian và có thể theo các phiên bản của sản phẩm trong dự án. Nghĩa là chúng ta có thể thêm và cải tiến các chức năng so với phiên bản trước của sản phẩm trong cùng một giai đoạn. Việc cho phép thêm và cải tiến chức năng này gọi là sự phát triển tăng dần của các giai đoạn theo thời gian.

UP là quá trình phát triển hệ thống theo hai chiều được mô tả bởi tập các *pha* (phase) và các *công đoạn* (workflow). Các pha bao gồm: khởi đầu, triển khai, xây dựng và chuyển giao. Các công đoạn bao gồm 7 công đoạn kỹ thuật và 3 công đoạn hỗ trợ. Các công đoạn kỹ thuật bao gồm: mô hình nghiệp vụ, xác định yêu cầu, phân tích yêu cầu, thiết kế, cài đặt, kiểm thử, triển khai. Các công đoạn hỗ trợ bao gồm: quản lý dự án, quản lý cấu hình và thay đổi và môi trường. Trong phần còn lại trong mục này, chúng ta sẽ mô tả các pha và các công đoạn của UP (Xem Hình 1.9).

Các pha (Phase)

Các pha của UP mô tả cách tiến hóa của hệ thống theo thời gian. Phụ thuộc vào pha phát triển nào của hệ thống đang tiến hành mà mức độ hoạt động sẽ thay đổi tương ứng với luồng công việc. Ví dụ, pha khởi động liên quan chủ yếu đến mô hình nghiệp vụ và luồng xác định yêu cầu mà bỏ qua luồng công việc kiểm thử và triển khai. Mỗi pha có một số bước lặp và mỗi bước lặp sử dụng những luồng công việc khác nhau để tạo nên các phiên bản mới của hệ thống. Khi hệ thống tiến hóa qua các pha, nó được cải tiến và trở nên đầy đủ hơn. Các pha được mô tả chi tiết hơn dưới đây:

Pha khởi động (inception)

Đây là pha đầu tiên và ngắn nhất trong vòng đời dự án. Khi áp dụng thực tế, pha này nên được triển khai càng ngắn gọn càng tốt vì nếu kéo dài có nghĩa là đang có sự dư thừa trong việc đặc tả hệ thống. Mốc bàn giao tài liệu yêu cầu và mục đích của dự án đánh dấu sự kết thúc của pha khởi động. Trong pha này, chúng ta cần trả lời các câu hỏi sau đây:

- *Tính khả thi về mặt kỹ thuật*: Chúng ta có khả năng về mặt kỹ thuật để xây dựng hệ thống hay không?
- *Tính khả thi về mặt kinh tế*: Nếu hệ thống được triển khai, nó có đem lại giá trị về mặt kinh tế hay không?
- *Tính khả thi về mặt tổ chức*: Nếu hệ thống được xây dựng, tổ chức hay doanh nghiệp có sử dụng nó hay không?

Để trả lời các câu hỏi này, nhóm phát triển cần tiến hành các công việc cơ bản liên quan đến mô hình nghiệp vụ, xác định yêu cầu và phân tích yêu cầu. Để hiểu những vấn đề có thể gặp phải về mặt kỹ thuật trong quá trình phát triển hệ thống, ta nên xây dựng một bản mẫu thử. Những sản phẩm chính của pha này bao gồm: (i) Tài liệu gồm các nội dung phạm vi của dự án, những yêu cầu và ràng buộc, kế hoạch dự án ban đầu và mô tả tính chấp nhận được của những rủi ro liên quan đến dự án; (ii) Môi trường cần thiết để phát triển hệ thống.

Pha thực thi (Elaboration)

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Những hoạt động trong pha thực thi của UP liên quan nhiều nhất đến phân tích và thiết kế hướng đối tượng. Các luồng phân tích và thiết kế cơ bản tập trung suốt trong pha này. Pha triển khai xây dựng tài liệu bao gồm: hoàn thiện các ca sử dụng nghiệp vụ, hiệu chỉnh đánh giá rủi ro, hoàn thiện kế hoạch dự án với đầy đủ chi tiết để các bên liên quan dự án có thể thỏa thuận xây dựng hệ thống thực sự cuối cùng.

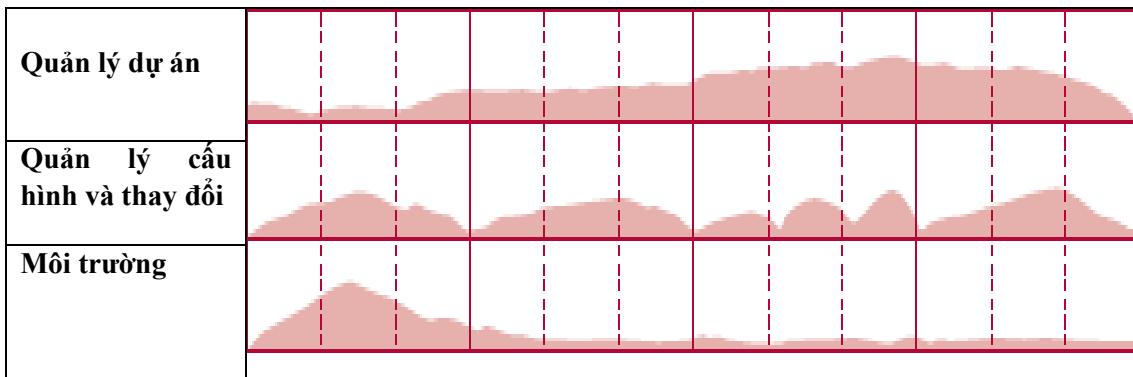
Pha này xây dựng các mô hình của bài toán theo các biểu đồ hành vi và cấu trúc của UML và đồng thời chi tiết hóa mô hình bài toán tương ứng với kiến trúc hệ thống. Vì phát triển lặp qua các luồng nên cần thiết phải xem xét vấn đề liên quan đến quản lý cấu hình và thay đổi cũng như các công cụ phát triển. Sản phẩm chính của pha này bao gồm: (i) Các biểu đồ cấu trúc và hành vi trong UML; (ii) Phiên bản cơ bản thực thi được của hệ thống. Phiên bản cơ bản này sẽ là cơ sở cho cả quá trình phát triển lặp sau này nên nếu có được cơ sở vững chắc tại thời điểm này thì những người phát triển có thể tiến hành hoàn thiện hệ thống trong các pha xây dựng và chuyển giao tiếp theo.

Pha xây dựng (Construction)

Pha xây dựng tập trung vào lập trình nền công việc chủ yếu liên quan đến luồng cài đặt. Tuy nhiên, các luồng xác định yêu cầu, phân tích và thiết kế cũng liên quan đến pha này. Chính pha này sẽ bổ sung những thiếu sót trong xác định yêu cầu và hoàn thiện mô hình phân tích và thiết kế. Luồng quản lý cấu hình và thay đổi rất quan trọng trong pha xây dựng này. Sản phẩm chính của pha này là phiên bản cài đặt beta của hệ thống với kiểm thử chấp nhận.

Pha Công đoạn	Khởi động	Thực thi	Xây dựng	Chuyển giao
Mô hình nghiệp vụ				
Xác định yêu cầu				
Phân tích yêu cầu				
Thiết kế				
Cài đặt				
Kiểm thử				
Triển khai				

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG



Hình 1.9: Các pha và các công đoạn phát triển UP

Pha chuyển giao (Transition)

Giống như pha xây dựng, pha chuyển giao bàn đến các khía cạnh liên quan đến pha cài đặt của cách tiếp cận phát triển phần mềm cổ điển. Khi chuyển đến pha này, các luồng mô hình nghiệp vụ, xác định yêu cầu và phân tích át đã được hoàn thiện trong các vòng lặp trước. Pha này tập trung chủ yếu vào các luồng kiểm thử và triển khai. Phụ thuộc vào kết quả kiểm thử, có thể cần phải thiết kế lại hay cài đặt lại nhưng phải là tối thiểu tại thời điểm này. Sản phẩm chính của pha này là hệ thống thông tin thực thi thực sự. Các sản phẩm khác bao gồm tài liệu hướng dẫn sử dụng, bản kế hoạch để hỗ trợ người dùng và nâng cấp hệ thống trong tương lai.

Các công đoạn

Các công đoạn (workflows) mô tả các công việc hay các hoạt động mà người xây dựng cần thực hiện theo thời gian để phát triển hệ thống. Các công đoạn trong UP được nhóm thành hai phạm trù: Các công đoạn kỹ thuật (engineering workflows) và các công đoạn hỗ trợ (supporting workflows).

Các công đoạn kỹ thuật

Các công đoạn kỹ thuật bao gồm mô hình nghiệp vụ, xác định yêu cầu, phân tích yêu cầu, thiết kế, cài đặt, kiểm thử và triển khai. Các công đoạn kỹ thuật liên quan đến các hoạt động để sinh ra sản phẩm kỹ thuật là hệ thống thông tin. Tiếp theo, chúng ta sẽ mô tả chi tiết các công đoạn kỹ thuật này.

Mô hình nghiệp vụ (business modeling)

Công đoạn *mô hình nghiệp vụ* nhằm phát hiện vấn đề và xác định các dự án tiềm năng bên trong tổ chức của người dùng. Luồng công việc này giúp hiểu được phạm vi của dự án và hướng đến cải tiến hiệu quả và hiệu suất của tổ chức người dùng.

Mục đích chủ yếu của mô hình nghiệp vụ là nhằm đảm bảo cả người phát triển và tổ chức người dùng hiểu được rằng hệ thống cần phải phát triển thích hợp ở đâu và như thế nào với tiến trình nghiệp vụ của tổ chức. Luồng công việc này chủ yếu được thực thi trong pha khởi đầu nhằm đảm bảo rằng hệ thống định phát triển có ý nghĩa về mặt nghiệp vụ. Những hoạt động xảy ra trong luồng này liên quan nhiều đến pha lập kế hoạch của tiến trình phát triển cổ

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

diễn. Tuy nhiên, các kỹ thuật thu thập yêu cầu, mô hình hóa tiến trình nghiệp vụ và ca sử dụng cũng được sử dụng để hiểu các tình huống nghiệp vụ.

Xác định yêu cầu (requirement determination)

Trong UP, công đoạn xác định yêu cầu nhằm làm rõ các yêu cầu chức năng và phi chức năng (sẽ trình bày chi tiết hơn trong Chương 3). Các yêu cầu được thu thập từ các bên tham gia dự án như người dùng cuối, người quản lý trong tổ chức người dùng cuối và ngay cả khách hàng như trong các phần mềm trên mạng hiện nay. Có nhiều cách để thu thập yêu cầu như phỏng vấn, quan sát, tổ chức hội thảo, phân tích tài liệu, lập phiếu điều tra... Công đoạn xác định yêu cầu được sử dụng trong các pha khởi đầu và triển khai. Các yêu cầu được xác định rõ ràng và đầy đủ sẽ rất hữu ích cho việc phát triển tài liệu và các ca sử dụng suốt trong quá trình phát triển hệ thống.

Phân tích yêu cầu (requirement analysis)

Công đoạn phân tích yêu cầu ưu tiên bàn đến việc tạo ra mô hình phân tích lĩnh vực mà dự án quan tâm. Trong UP, nhà phân tích bắt đầu với thiết kế kiến trúc liên quan đến miền bài toán và sử dụng các biểu đồ hành vi và cấu trúc của UML để mô tả các lớp và tương tác của chúng.

Mục tiêu đầu tiên của công đoạn phân tích là đảm bảo rằng cả nhóm phát triển và tổ chức sử dụng hiểu được những vấn đề cơ bản và giới hạn của miền mà dự án quan tâm. Mục tiêu thứ hai của phân tích là xác định những lớp thư viện mà có thể sử dụng lại để tránh phát triển dư thừa khi tạo ra những biểu đồ cấu trúc và hành vi. Giống như công đoạn xác định yêu cầu, công đoạn phân tích chiếm ưu thế trong pha triển khai nhưng nó có thể sử dụng trong mọi pha của quá trình phát triển hệ thống.

Thiết kế (design)

Công đoạn thiết kế nhằm chuyển mô hình phân tích thành *mô hình thiết kế* (design model) để có thể cài đặt hệ thống. Trong khi công đoạn phân tích quan tâm đến việc hiểu lĩnh vực của bài toán, công đoạn thiết kế tập trung vào phát triển một giải pháp có thể thực thi được trong một môi trường xác định. Nó liên quan chủ yếu đến các pha triển khai và xây dựng.

Công đoạn thiết kế mô tả hệ thống đầy đủ hơn bằng cách thêm các lớp liên quan đến môi trường hệ thống vào trong mô hình phân tích. Công đoạn này xem xét các hoạt động như thiết kế giao diện người sử dụng, thiết kế cơ sở dữ liệu, thiết kế kiến trúc vật lý, thiết kế chi tiết các lớp thực thể và tối ưu hóa hệ thống.

Cài đặt (implementation)

Mục đích chủ yếu của công đoạn cài đặt là tạo ra một giải pháp thực thi dựa trên mô hình thiết kế. Nó bao gồm ba giai đoạn chính: (i) Viết những lớp mới và kết nối với những lớp sử dụng lại từ các thư viện lớp để có được một giải pháp; (ii) Thực hiện kiểm thử các lớp mới và các tương tác với các lớp sử dụng lại; (iii) Khi có nhiều nhóm cùng tiến hành cài đặt, cần phải tiếp hành tích hợp các módun riêng lẻ đã được kiểm thử để tạo ra một phiên bản thực thi được của hệ thống. Công đoạn cài đặt chủ yếu liên quan đến các pha triển khai và xây dựng.

Kiểm thử (test)

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Mục đích chủ yếu của công đoạn kiểm thử là nhằm tăng cường chất lượng của hệ thống. Ngoài việc kiểm thử liên quan đến cài đặt, nó cũng bao gồm kiểm thử việc tích hợp các môđun, kiểm thử chấp nhận với người sử dụng và kiểm thử phần mềm triển khai thực sự. Thực tế, công đoạn kiểm thử được trải khắp các pha phát triển phần mềm. Kiểm thử các mô hình phân tích và thiết kế được thực hiện trong các pha triển khai và xây dựng. Kiểm thử cài đặt được tiến hành chủ yếu trong pha xây dựng và chuyển giao.

Triển khai (deployment)

Công đoạn triển khai liên quan nhiều nhất đến pha chuyển giao trong UP. Công đoạn triển khai bao gồm các hoạt động như đóng gói phần mềm, phân phối sản phẩm, cài đặt, kiểm thử. Khi triển khai thực sự hệ thống mới tại vị trí tổ chức người dùng, nhóm phát triển có thể phải chuyển đổi dữ liệu hiện thời, giao diện phần mềm mới với phần mềm hiện thời và tổ chức lớp huấn luyện sử dụng hệ thống mới.

Các công đoạn hỗ trợ

Các công đoạn hỗ trợ tập trung vào các khía cạnh quản lý bao gồm quản lý dự án, quản lý cấu hình và thay đổi, và môi trường.

Quản lý dự án (project management)

Trong khi các công đoạn kỹ thuật liên quan đến cả bốn pha trong UP, công đoạn quản lý dự án thực sự chỉ liên quan đến việc dịch chuyển giữa các pha. Tiến trình phát triển hỗ trợ phát triển lặp và gia tăng nền hệ thống sẽ tăng dần và tiến hóa theo thời gian. Tại cuối mỗi giai đoạn lặp, một phiên bản mới của hệ thống sẽ sẵn sàng cho phân phối. Do sự phức tạp của mô hình phát triển hai chiều (công đoạn và pha) của UP nên công đoạn quản lý dự án đóng vai trò rất quan trọng. Các hoạt động của công đoạn này bao gồm: xác định và quản lý rủi ro, ước lượng thời gian hoàn thành và chi phí cho mỗi bước lặp và toàn bộ dự án, theo dõi quá trình phát triển cho sản phẩm cuối cùng.

Quản lý cấu hình và thay đổi (configuration and change management)

Mục đích chủ yếu của công đoạn này là theo dõi hiện trạng sản phẩm của hệ thống đang phát triển. Một số hiện trạng sản phẩm bao gồm các biểu đồ, mã nguồn, mã thực thi. Suốt trong quá trình phát triển các sản phẩm này sẽ được sửa đổi. Thỏa thuận về các thông tin quản lý dự án cần được xác định như người thực hiện, thời gian và vị trí của mỗi sửa đổi. Công đoạn này phần lớn liên quan đến các pha xây dựng và chuyển giao.

Môi trường (environment)

Suốt trong quá trình phát triển dự án, nhóm phát triển cần sử dụng các công cụ và tiến trình khác nhau. Công đoạn môi trường tập trung xem xét những nhu cầu này. Ví dụ, các công cụ phân tích và thiết kế, các công cụ quản lý dự án và cấu hình, môi trường lập trình... Công đoạn này chủ yếu liên quan đến pha khởi đầu trong UP.

1.6.5 Các pha phát triển phần mềm

Các phương pháp luận dựa trên UP phân chia tiến trình phát triển thành các pha, các công đoạn lặp theo thời gian. Đó là những khuôn mẫu chung nên để có thể áp dụng vào quá trình phát triển phần mềm thực sự, chúng ta còn phải chỉ ra các bước, các công việc phải làm trong từng bước, biểu đồ UML nào sẽ sử dụng trong từng công đoạn và các sản phẩm của mỗi công đoạn. Phần này sẽ đề cập đến một bước cơ bản quen thuộc cần phải tiến hành trong mỗi công đoạn và pha phát triển phần mềm.

Các pha		Các bước thực hiện	UML
Xác định yêu cầu	Nghiệp vụ	1. Xác định và mô tả các tác nhân 2. Xây dựng bảng thuật ngữ 3. Xác định và mô tả các ca sử dụng 4. Xây dựng kịch bản 5. Xây dựng biểu đồ hoạt động (Tùy chọn) 6. Xây dựng biểu đồ giao tiếp (Tùy chọn)	Không Không Không Không Có Có
	Hệ thống	1. Xác định và mô tả các tác nhân 2. Xác định và mô tả các ca sử dụng 3. Xây dựng kịch bản 4. Xây dựng biểu đồ ca sử dụng 5. Xếp ưu tiên các ca sử dụng 6. Phác họa giao diện người dùng	Không Không Không Có Không Không
Phân tích yêu cầu	Phân tích tĩnh	1. Xác định các lớp 2. Xác định quan hệ giữa các lớp 3. Xây dựng biểu đồ lớp 4. Xác định thuộc tính lớp 5. Cập nhật bảng thuật ngữ và yêu cầu phi chức năng	Không Có Có Không Không
	Phân tích động	1. Xây dựng biểu đồ giao tiếp hay tuần tự 2. Gán phương thức cho các lớp	Có Không
Thiết kế	Hệ thống	1. Lựa chọn công nghệ mạng cho hệ thống 2. Thiết kế tương tranh và an toàn-bảo mật 3. Phân rã hệ thống thành các hệ thống con 4. Xây dựng biểu đồ gói	Không Không Không Có
	Chi tiết	1. Xây dựng mô hình lớp thiết kế 2. Xây dựng lược đồ cơ sở dữ liệu 3. Thiết kế giao diện người sử dụng	Có Không Không

Bảng 1.1: Các pha phát triển phần mềm

Mục đích của giáo trình này là tập trung trình bày ba công đoạn *xác định yêu cầu, phân tích yêu cầu và thiết kế* mà chủ yếu được thực hiện tập trung trong các pha khởi động, triển khai và xây dựng. Để tránh việc sử dụng nhiều thuật ngữ có thể dẫn đến nhầm lẫn, chúng ta sẽ gọi theo truyền thống các công đoạn này là các pha. Nghĩa là, từ bây giờ trở đi tài liệu này sẽ chỉ đề cập đến các pha phát triển phần mềm. Tóm lại, tài liệu này tập trung trình bày các pha:

- *Pha xác định yêu cầu* bao gồm hai giai đoạn: Xác định yêu cầu nghiệp vụ (mô hình nghiệp vụ hay miền ứng dụng hiện thời) và xác định yêu cầu hệ thống (mô hình hóa các yêu cầu cho hệ thống dự định phát triển).

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- *Pha phân tích yêu cầu* tập trung mô hình hóa phần mềm với UML để trả lời cho câu hỏi hệ thống mới có thể thực thi những chức năng nào.
- *Pha thiết kế* bao gồm hai giai đoạn: thiết kế hệ thống (phân rã hệ thống thành các hệ thống con và triển khai về mặt vật lý cho hệ thống) và thiết kế chi tiết (thiết kế các hệ thống con đã được phân rã).

Các pha sẽ tương ứng với một số bước cần phải thực hiện và các biểu đồ UML sẽ sử dụng được cho trong Bảng 1.1. Các pha, các bước và biểu đồ sẽ được trình bày chi tiết trong các Chương tiếp theo.

1.7 KẾT LUẬN

Chương này đã trình bày một số vấn đề cơ bản để làm cơ sở cho phát triển các hệ phần mềm hướng đối tượng. Nội dung tập trung xem xét khái niệm hệ thống và những vấn đề liên quan đến hệ thống; những kiểu hệ thống phần mềm; những khái niệm cơ bản của các hệ hướng đối tượng. Những khái niệm này đã được trình bày ngắn gọn với những ví dụ minh họa để bạn đọc hiểu được sâu sắc hơn về quan hệ giữa các lớp đã được học trong môn lập trình hướng đối tượng. Phần phương pháp luận trình bày vừa đủ để bạn đọc có thể hình dung các pha và công đoạn phát triển phần mềm. Cuối cùng, tài liệu trình bày các bước trong các pha xác định yêu cầu, phân tích và thiết kế mà sẽ là những chủ đề chính sẽ được trình bày đầy đủ hơn trong các chương sau.

BÀI TẬP

1. Tìm hiểu và khảo sát các hệ thương mại điện tử, hệ quản lý thư viện, hệ quản lý học tập theo tín chỉ. Liệt kê các tác nhân sử dụng hệ thống và các chức năng mà hệ thống này thực hiện.
2. Khảo sát các công nghệ, kỹ thuật để xây dựng các hệ thống thương mại điện tử như eBay, Amazon...
3. Trong ví dụ 1.1, tạo lớp mới *Fullname* gồm các thuộc tính *firstName*, *midName*, *lastName*. Hãy vẽ biểu đồ quan hệ *Fullname* và *Employee*. Hoàn thiện chương trình để có thể nhập thông tin họ và tên nhân viên và lưu vào cơ sở dữ liệu.
4. Hãy thêm các thuộc tính và các phương thức cập nhật, xóa nhân viên vào các lớp được chọn trong câu 3. Giải thích lý do chọn lựa các mức độ truy nhập các thuộc tính trên và lý do gán phương thức vào các lớp.
5. Sử dụng khái niệm *interface* và *abstract* trong Java để cài đặt các lớp với phương thức chèn thêm sinh viên, chèn thêm môn học, chèn thêm đăng ký học trong hệ quản lý học theo tín chỉ.
6. Tương tự như Câu 5 cho các chức năng khác với các hệ thống khác.
7. Hãy cho một ví dụ cho từng trường hợp sử dụng lại trình bày trong phần 1.4.
8. Sinh viên tham khảo thêm các tài liệu để viết một số ví dụ về mẫu thiết kế.

CHƯƠNG 1. CƠ SỞ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

9. Hãy thêm lớp mặt hàng sách Book với các thuộc tính tên sách, tác giả, nhà xuất bản, năm xuất bản, đơn giá vào quan hệ các lớp được trình bày trong phần 1.3.5.1.
10. Hãy thể hiện mã nguồn của câu 9 với Java.

CHƯƠNG 2

MÔ HÌNH HÓA HỆ PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

Chương này tập trung trình bày ngôn ngữ mô hình hóa thống nhất UML. Nội dung cụ thể bao gồm:

- Giới thiệu ngôn ngữ mô hình hóa thống nhất UML
- Các biểu đồ trong UML
- Giới thiệu công cụ phát triển phần mềm

2.1 GIỚI THIỆU VỀ UML

2.1.1 Vài nét lịch sử phát triển của UML

Các ngôn ngữ lập trình 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)... Tuy nhiên, mãi cho đến năm 1995, các nhóm nghiên cứu mới tập trung xem xét những phương pháp luận và ngôn ngữ mô hình với ký hiệu khác nhau, như Booch của Grady Booch, OMT của James Rumbaugh, OOSE của Ivar Jacobson, hay OOA & OOD của Coad và Yordon... Vấn đề làm thế nào xây dựng được một ngôn ngữ mô hình hóa thống nhất như một chuẩn cho những người phát triển phần mềm hướng đối tượng đã trở thành nhu cầu cấp thiết. 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 nhau xây dựng một Ngôn Ngữ Mô Hình Hoá Thống Nhất UML (Unified Modeling Language) (Hình 2.1). Phiên bản UML đầu tiên được đưa ra năm 1997 và sau đó được chuẩn hoá để trở thành phiên bản 1.0. UML đã không ngừng phát triển cho đến nay với phiên bản 2.4 (chi tiết tham khảo <http://www.omg.org/spec/UML/>). Phiên bản 2.0 với nhiều công cụ hỗ trợ đã được sử dụng rộng rãi trong công nghiệp phần mềm. Chương này tập trung trình bày ngôn ngữ mô hình hóa UML.

2.1.2 Một số đặc điểm của ngôn ngữ mô hình hóa UML

UML là ngôn ngữ mô hình hóa được xây dựng để đặc tả, phát triển và viết tài liệu cho các hệ phần mềm hướng đối tượng. Nó bao gồm một tập các khái niệm, các ký hiệu, các biểu đồ và hướng dẫn sử dụng. Mục đích của ngôn ngữ UML là: (i) Mô hình hóa các hệ thống bằng cách sử dụng các khái niệm hướng đối tượng; (ii) Thiết lập sự liên hệ từ nhận thức của con người đến các sự kiện cần mô hình hóa; (iii) Giải quyết vấn đề về mức độ thừa kế trong các hệ thống phức tạp với nhiều ràng buộc khác nhau; (iv) Tạo một ngôn ngữ mô hình hóa có thể sử dụng được bởi người và máy. UML hỗ trợ phân rã hệ hướng đối tượng dựa trên cấu trúc tĩnh và hành vi động của hệ thống.

- Các cấu trúc tĩnh (static structure) xác định các kiểu đối tượng quan trọng của hệ thống và mối quan hệ giữa các đối tượng đó nhằm đến cài đặt sau này.
- Các hành vi động (dynamic behavior) xác định các hành động của các đối tượng theo thời gian và tương tác giữa các đối tượng.

Những đặc điểm sau đây mô tả khái quát về UML:

- *Khung nhìn*: là một trùu tượng hóa để thể hiện những khía cạnh khác nhau của hệ thống. Mỗi khung nhìn gồm một số biểu đồ và các khung nhìn khác nhau sẽ cung cấp một bức tranh đầy đủ về hệ thống.
- *Biểu đồ*: các biểu đồ bao gồm các phần tử là đồ thị mô tả các nội dung theo khung nhìn. UML 2.4 có khoảng 14 kiểu biểu đồ khác nhau và tùy theo hệ thống khi kết hợp một số biểu đồ này sẽ cho ta toàn bộ khung nhìn hệ thống.
- *Các phần tử mô hình*: các khái niệm dùng trong biểu đồ là những phần tử mô hình nhằm biểu diễn các khái niệm hướng đối tượng quen thuộc như lớp, đối tượng, thông điệp và các quan hệ như tổng quát hóa, liên kết, phụ thuộc, kết hợp, hợp thành.
- *Cơ chế tổng quát*: các cơ chế tổng quát cung cấp cách mở rộng UML cho các miền đặc biệt và cách thêm chú thích, thông tin, ngữ nghĩa cho phần tử mô hình.
- *Kiến trúc hướng mô hình*: nhóm quản lý UML đặt UML trong một tập đặc điểm rộng hơn MDA (Model Driven Architecture). MDA tìm cách làm cho UML khả thi và tích hợp tốt hơn với các công cụ hỗ trợ phát triển.

2.1.3 Các khung nhìn trong UML

Rõ ràng rằng không thể sử dụng một biểu đồ duy nhất để thể hiện toàn bộ đặc trưng và hoạt động của một hệ thống phức tạp. Mỗi hệ thống có nhiều khía cạnh khác nhau: chức năng (cấu trúc tĩnh và tương tác động), phi chức năng (yêu cầu thời gian đáp ứng, tin cậy...) và những khía cạnh tổ chức (tổ chức hoạt động, phân chia phát triển các module...). Do đó, cần có các khung nhìn khác nhau để có thể mô tả đầy đủ hệ thống. Mỗi khung nhìn thể hiện bởi một số biểu đồ và phản ánh một phần của hệ thống qua khung nhìn đó. UML sử dụng 5 khung nhìn sau đây:

- *Khung nhìn ca sử dụng*: Khung nhìn này thể hiện chức năng của hệ thống đối với tác nhân bên ngoài. Tác nhân có thể là con người hay hệ thống khác. Khung nhìn ca sử dụng được sử dụng bởi khách hàng, nhân viên thiết kế, nhân viên phát triển và kiểm thử. Nó được mô tả bởi biểu đồ ca sử dụng và đôi khi cùng với biểu đồ hoạt động. Mỗi ca sử dụng là một mô tả tổng quát chức năng yêu cầu. Khung nhìn ca sử dụng là trung tâm và định hướng các khung nhìn khác.
- *Khung nhìn logic*: Khung nhìn này thể hiện cách thiết kế chức năng bên trong hệ thống theo cấu trúc và hành vi. Khung nhìn này chủ yếu dành cho người thiết kế và phát triển phần mềm. Nó mô tả cấu trúc tĩnh (lớp, đối tượng và quan hệ) và mô hình động qua tương tác bằng phương thức giữa các đối tượng. Cấu trúc tĩnh được mô tả

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

bởi các biểu đồ lớp và đối tượng; mô hình động được mô tả bởi máy trạng thái, biểu đồ tương tác và biểu đồ hoạt động.

- *Khung nhìn cài đặt:* Khung nhìn này thể hiện cách tổ chức mã nguồn và mã thực thi. Nó mô tả những môđun chính và phụ thuộc của chúng. Nó chủ yếu dành cho người phát triển và bao gồm các sản phẩm phần mềm như các môđun mã nguồn với cấu trúc và phụ thuộc hay thông tin quản lý và báo cáo tiến độ...
- *Khung nhìn tiến trình:* Khung nhìn này chỉ ra các phần tử chính như thông lượng, khả năng mở rộng...trong hệ thống liên quan đến hiệu năng của tiến trình. Nó liên quan đến tính phi chức năng của hệ thống. Khung nhìn này thể hiện bởi các biểu đồ động (máy trạng thái, các biểu đồ hoạt động và tương tác), biểu đồ cài đặt (các biểu đồ tương tác và triển khai) và biểu đồ thời điểm.
- *Khung nhìn triển khai:* Khung nhìn này thể hiện việc triển khai hệ thống trên các máy tính và các thiết bị. Nó được sử dụng bởi các nhân viên phát triển, tích hợp và kiểm thử. Khung nhìn này được thể hiện bởi biểu đồ triển khai.

Các công cụ hiện nay cho phép chúng ta vẽ và quản lý các biểu đồ cũng như chuyển từ khung nhìn này sang khung nhìn khác.

2.1.3 Các khái niệm cơ bản trong UML

Khái niệm mô hình

Mô hình (model) là một biểu diễn của sự vật, đối tượng hay một tập các sự vật trong một lĩnh vực ứng dụng theo một quan điểm nào đó. Mục đích của mô hình:

- Mô tả các đặc điểm hay khía cạnh quan trọng của sự vật cần quan tâm và biểu diễn theo một tập ký hiệu hoặc quy tắc nào đó. Do đó, cùng một đối tượng nhưng trong hai hệ thống khác nhau có thể có các đặc điểm cần quan tâm khác nhau.
- Thể hiện các mô tả bởi những biểu thức toán học hoặc bởi các biểu đồ dựa trên một tập ký hiệu và quy tắc xác định. Trong hệ phần mềm hướng đối tượng, các biểu đồ UML được sử dụng rộng rãi để mô hình hệ thống và sẽ được trình bày trong chương này.

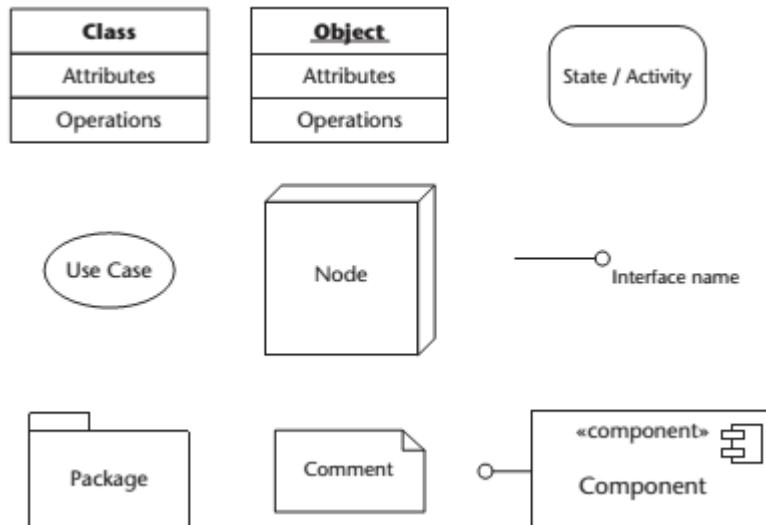
Ví dụ, khi xây dựng Hệ quản lý bán hàng, ta chỉ cần quan tâm đến các thuộc tính như họ tên, địa chỉ, phone, email...của đối tượng khách hàng. Trong khi xây dựng hệ Quản lý Học tập theo tín chỉ ngoài các thông tin liên quan đến đối tượng sinh viên như họ tên, địa chỉ, email, phone...như của khách hàng, ta còn phải quan tâm đến các thuộc tính như điểm, lớp học, môn học, khoa mà sinh viên đăng ký.

Các phần tử mô hình

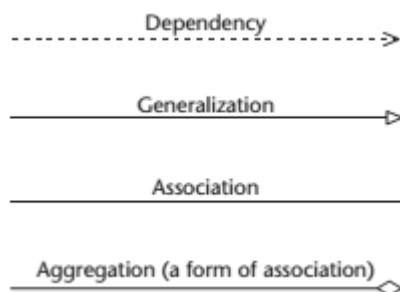
Một số ký hiệu để mô hình hóa hướng đối tượng thường gặp trong UML được biểu diễn trong Hình 2.2. Lớp (class) và đối tượng (object) được mô tả bởi một hộp với 3 gian là tên lớp, thuộc tính (attributes), thao tác (operations) trong đó thể hiện đối tượng có gạch chân. Ca sử dụng (use case) thể hiện bởi hình quả trám gắn với tên ca sử dụng. Trạng thái (state) hay hoạt động (activity) biểu diễn bởi hình chữ nhật góc tròn.

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

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 ca sử dụng, quan hệ trong biểu đồ lớp...Hình 2.3 biểu diễn một số quan hệ giữa các lớp: phụ thuộc (dependency), tổng quát hóa (generalization), liên kết (association) và kết hợp (aggregation).



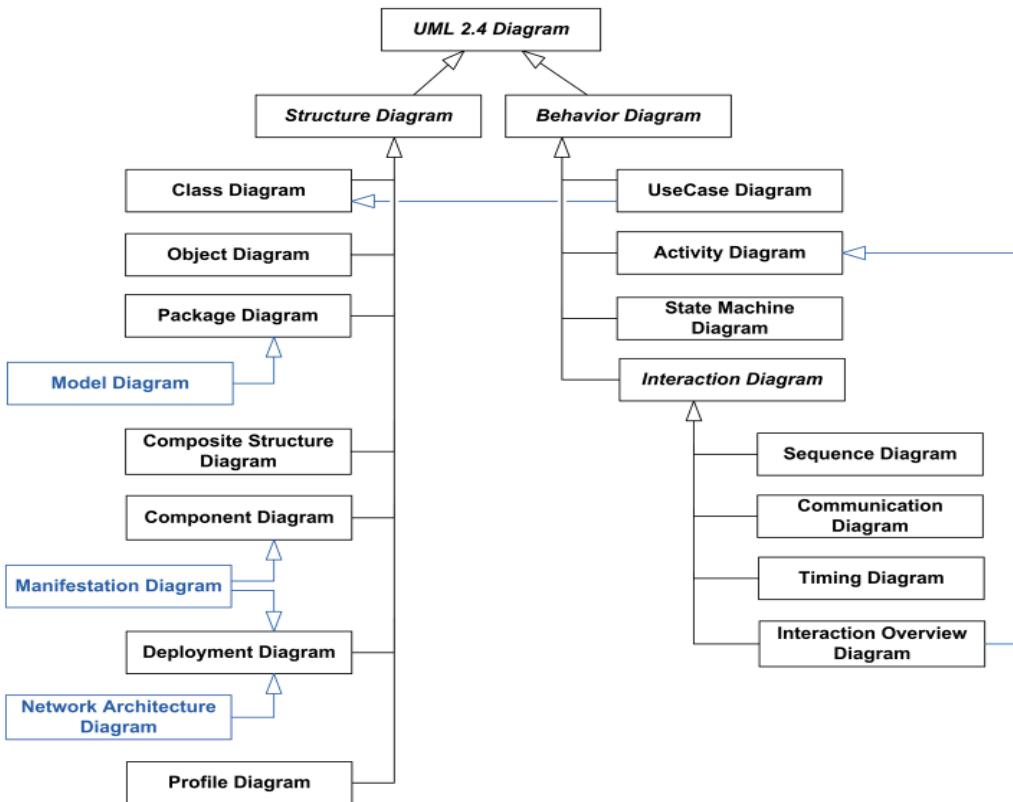
Hình 2.2: Một số phần tử mô hình thường gặp trong UML



Hình 2.3: Một số quan hệ giữa các lớp

2.2 CÁC BIỂU ĐỒ TRONG UML

UML chia các biểu đồ thành hai nhóm: Nhóm các biểu đồ cấu trúc và Nhóm các biểu đồ hành vi. Chúng được biểu diễn thành cây phân cấp như trong Hình 2.4.



Hình 2.4: Cây phân cấp các biểu đồ UML

Biểu đồ cấu trúc (Structure Diagram): Nhóm các biểu đồ này biểu diễn các cấu trúc tĩnh của hệ phần mềm cần được mô hình hóa. Các biểu đồ trong mô hình tĩnh tập trung biểu diễn khía cạnh tĩnh liên quan đến cấu trúc cơ bản cũng như các phần tử chính của hệ thống. UML đề xuất bảy dạng biểu đồ trong mô hình tĩnh bao gồm:

- Biểu đồ lớp (class diagram)
- Biểu đồ đối tượng (object diagram)
- Biểu đồ thành phần (component diagram)
- Biểu đồ gói (package diagram)
- Biểu đồ triển khai (deployment diagram)
- Biểu đồ cấu trúc phức hợp (composite structure diagram)
- Biểu đồ gói mở rộng (profile package)

Biểu đồ hành vi (Behavior Diagram): Nhóm biểu đồ này nhằm thể hiện các hoạt động và hành vi của hệ thống, cũng như tương tác giữa các phần tử bên trong và bên ngoài hệ thống. UML đề xuất bảy dạng biểu đồ trong mô hình hành vi bao gồm:

- Biểu đồ ca sử dụng (use case diagram)
- Biểu đồ hoạt động (activity diagram)
- Biểu đồ tuần tự (sequence diagram)
- Biểu đồ giao tiếp (communication diagram)
- Biểu đồ trạng thái (state machine diagram)

- Biểu đồ tương tác tổng quát (interaction overview diagram)
- Biểu đồ thời khắc (timing diagram)

Trong phần tiếp theo chúng ta sẽ lần lượt xem xét chi tiết một số biểu đồ UML thường gặp, mỗi biểu đồ sẽ được trình bày về ý nghĩa, tập kí hiệu UML cho biểu đồ đó và ví dụ minh họa.

2.2.1 Biểu đồ ca sử dụng

Ý nghĩa

Biểu đồ ca sử dụng (use case diagram) biểu diễn các chức năng của hệ thống và thể hiện sự tương tác giữa các tác nhân và hệ thống cần xây dựng. Nó bao gồm một tập các *tác nhân* (actor), các *ca sử dụng* (use case) và *các mối quan hệ* (relationship) giữa các ca sử dụng. Tác nhân có thể là con người, một hệ thống khác cung cấp thông tin hay tác động tới hệ thống.

Cách xây dựng

Biểu đồ ca sử dụng có thể được phân rã theo nhiều mức khác nhau. Từ tập yêu cầu xác định được, biểu đồ ca sử dụng sẽ 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 ca sử dụng là các *kịch bản* (scenario) nhằm mô tả chi tiết quá trình thực hiện ca sử dụng đó.

Mô hình chức năng với biểu đồ ca sử dụng được xây dựng qua quá trình thảo luận giữa bên phát triển và khách hàng/người sử dụng. Biểu đồ này được xem là một thỏa thuận về đặc tả yêu cầu giữa các bên liên quan.

Các đối tượng quan tâm ca sử dụng

- Khách hàng/người sử dụng: Các biểu đồ ca sử dụng mô tả các chức năng và cách sử dụng hệ thống nên cần thiết cả cho nhân viên bán hàng, nhóm hỗ trợ hay viết tài liệu.
- Người phát triển: Cần có ca sử dụng để hiểu hệ thống phải thực hiện chức năng gì và là cơ sở để mô hình chi tiết hệ thống cũng như mã hóa sau này.
- Quản lý dự án: Cần ca sử dụng để xây dựng kế hoạch từ phân tích đến thiết kế và cài đặt cho vòng đời phát triển phần mềm.
- Nhóm kiểm thử và tích hợp hệ thống: Cần ca sử dụng để kiểm tra các chức năng trong hệ thống có hoạt động như thể hiện trong ca sử dụng hay không.

Tập kí hiệu UML cho biểu đồ ca sử dụng

Chúng ta sẽ lần lượt xem xét các phần tử mô hình này:

- *Hệ thống*: Với vai trò là thành phần của biểu đồ ca sử dụng, hệ thống thể hiện ranh giới giữa phần mềm chúng ta đang xây dựng và phần bên ngoài của hệ thống. Một hệ thống trong biểu đồ ca sử dụng không phải bao giờ cũng nhất thiết là một hệ phần mềm; nó có thể là một chiếc máy, hoặc là một hệ thống thực (như một hệ quản lý doanh nghiệp hay trường đại học, ...).
- *Tác nhân*: là người sử dụng 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. Một

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

tác nhân có thể thực hiện nhiều ca sử dụng và ngược lại một ca sử dụng cũng có thể được thực hiện bởi nhiều tác nhân. Các tác nhân có thể có quan hệ tổng quát hóa.

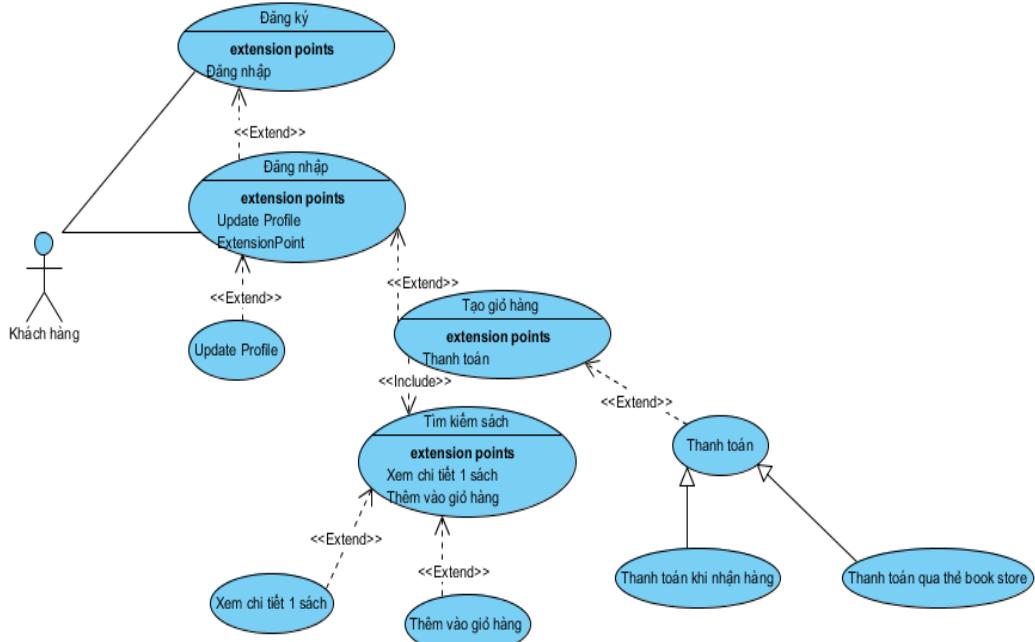
- *Các ca sử dụng:* Đây là thành phần cơ bản của biểu đồ ca sử dụng. Các ca sử dụng được biểu diễn bởi các hình elip thể hiện một chức năng xác định của hệ thống.
- *Quan hệ giữa các ca sử dụng:* giữa các ca sử dụng có thể có các quan hệ như sau:
 - *Bao hàm (Include):* Ca sử dụng UC1 có một số bước được cung cấp bởi ca sử dụng UC2 thì ta bảo UC1 bao hàm hay chứa UC2.
 - *Mở rộng (Extend):* Ca sử dụng UC1 mở rộng ca sử dụng UC2 bằng cách thêm vào một số chức năng cụ thể nào đó.
 - *Đặc biệt hóa (Specialization):* Ca sử dụng UC1 kế thừa các chức năng từ ca sử dụng UC2 thì UC1 gọi là *đặc biệt hóa* của UC2 và UC2 là gọi là *tổng quát hóa* (Generalization) của UC1.

Các phần tử mô hình ca sử dụng cùng với ý nghĩa và cách biểu diễn của nó được cho trong Bảng 2.1. Việc xác định quan hệ giữa các ca sử dụng là khá tinh tế, phụ thuộc vào yêu cầu hệ thống và ngữ nghĩa cần thể hiện. Bạn đọc sẽ hiểu rõ hơn các quan hệ giữa các ca sử dụng sẽ được trình bày sau này trong Pha xác định yêu cầu ở Chương 3.

Phần tử mô hình	Ý nghĩa	Cách biểu diễn	Ký hiệu trong biểu đồ
Ca sử dụng	Biểu diễn một chức năng xác định của hệ thống	Hình elip chứa tên của ca sử dụng	
Tác nhân	Là một đối tượng bên ngoài hệ thống tương tác trực tiếp với các use case	Biểu diễn hình người	
Quan hệ giữa các ca sử dụng	Tùy từng dạng quan hệ	Extend và include có dạng các mũi tên kèm theo tên. 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 2.1: Các phần tử mô hình trong biểu đồ ca sử dụng

Ví dụ 2.1: Trong hệ thống quản lý mua-bán sách trực tuyến, khách hàng có thể sử dụng hệ thống để đăng ký, đăng nhập, tạo giỏ hàng, tìm kiếm sách và thanh toán. Để được mua sách, khách hàng phải là thành viên hệ thống và phải đăng nhập trước. Các ca sử dụng và quan hệ của chúng được cho như Hình 2.5.



Hình 2.5: Biểu đồ ca sử dụng

2.2.2 Biểu đồ lớp

Ý nghĩa

Trong phương pháp hướng đối tượng, các đối tượng có chung một số thuộc tính và phương thức được nhóm thành một lớp. 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* (class diagram). Biểu đồ lớp là thể hiện mô hình tĩnh của hệ thống nhằm mô tả khung nhìn về một hệ thống bằng các khái niệm lớp, các thuộc tính, phương thức của lớp và mối quan hệ giữa chúng với nhau.

Tập kí hiệu UML cho biểu đồ lớp

Trong phần này, chúng ta sẽ xem xét các vấn đề liên quan đến biểu diễn biểu đồ lớp trong UML.

- Kí hiệu lớp*: trong UML, mỗi lớp được biểu diễn bởi hình chữ nhật gồm ba gian: 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:

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

phạm vi cho biết phạm vi truy nhập của thuộc tính. Có bốn kiểu xác định phạm vi thuộc tính 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 (+, #, -, ~) như trong UML hoặc biểu diễn dưới dạng các từ khoá (public, protected, private) như trong các ngôn ngữ lập trình.

Tên: là xâu 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:

phạm vi tên Phương thức(danh sách tham số): kiểu trả lại { kiểu Phương thức}

Trong đó:

phạm vi biểu diễn phạm vi cho phương thức. Giống như đối với thuộc tính, có bốn 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 xâu 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*.

kiểu Phương thức: không bắt buộc, cho biết kiểu phương thức. Phương thức có thể có 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 ba kiểu lớp dựa trên vai trò của nó trong hệ thống bao gồm:

- *Lớp thực thể (entity class)*: là lớp đại diện cho các thực thể thuộc miền quan tâm và chứa thông tin về các đối tượng xác định nào đó. Ví dụ, Khách hàng, Hóa đơn, Sinh viên, Môn học
- *Lớp biên (boundary class)*: là lớp nằm ở ranh giới giữa hệ thống với môi trường bên ngoài nhằm 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 (control class)*: thực hiện các chức năng điều khiển hoạt động của hệ thống tương ứng với các chức năng cụ thể nào đó của một nhóm các lớp biên hoặc nhóm các lớp thực thể.

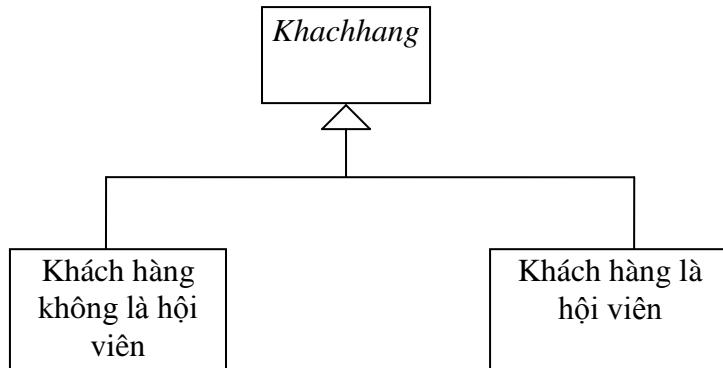
STT	Kiểu lớp	Kí hiệu UML
1	<i>Lớp thực thể</i>	
2	<i>Lớp biên (lớp giao diện)</i>	
3	<i>Lớp điều khiển</i>	

Bảng 2.2: Các kiểu lớp trong UML

Các quan hệ trong biểu đồ lớp thực thể

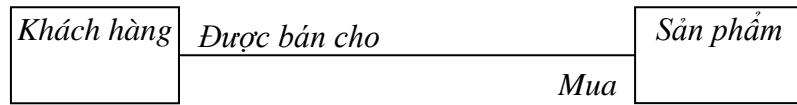
Như đã trình bày trong Chương 1, giữa các lớp thực có thể có bốn dạng quan hệ cơ bản. Phần tiếp theo sẽ trình bày chi tiết hơn cách biểu diễn quan hệ này trong UML:

- *Ké thừa (Inheritance)*: Ké thừa là mối quan hệ giữa một lớp có các đặc trưng mang tính khái quát cao hơn và một lớp có các tính chất đặc biệt hơn. Trong biểu đồ lớp, quan hệ kế thừa được biểu diễn bằng một mũi tên có tam giác rỗng gắn ở đầu. Ví dụ Hình 2.6, Khách hàng *Khachhang* có 2 lớp kế thừa từ nó là Khách hàng hội viên *KhachhangHoivien* và không hội viên



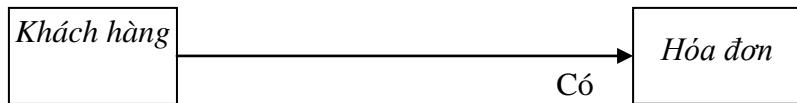
Hình 2.6: Quan hệ kế thừa

- **Liên kết (Association):** Một liên kết (association) là một sự kết nối giữa các lớp, cũng có nghĩa là sự kết nối giữa các đối tượng của các lớp này. Trong UML, quan hệ này nhằm mô tả mối liên quan về mặt ngữ nghĩa (semantic) giữa hai nhóm đối tượng được biểu diễn bởi các lớp tương ứng. Quan hệ liên kết được biểu diễn bởi đoạn thẳng hai chiều nối hai đối tượng và có thể kèm theo nghĩa của quan hệ tại hai đầu của đoạn thẳng. Ví dụ Hình 2.7, lớp khách hàng có quan hệ liên kết với lớp sản phẩm qua quan hệ mua, bán. Ngữ nghĩa của quan hệ này thể hiện ở chỗ: khách hàng *mua* sản phẩm, còn sản phẩm *được bán cho* khách hàng.



Hình 2.7: Quan hệ liên kết hai chiều

Quan hệ liên kết cũng có thể có dạng một chiều. Xem ví dụ Hình 2.8.



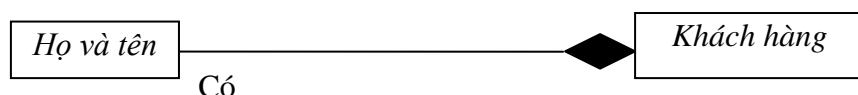
Hình 2.8: Quan hệ liên kết một chiều

- **Kết hợp (Aggregation):** là dạng quan hệ mô tả một lớp A là một bộ phận của lớp B và lớp A có thể tồn tại độc lập. Quan hệ kết hợp được biểu diễn bằng một mũi tên gắn hình thoi rỗng ở đầu hướng về lớp bao hàm. Xem ví dụ Hình 2.9. Lớp Địa chỉ là một phần của lớp Khách hàng nhưng đối tượng Địa chỉ vẫn có thể tồn tại độc lập với đối tượng khách hàng.



Hình 2.9: Quan hệ kết hợp

- **Hợp thành (Composition):** Quan hệ hợp thành biểu diễn một quan hệ kiểu tổng thể-bộ phận nhưng mạnh hơn quan hệ kết hợp. Lớp A có quan hệ hợp thành 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 2.10.



Hình 2.10: Quan hệ hợp thành

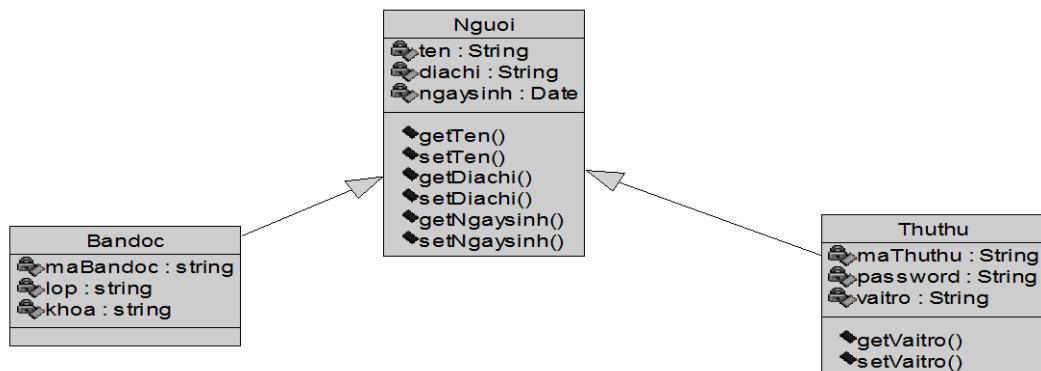
Bảng 2.3 tổng kết các phần tử trong ngôn ngữ mô hình UML được sử dụng để mô hình lớp, ý nghĩa và ký hiệu tương ứng trong các biểu đồ.

Phần tử mô hình	Ý nghĩa	Cách biểu diễn	Ký hiệu trong biểu đồ
<i>Lớp (class)</i>	Biểu diễn tên lớp, các thuộc tính và phương thức của lớp đó.	Một hình chữ nhật gồm 3 phần tách biệt.	Tên lớp
			Các thuộc tính
			Các phương thức
<i>Quan hệ kế thừa</i>	Lớp này thừa hưởng các thuộc tính - phương thức của lớp kia	Mũi tên tam giác.	
<i>Quan hệ kiểu liên kết</i>	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.	
<i>Quan hệ kết hợp</i>	Biểu diễn quan hệ kiểu bộ phận - tổng thể.	Đường kẻ liền nét có hình thoi ở đầu.	

<i>Quan hệ hợp thành</i>	Biểu diễn quan hệ kiểu bộ phận – tổng thể mạnh.	Đường kẻ liền nét có hình thoi đặc ở đầu.	
--------------------------	---	---	--

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

Ví dụ 2.2: Hình 2.11 biểu diễn một phần của biểu đồ lớp trong Hệ thống quản lý thư viện trong đó các lớp Thủ thư *Thuthu* và Bạn đọc *Bandoc* kế thừa từ lớp tổng quát là lớp Người *Nguoi*.



Hình 2.11: Biểu đồ lớp

2.2.3 Biểu đồ trạng thái

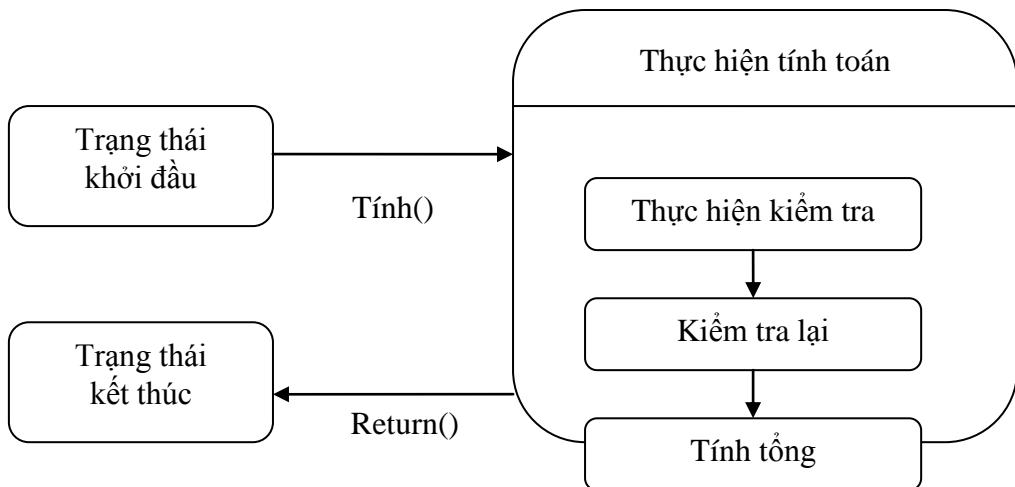
Ý nghĩa

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. Thông thường, mỗi lớp sẽ có một biểu đồ trạng thái (trừ lớp trừu tượng là lớp không có đối tượng) được biểu diễn dưới dạng máy trạng thái hữu hạn với các trạng thái và sự chuyển tiếp giữa các trạng thái đó.

Tập kí 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 thẻ mô 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. Ví dụ có trạng thái con thể hiện trong Hình 2.12.



Hình 2.12: 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 một biểu đồ trạng thái của UML có thể có một số loại sự kiện sau đây:

- *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.

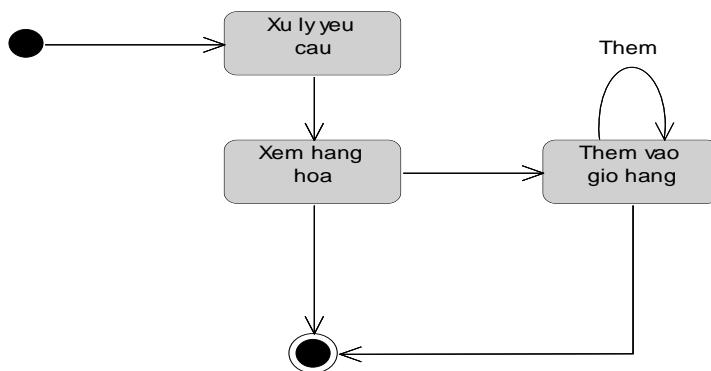
Các phần tử mô hình UML và ký hiệu tương ứng cho biểu đồ trạng thái được tổng kết như trong Bảng 2.5.

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 tròn 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</i>	Kết thúc vòng đời	Hai hình tròn lồng	

kết thúc	của đối tượng.	nhau	
Chuyển tiếp (transition)	Chuyển từ trạng thái này sang trạng thái khác	Mũi tên liền nét với tên gọi để biểu diễn chuyển tiếp.	chuyển tiếp →

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

Ví dụ 2.3: Để minh họa cho biểu đồ trạng thái, ta xét một hệ thống mua bán hàng. Một khách hàng gửi yêu cầu cần tìm mua một số mặt hàng đến hệ thống. Đối tượng hàng hóa được tạo ra và chuyển sang trạng thái *xử lý yêu cầu*. Tùy thuộc vào yêu cầu, đối tượng này sẽ chuyển sang trạng thái *xem hàng hóa*. Với mỗi hàng hóa được lựa chọn, đối tượng sẽ chuyển sang trạng thái *Thêm vào giỏ hàng* cho đến khi kết thúc quá trình lựa chọn.



Hình 2.13: Biểu đồ trạng thái

2.2.4 Biểu đồ tuần tự

Ý nghĩa

UML cung cấp bốn kiểu biểu đồ tương tác: Biểu đồ tuần tự, biểu đồ giao tiếp, biểu đồ tương tác tổng quát và biểu đồ thời khắc. *Biểu đồ tuần tự* biểu diễn tương tác giữa các đối tượng trong hệ thống và giữa các đối tượng với các tác nhân bên ngoài theo thời gian. Biểu đồ tuần tự nhấn mạnh thứ tự thực hiện của các tương tác.

Tập kí 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 tất cả các đối tượng đều cùng lúc xuất hiện trên một biểu đồ mà chúng chỉ xuất hiện (về mặt thời gian) khi thực sự tham gia vào tương tác.
- *Các thông điệp:* được biểu diễn bằng các mũi tên hướng từ đối tượng gửi sang đối tượng nhận. Tên các thông điệp 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

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

thức). Rõ ràng rằng *biểu diễn dưới dạng hình thức* sẽ rất hữu ích cho việc xác định các *phương thức cho lớp*. Biểu đồ tuần tự cho phép thể hiện các thông điệp từ một đối tượng tới chính bản thân nó.

- Có thể sử dụng từ khóa *alt* để thể hiện hành vi có điều kiện và được đặt một trong khung hình chữ nhật tách biệt 2 phần *if...else* bởi đường đứt nét -----
- Có thể sử dụng từ khóa *loop* để thể hiện hành vi lặp đặt trong khung hình chữ nhật.
- Trong biểu đồ tuần tự có thể có nhiều loại thông điệp khác nhau tuỳ theo mục đích sử dụng và tác động của thông điệp đến đối tượng. Các dạng thông điệp được tổng kết trong Bảng 2.6 dưới đây:

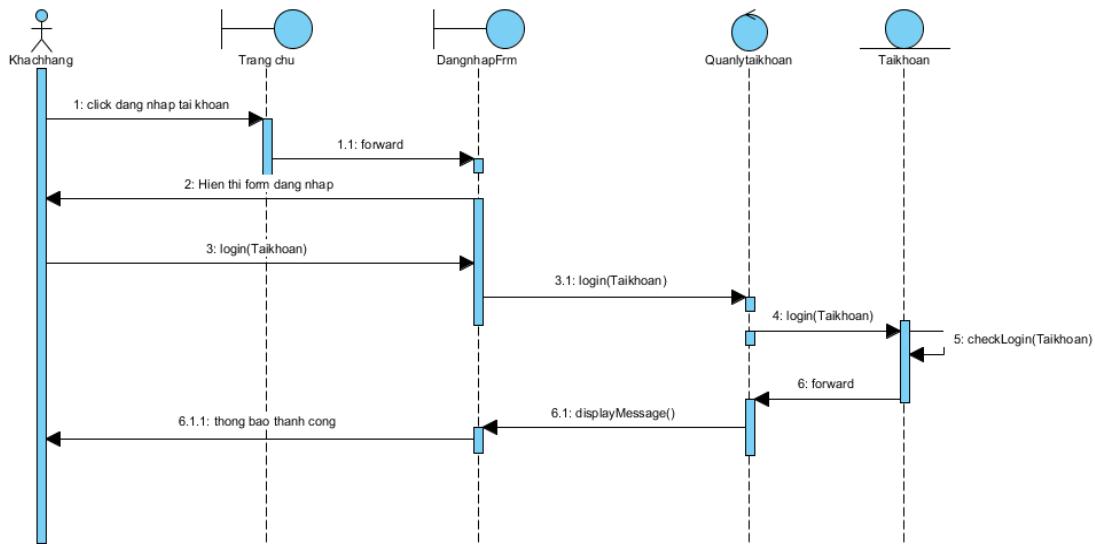
<i>STT</i>	<i>Loại thông điệp</i>	<i>Mô tả</i>	<i>Biểu diễn</i>
1	<i>Gọi (call)</i>	Mô tả một lời gọi từ đối tượng này đến đối tượng kia.	method()
2	<i>Trả về (return)</i>	Trả về giá trị ứng với lời gọi	← - - Giá trị trả về - →
3	<i>Gửi (send)</i>	Gửi một tín hiệu tới một đối tượng	send()
4	<i>Tạo (create)</i>	Tạo một đối tượng	<<create>>
5	<i>Huỷ (destroy)</i>	Huỷ một đối tượng	<<destroy>>

Bảng 2.6: Các dạng thông điệp trong biểu đồ tuần tự

- *Đường vòng đời*: là một đường kẻ nối dài phía dưới đối tượng, mô tả quá trình tồn tại của đối tượng trong quá trình tương tác.
- *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.

Ví dụ 2.4: Dưới đây là một ví dụ biểu đồ tuần tự cho chức năng Đăng nhập trong một hệ thống quản lý bán hàng. Các đối tượng tham gia trong tương tác là khách hàng, giao diện đăng nhập, điều khiển quản lý tài khoản và thực thể tài khoản. Xem Hình 2.14

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG



Hình 2.14: Biểu đồ tuần tự đăng nhập

2.2.5 Biểu đồ giao tiếp

Ý nghĩa

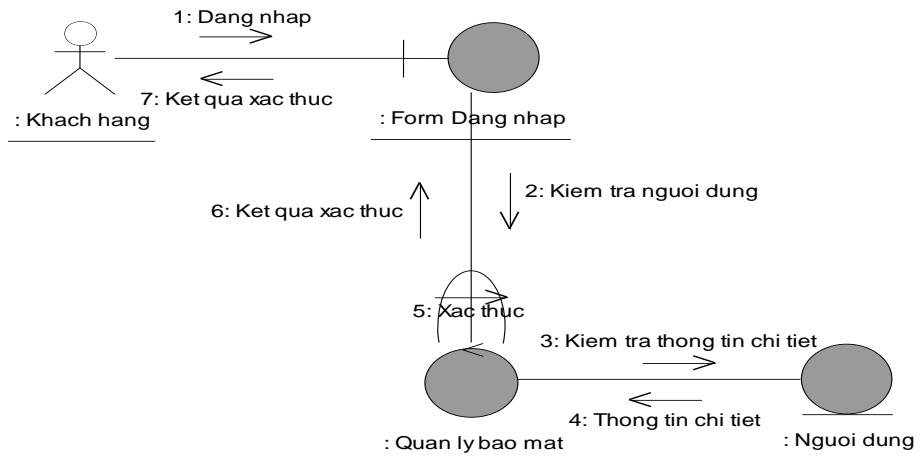
Biểu đồ giao tiếp là loại biểu đồ tương tác biểu diễn mối quan hệ giữa các đối tượng, giữa đối tượng và tác nhân. Biểu đồ giao tiếp cũng có các thông điệp với nội dung tương tự như trong biểu đồ tuần tự. Tuy nhiên, các đối tượng được đặt một cách tự do trong không gian của biểu đồ và không có đường vòng đời cho các đối tượng; các thông điệp được đánh số thể hiện thứ tự thời gian.

Tập kí hiệu UML cho biểu đồ giao tiếp

Các thành phần cơ bản của một biểu đồ giao tiếp 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 giao tiếp, các đối tượng tham gia tương tác luôn xuất hiện tại một vị trí xác định.
- *Các liên kết*: hai đối tượng có tương tác sẽ có một liên kết nối hai đối tượng đó.
- *Các thông điệp*: được biểu diễn bằng các mũi tên hướng từ đối tượng gửi sang đối tượng nhận bên cạnh liên kết giữa hai đối tượng đó. Trong biểu đồ giao tiếp, các thông điệp được đánh số theo thứ tự xuất hiện trong kịch bản mô tả ca sử dụng tương ứng.

Ví dụ 2.5: Hình 2.15 là một biểu đồ giao tiếp mô tả chức năng đăng nhập.



Hình 2.15: Biểu đồ giao tiếp

2.2.6 Biểu đồ hoạt động

Ý nghĩa

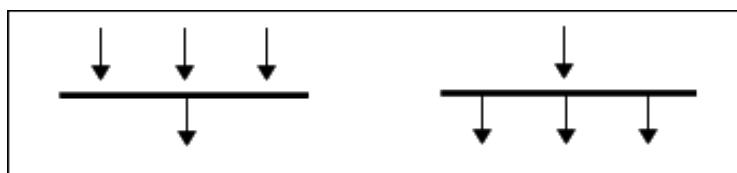
Biểu đồ hoạt động biểu diễn các hoạt động và sự đồng bộ, chuyển tiếp các hoạt động của hệ thống trong một lớp hoặc kết hợp giữa các lớp với nhau trong một chức năng cụ thể. Biểu đồ hoạt động có thể được sử dụng cho nhiều mục đích khác nhau, ví dụ như:

- Để xác định các hành động phải thực hiện trong phạm vi một phương thức.
- Để xác định công việc cụ thể bên trong 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 thế nào đến những đối tượng xung quanh.

Tập kí 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 phương thức 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.

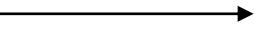


Hình 2.16: Thanh đồng bộ hóa trong biểu đồ động

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

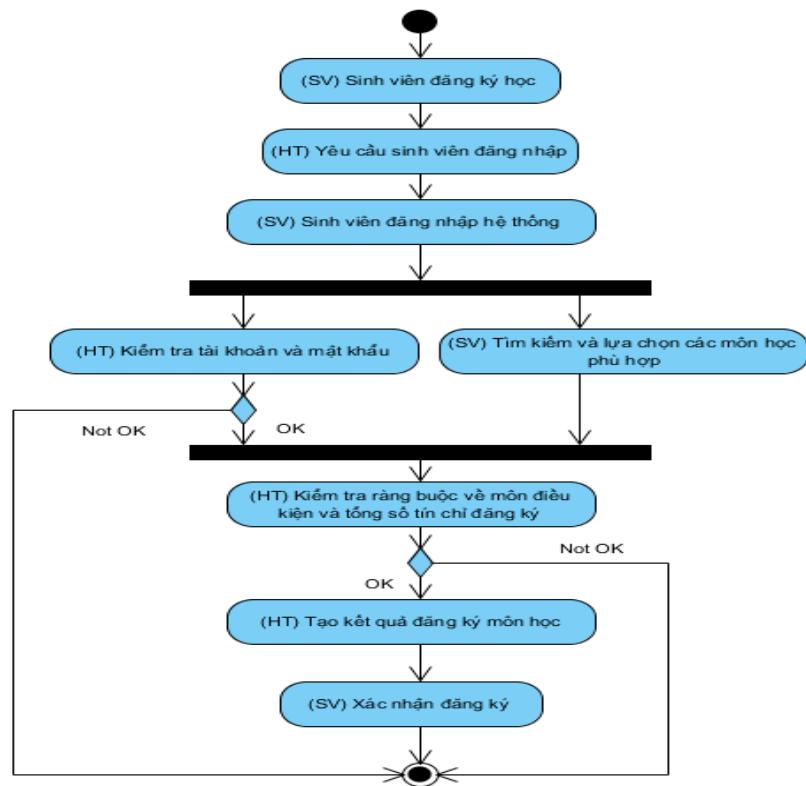
- *Điều kiện (Guard Condition)*: các biểu thức logic có giá trị đúng hoặc sai. Điều kiện được thể hiện trong ngoặc vuông, ví dụ: [Customer existing].
- *Các luồng (swimlane)*: Mỗi biểu đồ hoạt động có thể biểu diễn sự phối hợp hoạt động trong nhiều lớp khác nhau. Khi đó mỗi lớp được phân tách bởi một luồng (swimlane) riêng biệt và 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 đọc</i>	Mô tả thanh đồng bộ theo chiều thẳng đứng	
<i>Chuyển tiếp</i>	Mô tả sự chuyển tiếp	
<i>Quyết định</i>	Mô tả một lựa chọn điều kiện.	
<i>Các luồng</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 2.7: Các phần tử của biểu đồ hoạt động

Ví dụ 2.6: Dưới đây là ví dụ biểu đồ hoạt động mô tả chức năng sinh viên đăng ký môn học theo tín chỉ (Hình 2.17).



Hình 2.17: Biểu đồ hoạt động

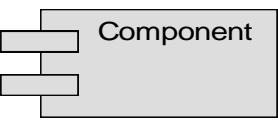
2.2.7 Biểu đồ thành phần

Ý nghĩa

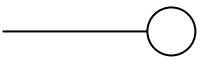
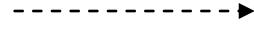
Biểu đồ thành phần được sử dụng để biểu diễn các thành phần phần mềm cấu thành nên hệ thống. Một hệ phần mềm có thể được xây dựng từ đầu bằng cách sử dụng mô hình lớp như đã trình bày trong các phần trước của tài liệu, hoặc cũng có thể được tạo nên từ các thành phần sẵn có. Mỗi thành phần có thể coi như một phần mềm cung cấp một số chức năng dưới dạng một hộp đen để xây dựng phần mềm lớn hơn. Nói cách khác, các thành phần là các gói được xây dựng để phát triển 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.

Tập kí hiệu UML

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

Phần tử mô hình	Ý nghĩa	Ký hiệu trong biểu đồ
<i>Thành phần</i>	Mô tả một thành phần của biểu đồ, mỗi thành phần có thể chứa nhiều lớp hoặc nhiều chương trình con.	 Component

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

<i>Giao tiếp</i>	Mô tả giao tiếp gắn với mỗi thành phần.	
<i>Mối quan hệ phụ thuộc giữa các thành phần</i>	Mối quan hệ phụ thuộc giữa các thành phần	

Bảng 2.8: Các ký hiệu của biểu đồ thành phần

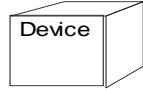
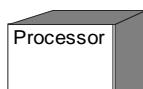
2.2.8 Biểu đồ triển khai

Ý nghĩa

Biểu đồ triển khai biểu diễn kiến trúc cài đặt và triển khai hệ thống dưới dạng các đỉnh và các mối quan hệ giữa các đỉnh đó. Thông thường, các đỉnh được kết nối với nhau thông qua các liên kết truyền thông như các giao thức kết nối mạng TCP/IP, IIOP...

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 CÔNG CỤ PHÁT TRIỂN PHẦN MỀM

Cho đến nay có rất nhiều công cụ hỗ trợ phát triển phần mềm. VP (Visual Paradigm (<http://www.visual-paradigm.com>) là một trong các bộ công cụ được sử dụng cho phát triển các hệ phần mềm hướng đối tượng theo ngôn ngữ mô hình hóa UML. VP cho phép chúng ta thiết kế hệ thống với tất cả các loại biểu đồ UML. VP-UML hỗ trợ các tiêu chuẩn mô hình chủ chốt như Unified Modeling Language (UML) 2.4, SoaML, SysML, ERD, DFD, BPMN 2.0, ArchiMate 2.0... Công cụ này hỗ trợ nhóm phát triển phần mềm trong việc xây dựng các biểu đồ, xác định yêu cầu, xây dựng ca sử dụng, lập kế hoạch cho phần mềm, mô hình hóa các lớp, mô hình hóa dữ liệu... Ngoài ra, VP còn cung cấp chức năng hỗ trợ quản lý dự án phát

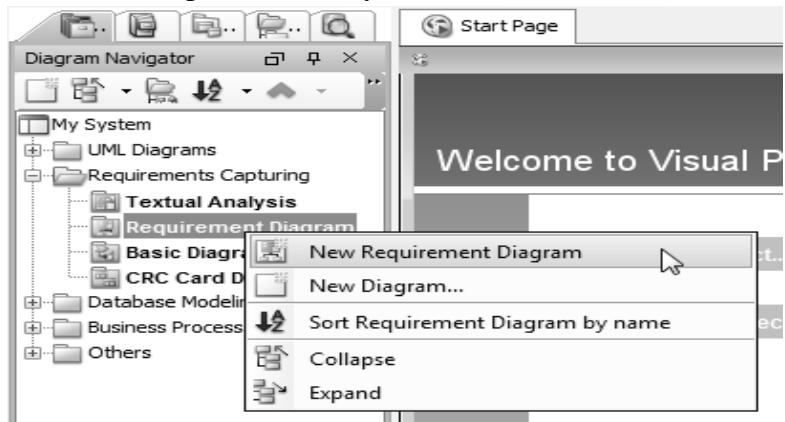
triển phần mềm, cung cấp các thư viện để hỗ trợ sinh khung mã cho hệ thống theo một ngôn ngữ lập trình nào đó. Bật VP-UML trong một workspace mới thì màn hình khởi động của VP 8.0 sẽ hiện ra như trong Hình 2.19.



Hình 2.18: Màn hình khởi động của VP

Tạo một project mới bằng cách chọn **File > New Project** từ menu chính và đặt tên cho project. Trong cửa sổ Diagram Navigator chứa nhiều View, mỗi view sẽ cho phép mô hình hoá hệ thống theo nhiều khía cạnh khác nhau. Cụ thể:

- **Các biểu đồ UML:** Mô hình chức năng hệ thống ở cấp cao với biểu đồ ca sử dụng, biểu đồ lớp, biểu đồ tuần tự, biểu đồ hoạt động... Cách tiếp cận mô hình hoá hệ phần mềm dựa trên ca sử dụng giúp cho nhóm phát triển phần mềm tập trung vào những gì mà người sử dụng muốn có hơn là các tính năng cần phát triển. VP-UML hỗ trợ tất cả các kí hiệu chuẩn nhất của các biểu đồ trong UML.
- **Xác định yêu cầu:** Nắm bắt yêu cầu hệ thống với các ca sử dụng, phân tích văn bản. Biểu đồ yêu cầu hỗ trợ xác định khả năng và các điều kiện phải được gán cho đối tượng. Khả năng thường đề cập đến các chức năng mà hệ thống phải hỗ trợ và gọi là yêu cầu chức năng. Điều kiện thường có nghĩa là hệ thống có thể chạy hoặc tạo ra kết quả với ràng buộc cụ thể gọi là yêu cầu phi chức năng. VP cung cấp một biểu đồ yêu cầu SysML để xác định và phân tích các yêu cầu.



Hình 2.19: Giao diện chính của VP

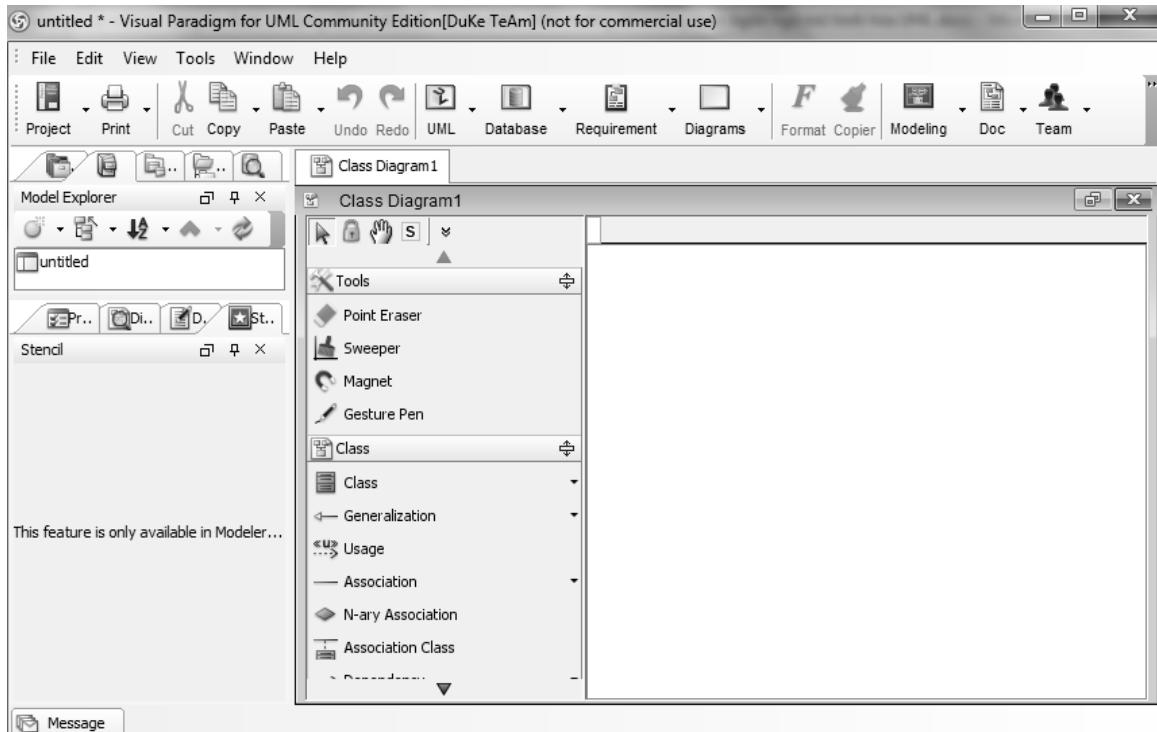
- *Mô hình cơ sở dữ liệu:* VP cho phép thiết kế cơ sở dữ liệu với biểu đồ quan hệ thực thể. VP hỗ trợ ba giai đoạn của mô hình hoá dữ liệu: mô hình khái niệm, mô hình logic và mô hình vật lý. Môi trường mô hình hoá trực quan của nó cho phép thiết kế cơ sở dữ liệu nhanh chóng và chính xác.
- *Mô hình tiến trình nghiệp vụ:* Biểu diễn luồng công việc nghiệp vụ bằng biểu đồ tiến trình nghiệp vụ và bản đồ tiến trình.

Cửa sổ phía bên phải của màn hình VP là cửa sổ được sử dụng để vẽ các biểu đồ bằng cách sử dụng các công cụ vẽ tương ứng trong ToolBox. Hầu hết các ký hiệu sử dụng để vẽ biểu đồ trong VP đều thống nhất với chuẩn của UML. Giao diện chính của VP trong các biểu đồ đều được chia thành các phần như trong Hình 3.20. Ý nghĩa chính của các thành phần này như sau:

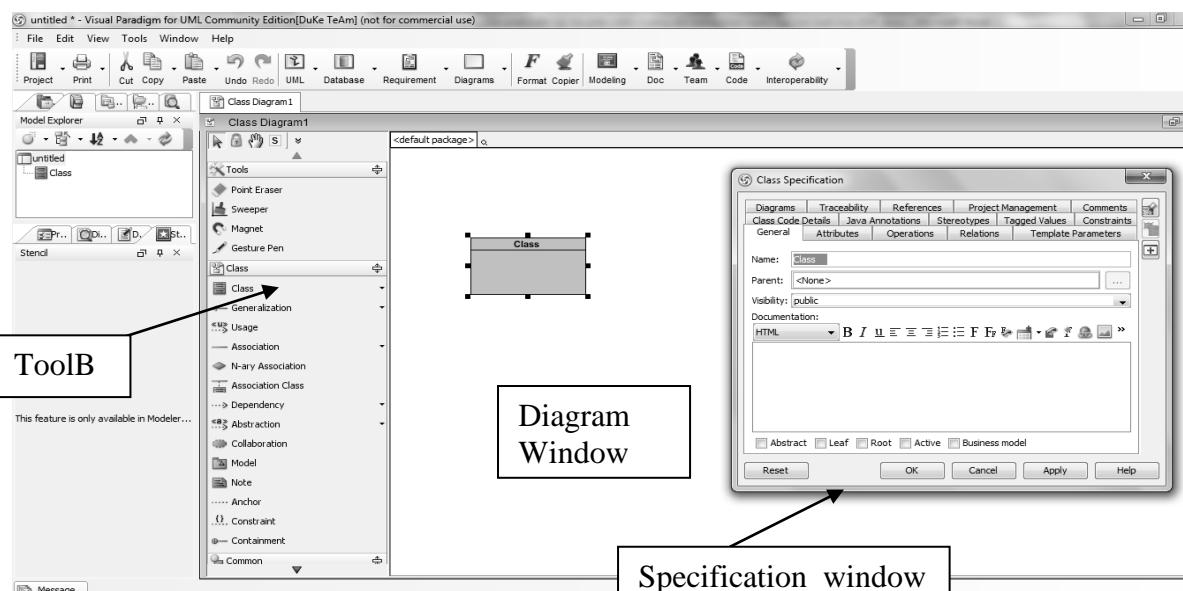
- Menu chính chứa các menu và công cụ tương tự như các ứng dụng Windows khác.
- Phần ToolBox chứa các công cụ dùng để vẽ biểu đồ. Ứng với mỗi dạng biểu đồ thì sẽ có một dạng toolbox tương ứng.
- Phần Diagram Window là không gian để vẽ và hiệu chỉnh các biểu đồ trong mô hình tương ứng.
- Cửa sổ Specification Window là đặc tả chi tiết của mỗi phần tử mô hình theo các trường thông tin tương ứng với dạng biểu đồ đó.

Sử dụng Visual Paradigm cho các bước cụ thể trong phân tích thiết kế hệ thống sẽ được trình bày chi tiết trong chương còn lại của tài liệu này. Tham khảo <https://www.visual-paradigm.com/features/uml-and-sysml-modeling/> để hiểu hơn về xây dựng các biểu đồ

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG



Hình 2.22: Giao diện chính của VP



Hình 2.21: Các thành phần trong giao diện VP

2.4 KẾT LUẬN

Chương này đã giới thiệu ngôn ngữ mô hình hóa các hệ thống hướng đối tượng UML. Các nội dung chính bao gồm:

- Lịch sử ra đời của UML với 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 hóa chuẩn cho công nghiệp phần mềm hướng đối tượng.

CHƯƠNG 2. MÔ HÌNH HÓA PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- Ngôn ngữ UML 2.2 hiện nay bao gồm 14 biểu đồ và phân làm hai nhóm: Nhóm biểu đồ cấu trúc và nhóm biểu đồ hành vi. 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.
- Giới thiệu công cụ VP để phát triển các dự án phần mềm đã được sử dụng rộng rãi trong môi trường giáo dục hiện nay.

BÀI TẬP

1. Trong hệ quản lý Bán hàng Sách online, trước hết khách hàng phải làm thủ tục đăng ký thành viên và khi đó hệ thống sẽ cung cấp một account để đăng nhập. Sau khi đăng nhập khách hàng có thể tìm kiếm, tạo giỏ hàng, đăng ký mua hàng, thanh toán... Hãy xác định các ca sử dụng và sử dụng VP để vẽ biểu đồ quan hệ giữa các ca sử dụng này.
2. Tiếp tục câu 2 với xây dựng các kịch bản tương ứng với các ca sử dụng. Sinh viên có thể sử dụng VP để xây dựng kịch bản.
3. Sử dụng VP để xây dựng biểu đồ tuần tự và biểu đồ giao tiếp cho đăng nhập và tạo giỏ hàng được trình bày trong câu 1.
4. Sử dụng VP để xây dựng biểu đồ hoạt động ca sử dụng đăng ký mua hàng được trình bày trong câu 1.
5. Từ các kịch bản trong Câu 2 hãy xác định các lớp: địa chỉ *Diachi*, họ tên *Hoten*, mã khách hàng *MaKhachhang*, hóa đơn *Hoadon*, mặt hàng *Mathang*. Sử dụng VP để xây dựng các lớp này cùng các thuộc tính tương ứng.
6. Xác định quan hệ giữa các lớp trong Câu 5 và sử dụng VP để xây dựng biểu đồ lớp.
7. Tương tự các Câu 1-6 cho Hệ Quản lý đăng ký học theo tín chỉ, Hệ Quản lý thư viện.

CHƯƠNG 3

XÁC ĐỊNH YÊU CẦU

3.1 GIỚI THIỆU

Vòng đời phát triển hệ thống là một tiến trình biến đổi hệ thống hiện thời của một tổ chức hay doanh nghiệp thành một hệ thống mới. Dù theo tiến trình phát triển nào, pha xác định yêu cầu luôn được xem là pha then chốt nhất của toàn bộ tiến trình phát triển phần mềm. Nhiều nghiên cứu đã chỉ ra rằng hơn một nửa dự án phần mềm thất bại là do xác định yêu cầu chưa đúng đắn. Đây cũng là lý do tại sao các cách tiếp cận lặp trong các phương pháp luận hướng đối tượng được xem là hiệu quả hơn các phương pháp luận truyền thống như mô hình thác nước hay xoắn ốc. Mục đích của pha xác định yêu cầu là:

- Xây dựng được luận cứ về sự cần thiết phải phát triển dự án phần mềm: Nghĩa là giải thích lý do tại sao phần mềm này cần phải phát triển. Khi đã quyết định cần thiết xây dựng dự án, chúng ta phải khảo sát đầy đủ chi tiết các nghiệp vụ hiện thời đang được tiến hành trong tổ chức hay doanh nghiệp và hiểu rõ được yêu cầu của khách hàng cho hệ thống mới.
- Mô tả các yêu cầu của hệ thống dự định phát triển: Xác định chính xác yêu cầu không những ảnh hưởng đến việc quyết định các chức năng của hệ thống mà còn ảnh hưởng đến việc phát hiện các ràng buộc như chi phí và thời gian phát triển, các công nghệ cần sử dụng...

3.2 CÁC BƯỚC TRONG PHA XÁC ĐỊNH YÊU CẦU

3.2.1 Yêu cầu là gì?

Yêu cầu (requirement) đơn giản là những phát biểu về những gì mà hệ thống phải thực hiện hay các chức năng nào hệ thống cần phải có. Trong pha xác định yêu cầu, các yêu cầu được thể hiện theo quan điểm của người sử dụng nghiệp vụ và tập trung vào "cái gì" mà hệ thống có thể thực hiện hơn là cách thức thực hiện thế nào. Các yêu cầu thường tập trung vào các nhu cầu của người sử dụng nghiệp vụ nên thường được gọi *yêu cầu nghiệp vụ* (business requirement) hay *yêu cầu người sử dụng* (user requirement).

Các yêu cầu nghiệp vụ thường sẽ được tiến hóa thành những biểu diễn có kỹ thuật hơn để mô tả yêu cầu hệ thống theo quan điểm của người phát triển và các thể hiện yêu cầu mới này được gọi là *yêu cầu hệ thống* (system requirement). Quyết định hệ thống có thể làm gì và không thể làm gì sẽ giúp chúng ta tập trung vào việc tạo ra những dòng mã cần thiết. Việc hiểu sai lệch về các yêu cầu hệ thống có thể dẫn đến lãng phí thời gian vào việc phát triển những dòng mã mà ta không được trả tiền để làm.

Cần nhấn mạnh rằng không có một ranh giới rõ ràng nào giữa yêu cầu nghiệp vụ và yêu cầu hệ thống nên một số nhà phát triển không phân biệt khi sử dụng những khái niệm này. Tuy nhiên, điều quan trọng cần phải chú ý rằng yêu cầu là những phát biểu về những gì mà hệ thống phải thực hiện và tiến hóa theo thời gian. Sự tiến hóa có thể xuất phát từ các phát biểu chi tiết về các chức năng nghiệp vụ mà hệ thống cần phải có đến các phát biểu chi tiết về mặt kỹ thuật các chức năng sẽ được cài đặt trong hệ thống mới.

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

Các yêu cầu hệ thống thường được chia làm hai nhóm: yêu cầu chức năng và yêu cầu phi chức năng. *Yêu cầu chức năng* (functional requirement) liên quan trực tiếp đến tiến trình mà hệ thống cần phải xử lý hay thông tin mà hệ thống cần lưu trữ. Ví dụ, các chức năng mà hệ thống bán hàng cung cấp như tạo giỏ hàng, tạo đơn đặt hàng, tìm kiếm... là các yêu cầu chức năng.

Yêu cầu phi chức năng (nonfunctional requirement) liên quan đến các tính chất của hành vi mà hệ thống phải có như khả năng truy nhập hệ thống qua các trình duyệt Web khác nhau hay hỗ trợ việc sử dụng video để quảng cáo... Yêu cầu phi chức năng chủ yếu được sử dụng trong pha thiết kế khi cần đưa ra quyết định về giao diện người dùng, các phần cứng và phần mềm cần hỗ trợ hay có liên quan, và kiến trúc vật lý cơ sở cho hệ thống. Hình 3.1 liệt kê một số yêu cầu phi chức năng và các ví dụ tiêu biểu (<http://www.systemsguild.com>)

Yêu cầu	Mô tả	Ví dụ
Phi chức năng		
Thao tác	Môi trường kỹ thuật và vật lý mà hệ thống sẽ hoạt động	<ul style="list-style-type: none"> Hệ thống có thể tích hợp với hệ thống quản lý bán hàng hiện thời Hệ thống có thể hoạt động với các trình duyệt web khác nhau
Hiệu năng	Tốc độ, khả năng và độ tin cậy của hệ thống	<ul style="list-style-type: none"> Tương tác giữa người dùng và hệ thống không nên vượt quá 2 giây Cơ sở dữ liệu của hệ thống quản lý bán hàng phải cập nhật theo thời gian thực Hệ thống cần sẵn sàng phục vụ người dùng bất kỳ lúc nào
Bảo mật	Ai có quyền truy nhập hệ thống cho một số chức năng nào đó?	<ul style="list-style-type: none"> Chỉ giám đốc mới có quyền xem hồ sơ cá nhân của nhân viên Khách hàng chỉ có thể xem lịch sử đặt hàng trong thời gian hành chính
Văn hóa Và chính trị	Các yếu tố văn hóa, chính trị và các yêu cầu luật pháp ảnh hưởng đến hệ thống	<ul style="list-style-type: none"> Hệ thống có thể phân biệt các loại tiền tệ khác nhau Chính sách công ty chỉ cho phép mua máy tính Dell Các nhà quản lý các nước được quyền địa phương hóa giao diện Hệ thống phải tương thích với chuẩn của các công ty bảo hiểm

Hình 3.1: Một số yêu cầu phi chức năng

3.2.2 Xác định yêu cầu

Đầu ra của pha *xác định yêu cầu* (requirement determination) thông thường là một báo cáo bằng văn bản cùng với một số biểu đồ trong UML để mô tả các yêu cầu chức năng và phi chức năng. Đây là công đoạn cần sự tham gia của cả chuyên gia phân tích trong nhóm phát triển và đại diện của tổ chức người sử dụng. Thực tế người sử dụng thường không biết chính xác những gì họ cần và do đó các nhà phân tích phải giúp họ khám phá ra những yêu cầu của họ.

Ba kỹ thuật quen thuộc hỗ trợ cho các nhà phân tích thực hiện công việc này¹: *Tự động hóa tiến trình nghiệp vụ* (BPA: business process automation), *cải tiến tiến trình nghiệp vụ* (BPI: business process improvement) và *kỹ nghệ lắp ráp tiến trình nghiệp vụ* (BPR: business process reengineering). Những kỹ thuật này là các công cụ mà các nhà phân tích có thể sử dụng khi họ cần hướng dẫn người sử dụng giải thích những gì họ muốn từ hệ thống. Nó có thể giúp cho những người sử dụng xem xét đánh giá tình trạng hiện thời của hệ thống và các tiến trình đang xảy ra, đồng thời xác định chính xác cái gì cần phải thay đổi và phát triển các khái niệm cho hệ thống mới định phát triển.

Ba kỹ thuật này được xem là tương tự nhau nhưng có sự khác biệt nhau về mức độ thay đổi. BPA tạo ra lượng thay đổi nhỏ; BPI tạo ra lượng thay đổi trung bình và BPR tạo ra lượng thay đổi lớn ảnh hưởng nhiều đến tổ chức hệ thống. Mặc dù các kỹ thuật này có thể giúp người sử dụng có cái nhìn về hệ thống mới nhưng không đủ để có thể trích được thông tin về yêu cầu nghiệp vụ chi tiết cần xây dựng hệ thống. Vì vậy, ngoài những kỹ thuật này, người ta còn có thể sử dụng nhiều kỹ thuật để thu thập yêu cầu như phỏng vấn, lập phiếu điều tra, quan sát, phân tích tài liệu... Việc trình bày đầy đủ nội dung về thu thập yêu cầu và những kỹ thuật yêu cầu nằm ngoài phạm vi của giáo trình và bạn đọc cần quan tâm có thể tham khảo ([2] [6] [8]).

Một khi đã thu thập được yêu cầu người sử dụng, người phát triển cần phải xem xét cách biểu diễn hay mô hình các yêu cầu này như thế nào. Như đã trình bày, thông thường có hai giai đoạn xác định yêu cầu: *Xác định yêu cầu nghiệp vụ* và *xác định yêu cầu hệ thống*. Khi đó, kết quả thực hiện của các giai đoạn này tương ứng sẽ được gọi các *mô hình nghiệp vụ* và *mô hình hệ thống*.

Phần còn lại của chương này sẽ tập trung trình bày chi tiết hai giai đoạn này cùng các bước cần phải thực hiện của mỗi giai đoạn. Một Case study về Hệ quản lý đăng ký học theo tín chỉ sẽ được sử dụng xuyên suốt để minh họa các bước thực hiện trong chương này và cũng sẽ được sử dụng trong các chương sau.

3.3 XÁC ĐỊNH YÊU CẦU NGHIỆP VỤ

Trong phần này, chúng ta sẽ xem xét cách xây dựng mô hình nghiệp vụ để làm tiền đề cho việc xây dựng mô hình chức năng của hệ thống sau này. *Mô hình nghiệp vụ* có thể đơn giản như là một biểu đồ lớp nhằm chỉ ra những quan hệ giữa các thực thể nghiệp vụ nên đôi khi còn được gọi là *mô hình miền*. Đối với các dự án nhỏ, mô hình miền có thể được xem là tạm dù. Tuy nhiên, trong đa phần các dự án phần mềm, chúng ta phải xây dựng toàn bộ mô hình

¹ Tham khảo http://en.wikipedia.org/wiki/Business_process_improvement

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

nghiệp vụ để biểu diễn cách mà các nghiệp vụ đang được vận hành hay một phần nghiệp vụ có liên quan đến hệ thống dự định phát triển.

Ca sử dụng không phải là cách duy nhất để mô hình hóa nghiệp vụ. Nhưng đó là cách đơn giản hơn so với các cách mô hình phức tạp như *mô hình hóa tiến trình nghiệp vụ* (business process modeling) hay *phân tích luồng công việc* (workflow analysis). Mô hình hóa nghiệp vụ dựa vào ca sử dụng được xem là đơn giản vì nó không đòi hỏi một sự am hiểu chuyên sâu về công nghệ hay kỹ thuật nào. Chỉ với sự hiểu biết thông thường và một chút logic là có thể hiểu được những gì nó thể hiện. Mô hình hóa nghiệp vụ với ca sử dụng bao gồm các bước sau đây:

1. *Xác định và mô tả các tác nhân*
2. *Xây dựng bảng thuật ngữ*
3. *Xác định và mô tả các ca sử dụng*
4. *Mô tả chi tiết ca sử dụng*
5. *Xây dựng biểu đồ giao tiếp (Tùy chọn)*
6. *Xây dựng biểu đồ hoạt động (Tùy chọn)*

Cần nhấn mạnh ở đây rằng trong quá trình phát triển hệ phần mềm hướng đối tượng, chúng ta cần phải lặp đi lặp lại những bước này cho đến khi có được một bức tranh đầy đủ về hệ thống mà ta muốn xây dựng.

3.3.1 Xác định và mô tả các tác nhân

Việc đầu tiên ta cần phải làm là xác định các tác nhân nghiệp vụ. *Một tác nhân* (actor) là một người hay một đối tượng giữ vai trò nào đó trong nghiệp vụ như một bộ phận hay một hệ phần mềm riêng biệt. Việc xác định tác nhân giúp chúng ta xác định được cách mà nghiệp vụ được sử dụng, từ đó chỉ ra được các trường hợp sử dụng. Trong thực tế, một tác nhân có thể đóng các vai trò khác nhau trong những thời điểm khác nhau.

Ví dụ, một nhân viên thư viện khi hết giờ làm việc có thể mượn sách về nhà đọc và khi đó anh ta trở thành bạn đọc; một nhân viên của Công ty cho thuê xe ôtô sẽ trở thành khách hàng khi hết giờ làm việc anh ta quyết định thuê một chiếc ô tô để đi nghỉ cuối tuần.

Vì vậy trong giai đoạn này của quá trình phát triển, ta nên làm việc với chính các khách hàng để tìm hiểu xem các hoạt động nghiệp vụ tương ứng với mỗi tác nhân và khi đó chúng ta sẽ xác định được các tác nhân một cách dễ dàng hơn. Chúng ta hãy xem xét cẩn thận hơn ví dụ về hệ quản lý đăng ký học chỉ sau đây.

Ví dụ: Hệ thống quản lý đăng ký học theo tín chỉ ở trường Đại học có các phòng ban, cá nhân là những tác nhân liên quan như sau:

- *Văn phòng khoa*: xác định các ràng buộc được đăng ký các môn học, mời giảng viên cho các lớp tín chỉ.
- *Phòng đào tạo*: tạo lớp tín chỉ, xây dựng kế hoạch lớp tín chỉ, xây dựng điều kiện được tốt nghiệp các ngành học đối với sinh viên.
- *Sinh viên*: xem các khóa học trong kỳ tới, đăng ký học, hủy đăng ký, xem lịch học, xem kết quả học tập các môn học.
- *Giảng viên*: xem danh sách môn học, đăng ký giảng dạy, xem lịch giảng dạy.

- *Phòng kế toán*: dựa trên số tín chỉ sinh viên đăng ký để thu học phí.
- *Hệ thống lập lịch phòng học*: sắp xếp lớp, phòng học, kíp theo sĩ số lớp tùy từng hệ, khoa, khóa học hay chuyên ngành.
- *Hệ thống quản lý sinh viên*: phân lớp, quản lý mã sinh viên, hồ sơ học bạ sinh viên, danh sách sinh viên...
- *Hệ thống quản lý kết quả học tập*: cập nhật điểm thành phần, danh sách sinh viên không đủ điều kiện dự thi, cập nhật điểm thi, tổng hợp điểm, danh sách sinh viên nhận học bổng...

3.3.2 Xây dựng Bảng Thuật ngữ

Nhiều nghiên cứu đã chỉ ra rằng việc tiến hành xây dựng *Bảng thuật ngữ* (Glossary) ngay khi khởi đầu dự án đóng vai trò quan trọng cho việc xác định chính xác các yêu cầu khách hàng. Từ *Bảng thuật ngữ* ngày nay được cộng đồng hướng đối tượng ưa thích hơn *từ điển dữ liệu* như từng được dùng trước đây.

Mục đích của bảng thuật ngữ là nhằm làm sáng tỏ các thuật ngữ được sử dụng cho một miền nào đó để mọi người hiểu được các sản phẩm trong quá trình phát triển phần mềm. Trong khi đó, từ điển dữ liệu ngũ ý rằng dữ liệu được mô hình hóa một cách độc lập và như vậy là đã tách dữ liệu ra khỏi các thao tác liên quan. Điều mà những người theo quan điểm hướng đối tượng đã tìm cách tránh tính cô lập này. Hơn nữa, bảng thuật ngữ cũng cho phép ta sắp xếp thành các nhóm từ đồng nghĩa để có thể dùng bất kỳ từ nào trong nhóm mà không gây ra sự hiểu nhầm khi biểu đạt yêu cầu.

Mỗi dòng trong *Bảng thuật ngữ* định nghĩa một thuật ngữ và nó có thể ngắn hoặc dài tùy theo các trường hợp. Tuy nhiên, việc mô tả tác nhân thường nên có tính chất tổng quát bởi vì nó có thể được áp dụng trong nhiều ngữ cảnh. Do đó, ta có thể ghi lại các quan hệ của mỗi thuật ngữ trong quá trình phát triển như tác nhân nghiệp vụ, tác nhân hệ thống... Sau đây là danh sách các mối quan hệ mà chúng ta có thể tham khảo trong quá trình xây dựng bảng thuật ngữ:

- *Tác nhân nghiệp vụ* (business actor): tác nhân xuất hiện trong các yêu cầu nghiệp vụ.
- *Đối tượng nghiệp vụ* (business object): đối tượng xuất hiện trong các yêu cầu nghiệp vụ.
- *Tác nhân hệ thống* (system actor): tác nhân xuất hiện trong các yêu cầu hệ thống.
- *Đối tượng hệ thống* (system object): đối tượng xuất hiện (bên trong hệ thống) trong các yêu cầu hệ thống.
- *Đối tượng phân tích* (analysis object): đối tượng xuất hiện trong mô hình phân tích.
- *Sản phẩm triển khai* (deployment artifact): những sản phẩm được triển khai trong hệ thống, ví dụ như một tệp tin.
- *Đối tượng thiết kế* (design object): đối tượng xuất hiện trong mô hình thiết kế.
- *Nút thiết kế* (design node): một máy tính hoặc một tiến trình tạo thành kiến trúc hệ thống.
- *Cụm thiết kế* (design layer): phân rã một hệ thống con theo chiều ngang.

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

- *Gói thiết kế* (design package): một nhóm các lớp liên quan nhau về mặt logic, được dùng để tổ chức công việc phát triển hệ thống.

Trong bảng liệt kê này, đối tượng được hiểu là một thực thể hoặc “dữ liệu và tiến trình đã được đóng gói” như cách hiểu thông thường trong cách tiếp cận hướng đối tượng. Tuy nhiên, mỗi loại đối tượng (nghiệp vụ, hệ thống, phân tích, thiết kế) là khác nhau.

Ví dụ, khi bạn đọc mượn một cuốn sách, công việc của thủ thư lúc này vừa liên quan đến đối tượng nghiệp vụ ở bên ngoài hệ thống là cuốn sách vật lý trên giá sách vừa liên quan đến đối tượng nằm trong chính hệ thống được khởi tạo từ lớp chúng ta cài đặt.

Các thực thể trong Bảng thuật ngữ đều tuân theo quy định đặt tên lớp như lớp bạn đọc *Bandoc*, lớp sách *Sach*, nhân viên thư viện *NhanvienThuvien*, thủ kho *Thukho*, thủ thư *Thuthu*... Rõ ràng khi chúng ta sử dụng cùng một quy định trong tất cả tài liệu của dự án, thì người đọc dễ dàng tham chiếu đến các định nghĩa trong bảng thuật ngữ hơn.

Ví dụ: Một số thuật ngữ trong Hệ đăng ký học theo tín chỉ:

STT	Tiếng Anh	Tiếng Việt	Giải thích nội dung
1	Branches	Ngành	Một cơ sở đào tạo, một trường có thể đào tạo một số ngành. Ví dụ ngành công nghệ thông tin, ngành viễn thông...
2	Class	Lớp học	Là cách tổ chức một số sinh viên cùng ngành, cùng khóa để dễ theo dõi, quản lý.
3	Credit	Tín chỉ	Được sử dụng để tính khối lượng học tập của SV. Một tín chỉ được quy định bằng 15 tiết học lý thuyết
4	Faculty	Khoa	Một bộ phận trong một cơ sở đào tạo một hay một số ngành học nhất định
5	Major	Chuyên ngành	Một ngành chia thành một số chuyên ngành. Ngành công nghệ thông tin có chuyên ngành công nghệ phần mềm, hệ thống thông tin...
6	Mark	Điểm	Con số 1, 2,...hay chữ số A, B,...đánh giá kết quả học tập của sinh viên về một môn học nào đó
7	Prerequisite subject	Môn học điều kiện	Những yêu cầu cần có để có thể học được môn học nêu ra
8	schoolYear	Năm học	Thường gồm 10 tháng, chia thành một số học kỳ
9	Semester	Học kỳ	Một khoảng thời gian được quy định trong một năm học. Năm học có thể được chia thành hai hay ba học kỳ
10	Student	Sinh viên	Thí sinh trúng tuyển vào học trong trường đại học
11	Subjects	Môn học	Gồm các thông tin chi tiết về môn học như tên môn, số tín chỉ, chuyên ngành, môn điều kiện...
12	Instructor	Giảng viên	Người trực tiếp giảng dạy các môn học
13	timeTables	Thời khóa biểu	Lịch học cho các lớp gồm môn học, giờ học, phòng học và giảng viên dạy môn tương ứng

3.3.3 Xác định và mô tả các ca sử dụng nghiệp vụ

Một khi đã xác định được các tác nhân, nhiệm vụ tiếp theo là xác định các ca sử dụng nghiệp vụ. Mỗi ca sử dụng là một phần nhỏ của nghiệp vụ. Tại thời điểm này, các ca sử dụng có thể

liên quan đến giao tiếp hai chiều giữa các tác nhân, đặc biệt tác nhân con người. Sau này, ta sẽ thấy các *ca sử dụng* hệ thống sẽ có cấu trúc hơn vì giao tiếp lúc này thường là giữa tác nhân và hệ thống hay giữa các chức năng liên quan nhau bên trong hệ thống.

Không có một quy tắc nào để chỉ ra cách chia nhỏ nghiệp vụ thành các ca sử dụng. Điều này phải dựa vào kinh nghiệm, tư duy logic và cảm nhận chung về nghiệp vụ. Những cuộc hội thảo, phỏng vấn với các nhà đầu tư, khách hàng hay người sử dụng...sẽ hỗ trợ rất nhiều trong việc xác định các ca sử dụng.Thêm vào đó, các tài liệu về đào tạo nhân viên hay người quản lý, các sổ tay hướng dẫn cho nhân viên như hướng dẫn giảng viên trong trường đại học, các quy định của đơn vị tổ chức cùng nhiều tài liệu khác sẽ là các nguồn hỗ trợ quan trọng.

Trong quá trình xác định các ca sử dụng, phải luôn luôn nắm lòng câu hỏi sau đây: “*Các hành động chủ yếu làm cho nghiệp vụ đó hoạt động là gì?*” Chú ý rằng khi mô hình hóa nghiệp vụ, chúng ta không quan tâm tới cách mà hệ thống định phát triển có thể vận hành như thế nào. Giai đoạn này chủ yếu tập trung mô tả cách mà *nghiệp vụ hiện thời diễn ra* thế nào. UML không quy định nội dung của các ca sử dụng nghiệp vụ hay cách đánh số thứ tự thế nào. Do đó, chúng ta có thể sử dụng ngôn ngữ tự nhiên để mô tả trình tự các bước hay ngôn ngữ có cấu trúc (ngôn ngữ tự nhiên với *if...then...else* và cấu trúc lặp *loop* quen thuộc trong ngôn ngữ lập trình) hay bất cứ thứ gì có thể diễn đạt và dễ hiểu nhau.

Tuy nhiên, ta nên viết theo trình tự các bước với ngôn ngữ tự nhiên để làm các ca sử dụng rõ ràng và độc lập với cài đặt. Việc sử dụng ngôn ngữ cấu trúc có nguy cơ làm cho khách hàng đầu tư dự án khó hiểu vì mô tả của chúng ta mang tính chất “chuyên môn” quá. Sau khi đã có danh sách đề xuất các ca sử dụng, chúng ta có thể liệt kê các bước có liên quan trong mỗi ca.

Ví dụ: Danh sách một số ca sử dụng trong Hệ thống đăng ký học theo tín chỉ

- *Đăng ký khóa học*: sinh viên đăng ký các khóa học trong kỳ tới với phòng đào tạo.
- *Hủy khóa học*: sinh viên hủy khóa học đã đăng ký với phòng đào tạo.
- *Xem điểm*: sinh viên xem điểm tổng kết các khóa học của mình.
- *Xem kết quả đăng ký học*: sinh viên xem danh sách các khóa học đã đăng ký.
- *Xem chương trình học*: sinh viên xem danh sách các môn học của từng khoa, chuyên ngành, hệ đào tạo
- *Đăng ký thi đi*: sinh viên đăng ký thi khi kết thúc khóa học.
- *Đăng ký thi lại*: sinh viên đăng ký các khóa học sẽ thi lại.
- *Xem lịch thi cá nhân*: sinh viên xem lịch thi các khóa học mình đã đăng ký thi.
- *Đăng ký giảng dạy*: giảng viên đăng ký các môn học có thể đảm nhiệm trong kỳ tới.
- *Hủy đăng ký giảng dạy*: giảng viên hủy đăng ký các môn học có thể đảm nhiệm trong kỳ tới.
- *Xem lịch giảng dạy*: giảng viên xem lịch giảng dạy của mình trong kỳ tới.

3.3.4 Mô tả chi tiết ca sử dụng

Tại thời điểm này, việc mô tả chi tiết ca sử dụng chỉ là bước đầu để hiểu được hoạt động nghiệp vụ hiện thời mà người sử dụng đang tiến hành. Chi tiết hóa ca sử dụng bằng kịch bản

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

(scenario) sẽ được trình bày một cách đầy đủ hơn trong phần sau khi đề cập đến giai đoạn xác định yêu cầu hệ thống.

Ví dụ: Ca sử dụng: *Đăng ký khóa học*

Ca sử dụng: Đăng ký khóa học

1. Sinh viên (SV) truy cập vào website nhà trường để đăng ký học
2. Hệ thống yêu cầu SV đăng nhập trước khi đăng ký
3. Sinh viên đăng nhập hệ thống
4. Hệ thống kiểm tra các thông tin liên quan đến SV này.
 - 4.1. Nếu đúng, SV đề nghị Danh sách các môn học muốn đăng ký.
 - 4.1.1. Nếu các yêu cầu đối với các khóa học được thỏa mãn, SV được ghi danh vào các lớp học.
 - 4.1.2. Nếu không, Hệ thống thông báo SV chưa đủ điều kiện học môn học.
 - 4.2. Nếu sai, Hệ thống yêu cầu SV nhập lại tài khoản và mật khẩu truy cập.

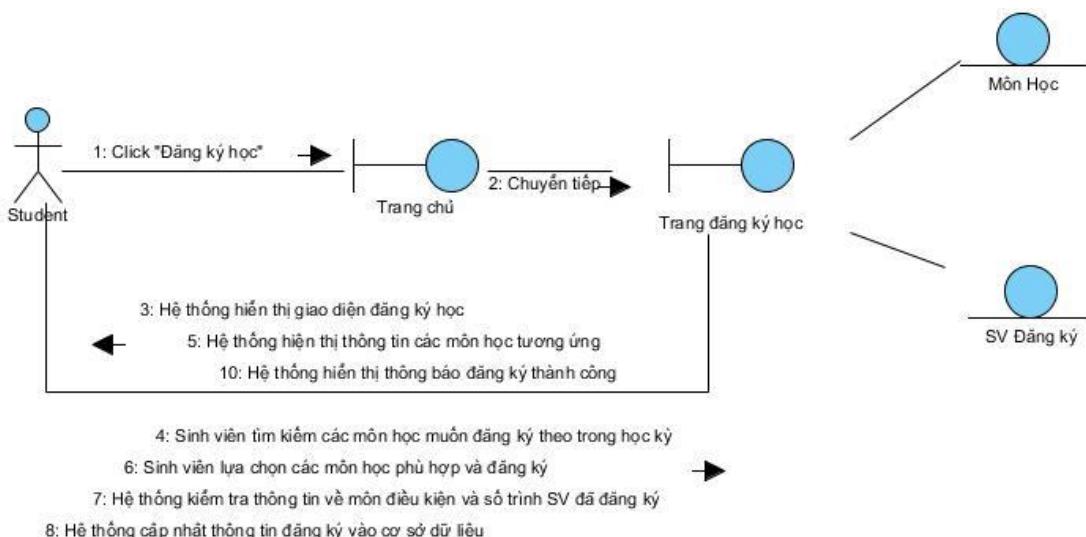
3.3.5 Xây dựng biểu đồ giao tiếp

Biểu đồ giao tiếp trong UML là một thể hiện chi tiết của một ca sử dụng bằng một loạt các tương tác theo thứ tự xảy ra giữa các tác nhân và hệ thống và các đối tượng bên trong hệ thống. Do không sử dụng quá nhiều chi tiết kỹ thuật trong giai đoạn đầu của quá trình phát triển, nên biểu đồ giao tiếp thường được sử dụng để mô hình hóa nghiệp vụ. Chuỗi các tương tác trong biểu đồ có thể không hoàn toàn tương ứng với chuỗi các bước trong mô tả chi tiết ca sử dụng. Mỗi tương tác sẽ biểu diễn một cách khái quát về một hoặc nhiều bước. Trong biểu đồ giao tiếp, *thông điệp* (message) truyền đi giữa các đối tượng được biểu diễn bằng một mũi tên nhỏ, vẽ dọc theo đường kết nối giữa hai đối tượng, với hàm ý rằng nhờ kết nối đó mà bên gửi biết bên nhận để có thể gửi thông điệp đi.

Ví dụ: Biểu đồ giao tiếp cho ca sử dụng Sinh viên đăng ký môn học (Hình 3.2)

Có thể cho rằng thứ tự của các tương tác nên tương ứng một cách chính xác với thứ tự các bước trong chi tiết ca sử dụng. Tuy nhiên, bởi ngôn ngữ tự nhiên không phải là một dãy tuần tự các bước. Do đó, nhiều khả năng mỗi thao tác sẽ mô tả tổng hợp của một hay nhiều bước. Mặc dù không chính xác như ca sử dụng, nhưng biểu đồ giao tiếp vẫn là cần thiết bởi vì nó chi tiết hóa hơn các ca sử dụng. Hơn nữa, nó có thể giúp chúng ta có được một cái nhìn đầy đủ và rõ ràng hơn về hệ thống sau này ngay từ giai đoạn đầu phát triển. Vì sự tương tác được đưa ra trong giai đoạn này là không quá phức tạp, nên yêu cầu xây dựng mỗi biểu đồ giao tiếp cho mỗi ca sử dụng là hoàn toàn hợp lý.

Để đơn giản, các tương tác trong biểu đồ giao tiếp nên là *đường đi chuẩn mực* xuyên suốt ca sử dụng. Sau này, khi đề cập đến ca sử dụng hệ thống, chúng ta có thể cần xác định thêm các *đường đi khác thường* hay *ngoại lệ*. Tuy nhiên, bây giờ chúng ta chỉ cần hiểu sự tồn tại của chúng tương ứng với mỗi ca sử dụng là được.



Hình 3.2: Biểu đồ giao tiếp sinh viên đăng ký môn học

3.3.6 Xây dựng biểu đồ hoạt động

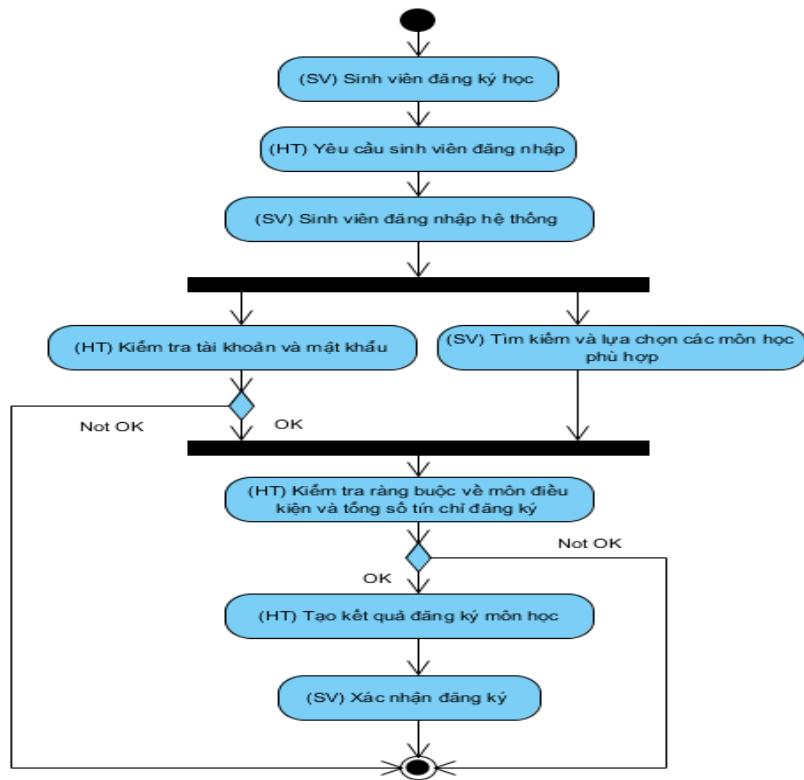
Biểu đồ hoạt động trong UML được sử dụng để chỉ ra sự phụ thuộc giữa các hoạt động khi chuyển từ điểm bắt đầu tới một điểm kết thúc của một tiến trình. Giống như biểu đồ giao tiếp, các hành động trong biểu đồ hoạt động có thể không tương ứng với từng bước trong mô tả chi tiết của ca sử dụng. Trong thực tế phát triển phần mềm, biểu đồ hoạt động có thể được sử dụng cho nhiều mục đích khác nhau như xây dựng toàn bộ mô hình nghiệp vụ hoặc viết tài liệu cho một phương thức để thể hiện hành vi của một đối tượng phần mềm.

Biểu đồ hoạt động là một đồ thị có hướng, trong đó các nút (đỉnh) là các hoạt động và các cung là các dịch chuyển:

- **Hoạt động:** là một công việc có thể được xử lý bằng tay như Điền mẫu, hoặc bằng máy như Hiển thị màn hình đăng nhập. Một hoạt động được biểu diễn trong biểu đồ bằng một hình chữ nhật tròn góc có mang tên của hoạt động
- **Chuyển dịch:** là sự chuyển từ hoạt động này sang hoạt động khác được thể hiện bằng một mũi tên nối giữa hai hoạt động.
- **Nút khởi đầu:** thể hiện điểm bắt đầu của hoạt động được ký hiệu bởi một hình tròn đặc.
- **Nút kết thúc:** thể hiện điểm kết thúc các hoạt động được ký hiệu hình tròn đặc có viền bao quanh. Tùy trường hợp có thể có một hoặc nhiều nút kết thúc
- **Các điều kiện chuyển dịch hoạt động:** được ký hiệu bởi một hình thoi để thực hiện sự rẽ nhánh các hoạt động.
- **Thanh đồng bộ hóa:** để mở hay đóng các nhánh thực hiện song song.

Ví dụ: Biểu đồ hoạt động cho ca sử dụng Sinh viên đăng ký môn học thể hiwwjn trong Hình 3.3

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU



Hình 3.3: Biểu đồ hoạt động

3.4 XÁC ĐỊNH YÊU CẦU HỆ THỐNG

Giai đoạn thứ hai của pha xác định yêu cầu là *mô hình hóa yêu cầu hệ thống* (system requirement modeling). Mục đích là mô hình hệ thống mà chúng ta định phát triển nhằm cài tiến nghiệp vụ hiện thời của khách hàng.

Mô hình hóa yêu cầu hệ thống với ca sử dụng được thể hiện bằng *mô hình ca sử dụng*. Cách tiếp cận này đã trở thành quen thuộc trong phát triển phần mềm hướng đối tượng hiện nay và sẽ được sử dụng trong giáo trình này vì nó dễ xây dựng và dễ hiểu đối với mọi người. Mô hình ca sử dụng trong giai đoạn xác định yêu cầu hệ thống sẽ được chi tiết hóa hơn và có tính “kỹ thuật” hơn so với ca sử dụng trong giai đoạn xác định yêu cầu nghiệp vụ đã được trình bày trong mục trước. Xác định yêu cầu hệ thống bao gồm các bước sau đây:

1. Xác định và mô tả các tác nhân
2. Xác định và mô tả các ca sử dụng
3. Xây dựng biểu đồ ca sử dụng
4. Xây dựng kịch bản
5. Xếp ưu tiên các ca sử dụng
6. Phác họa giao diện người dùng

3.4.1 Xác định và mô tả các tác nhân

Trước hết điều ta cần làm là xác định và mô tả các tác nhân của hệ thống với sự giúp đỡ của khách hàng. Các tác nhân ở đây bao gồm con người hay các hệ thống bên ngoài *tương tác trực tiếp* với hệ thống sẽ xây dựng chứ không bao gồm các tác nhân gián tiếp như trong giai đoạn xác định yêu cầu nghiệp vụ.

Quan hệ giữa các tác nhân

Các tác nhân trong hệ thống thường có một mối quan hệ nào đó và việc phát hiện các quan hệ này sẽ giúp cung cấp cách nhìn tổng quát về hệ thống. Một tác nhân có thể là một *đặc biệt hóa* của một *tác nhân tổng quát* khác. Khi đó, các tác nhân đặc biệt hóa sẽ kế thừa các hành vi từ các tác nhân tổng quát hóa. Điều này tạo thêm sức mạnh cho mô hình ca sử dụng sau này. Trong UML chúng ta cũng sử dụng ký hiệu tương tự như quan hệ kế thừa giữa các lớp. Biểu đồ quan hệ các tác nhân như được trình bày trong Hình 3.4.

Ví dụ, Hệ quản lý đăng ký học theo tín chỉ có các nhân Sinh viên, Giảng viên và Nhân viên Phòng đào tạo. Ta có thể tiếp tục đặc biệt hóa Nhân viên Phòng đào tạo thành Nhân viên xếp lịch dạy, Nhân viên nhập điểm, Nhân viên theo dõi tình hình học tập... Hay có thể tổng quát hóa Sinh viên, Giảng viên, Nhân viên thành Thành viên sử dụng hệ thống. Điều này có thể hơi phức tạp nhưng nó sẽ mang lại nhiều lợi ích hơn. Vì cách làm như vậy đảm bảo rằng bạn hiểu chính xác miền bài toán và khách hàng hy vọng sẽ có được một hệ thống với những chức năng mà họ thực sự mong muốn.

Ví dụ: Hệ thống đăng ký học theo tín chỉ bao gồm các tác nhân sau đây:

- *Nhân viên*: cập nhật (môn học, thông tin về khoa, chuyên ngành, lớp học, điểm thi...), hủy (môn học, chuyên ngành, khoa...)
- *Sinh viên*: xem lịch học, đăng ký học, hủy đăng ký...
- *Giảng viên*: đăng ký dạy, hủy đăng ký, xem lịch giảng dạy...
- *Thành viên*: sinh viên hoặc giảng viên có thông tin cá nhân được lưu trong cơ sở dữ liệu. Mỗi thành viên sẽ được cấp một mật khẩu để truy cập tài khoản cá nhân của mình.

3.4.2 Xác định và mô tả các ca sử dụng

Một khi đã có được danh sách các tác nhân, với sự trợ giúp của các khách hàng ta sẽ tiếp tục xác định được sự tương ứng giữa các ca sử dụng với các tác nhân đó. Đồng thời khi đề xuất ca sử dụng, ta phải viết một mô tả ngắn gọn về các ca sử dụng đó.

Ví dụ: Một số ca sử dụng của Hệ thống đăng ký học theo tín chỉ:

U1 Đăng nhập: các tác nhân đăng nhập hệ thống.

U2 Thoát: các tác nhân thoát khỏi hệ thống.

U3 Đăng ký học: sinh viên đăng ký các môn học trong kỳ tới.

U4 Xem lịch học: sinh viên xem lịch về các lớp học đã đăng ký

U5 Xem lịch thi học kỳ: sinh viên xem lịch thi học kỳ các môn đã đăng ký

U6 Xem điểm học tập: sinh viên xem kết quả học tập của mình.

U7 In bảng điểm: sinh viên in bảng điểm các khóa học của mình.

U8 Tìm kiếm thông tin: sinh viên tìm kiếm các môn học của từng khoa, chuyên ngành, hệ đào tạo...

U9 Xem lịch giảng dạy: giảng viên xem lịch giảng dạy của mình trong kỳ tới.

U10 Đổi mật khẩu: người dùng thay đổi mật khẩu truy cập hệ thống.

U11 Thay đổi thông tin cá nhân: người dùng thay đổi thông tin cá nhân sau khi đăng nhập hệ thống

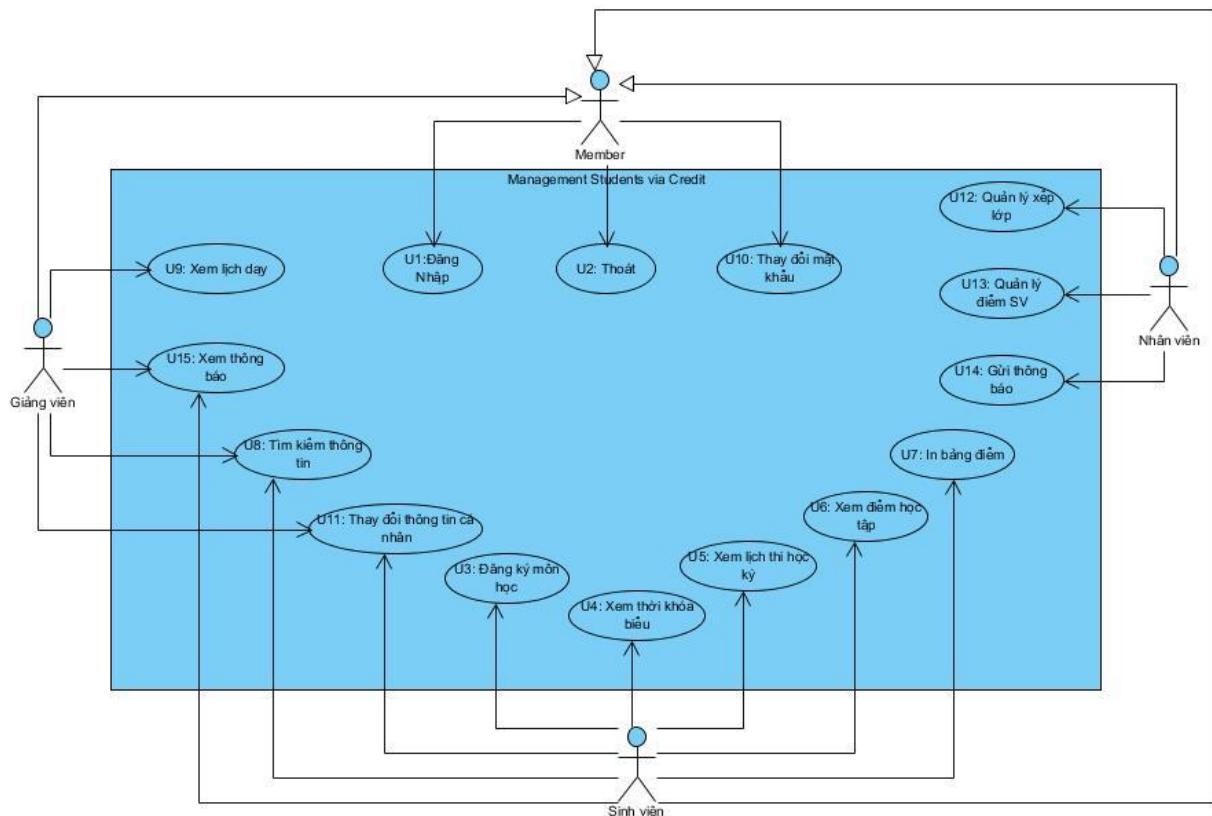
U12 Quản lý xếp lớp: nhân viên phòng đào tạo thực hiện chức năng phân chia và xếp lớp cho các môn học sinh viên đăng ký

U13 Quản lý điểm sinh viên: nhân viên phòng đào tạo cập nhật kết quả điểm của sinh viên

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

U14 Gửi thông báo: phòng đào tạo gửi các thông báo tới sinh viên.

U15 Xem thông báo: giảng viên, sinh viên xem các thông báo của phòng đào tạo



Hình 3.4: Xác định ca sử dụng

3.4.3 Xây dựng biểu đồ ca sử dụng

Biểu đồ ca sử dụng được dùng để chỉ ra những quan hệ của tác nhân với các ca sử dụng cụ thể và *quan hệ* giữa các ca sử dụng với nhau. Điều này giúp chúng ta có một cái nhìn tổng quát đầy đủ hơn về hệ thống sẽ được sử dụng như thế nào sau này. Tiếp theo chúng ta sẽ xem xét các kiểu quan hệ có thể có giữa các ca sử dụng và cách phân rã thành các biểu đồ ca sử dụng.

Quan hệ giữa các ca sử dụng

Có ba kiểu quan hệ giữa các ca sử dụng: *đặc biệt hóa*, *bao hàm* và *mở rộng*. Các quan hệ này cho phép chúng ta nhóm các ca sử dụng có liên quan nhau, phân rã các ca sử dụng, sử dụng lại hành vi và xác định hành vi không bắt buộc.

- *Đặc biệt hóa (Specialize)*: giống như tác nhân, một ca sử dụng có thể *ké thừa* từ một ca sử dụng khác. Để tránh những phức tạp liên quan đến việc định nghĩa lại các bước và thêm các bước phụ, ta có thể đặc biệt hóa các *ca sử dụng trùu tượng*. *Ca sử dụng trùu tượng* là ca sử dụng không có bước nào, nhiệm vụ của nó là nhóm một số ca sử dụng lại với nhau.
- *Bao hàm (Include)*: một ca sử dụng UC1 có một số bước được cung cấp bởi một ca sử dụng khác UC2 thì ta nói UC1 bao hàm UC2. Khi đó, UC1 gọi là *ca sử dụng nguồn* và UC2 gọi là *ca sử dụng đích*. Quan hệ bao hàm được dùng để rút ra các bước chung cho một số ca sử dụng, hoặc để chia ca sử dụng lớn thành các ca sử dụng nhỏ hơn.

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

- *Mở rộng (Extend)*: một ca sử dụng UC1 thêm thông tin vào một ca sử dụng khác UC2 thì UC1 được gọi là mở rộng của UC2. Thông thường các thông tin đó xuất hiện ở cuối ca sử dụng nhưng chúng cũng có thể xảy ra ở đầu hoặc đôi khi là ở giữa.

Có một sự khác biệt cơ bản giữa bao hàm và mở rộng. Với bao gồm, ca sử dụng nguồn sẽ không hoạt động nếu không có ca sử dụng đích; với mở rộng, ca sử dụng nguồn sẽ làm việc tốt kể cả không có ca sử dụng đích. Nói cách khác, ca sử dụng được bao gồm có sự tồn tại độc lập của nó, còn ca sử dụng mở rộng chỉ tồn tại như là một mở rộng.

Thực tế không dễ xác định chính xác quan hệ giữa các ca sử dụng trong lần đầu tiên qua mô hình yêu cầu hệ thống. Hơn nữa, việc xác định quan hệ này phụ thuộc vào chúng ta quyết định xem cái gì là thực sự cần thiết và khách hàng của chúng ta chấp nhận nó hay không. Trong cách tiếp cận hướng đối tượng, có nhiều cách phân chia ca sử dụng thành các mối quan hệ bao hàm, mở rộng hay kế thừa. Thật ra không có cách nào là chính xác, vẫn đề là chúng ta cần phải phát triển một mô hình sao cho phù hợp với cảm nhận của chúng ta và khách hàng.

Phân rã biểu đồ ca sử dụng

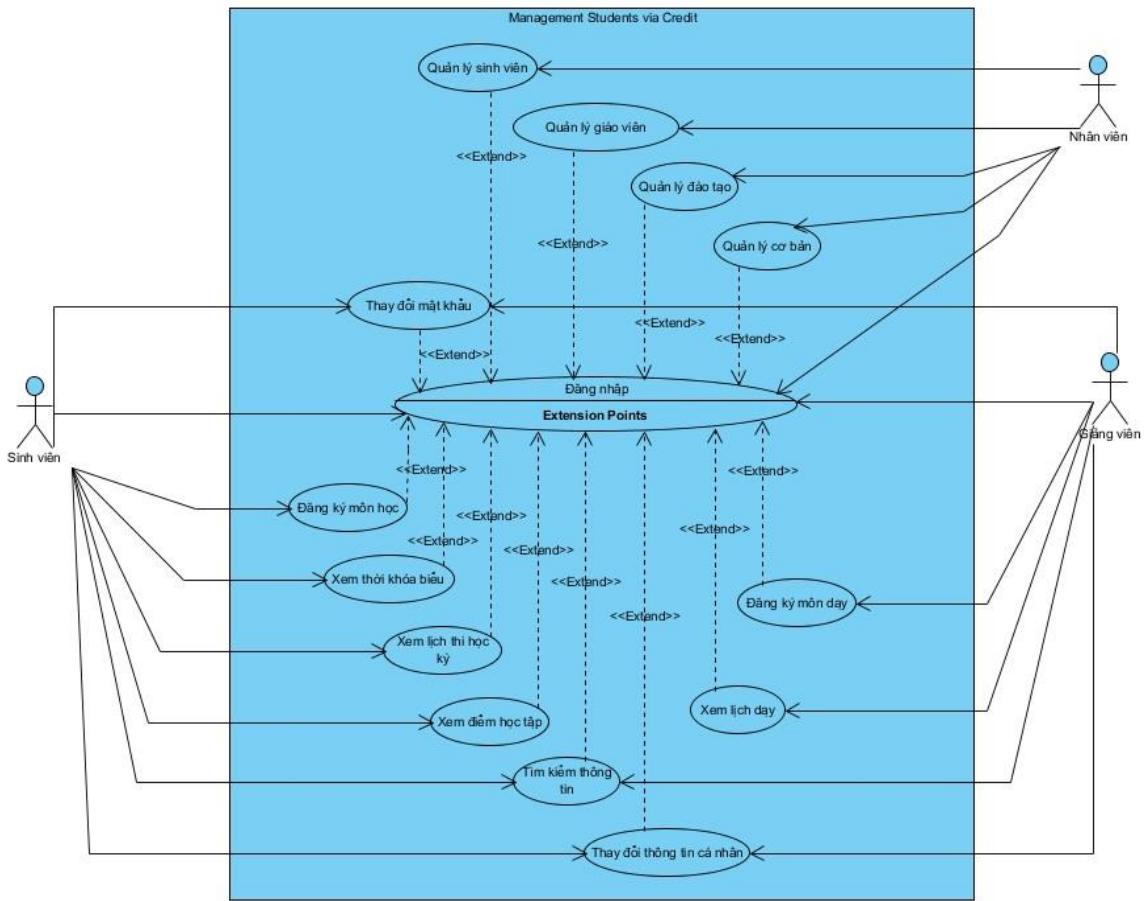
Thông thường để thuận lợi cho việc biểu diễn, người ta chia biểu đồ ca sử dụng thành các mức dựa trên quan hệ ca sử dụng: biểu đồ ca sử dụng mức tổng quát và biểu đồ ca sử dụng các mức thấp hơn.

Biểu đồ ca sử dụng mức tổng quát: từ tập các tác nhân và ca sử dụng đã xác định được, ta xây dựng biểu đồ ca sử dụng mức tổng quát. Thông thường biểu đồ ca sử dụng mức tổng quát bao gồm các tác nhân tương ứng ca sử dụng trừu tượng và một số ca sử dụng cụ thể khác.

Biểu đồ ca sử dụng mức thấp: các ca sử dụng trừu tượng được phân rã thành các ca sử dụng cụ thể hơn nhờ quan hệ extend. Quá trình này được tiếp tục cho đến ca sử dụng mức thấp nhất là các chức năng cụ thể mà tác nhân có thể tương tác với hệ thống.

Ví dụ: Hệ quản lý học theo tín chỉ có biểu đồ ca sử dụng tổng quát được thể hiện trong Hình 3.5. Một phân rã mức thấp hơn được cho trong Hình 3.6.

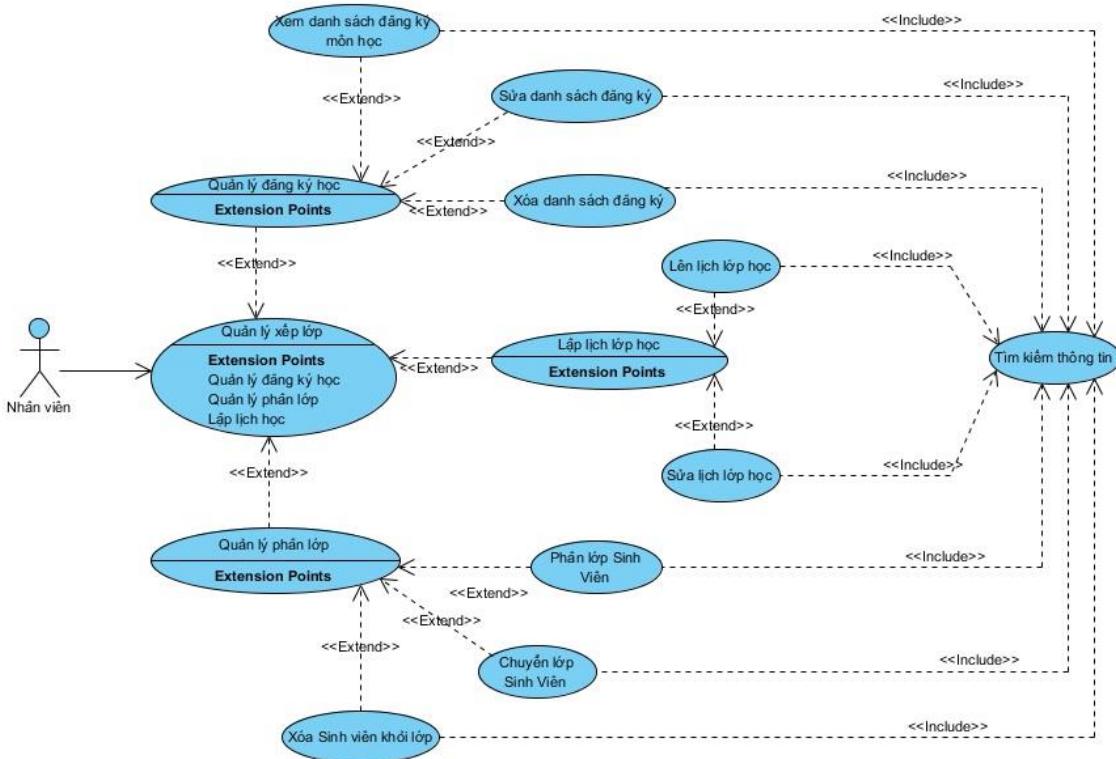
CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU



Hình 3.5: Biểu đồ ca sử dụng mức tổng quát

Có thể nhận thấy trong biểu đồ ca sử dụng Hình 3.6, ca sử dụng Quản lý xếp lớp được phân rã thành các ca sử dụng: Quản lý Đăng ký học, Quản lý phân lớp, Lập lịch lớp học. Ca sử dụng Quản lý xếp lớp là ca sử dụng trùu tượng có nhiệm vụ chính là nhóm các ca sử dụng Quản lý Đăng ký học, Quản lý phân lớp, Lập lịch.

Các ca sử dụng xem danh sách đăng ký môn học, sửa danh sách đăng ký... khi thực hiện phải bao gồm thao tác tìm kiếm thông tin cần thiết. Do đó, các ca sử dụng này bao gồm <<include>> ca sử dụng tìm kiếm thông tin. Các ca sử dụng Đăng ký môn học, Xem thời khóa biểu, Xem điểm học tập... là mở rộng của ca sử dụng Đăng nhập. Người dùng sẽ không thể thực hiện các chức năng đó nếu ca sử dụng Đăng nhập chưa được thực hiện.



Hình 3.6: Biểu đồ phân rã ca sử dụng Quản lý xếp lớp

3.4.4 Xây dựng kịch bản

Khác với tính đơn giản của chi tiết hóa ca sử dụng trong mô hình nghiệp vụ, chi tiết hóa ca sử dụng mô hình hệ thống, gọi là *kịch bản*, sẽ gồm nhiều thông tin liên quan đến quá trình hiện thực hóa hệ thống sau này. Do đó, chúng ta cần phải cẩn thận khi xây dựng các kịch bản vì cho đến thời điểm này nó là phương tiện duy nhất và là điểm khởi đầu để giúp cho ta và khách hàng hình dung về hệ thống. Sau đây, chúng ta sẽ tìm hiểu đầy đủ hơn các khái niệm liên quan đến việc xây dựng kịch bản.

Chi tiết hóa ca sử dụng

UML không xác định chi tiết hóa các ca sử dụng hệ thống nên bao gồm cái gì và nên được sắp xếp như thế nào. Sau đây là một cách viết chi tiết ca sử dụng đã được dùng phổ biến trong công nghiệp phần mềm hiện nay. Định dạng chi tiết ca sử dụng có thể gồm các thành phần:

- Số thứ tự và tên các ca sử dụng
- Tác nhân chính là tác nhân thực hiện ca sử dụng
- Tiền điều kiện là điều kiện đảm bảo trước khi thực hiện ca sử dụng
- Đảm bảo tối thiểu là điều kiện đảm bảo khi xảy ra dị thường (ngoại lệ)
- Hậu điều kiện là điều kiện sau khi thực hiện xong ca sử dụng
- Chuỗi sự kiện chính
- Các đường đi dị thường (ngoại lệ)

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

Tiền điều kiện (preconditions), Hậu điều kiện (postconditions)

Khi xem xét sự kế thừa giữa các ca sử dụng, ta cũng phải quan tâm đến việc đặc biệt hóa có ảnh hưởng gì đến điều kiện trước và điều kiện sau không. Mặc dù các ca sử dụng chỉ kế thừa từ ca sử dụng trùu tượng nhưng chúng vẫn có điều kiện trước và điều kiện sau. Sau đây là một số quy tắc:

1. Khi một ca sử dụng cụ thể hóa một ca sử dụng khác, nó kế thừa các tiền điều kiện của ca sử dụng cha như là một điểm khởi đầu. Bất cứ tiền điều kiện nào của ca sử dụng con phải làm yếu đi tiền điều kiện ca sử dụng cha, nghĩa là chúng được kết hợp với các điều kiện trước của ca sử dụng con bởi phép toán “hoặc”.
2. Với các hậu điều kiện, điểm khởi đầu của ca sử dụng con là hậu điều kiện của ca sử dụng cha. Các hậu điều kiện mà ca sử dụng con thêm vào phải làm mạnh ca sử dụng cha, nghĩa là nó sẽ kết hợp với hậu điều kiện của ca sử dụng cha bằng phép toán “và”.
3. Các tiền điều kiện và hậu điều kiện được thêm vào bởi các ca sử dụng con sẽ không có ảnh hưởng đến các tiền và hậu điều kiện của ca sử dụng cha.

Trong quy tắc trên, quy tắc thứ ba là hiển nhiên đối với những gì chúng ta đã biết về lý thuyết hướng đối tượng (các con không ảnh hưởng đến hành vi của cha). Quy tắc thứ nhất và thứ hai hàm ý rằng nếu ca sử dụng cha không có tiền điều kiện, thì ca sử dụng con cũng phải không có tiền điều kiện; nếu ca sử dụng cha không có hậu điều kiện (không có sự bảo đảm rõ ràng về điều ra), thì khi đó ca sử dụng con có thể có bất cứ hậu điều kiện nào. Tóm lại, khi một ca sử dụng đặc biệt hóa một ca sử dụng khác, chúng ta phải xem xét cẩn thận các tiền điều kiện và hậu điều kiện của ca sử dụng cha.

Khảo sát các ca sử dụng

Khảo sát các ca sử dụng là cách mô tả phi hình thức về một nhóm các ca sử dụng để xem các ca sử dụng có liên quan với nhau thế nào. Bản khảo sát là một kiểu tường thuật mà nhà phát triển có thể đưa ra để dẫn dắt khách hàng đầu tư xuyên suốt qua biểu đồ ca sử dụng. Bản khảo sát các ca sử dụng giúp cho các nhà đầu tư có sự hiểu thấu đáo hơn về các ca sử dụng mà không cần bất kỳ sự giới thiệu nào từ nhà phát triển hệ thống. Hơn nữa, nó là tiền đề để viết chi tiết các kịch bản sau này.

Ví dụ: Khảo sát ca sử dụng Hệ thống quản lý học tập theo tín chỉ:

- Khi truy cập vào hệ thống đăng ký học theo tín chỉ, yêu cầu người dùng phải đăng nhập vào hệ thống (U1) trước khi có thể sử dụng các chức năng của hệ thống. Sau khi đăng nhập hệ thống, người dùng có thể sửa đổi mật khẩu (U10) tài khoản của mình.
- Giảng viên sau khi đăng nhập hệ thống (U1) có thể Tìm kiếm thông tin (U8), xem chi tiết lịch giảng dạy của mình (U9), thay đổi thông tin cá nhân (U11) và nhận thông báo từ phòng đào tạo (U15).
- Sinh viên sau khi đăng nhập vào hệ thống (U1) có các tùy chọn sử dụng chức năng của hệ thống như Tìm kiếm thông tin (U8), Đăng ký môn học trong học kỳ (U3). Tùy vào các ràng buộc đối với môn học về các môn điều kiện, số tín chỉ tích lũy, hệ thống sẽ xem xét sinh viên có đủ điều kiện học môn đó hay không và cập nhật lịch học cũng như lịch thi học kỳ (U12). Khi sinh viên không đủ điều kiện học môn nào đó, hệ thống có thể gửi thông báo (U14) về các môn điều kiện còn thiếu tới sinh viên để sinh viên

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

hoàn tất các môn học đó. Sinh viên có thể xem thông báo (U15), xem thời khóa biểu (U4) là lịch học các môn học đã đăng ký mà sinh viên có thể theo học cũng như lịch thi học kỳ các môn đó.

- Phòng đào tạo có nhiệm vụ cập nhật bảng điểm (U13) đối với sinh viên và sinh viên có thể xem kết quả học tập của mình (U6) sau khi đăng nhập hệ thống (U1). Bên cạnh đó hệ thống còn có các tùy chọn cho sinh viên như In bảng điểm (U7) hay tìm kiếm thông tin (U8).
- Cán bộ phòng đào tạo sau khi đăng nhập vào hệ thống (U1) có thể cập nhật thông tin quản lý xếp lớp (U12), quản lý điểm sinh viên (U13) và gửi thông báo tới sinh viên (U14).

Các yêu cầu phụ

Đôi khi chúng ta phải kết hợp các yêu cầu phi chức năng với một ca sử dụng cụ thể. Các yêu cầu phi chức năng không phù hợp với ca sử dụng nào sẽ được ghi lại thành tài liệu về các yêu cầu phụ. Ví dụ, Hệ thống đăng ký học theo tín chỉ có thể có các yêu cầu phụ sau đây:

S1. Hệ thống phải đáp ứng với hàng nghìn sinh viên cùng lúc đăng ký các khóa học.

S2. Người dùng nên sử dụng trình duyệt FireFox 3.5 hoặc mới hơn

S3. Thông tin về chương trình đào tạo, mỗi chuyên ngành nên hiển thị đầy đủ thông tin trên trang web hơn là dưới dạng một tệp tin download.

Kịch bản

Sau khi đã hoàn thành được biểu đồ phân rã ca sử dụng, người phát triển tiến hành xây dựng các kịch bản. Sau đây là một kịch bản tiêu biểu bao gồm các yêu cầu được trình bày ở trên.

Tên ca sử dụng	
Tác nhân chính	
Tiền điều kiện	
Đảm bảo tối thiểu	
Hậu điều kiện	
Chuỗi sự kiện chính	<ol style="list-style-type: none">1.2.3.
Ngoại lệ	<ol style="list-style-type: none">2.1. Ngoại lệ xảy ra ở bước 2<ol style="list-style-type: none">2.1.1.3.1. Ngoại lệ xảy ra ở bước 3<ol style="list-style-type: none">3.1.1.

Ví dụ: Kịch bản của ca sử dụng Đăng nhập

Tên Use Case	Đăng nhập
Tác nhân chính	Người dùng hệ thống
Điều kiện trước	Người dùng đã có tài khoản để đăng nhập hệ thống
Đảm bảo tối thiểu	Hệ thống cho phép người dùng đăng nhập lại
Điều kiện sau	Người dùng đăng nhập được vào hệ thống
Chuỗi sự kiện chính	<ol style="list-style-type: none">1. Người dùng chọn chức năng đăng nhập trên giao diện chính của hệ thống2. Hệ thống hiển thị form đăng nhập3. Người dùng nhập tài khoản và mật khẩu của mình

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

- | |
|---|
| 4. Hệ thống kiểm tra tính hợp lệ của tài khoản và mật khẩu |
| 5. Hệ thống hiển thị giao diện chính tương ứng với các chức năng của tác nhân |

Ngoại lệ:

- | |
|--|
| 4.1.Người dùng nhập tài khoản hay mật khẩu không chính xác |
| 4.1.1.Hệ thống thông báo lỗi và yêu cầu nhập lại |
| 4.2. Tài khoản người dùng đăng nhập không tồn tại |
| 4.2.1. Hệ thống thông báo lỗi và yêu cầu người sử dụng đăng ký |

Chú ý :

Ca sử dụng hệ thống chi tiết hơn nhiều so với ca sử dụng nghiệp vụ mà chúng ta đã thảo luận trong phần trước. Điều này phản ánh một thực tế rằng chúng ta hiện đang cố gắng phát triển hệ thống một cách chi tiết hơn; chúng ta muốn xác định chính xác về dịch vụ mà hệ thống sẽ cung cấp để xóa đi những sự phỏng đoán về phân tích và thiết kế hệ thống.

Khi viết chi tiết ca sử dụng, điều quan trọng là chúng ta phải xác định các *chức năng của hệ thống nhưng không phải cách thức thực hiện các chức năng ấy*. Ví dụ nếu chúng ta khai báo các bước như “Sinh viên lựa chọn button Chi tiết môn học”, chúng ta sẽ hạn chế cách thể hiện trong thiết kế giao diện người dùng sau này. Trừ phi đó là một yêu cầu bắt buộc, nếu không chúng ta nên cố gắng luôn sử dụng các từ ngữ trung lập như Chọn, Thêm, Xóa, Hiển thị...

3.4.5 Xếp ưu tiên các ca sử dụng

Sắp xếp các ca sử dụng trong yêu cầu hệ thống theo thứ tự ưu tiên về mặt cài đặt là một ý tưởng tốt đặc biệt khi tuân theo phương pháp luận phát triển gia tăng. Sắp xếp thứ tự các ca sử dụng có nghĩa là gán cho mỗi ca sử dụng một trọng số và được dùng trong việc lập kế hoạch phát triển và tăng dần trong tương lai. Một kỹ thuật sắp xếp ưu tiên thông dụng là dùng các *màu đèn giao thông*.

- Các ca sử dụng có mức độ ưu tiên màu xanh phải được cài đặt trong giai đoạn hiện tại.
- Các ca sử dụng có mức độ ưu tiên màu vàng không bắt buộc phải được cài đặt trong giai đoạn hiện tại và chỉ được cài đặt khi các ca sử dụng màu xanh đã hoàn thành.
- Các ca sử dụng có mức độ ưu tiên màu đỏ không được cài đặt trong giai đoạn hiện tại.

Trong thực tế, độ ưu tiên của các ca sử dụng không chỉ phụ thuộc vào sự mong muốn của khách hàng mà còn phụ thuộc vào kiến trúc hệ thống và việc cài đặt ca sử dụng trong giai đoạn hiện tại. Việc lựa chọn độ ưu tiên cho các ca sử dụng đòi hỏi nhiều kỹ năng và kinh nghiệm. Cách tốt nhất là ta nên đặt những ca sử dụng dễ dàng cài đặt hơn lên đầu bởi chúng sẽ giúp nhóm phát triển hiểu thêm về hệ thống và tất nhiên sẽ ít rủi ro hơn. Nếu dự án có thời gian vào thời điểm cuối (sau khi đã kết thúc những ca sử dụng màu xanh và màu vàng), ta nên xem xét lại tình hình dự án cũng như kế hoạch cho giai đoạn sắp tới. Ví dụ, sắp xếp ưu tiên lại cho những ca sử dụng chưa được thực thi.

Ví dụ: Sắp xếp ưu tiên cho Hệ thống quản lý học theo tín chỉ

- **Xanh:**

- U1: Đăng nhập
- U3: Đăng ký khóa học
- U4: Xem thời khóa biểu

- U5: Xem lịch thi
- U8: Tìm kiếm thông tin
- U9: Xem lịch giảng dạy
- U14: Gửi thông báo
- **Vàng:**
 - U2: Thoát
 - U11: Thay đổi thông tin cá nhân
 - U12: Quản lý xếp lớp
 - U13: Quản lý điểm
 - U15: Xem thông báo
- **Đỏ:**
 - U6: Xem điểm học tập
 - U7: In bảng điểm
 - U10: Đổi mật khẩu

3.4.6 Phác họa giao diện người dùng

Suy nghĩ cẩn thận về giao diện người dùng của hệ thống có thể giúp ta hiểu rõ ràng hơn các ca sử dụng. Các giao diện này có thể được phác họa cùng với nhà tài trợ/khách hàng ở giai đoạn đầu và kết quả được lưu lại thành *các bản phác họa giao diện người dùng*.

Các phác họa này nên được xem như là những hướng dẫn hơn là bản thiết kế chuyên nghiệp. Nó có thể giúp chúng ta xác định và phân rã các chức năng mà có thể cài đặt theo sở thích cá nhân. Do ca sử dụng và giao diện người dùng đều cùng biểu diễn một phân rã chức năng hệ thống, nên chúng ta cần chỉ rõ sự tương ứng giữa hai khái niệm này để có thể phục vụ quá trình cài đặt sau này và khi trình bày phác thảo giao diện nên kèm theo kịch bản tương ứng. Ví dụ, trong Hệ quản lý học theo tín chỉ, có ba loại đối tượng truy nhập: sinh viên, giảng viên, nhân viên. Vì vậy cần phải có ba giao diện riêng biệt tương ứng với mỗi loại đối tượng.

Ví dụ: Phác thảo giao diện người dùng với kịch bản Sinh Viên đăng ký môn học

Tên Use Case	Sinh viên đăng ký môn học
Tác nhân chính	Sinh viên
Điều kiện trước	Sinh viên đã có tài khoản để đăng nhập hệ thống
Đảm bảo tối thiểu	Hệ thống cho phép sinh viên đăng ký lại
Điều kiện sau	Sinh viên đăng ký thành công các môn học đủ điều kiện
Chuỗi sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng chọn chức năng đăng ký học trên giao diện chính của hệ thống 2. Hệ thống hiển thị form đăng ký học cho sinh viên với các thông tin cần thiết 3. Sinh viên chọn khóa học muốn đăng ký học cùng (có thể đăng ký học cùng một số môn học với khóa trước nếu sinh viên phải học lại môn đó) và chọn “Hiển thị danh sách môn học” 4. Hệ thống hiển thị danh sách các môn học mà sinh viên có thể đăng ký học tương ứng với mỗi khóa học cùng 5. Sinh viên lựa chọn các môn học mình muốn đăng ký học và chọn “Đăng ký” 6. Hệ thống kiểm tra thông tin về các môn điều kiện của sinh viên và hiển thị thông báo “Đăng ký môn học thành công”

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

Ngoại lệ

- 5.1. Sinh viên đăng ký môn học trong khi chưa học đủ các môn điều kiện hoặc còn nợ môn từ các kỳ trước
- 5.1.1. Hệ thống thông báo sinh viên không đủ điều kiện học và yêu cầu sinh viên đăng ký lại.



Hình 3.7: Giao diện Sinh viên Đăng ký môn học

3.5 CASE STUDY: HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Phản này tập trung trình bày các bước trong giai đoạn xác định yêu cầu hệ thống và bỏ qua phần xác định yêu cầu nghiệp vụ.

Xác định và mô tả các tác nhân

Sinh viên: người sử dụng có những thông tin được lưu trữ trong CSDL sinh viên như: mã sinh viên, họ tên, lớp, khóa... Mỗi sinh viên phải được cung cấp mật khẩu truy nhập phù hợp với mã sinh viên để có thể truy cập vào hệ thống.

Giảng viên: người sử dụng có những thông tin được lưu trữ trong CSDL giáo viên như: mã giáo viên, họ tên, khoa... Mỗi giảng viên phải được cung cấp mật khẩu truy nhập phù hợp với mã giảng viên để có thể truy cập vào hệ thống.

Nhân viên phòng đào tạo: người chịu trách nhiệm điều hành hệ thống.

Xác định và mô tả ca sử dụng

- U1: Đăng nhập: các tác nhân đăng nhập hệ thống.
- U2: Thoát: các tác nhân thoát khỏi hệ thống.
- U3: Đăng ký môn học: sinh viên đăng ký các môn học trong kỳ tới.
- U4: Xem thời khóa biểu: sinh viên xem thời khóa biểu chi tiết các lớp học đã đăng ký của mình
- U5: Xem lịch thi học kỳ: sinh viên xem lịch thi học kỳ các môn đã đăng ký
- U6: Xem điểm học tập: sinh viên xem kết quả học tập của mình.
- U7: In bảng điểm: sinh viên in bảng điểm các khóa học của mình.
- U8: Tìm kiếm thông tin: sinh viên xem danh sách các môn học của từng khoa, chuyên ngành, hệ đào tạo
- U9: Đăng ký môn dạy: giảng viên đăng ký môn sẽ dạy trong kỳ tới.
- U10: Xem lịch giảng dạy: giảng viên xem lịch giảng dạy của mình trong kỳ tới.
- U11: Đổi mật khẩu: người dùng thay đổi mật khẩu truy cập hệ thống.
- U12: Thay đổi thông tin cá nhân: Người dùng thay đổi thông tin cá nhân sau khi đăng nhập hệ thống
- U13: Quản lý sinh viên: Nhân viên phòng đào tạo thực hiện chức năng quản lý sinh viên với các thao tác cơ bản: thêm sinh viên, sửa thông tin sinh viên, xóa sinh viên.
- U14: Quản lý giảng viên: Nhân viên phòng đào tạo thực hiện chức năng quản lý Giảng viên với các thao tác cơ bản: thêm giảng viên, sửa thông tin giảng viên, xóa giảng viên.
- U15: Quản lý cơ bản: Nhân viên phòng đào tạo thực hiện các chức năng quản lý cơ bản như thêm Khoa, thêm Chuyên ngành...
- U16: Quản lý đào tạo: Nhân viên phòng đào tạo thực hiện các chức năng quản lý đào tạo như: quản lý học phần, quản lý đăng ký học và xếp lớp cho sinh viên, quản lý đăng ký dạy của giảng viên, quản lý kết quả học tập của sinh viên.
- U17: Gửi thông báo: phòng đào tạo gửi các thông báo tới sinh viên.
- U18: Xem thông báo: giảng viên, sinh viên xem thông báo của phòng đào tạo gửi đến

Khảo sát các ca sử dụng

Khi truy cập vào hệ thống đăng ký học theo tín chỉ, yêu cầu người dùng phải đăng nhập vào hệ thống (U1) trước khi có thể sử dụng các chức năng của hệ thống. Sau khi đăng nhập hệ thống, người dùng có thể sửa đổi mật khẩu (U11) tài khoản của mình. Giảng viên sau khi đăng nhập hệ thống (U1) có thể đăng ký môn dạy và lớp dạy trong học kỳ (U9) và có thể xem lịch giảng dạy của mình (U10)

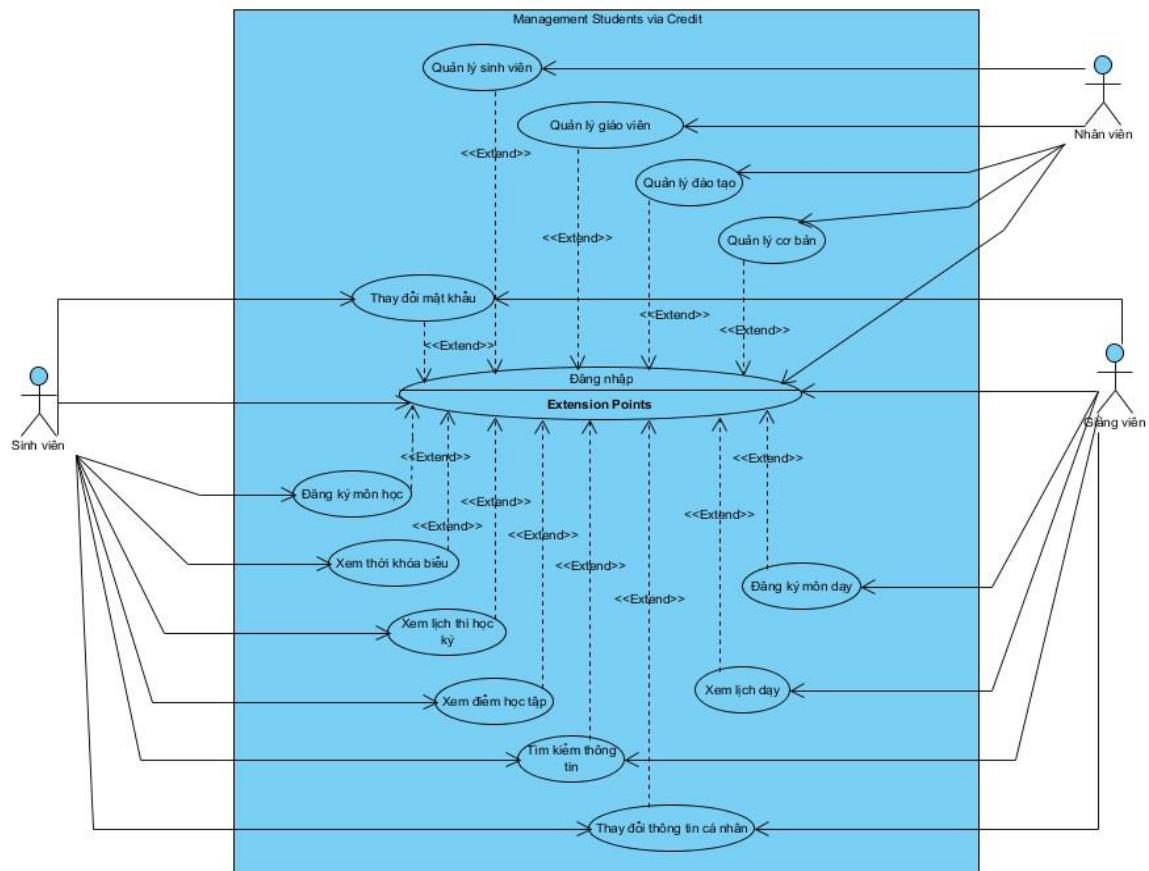
Sinh viên sau khi đăng nhập vào hệ thống (U1) có các tùy chọn sử dụng chức năng của hệ thống như Tìm kiếm thông tin (U8), Đăng ký môn học trong học kỳ (U3). Tùy vào các ràng buộc đối với môn học về các môn điều kiện, số tín chỉ tích lũy, cán bộ phòng đào tạo có thể xét sinh viên có đủ điều kiện học môn đó hay không và cập nhật lịch học cũng như lịch thi học kỳ (U16). Khi sinh viên không đủ điều kiện học môn nào đó, phòng đào tạo có thể gửi thông báo (U17) về các môn điều kiện còn thiếu tới sinh viên để sinh viên hoàn tất các môn học đó. Sinh viên có thể xem thông báo (U18), xem thời khóa biểu (U4) là lịch học các môn học đã đăng ký mà sinh viên có thể theo học cũng như lịch thi học kỳ các môn đó.

Phòng đào tạo có nhiệm vụ cập nhật bảng điểm (U16) đối với sinh viên, sinh viên có thể xem kết quả học tập của mình (U6) khi đăng nhập hệ thống (U1). Bên cạnh đó hệ thống còn có các tùy chọn cho sinh viên như In bảng điểm (U7) hay tìm kiếm thông tin (U8). Cán bộ Đào tạo sau khi đăng nhập vào hệ thống (U1) có thể cập nhật thông tin quản lý cơ bản (U15), thông tin về sinh viên (U13), thông tin về giảng viên (U14).

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

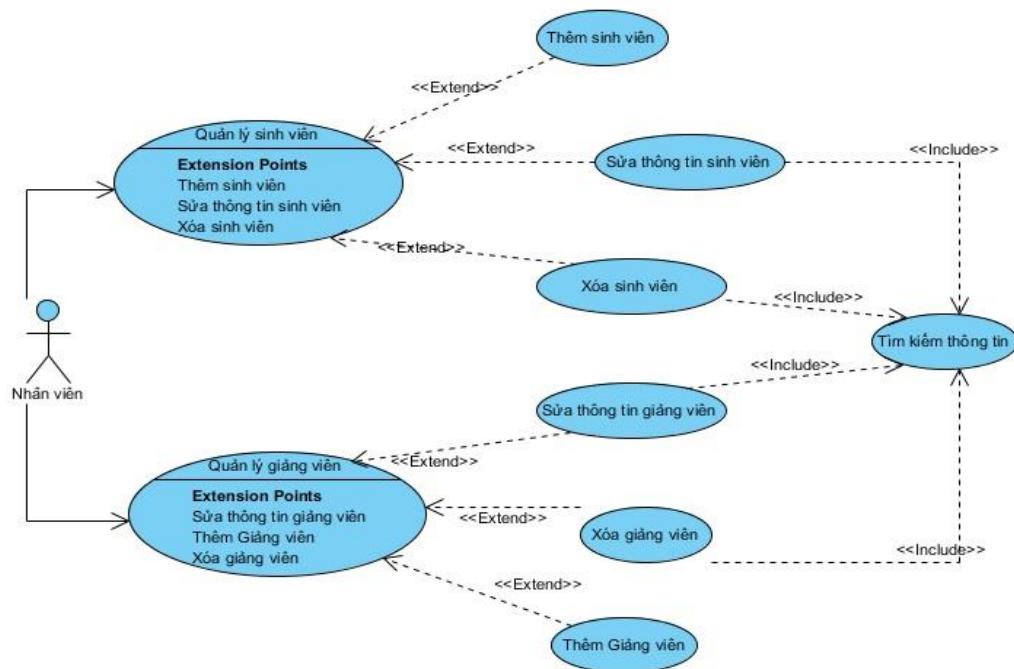
Qua chức năng quản lý đào tạo (U16), cán bộ đào tạo chia lớp học theo Học phần, chuyển đổi học phần cho sinh viên đã đăng kí vào những học phần mà có số lượng sinh viên đăng kí nhỏ hơn điều kiện tối thiểu.

Xây dựng Biểu đồ ca sử dụng

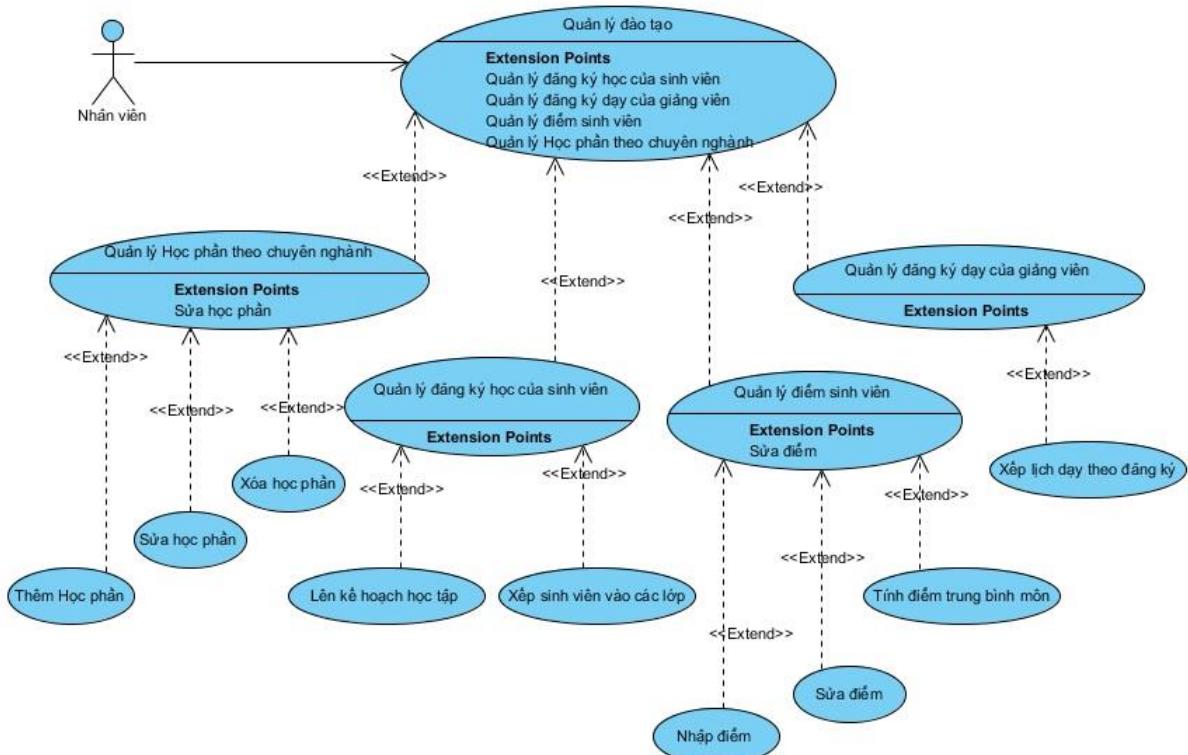


Hình 3.8: Biểu đồ ca sử dụng tổng quát

Biểu đồ ca sử dụng phân rã cấp 1

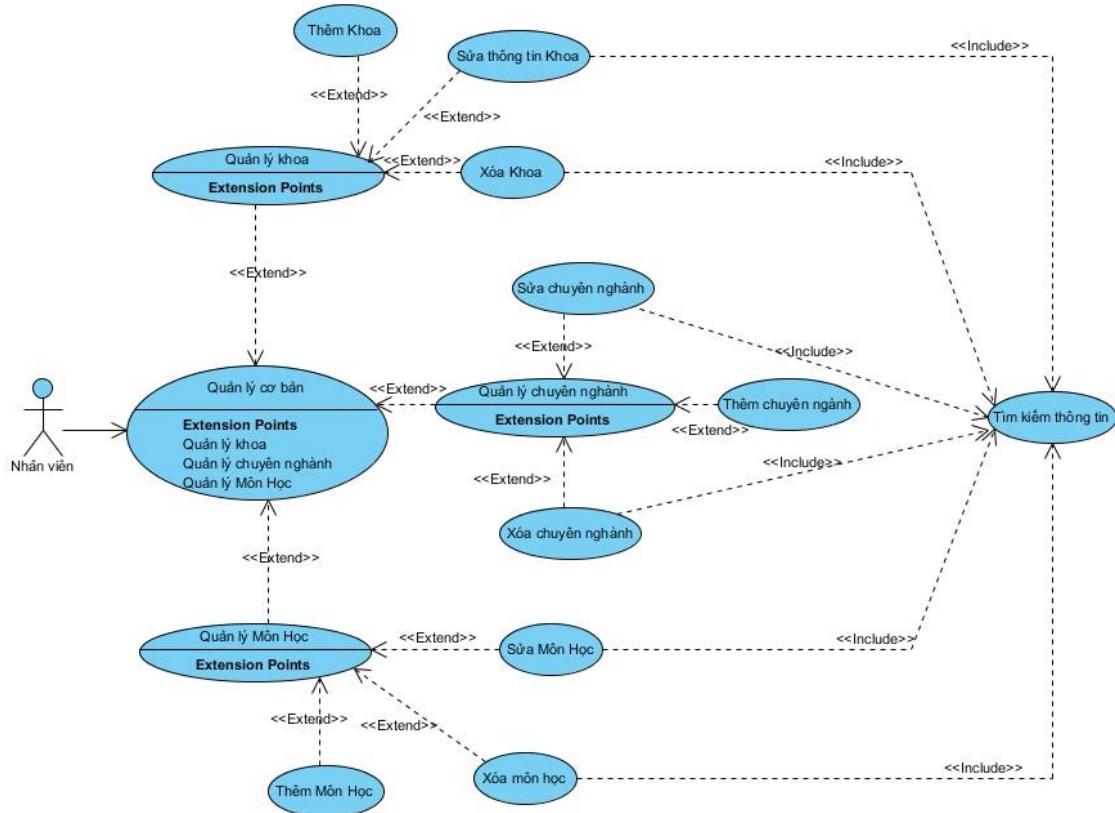


Hình 3.9: Phân rã ca sử dụng Quản lý sinh viên – Quản lý giảng viên



Hình 3.10: Phân rã ca sử dụng Quản lý đào tạo

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU



Hình 3.11: Phân rã ca sử dụng Quản lý cơ bản

Kịch bản và phác thảo giao diện người dùng

Quản lý khoa

Thêm khoa

Tên Use Case	Thêm Khoa
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Khoa
Kích hoạt	Button “Thêm Khoa” trên Form Quản lý Khoa

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt form Quản lý cơ bản
2. Hệ thống hiện thị ba tùy chọn: Quản lý **Khoa**, Quản lý **Chuyên Ngành**, Quản lý **Môn học**.
3. **Cán bộ đào tạo** lựa chọn Quản lý **Khoa**
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về **Khoa** như **Mã Khoa**, **Tên Khoa**, **Trưởng Khoa**, **Mô tả chung**
5. **Cán bộ đào tạo** nhập thông tin về **Khoa** và chọn “Thêm Khoa”
6. Hệ thống kiểm tra thông tin và lưu vào cơ sở dữ liệu
7. Hệ thống thông báo thêm khóa thành công

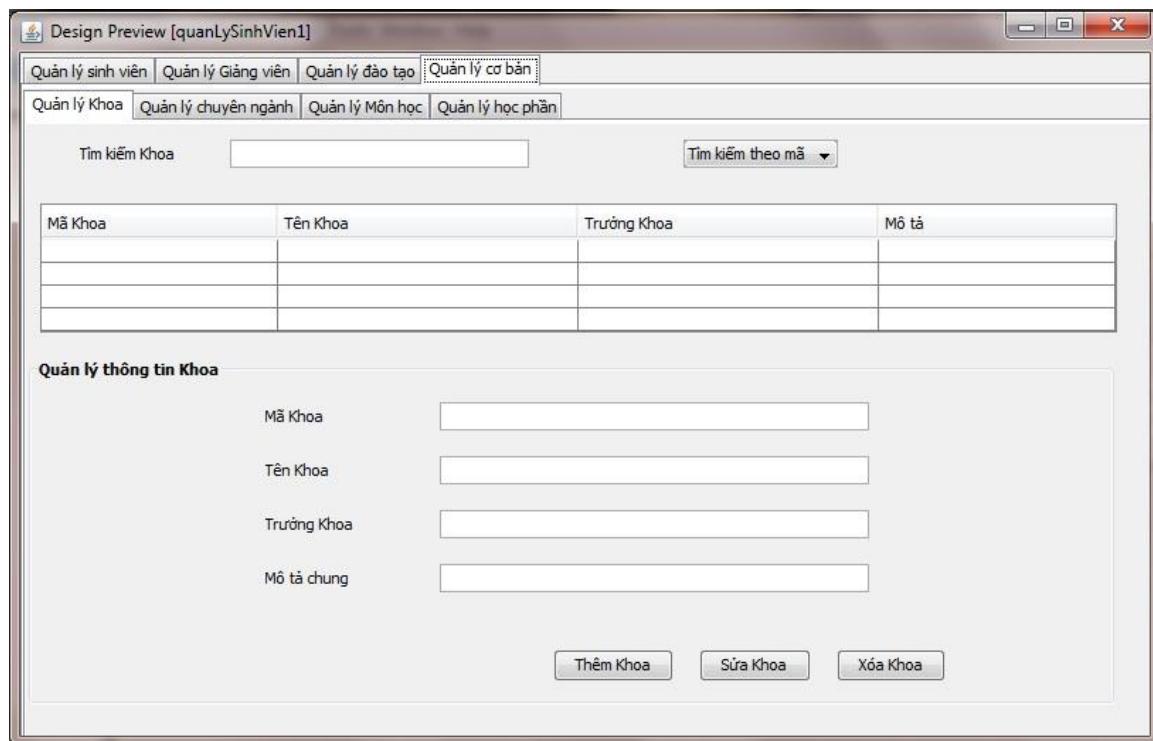
Ngoại lệ:

6.1. Hệ thống thông báo **Mã Khoa** không hợp lệ (trùng, không đúng dạng)

6.1.1. Hệ thống yêu cầu nhập lại **Mã Khoa**.

6.1.2. **Cán bộ đào tạo** nhập lại và tiếp tục các bước sau

Phác thảo giao diện Quản lý Khoa



Hình 3.12: Giao diện Quản lý Khoa

Quản lý Môn Học

Thêm Môn học

Tên Use Case	Thêm Môn Học
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Môn Học
Kích hoạt	Chọn “Thêm Môn Học” trên Form Quản lý Môn Học

Chuỗi sự kiện chính

1. **Cán bộ đào tạo** kích hoạt form Quản lý cơ bản
2. Hệ thống hiện thị ba tùy chọn: Quản lý **Khoa**, Quản lý **Chuyên Ngành**, Quản lý **Môn học**. Quản lý **Học phần**.
3. Cán bộ đào tạo lựa chọn Quản lý **Môn Học**
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về **Môn Học** như **Mã môn**, **tên môn**, **số tín chỉ**, **số tiết Lý thuyết**, **số tiết Thực hành**, **số tiết Bài tập**, **Khoa phụ trách**, **Môn điều kiện**.
5. **Cán bộ đào tạo** nhập thông tin về Môn Học và chọn “Thêm Môn Học”
6. Hệ thống kiểm tra thông tin và lưu vào cơ sở dữ liệu
7. Hệ thống thông báo thêm **Môn Học** thành công

Ngoại lệ:

- 6.1. Hệ thống thông báo **Mã Môn Học** không hợp lệ (trùng, không đúng dạng)
 - 6.1.1. Hệ thống yêu cầu nhập lại **Mã Môn Học**
 - 6.1.2. **Cán bộ đào tạo** nhập lại và tiếp tục các bước sau

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

Phác thảo giao diện Quản lý Môn học

Hình 3.13: Giao diện Quản lý Môn học

Quản lý sinh viên

Thêm sinh viên

Tên Use Case	Thêm Sinh Viên
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Sinh Viên
Kích hoạt	Button “Thêm Sinh viên” trên Form Quản lý Sinh Viên

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt **form Quản lý Sinh viên**
2. Hệ thống hiển thị hai tùy chọn “Thêm Sinh Viên” và “Sửa/xóa sinh viên”
3. Cán bộ đào tạo lựa chọn **form “Thêm Sinh viên”**
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Sinh viên và các tùy chọn: Tìm kiếm, Thêm Sinh viên, Sửa Sinh viên, Xóa Sinh viên.
5. Cán bộ đào tạo nhập thông tin về **Sinh viên** gồm có **Mã SV, Họ tên, Ngày sinh, Giới tính, Khoa, Chuyên ngành, Khóa, Lớp** và chọn “Thêm Sinh viên”
6. Hệ thống kiểm tra thông tin và lưu vào cơ sở dữ liệu

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

7. Hệ thống thông báo thêm Sinh viên thành công

Ngoại lệ:

- 6.1 Hệ thống thông báo Mã Sinh viên nhập thông tin không hợp lệ
 - 6.1.1 Hệ thống yêu cầu nhập lại.
 - 6.1.2 Cán bộ đào tạo nhập lại và tiếp tục các bước sau

Phác thảo giao diện quản lý sinh viên:

Hình 3.14: Giao diện Thêm Sinh Viên

Hình 3.15: Giao diện Sửa/xóa Sinh viên

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

Giảng viên đăng ký dạy

Tên use case:	Giảng viên đăng ký môn dạy
Ngữ cảnh	Giảng viên đăng ký môn dạy thành công
Tác nhân chính	Giảng viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Giảng viên đăng ký thành công môn dạy
Đảm bảo tối thiểu	Trở lại màn hình đăng ký để giảng viên có thể đăng ký lại
Kích hoạt	Người dùng chọn chức năng đăng ký môn dạy
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Giảng viên kích hoạt form đăng ký môn dạy 2. Hệ thống hiển thị danh sách và thông tin chi tiết các môn và khóa học giảng viên có thể đăng ký dạy theo chuyên ngành 3. Giảng viên lựa chọn các môn dạy và khóa dạy tương ứng với mỗi môn đó 4. Hệ thống kiểm tra và cập nhật thông tin giảng viên đã đăng ký vào cơ sở dữ liệu và hiển thị đăng ký thành công 5. Hệ thống thông kê và hiển thị chi tiết thông tin các môn dạy và khóa dạy mà giảng viên đã đăng ký dưới dạng bảng 	
Ngoại lệ	
<ol style="list-style-type: none"> 4.1. Thông tin đăng ký không đủ điều kiện: <ul style="list-style-type: none"> • Khóa học đã có giảng viên đăng ký trước đó • Thời gian một số môn dạy giảng viên đăng ký bị trùng nhau • Giảng viên (thỉnh giảng) không được dạy môn đăng ký 4.2. Hệ thống thông báo đăng ký không thành công và quay lại bước 2 	

Phác thảo giao diện Giáo viên đăng ký môn dạy

Hình 3.16: Giáo viên đăng ký môn dạy**Sinh viên đăng ký môn học**

Tên use case:	Sinh viên đăng ký môn học
Tác nhân chính	Sinh viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Sinh viên đăng ký thành công môn học
Đảm bảo tối thiểu	Trở lại màn hình đăng ký để sinh viên có thể đăng ký lại
Kích hoạt	Người dùng chọn chức năng đăng ký môn học
Chuỗi sự kiện chính	
1. Sinh viên kích hoạt form đăng ký môn học 2. Hệ thống hiển thị danh sách và thông tin chi tiết các khóa học Sinh viên có thể đăng ký học trong học kỳ gồm có thông tin về Mã môn , Tên môn , Số tín chỉ , Loại môn học , Môn điều kiện , Môn nợ . 3. Sinh viên lựa chọn các môn học và khóa học tương ứng với mỗi môn học 4. Hệ thống kiểm tra và cập nhật thông tin Sinh viên đã đăng ký vào cơ sở dữ liệu và hiển thị đăng ký thành công 5. Hệ thống thống kê và hiển thị chi tiết thông tin các môn học và khóa học mà Sinh viên đã đăng ký dưới dạng bảng	

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

Ngoại lệ

- 4.3. Thông tin đăng ký không đủ điều kiện:
 - Sinh viên chưa học đủ các môn điều kiện của môn học
 - Môn học đã đủ l López và đủ số lượng **Sinh viên** trong 1 lớp.
 - **Sinh viên** đăng ký hai khóa học của cùng một môn học
 - Thời gian một số buổi học môn học **Sinh viên** đăng ký bị trùng nhau
- 4.4. Hệ thống thông báo đăng ký không thành công và quay lại bước 2

Phác thảo giao diện sinh viên đăng ký môn học



Hình 3.18: Giao diện sinh viên đăng ký học

3.6 KẾT LUẬN

Trong chương này chúng ta đã trình bày

- Tầm quan trọng của quá trình xác định yêu cầu chức năng và yêu cầu phi chức năng trong pha xác định yêu cầu.
- Mô hình hóa yêu cầu nghiệp vụ và chức năng hệ thống bằng cách dùng các tác nhân và ca sử dụng nghiệp vụ mức khái quát.
- Mô hình hóa yêu cầu hệ thống với mô hình ca sử dụng đầy đủ bao gồm các ca sử dụng, độ ưu tiên các ca sử dụng, biểu đồ ca sử dụng, chi tiết ca sử dụng bằng kịch bản và phác thảo giao diện người dùng. Biểu đồ giao tiếp và biểu đồ hoạt động có thể được

CHƯƠNG 3. XÁC ĐỊNH YÊU CẦU

cân nhắc tùy chọn trong giai đoạn này, nhưng một bảng thuật ngữ để mô tả về các thuật ngữ liên quan đến miền của hệ thống luôn luôn là cần thiết.

Mặc dù trong giáo trình, một số khía cạnh về xác định yêu cầu được trình bày theo các bước, nhưng mỗi phương pháp luận khác nhau sẽ có các yêu cầu tài liệu tương ứng với từng pha/công đoạn phát triển phần mềm mà phương pháp đó quy định. Vì vậy, trong quá trình phát triển phần mềm thực tế, chúng ta không bắt buộc phải tuân theo đúng các bước như trên.

BÀI TẬP

1. Hãy tham khảo các trang link sau đây và tìm hiểu các nguyên nhân dẫn đến thất bại của các dự án phần mềm.
 - Lorin J. May, *Major Causes of Software Project Failures*
<http://info.psu.edu.sa/psu/cis/biq/se501/a/a1/MajorCausesofSoftwareProjectFailures.pdf>
 - Khaled El Emam and A. Günes, Koru A, *Replicated Survey of IT Software Project Failures*
http://www.ruor.uottawa.ca/fr/bitstream/handle/10393/12988/El_Emam_Khaled_2008_A_repli_cated_survey_of_IT_software.pdf
2. Hãy trình bày sự khác biệt trong việc xây dựng yêu cầu mô hình nghiệp vụ và xây dựng yêu cầu mô hình hệ thống.
3. Xây dựng các ca sử dụng của hệ quản lý email và biểu diễn quan hệ giữa các ca sử dụng này. Sử dụng VP để hoàn thiện biểu đồ ca sử dụng. Hãy khảo sát và nêu nhận xét về các giao diện của các hệ email như GMail, yahooMail.
4. Khảo sát trên mạng các hệ quản lý tài liệu điện tử như quản lý các bài báo khoa học. Bạn đọc sau khi tìm kiếm có thể đọc tóm tắt bài báo nhưng khi ấn vào download thì hệ thống đòi hỏi đăng nhập. Hãy xây dựng biểu đồ ca sử dụng của hệ thống này.
5. Khảo sát các hệ quản lý thương mại điện tử như Amazon và eBay về các mặt sau đây: các chức năng, giao diện. Hãy xây dựng biểu đồ ca sử dụng của hệ thống này.
6. Xác định yêu cầu cho Hệ thống Quản lý đặt chỗ thuê xe hơi từ quan điểm nghiệp vụ và quan điểm hệ thống như trình bày trong tài liệu. Sử dụng Tool VP để hoàn thiện bài tập của mình.
7. Xác định yêu cầu cho Hệ thống Quản lý Thư viện tài liệu điện tử từ quan điểm nghiệp vụ và quan điểm hệ thống như trình bày trong tài liệu. Sử dụng Tool VP để hoàn thiện bài tập của mình.
8. Xác định yêu cầu cho Hệ thống Quản lý Thư viện thường từ quan điểm nghiệp vụ và quan điểm hệ thống như trình bày trong tài liệu. Sử dụng Tool VP để hoàn thiện bài tập của mình.
9. Xác định yêu cầu cho Hệ thống Quản lý cửa hàng bán sách từ quan điểm nghiệp vụ và quan điểm hệ thống như trình bày trong tài liệu. Sử dụng Tool VP để hoàn thiện bài tập của mình.
10. Xác định yêu cầu cho Hệ thống Quản lý siêu thị điện máy từ quan điểm nghiệp vụ và quan điểm hệ thống như trình bày trong tài liệu. Sử dụng Tool VP để hoàn thiện bài tập của mình.

CHƯƠNG 4

PHÂN TÍCH YÊU CẦU

4.1 GIỚI THIỆU

Phân tích yêu cầu một hệ thống là nhằm trả lời câu hỏi hệ thống sẽ làm *những gì*, hơn là chỉ ra *cách thức* làm những việc đó. Vì vậy, chúng ta cần phải phân rã các yêu cầu phức tạp được trình bày trong pha xác định yêu cầu thành các nhân tố chính cùng mối quan hệ giữa chúng để làm cơ sở cho giải pháp được trình bày trong pha thiết kế sau này. Phân tích là pha đầu tiên giúp chúng ta hiểu được mô hình thực sự của hệ thống là gì. Quá trình phân tích yêu cầu thường được chia thành hai giai đoạn:

- Xây dựng *mô hình phân tích tĩnh*: Mô hình phân tích tĩnh được thể hiện bởi một *biểu đồ lớp* để chỉ ra các đối tượng mà hệ thống sẽ sử dụng và cách thức chúng tương tác với nhau.
- Xây dựng *mô hình phân tích động*: Mô hình phân tích động có thể được thể hiện bởi biểu đồ tương tác như *biểu đồ giao tiếp* hay *biểu đồ tuần tự* nhằm chỉ ra tính khả thi của mô hình phân tích tĩnh.

Pha phân tích là pha tiếp theo của pha xác định yêu cầu nên có hai đầu vào cho pha phân tích như đã trình bày trong Chương 3. Một là *mô hình yêu cầu nghiệp vụ* nhằm mô tả các thao tác bằng tay và tự động hóa của luồng công việc được thể hiện theo *hướng nghiệp vụ* các tác nhân, ca sử dụng, đối tượng, bản thuật ngữ và có thể có biểu đồ giao tiếp và hoạt động. Hai là *mô hình yêu cầu hệ thống* nhằm mô tả cái nhìn tổng quát bên ngoài hệ thống được thể hiện theo bởi các tác nhân, các ca sử dụng và biểu đồ ca sử dụng, phác họa giao diện người dùng, bản thuật ngữ đã bổ sung và các yêu cầu phi chức năng.

Những đầu vào này cần phải được chuyển thành mô hình các đối tượng cùng với các thuộc tính và các mối quan hệ giữa chúng để được xử lý bởi hệ thống đề xuất. Những đối tượng này sẽ tồn tại bên trong chính hệ thống hoặc tại biên của hệ thống và có thể truy nhập được thông qua một hay nhiều giao diện. Hầu hết các đối tượng mà chúng ta khám phá được ở giai đoạn này sẽ tương ứng với các đối tượng vật lý hay các khái niệm của thế giới hiện thực. Một khi có được mô hình các đối tượng của hệ thống, chúng ta cũng cần phải tự kiểm định xem chúng có thể hỗ trợ cho một giải pháp không.

4.2 CÁC BƯỚC TRONG PHA PHÂN TÍCH

Câu hỏi đặt ra là tại sao cần phải thực hiện pha phân tích trong các dự án phần mềm trong khi mục đích của chúng ta là cài đặt hệ thống cần xây dựng. Hãy hình dung một ví dụ tương tự về xây dựng nhà như sau. Khi xây một ngôi nhà, chúng ta muốn có được một kế hoạch chu đáo trước khi thực hiện những nhát đầu tiên. Thông thường, để biết ngôi nhà sau này có giống như ý định mong muốn hay không chúng ta phải thuê kiến trúc sư và kỹ sư xây dựng bắn vẽ cho ngôi nhà đó. Nhưng để xây dựng bắn vẽ khả thi sau này, các nhà xây dựng không những phải hiểu rõ yêu cầu của khách hàng, mà còn phải nắm vững địa hình của khu vực xây dựng, các công nghệ có thể sử dụng, thời gian, giá thành...

Cho đến thời điểm hiện thời, tài liệu của cả hai mô hình nghiệp vụ và mô hình hệ thống từ pha xác định yêu cầu chưa thể giúp chúng ta hiểu được đầy đủ cách thức hoạt động của hệ

thống sau này. Vì mô hình yêu cầu nghiệp vụ chỉ mô tả thực tại của tổ chức mà chưa nói gì được hiện thực khi phần mềm được thêm vào. Mô hình hệ thống với ca sử dụng cũng chưa thể giúp chúng ta hiểu vấn đề một cách đầy đủ. Lý do là các ca sử dụng chỉ mô tả tương tác giữa tác nhân và biến của hệ thống nên hệ thống chỉ là một hộp đen khi nhìn từ bên ngoài. Hơn nữa, chi tiết các ca sử dụng được viết bằng ngôn ngữ tự nhiên nên có thể dẫn đến cách hiểu không rõ ràng hay nhập nhằng giữa nhóm phát triển và đối tác khách hàng.

Một khi đã hoàn thành quá trình phân tích tĩnh, khách hàng mới có thể xác nhận được rằng hiểu biết của chúng ta về đối tượng nghiệp vụ là chính xác để chúng ta có thể tiến hành thực hiện pha thiết kế. Sau khi tiến hành phân tích động, chúng ta mới có thể xác định được những đối tượng phân tích thực sự hỗ trợ chức năng mà hệ thống yêu cầu. Phân tích yêu cầu được chia làm hai giai đoạn và bao gồm các bước sau đây:

Phân tích tĩnh

1. Xác định các lớp
2. Xác định quan hệ giữa các lớp
3. Xây dựng biểu đồ lớp
4. Xác định thuộc tính lớp
5. Cập nhật bảng thuật ngữ và các yêu cầu phi chức năng (Tùy chọn)

Phân tích động

1. Xây dựng biểu đồ giao tiếp (pha phân tích)
2. Gán phương thức cho các lớp

Trước hết chúng ta sẽ trình bày vắn tắt các bước để có cái nhìn ban đầu về các bước phân tích. Thông thường chúng ta sẽ sử dụng mô hình yêu cầu hệ thống để tìm các lớp ứng cử mô tả những đối tượng liên quan tới hệ thống, các quan hệ giữa các lớp này và biểu diễn mối quan hệ của chúng bằng biểu đồ lớp và đối tượng như đã trình bày trong Chương 2. Sau đó, xác định thuộc tính của các lớp với tên thuộc tính và kiểu của thuộc tính bằng cách tiến hành xem xét các ca sử dụng trong mô hình hệ thống, kiểm tra xem chúng có được hỗ trợ bởi các đối tượng mà chúng ta đề ra hay không, bổ sung các thuộc tính và mối quan hệ giữa chúng. Các thao tác sẽ được bổ sung sau khi ta đi đến giai đoạn hiện thực hóa ca sử dụng.

Thuật ngữ *hiện thực hóa ca sử dụng* (use case realization) có nghĩa là làm cho các ca sử dụng trở thành cài đặt được. Những thao tác được khám phá trong suốt quá trình hiện thực hóa ca sử dụng có thể sẽ bị bỏ qua trong quá trình thiết kế. Vì vậy, trong giai đoạn này, chúng ta cố gắng xây dựng sự tin cậy về tính khả thi mà không thiết kế giải pháp.

Chúng ta cần cho khách hàng xem các biểu đồ lớp với đầy đủ thuộc tính để họ giúp phát hiện lỗi vì những người này chắc chắn hiểu biết về nghiệp vụ hơn chúng ta. Như vậy, một thành viên của nhóm nên tóm lược những thông tin thể hiện trên biểu đồ lớp để khách hàng dễ dàng theo dõi. Vì biểu đồ lớp là tương đối dễ dàng để những người không phải là lập trình viên có thể hiểu được và điều này sẽ giúp đưa ra những bình luận hữu ích. Quyết định khi nào trình bày biểu đồ lớp cho khách hàng là do nhóm phát triển dự án, nhưng thường nên làm điều này ít nhất hai lần: trong quá trình phát hiện lỗi và khi đã sửa lỗi đó.

Các biểu đồ giao tiếp và các thao tác của đối tượng không nên đưa cho khách hàng xem vì nó hơi phức tạp và thật ra là không cần thiết. Tuy nhiên, một số khách hàng như các nhà quản

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

lý kỹ thuật có thể thích được chỉ ra một vài phân tích động để tăng thêm sự tin cậy của họ nhưng nên làm trong cuộc gặp khác. Trong phần còn lại của chương này, chúng ta sẽ xem xét chi tiết các bước trong phân tích động và phân tích tĩnh.

4.3 PHÂN TÍCH TĨNH

Mô hình phân tích tĩnh liên quan đến việc phân rã hệ thống thành các thành phần logic và vật lý và cách mà các thành phần này tương tác với nhau. Mục đích của nó là nhằm mô tả cách chúng ta *xây dựng* và *khởi tạo* hệ thống như thế nào. Sau đây chúng ta sẽ trình bày chi tiết các bước trong phân tích tĩnh đã trình bày ở phần 4.2.

4.3.1. Xác định các lớp

Hầu như không có một quy tắc chung cho việc phát hiện ra các lớp. Tìm các lớp là một công việc đòi hỏi sự sáng tạo và cần được thực hiện với sự trợ giúp của chuyên gia miền ứng dụng. Vì tiến trình phân tích và thiết kế là tiến trình lặp đi lặp lại, nên danh sách các lớp sẽ thay đổi theo thời gian. Tập hợp của các lớp tìm ra ban đầu chưa chắc đã là tập hợp cuối cùng của các lớp sẽ được thực thi và biến đổi thành code sau này. Vì thế, người ta thường hay sử dụng khái niệm các *lớp ứng viên* (candidate class) để chỉ tập hợp những lớp đầu tiên được tìm ra cho hệ thống. Sau đây là một số gợi ý để tìm các lớp ứng viên:

- Lớp ứng viên thường là các *danh từ* được trích ra từ các kịch bản. Ví dụ, các từ hệ thống *System*, tín chỉ *Credit*, Môn học *Subject*... là các danh từ nhưng với chúng ta thì hệ thống *System* chỉ là một ranh giới của phần phát triển và bên ngoài nên không thể trở thành lớp ứng viên.
- Tác nhân (Actor), ví dụ sinh viên *Student* hoặc giảng viên *Instructor*, thành viên *Member* là các lớp ứng cử.
- Biên (Boundaries), ví dụ giao diện *Interface* liên quan đến các đối tượng nghiệp vụ thực hiện hành động như nhập thông tin môn học cùng với các thông tin về sở thích và hành vi. *Boundaries* là một phần riêng của phần mềm cho phép các tác nhân truy xuất được các đối tượng.
- Các kiểu thông dụng như *String*, *int*, *float*... thường được cung cấp bởi ngôn ngữ cài đặt hoặc từ thư viện của ngôn ngữ.

Chi tiết cách xác định các lớp sẽ được trình bày trong Case study.

4.3.2 Xác định quan hệ giữa các lớp

Với một danh sách các lớp ứng viên, chúng ta sẽ xem xét các quan hệ giữa chúng. Như đã trình bày trong Chương 1, có 4 loại quan hệ chính sau đây:

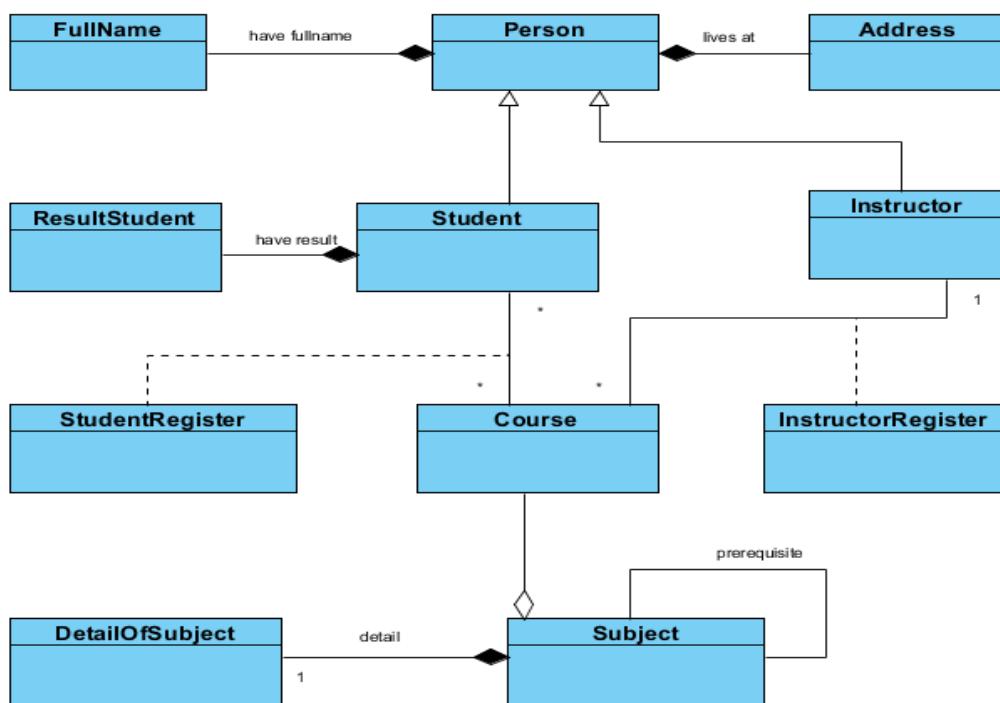
- Kế thừa (Inheritance): một lớp con sẽ kế thừa tất cả các thuộc tính và hành vi của lớp cha của nó.
- Liên kết (Association): các đối tượng của một lớp được liên kết với các đối tượng của lớp khác.
- Kết hợp (Aggregation): một loại liên kết mạnh, nghĩa là một thể hiện của một lớp được tạo ra bởi các thể hiện của một lớp khác.

- Hợp thành (Composition): quan hệ kết hợp mạnh, nghĩa là một đối tượng hợp thành không thể được dùng chung với các đối tượng khác và nó sẽ mất cùng với đối tượng được hợp thành.

Ké thừa là một kiểu quan hệ khác với ba loại quan hệ kia. Ké thừa mô tả một quan hệ trong thời gian biên dịch giữa các lớp trong khi các quan hệ kia mô tả kết nối trong thời gian chạy giữa các đối tượng. Theo chuẩn UML, tất cả các quan hệ trong thời gian chạy đều hình thành từ liên kết. Tuy nhiên, nhiều người sử dụng thuật ngữ liên kết với ý nghĩa khác với kết hợp và hợp thành. Lựa chọn giữa các quan hệ đòi hỏi một sự khéo léo, nghĩa là cần sử dụng khả năng trực giác, kinh nghiệm và sự phỏng đoán.

4.3.3 Xây dựng biểu đồ lớp

Biểu đồ lớp biểu diễn các lớp và quan hệ giữa chúng. Biểu đồ lớp cũng thể hiện các thuộc tính và hành động tương ứng với các lớp nhưng điều đó đòi hỏi nhiều không gian hơn. Hình 4.1 biểu diễn một biểu đồ lớp UML cho Hệ thống quản lý học tập theo tín chỉ (việc thêm thuộc tính và phương thức sẽ được đề cập sau này). Mỗi lớp được thể hiện bởi một hình chữ nhật có 3 gian với tên lớp bên trong (in đậm). Nếu lớp thuộc loại trừu tượng, tên lớp in nghiêng hay có thể thêm từ khóa *{abstract}* bên trên hoặc bên phải lớp thay cho chữ in nghiêng.



Hình 4.1: Biểu đồ Lớp

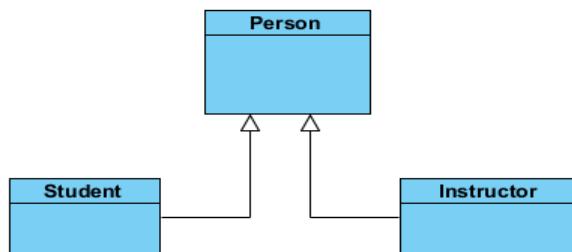
Quan hệ giữa các lớp được thể hiện bằng các đường nối với các chú thích khác nhau. Nếu không có kiến thức chuyên môn về UML, ta cũng có thể dễ dàng rút ra được thông tin từ biểu đồ lớp. Ví dụ, chúng ta có thể thấy rằng “một thành viên có thể có Họ tên và Địa chỉ”, “Mỗi sinh viên sẽ có kết quả học tập của mình”...

Các biểu diễn quan hệ lớp trong UML đã được trình bày trong Chương 1. Tuy nhiên, vì các quan hệ này không phải dễ dàng nắm bắt và nó đóng vai trò rất quan trọng liên quan chặt chẽ đến thiết kế và cài đặt sau này nên phần còn lại trong mục này sẽ dành tìm hiểu thêm về các quan hệ giữa các lớp.

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

Biểu diễn các quan hệ

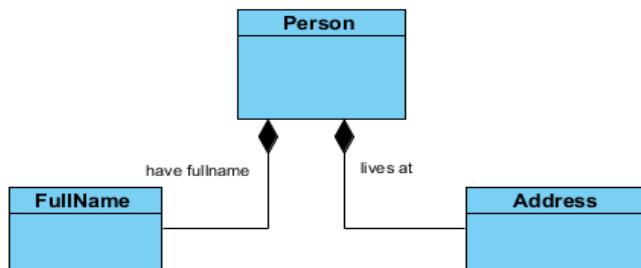
Xét một số quan hệ trong hệ thống quản lý học tập theo tín chỉ với cách biểu diễn cụ thể như sau:



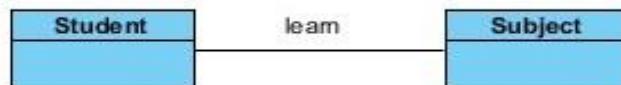
Hình 4.2: Quan hệ kế thừa trong UML



Hình 4.3: Quan hệ kết hợp trong UML



Hình 4.4: Quan hệ hợp thành trong UML



Hình 4.5: Quan hệ liên kết trong UML

- Quan hệ kế thừa được mô tả trong biểu đồ lớp bởi một đường với đầu mũi tên màu trắng được vẽ từ lớp con hướng tới lớp cha (lớp Student và Instructor là kế thừa của lớp Member)
- Quan hệ kết hợp trong UML được mô tả bởi một đường nối giữa hai lớp với một đầu có hình quả trám trắng ở cuối quan hệ chỉ quan hệ bộ phận trên lớp tổng thể. Ví dụ, Course là một phần của Subject.
- Quan hệ hợp thành trong UML tương tự như quan hệ kết hợp, nhưng đầu hình quả trám là màu đen ở cuối. Ví dụ, FullName và Address luôn là một phần của Member.
- Quan hệ liên kết được mô tả bởi một đường thẳng nối giữa hai lớp có liên quan về nghĩa. Nghĩa đó có thể được mô tả trên đường nối các quan hệ. Ở đây Sinh viên Student có quan hệ liên kết với Môn học Subject, nghĩa là Sinh viên sẽ đăng ký học môn học nào đó. Rõ ràng Môn Học không phải là một thành phần của Sinh viên và ngược lại.

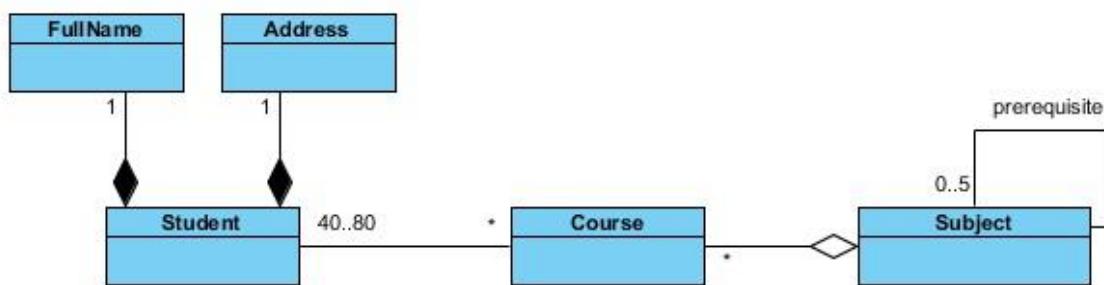
Khi phát triển mô hình lớp phân tích, chúng ta cần phải đảm bảo rằng *thông tin chỉ có thể suy diễn theo một cách*. Ví dụ, biểu đồ lớp 4.1 cho phép chúng ta tính số môn học mà sinh viên đăng ký từ StudentRegister qua liên kết với lớp Course và Student. Do đó, sẽ là dư thừa nếu cho rằng số môn học này có thể tính qua liên kết với lớp InstructorTeach mặc dù có thể có biểu diễn liên kết giữa Student và InstructorTeach.

Tính nhiều (Multiplicity)

Ngoài trừ quan hệ ké thừa, tất cả các quan hệ đều có thể hiện số lượng đối tượng được phép tham gia vào các quan hệ tại cuối mỗi liên kết. Số lượng đối tượng có thể có các dạng sau đây:

- n : lấy chính xác n
- $m...n$: số nào trong dải từ m đến n
- $p...*$: số nào trong dải từ p đến vô cùng
- $*$: viết gọn của $0...*$
- $0...1$: tùy chọn

Với quan hệ hợp thành, tính nhiều tại cuối hợp thành luôn luôn là 1.



Hình 4.6: Mô tả tính nhiều trong UML

Một Student có duy nhất một FullName và một Address

Một Student có thể tham gia học nhiều Course

Một Course chỉ được học bởi tối thiểu 40 Student hoặc tối đa 80 Student;

Một Course sẽ thuộc về một Subject nhất định

Một Subject có thể có nhiều Course (Course 3 đvht hay 4 đvht)

Một Subject có thể không có môn điều kiện nào (Prerequisite Subject)

Một Subject có thể có tối đa 5 môn học điều kiện

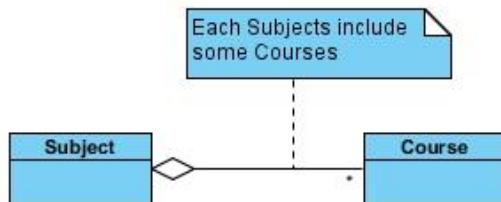
Quan hệ giữa FullName và Address với Student là hợp thành, khi đối tượng sinh viên được tạo ra thì đối tượng FullName và Address cũng đồng thời được tạo ra theo. Tương tự, quan hệ giữa Course và Subject là kết hợp, course là thành phần của Subject. Sự khác nhau giữa hợp thành và kết hợp có thể thấy ở đây là lớp FullName và Address là thành phần của Student nhưng nó không thể tồn tại độc lập mà không có Student. Sự tồn tại của nó phụ thuộc vào sự

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

tồn tại của lớp *Student*. Trong khi *Course* là thành phần của *Subject*, nhưng sự tồn tại của *Subject* không quy định sự tồn tại của *Course*.

Các nhãn, vai trò và chú thích cho quan hệ

Tất cả các quan hệ, ngoại trừ kế thừa, đều có thể sinh ra một nhãn liên kết, nhằm chỉ ra bản chất của liên kết. Ví dụ trong hình 4.7, *Course* là một phần của *Subject*, một *Subject* sẽ bao gồm nhiều *Course* như là thành phần của nó.

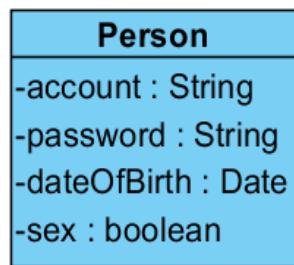


Hình 4.7: Nhãn và chú thích cho liên kết trong UML

Trong các quan hệ, chúng ta có thể biểu diễn các vai trò (role) nhằm mô tả nhãn gần đới tượng thể hiện vai trò đó. Ví dụ, trong Hình 4.4, Thành viên *Member* sống tại *live at* địa chỉ *Address*. Về nguyên tắc, nhãn và vai trò có thể được kết hợp trên cùng một liên kết nhưng nên cân nhắc thay thế để tránh nhầm lẫn lộn xộn. Phần chú thích, một phần văn bản tùy ý được đặt trong một biểu tượng trống như trang giấy, dùng để cung cấp thông tin về gì mà nó cần giải thích thêm.

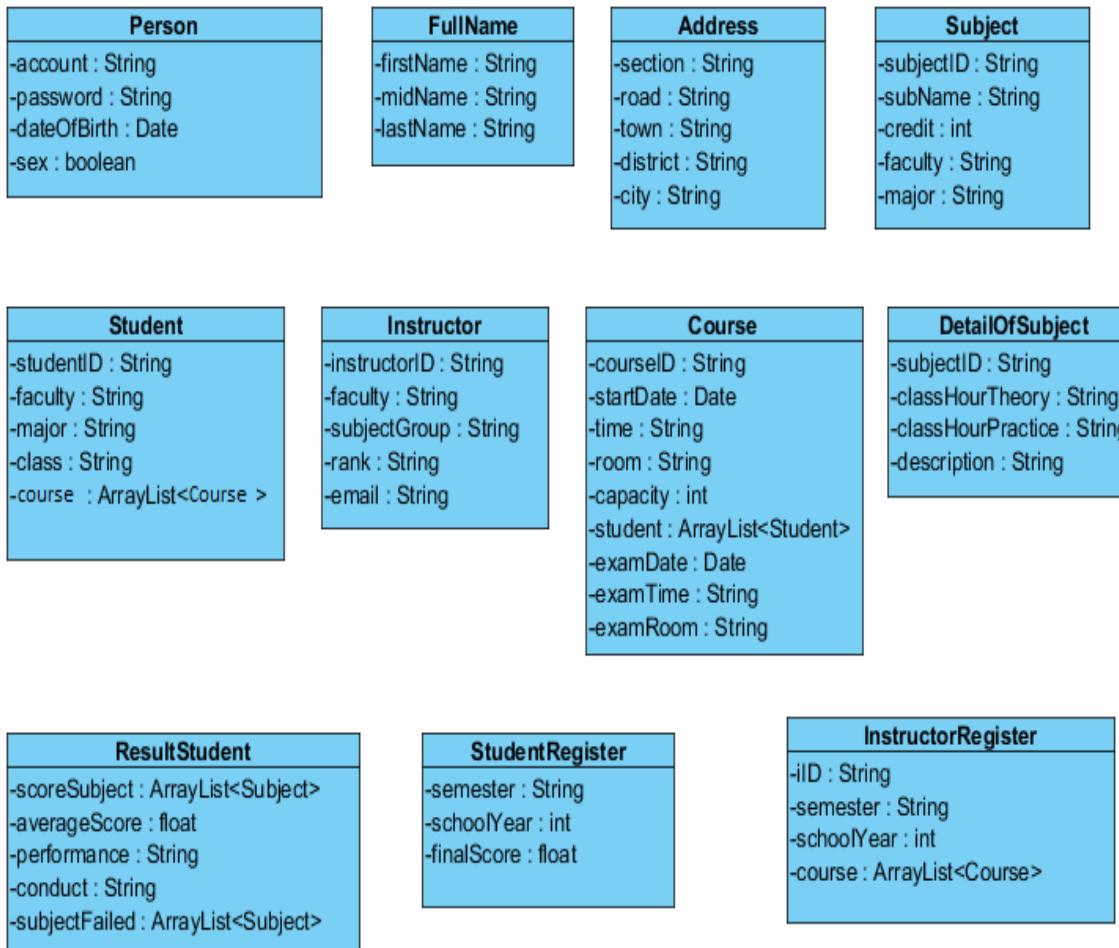
4.3.4 Xác định thuộc tính lớp

Một thuộc tính là một đặc tính hay tính chất của một đối tượng, như là kích cỡ, vị trí, tên, giá, sở thích, giới tính hay bất kì điều gì. Trong UML, mỗi thuộc tính có thể được cho bởi kiểu (type) được xác định bởi một lớp hay kiểu cơ bản. Kiểu được biểu diễn sau dấu “:” và đặt sau tên thuộc tính.



Hình 4.8: Biểu diễn một lớp cơ bản trong UML

Thuộc tính có thể được hiển thị trong biểu đồ lớp bằng cách thêm một ngăn ở dưới tên lớp. Để tiết kiệm không gian, chúng ta có thể viết tài liệu riêng biệt cho chúng thay vì một danh sách thuộc tính hoàn chỉnh với các mô tả. Nếu chúng ta sử dụng công cụ phát triển phần mềm, chúng ta hy vọng có thể phóng to để nhìn các thuộc tính (và mô tả của chúng) hoặc thu nhỏ để chỉ nhìn tên lớp. Hình 4.8 biểu diễn các thuộc tính của lớp *Member* trong Hệ thống quản lý học tập theo tín chỉ với các thuộc tính *Account*, *password*, *dateOfBirth*, *sex* cùng với kiểu của các thuộc tính đó (*String*, *date*, *Boolean*...). Hình 4.9 là một biểu diễn của một số lớp với các thuộc tính của hệ quản lý đăng ký học theo tín chỉ.



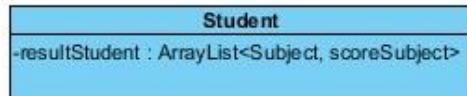
Hình 4.9 Biểu diễn lớp và các thuộc tính

Thuộc tính hay quan hệ

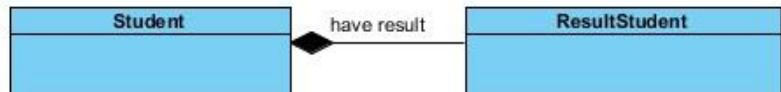
Thông thường chúng ta cần phải đưa ra những lựa chọn khác nhau cho mô hình hóa thông tin. Ví dụ, mô hình kết quả học tập của sinh viên như thế nào được cho là phù hợp nhất với hệ thống. Hình 4.10 thể hiện 3 cách khác nhau:

- Thêm một thuộc tính cho lớp *Student* gọi là *resultStudent* với kiểu mảng chứa danh sách môn học và kết quả môn học sinh viên đăng ký.
- Quan hệ kiểu hợp thành giữa *Student* và *ResultStudent*
- Quan hệ kiểu liên kết giữa *Student* và *ResultStudent*

Lựa chọn 1

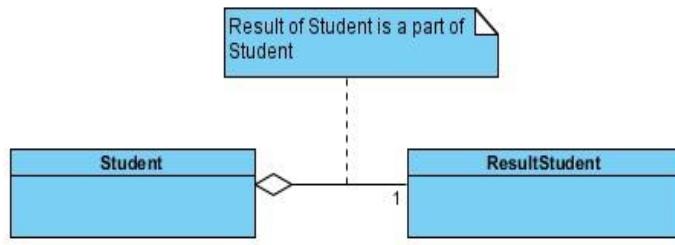


Lựa chọn 2



CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

Lựa chọn 3



Hình 4.10: Lựa chọn giữa thuộc tính hay quan hệ

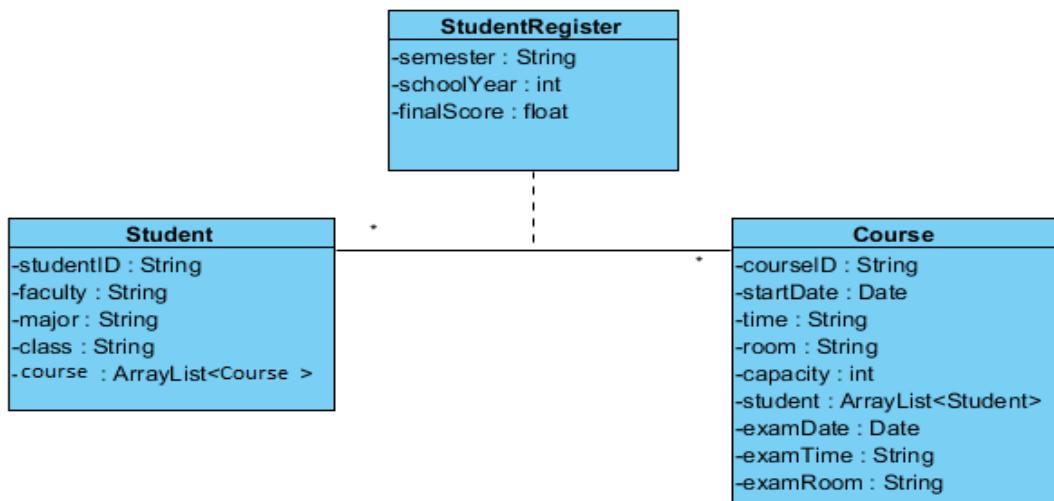
Trong các mô hình này, mô hình nào sẽ được chọn phù hợp với ngữ cảnh nhất.

- **Lựa chọn 1** có thể dẫn đến xây dựng một cơ sở dữ liệu phức tạp cho *Student*.
- **Lựa chọn 2** khá hợp lý khi *ResultStudent* là một phần của *Student* và sự tồn tại của lớp *Student* điều khiển sự tồn tại của *ResultStudent*, *ResultStudent* chỉ tồn tại khi *Student* tồn tại.
- **Lựa chọn 3** *ResultStudent* vẫn là một thành phần của *Student* nhưng đường như không chính xác khi cho rằng *ResultStudent* có thể tồn tại độc lập với *Student*.

Nói chung lựa chọn 2 là tốt nhất trong ngữ cảnh quản lý học tập theo tín chỉ. Một lưu ý khi phân tích là cần phải lựa chọn những trường hợp biểu diễn thỏa mãn với ngữ cảnh hiện tại nhất nhưng không phải bao giờ ta cũng có một câu trả lời chính xác duy nhất. Lời khuyên tốt nhất là không nên lo lắng về mặt ý nghĩa quá nhiều. Thay vì, hãy sử dụng những phán đoán chung, những kinh nghiệm, trực giác, lặp đi lặp lại và mở rộng cho đến khi thực hiện thành công.

Thuộc tính của lớp liên kết

Một lớp được đính kèm trong quan hệ liên kết được gọi là một *lớp liên kết* (association class). Một lớp liên kết không được nối tới bất kỳ lớp nào của quan hệ mà nó nối tới chính bản thân quan hệ. Cũng giống như một lớp bình thường, lớp liên kết có thể có thuộc tính, phương thức và các quan hệ khác. Lớp liên kết được sử dụng để bổ sung thêm thông tin cho quan hệ liên kết. Hình 4.11 chỉ ra rằng lớp *Student* có thể được kết nối với một số bất kỳ đối tượng *Course* và ngược lại.

**Hình 5.11: Lớp liên kết**

Lớp liên kết mô tả các thuộc tính và hành động của thực thể và chỉ tồn tại khi sự liên kết tồn tại: các thuộc tính và các hành động không liên quan đến các đối tượng trong liên kết. Trong ví dụ trên, khi *Student* đăng ký học một *Course*, một liên kết mới được tạo *trong thời gian chạy giữa Student và Course* tương ứng. Khi đó, một số thuộc tính mới này sinh như điểm kết thúc môn học *finalScore*, danh sách sinh viên...

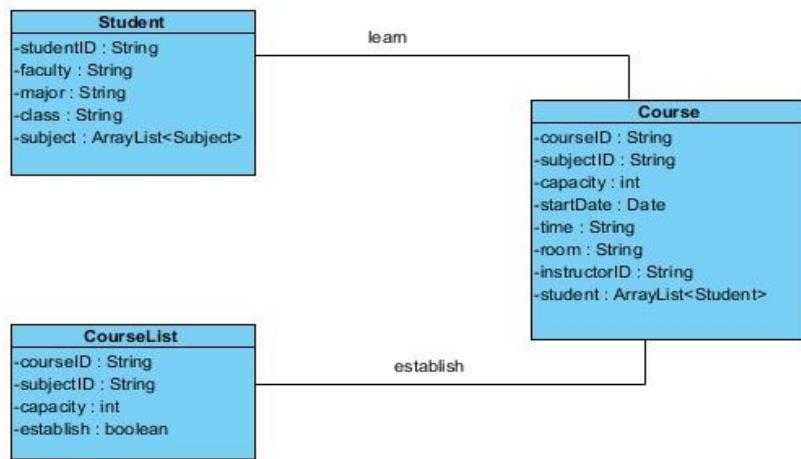
Khi tiến hành pha thiết kế, chúng ta sẽ thay thế các lớp liên kết bởi cái gì cụ thể hơn, bởi vì chúng không được hỗ trợ trực tiếp trong nhiều ngôn ngữ lập trình. Tuy nhiên, các lớp liên kết tỏ ra rất hữu ích trong quá trình phân tích.

Đối tượng hữu hình và đối tượng vô hình

Trong các mô hình hướng đối tượng thường gồm những đối tượng hữu hình và những đối tượng vô hình. Ví dụ, một *Course* trong *CourseList* có thể là một đối tượng vô hình vì nó có thể được tạo ra hoặc không tùy theo số lượng sinh viên đăng ký tương ứng với *Course* đó. Các *Course* trong danh sách *CourseList* là những course có thể được mở, chứ không nhất thiết phải bắt buộc mở. Khi *Course* được mở cho sinh viên, *Course* sẽ trở thành một đối tượng hữu hình. Nói chung, có thể có nhiều đối tượng hữu hình trong một đối tượng vô hình. Có một nhầm lẫn phổ biến là ghép cặp đối tượng hữu hình và vô hình như một đối tượng đơn.

Ví dụ, trong hệ thống quản lý học tập theo tín chỉ, ta sẽ thấy trong suốt quá trình sinh viên đăng ký học, chúng ta có thể lên kế hoạch dự kiến các lớp học cho các môn học đó để mở lớp. Tuy nhiên, danh sách các lớp học khác với một đối tượng lớp học hữu hình được mở và sinh viên tham gia học. Như vậy *Course* trong *CourseList* là đối tượng vô hình, nhưng *Course* mà sinh viên tham gia học là đối tượng hữu hình. Sự khác biệt giữa các đối tượng này dễ dàng thấy được khi bổ sung thêm thuộc tính.

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU



Hình 4.12: Đối tượng hữu hình và đối tượng vô hình

Các thông tin liên quan đến *Course* vô hình trong *CourseList* bao gồm:

- *CourseID*: Mã Course
- *SubjectID*: Mã môn học tương ứng với Course
- *Capacity*: số lượng dự kiến
- *Establish*: biến Boolean cho biết lớp có được tổ chức hay không (có thể có hoặc không)

Các thông tin liên quan đến *Course* hữu hình bao gồm:

- *CourseID*: mã Course
- *SubjectID*: mã môn học tương ứng với Course
- *Capacity*: số lượng thực tế
- *startDate*: ngày bắt đầu học
- *Time*: thời gian học
- *Room*: phòng học
- *InstructorID*: ID giảng viên cho lớp học
- *Student*: mảng danh sách các sinh viên tham gia học. Số phần tử của mảng chính là giá trị của Capacity.

4.4 PHÂN TÍCH ĐỘNG

Mục đích của phân tích động là:

- Nhằm khẳng định rằng biểu đồ lớp của chúng ta đã hoàn thiện và chính xác để có thể cố định nó sớm hơn. Lý do là vì điều này liên quan đến việc thêm, xóa, sửa các lớp, các quan hệ, các thuộc tính và các thao tác trong các lớp.
- Để các nhà phát triển lẫn khách hàng có thể tin được rằng tiến trình mô hình hóa hệ thống của chúng ta đến nay có thể cài đặt thành phần mềm.
- Để kiểm định tính năng của giao diện người dùng sẽ xuất hiện trong hệ thống cuối cùng. Đây là một ý tưởng tốt để phân quyền truy cập hệ thống thành các giao diện tách biệt theo các dòng kịch bản trước khi đi tiên hành thiết kế chi tiết.

Theo Jacobson [7], phần quan trọng nhất của phân tích động là *hiện thực hóa ca sử dụng* (use case realization) nghĩa là làm cho các ca sử dụng trở nên hiện thực hơn để cài đặt sau này bằng thể hiện cách cài đặt hệ thống như tập các đối tượng tương tác với nhau. Hiện thực hóa bao gồm các bước sau :

- Duyệt qua các ca sử dụng của hệ thống, mô phỏng các thông điệp được gửi giữa các đối tượng và ghi lại các kết quả bằng biểu đồ giao tiếp hoặc biểu đồ tuần tự.
- Gán các thao tác cho các lớp tương ứng.
- Thêm các lớp để biểu diễn các lớp biên (các giao diện hệ thống) và các lớp điều khiển (dành cho các tiến trình nghiệp vụ phức tạp hoặc cho việc tạo và truy xuất các đối tượng) khi cần thiết.

4.4.1 Xây dựng biểu đồ giao tiếp

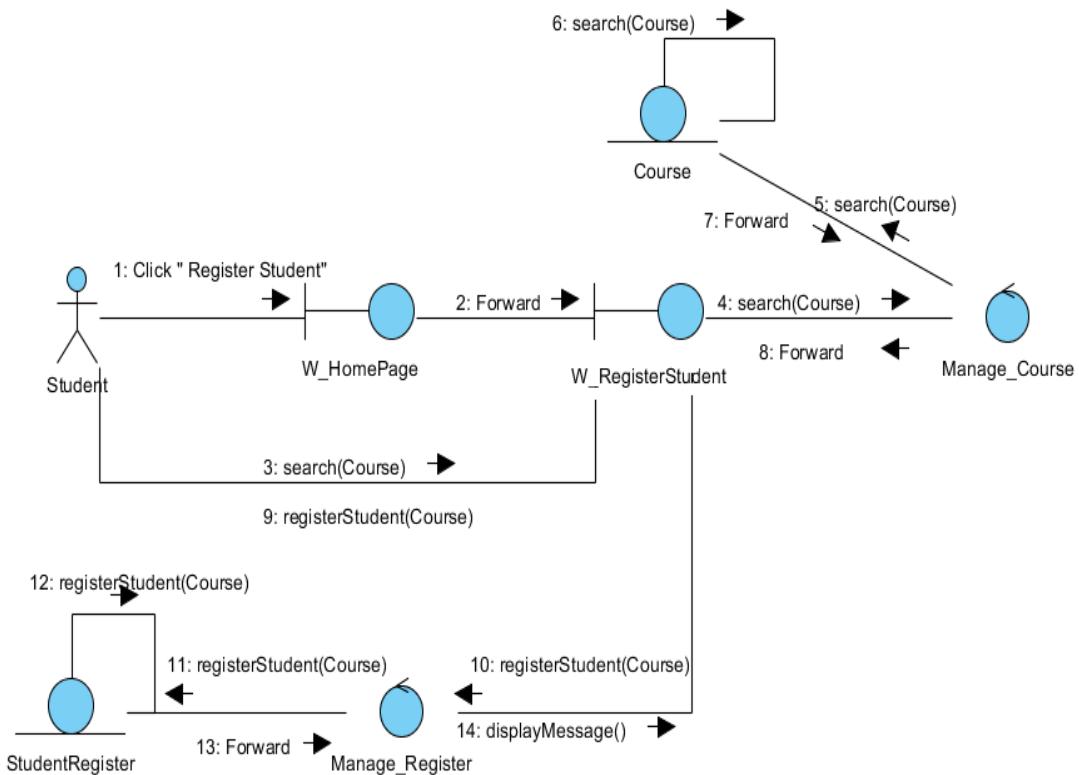
Khi mô phỏng các thông điệp được gửi đi giữa các đối tượng trong pha phân tích, chúng ta cần ghi lại các kết quả tương tác này bằng biểu đồ giao tiếp hay biểu đồ tuần tự trong UML. Mặc dù các thông tin thể hiện trên hai biểu đồ này được xem là tương tự nhau, nhưng biểu đồ giao tiếp được xem là tốt hơn cho hiện thực hóa ca sử dụng và dễ dàng xây dựng hơn. Lý do là biểu đồ giao tiếp chỉ chú trọng vào các đối tượng và các tương tác giữa chúng nên phù hợp trong giai đoạn này hơn là thứ tự các thông điệp được gửi đi như biểu đồ tuần tự. Trong pha xác định yêu cầu, chúng ta có thể cập nhật biểu đồ giao tiếp nhưng là tùy chọn và có biểu diễn đơn giản hơn nhằm mô tả hoạt động nghiệp vụ hiện thời của khách hàng. Trong khi đó, biểu đồ giao tiếp mức phân tích thể hiện đầy đủ hơn và bao gồm các thành phần:

- Các tác nhân tương tác với các đối tượng biên.
- Các đối tượng biên tương tác với các đối tượng bên trong hệ thống.
- Các đối tượng bên trong hệ thống tương tác với các đối tượng biên của hệ thống bên ngoài.
- Các thông điệp *thể hiện như kiểu phương thức* để có thể gán cho các đối tượng tương ứng sau này.

Chúng ta không cần phải chỉ ra bất kì các đối tượng nghiệp vụ nào nằm bên ngoài hệ thống cũng như các tác nhân không tương tác trực tiếp với hệ thống.

Ví dụ: Biểu đồ giao tiếp cho ca sử dụng sinh viên đăng ký học trong Hệ quản lý học theo tín chỉ (Hình 4.13)

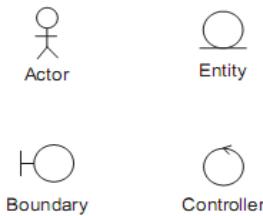
CHƯƠNG 4. PHÂN TÍCH YÊU CẦU



Hình 4.13: Biểu đồ giao tiếp cho Sinh viên đăng ký học

Một số khái niệm, ký hiệu của biểu đồ giao tiếp UML đã được giới thiệu trong Chương 2, phần còn lại trong mục này sẽ giải thích kỹ càng hơn các khái niệm này.

Các đối tượng biên, thực thể và điều khiển



Hình 4.14: Biểu diễn các kiểu lớp

Tác nhân: Là một người hoặc hệ thống đang tồn tại bên ngoài hệ thống của chúng ta.

Đối tượng biên: một đối tượng nằm tại biên hệ thống nghĩa là giữa hệ thống và tác nhân. Với tác nhân là một hệ thống, đối tượng biên cung cấp một đường giao tiếp. Với tác nhân là con người, đối tượng biên là giao diện có nhiệm vụ nhận lệnh, truy vấn và hiển thị phản hồi cũng như kết quả. Mỗi đối tượng biên thường tương ứng với một ca sử dụng hay nhóm ca sử dụng có liên quan nhau. Các đối tượng biên như vậy thường tương ứng với phác họa giao diện (có thể toàn bộ hay cửa sổ con). Vì vậy, các đối tượng biên thường tồn tại trong suốt pha thiết kế sau này.

Đối tượng thực thể: Một đối tượng nằm bên trong hệ thống, mô tả các khái niệm nghiệp vụ như Sinh viên *Student*, Khách hàng *Customer*, Môn học *Subject*... và chứa các thông tin hữu ích về nghiệp vụ cần xử lý. Thông thường các lớp thực thể xuất hiện trong biểu đồ giao tiếp và thường tồn tại trong suốt quá trình thiết kế sau này.

Đối tượng điều khiển: Một đối tượng bên trong hệ thống đóng gói một tiến trình phức tạp. Đối tượng điều khiển thường là một đối tượng dịch vụ để cung cấp các dịch vụ như: điều khiển tất cả hoặc một phần của tiến trình hệ thống, tạo các thực thể mới, triệu hồi các thực thể đang tồn tại. Vì các lớp điều khiển chỉ tiện lợi cho phân tích, chúng ta không hy vọng tất cả các lớp này tồn tại trong quá trình thiết kế ngoại trừ lớp điều khiển được sử dụng để tạo ra những thực thể mới và truy xuất đến các thực thể đang tồn tại. Vì nó là khái niệm rõ ràng nên thường tồn tại trong quá trình thiết kế.

Theo phương pháp luận phát triển RUP (trình bày trong Chương 1), tiến trình phát triển sẽ giữ lại tất cả các lớp điều khiển cho pha thiết kế. Như vậy, nhiệm vụ pha thiết kế là hoàn thiện tất cả các thao tác mà chúng ta đã tìm ra trong quá trình phân tích. RUP không phân biệt giữa mô hình phân tích và mô hình thiết kế, mà đơn giản là bắt đầu với mô hình phân tích và sẽ mở rộng lặp đi lặp lại nhiều lần cho đến khi nó có thể chuyển thành một mô hình thiết kế có thể cài đặt được.

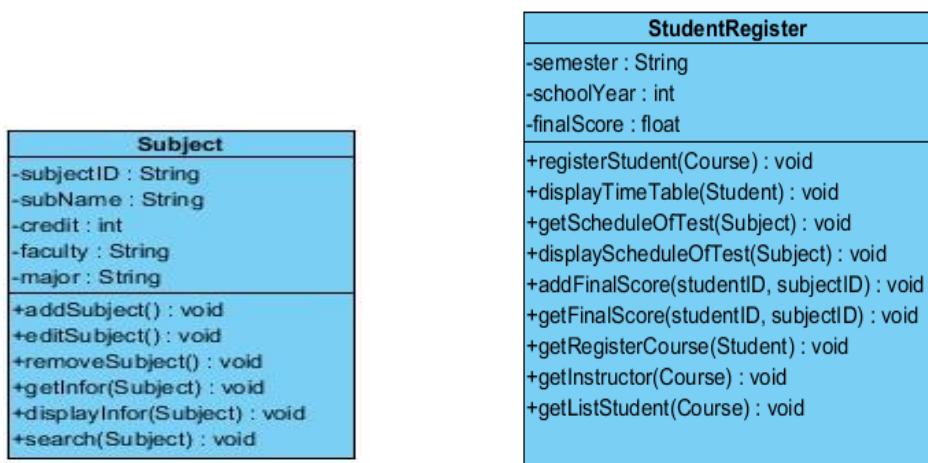
Các thành phần của biểu đồ giao tiếp

- *Tác nhân:* được biểu diễn giống trong biểu đồ ca sử dụng.
- *Đối tượng:* được biểu diễn theo Hình 4.14 và gán tên.
- Một đường thẳng nối giữa các đối tượng chỉ ra thông điệp được gửi đến đâu.
- Thông điệp được thể hiện với số tuần tự để chỉ ra vị trí thông điệp trong giao tiếp, tên thông điệp và danh sách tham số để trong dấu ngoặc.
- Mũi tên chỉ hướng thông điệp được gửi đi.
- Nhãn được sử dụng để định danh cho đối tượng và tham số như *tên:kiểu*, ví dụ *s: Student*.
- Gán giá trị trả về cho một cái tên và được biểu diễn như sau: *n = getNumber();*
- Thông điệp có điều kiện: biểu diễn: *[điều kiện] thông điệp*.
- Lặp được thể hiện bằng ký hiệu * và theo sau là số lần lặp. Ví dụ, *3 (có nghĩa là lặp 3 lần).

4.4.2 Gán phương thức cho các lớp

Mỗi thông điệp trong biểu đồ giao tiếp tương ứng với một phương thức trên một lớp, chính vì vậy chúng ta nên ghi lại các phương thức để có một tập hoàn thiện các hiện thực hóa ca sử dụng. Các phương thức có thể được biểu diễn trong biểu đồ lớp với một ngăn tách biệt phía dưới ngăn thuộc tính:

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU



Hình 4.15: Thêm các phương thức vào lớp

Dạng tổng quát của cách biểu diễn phương thức trong UML, trong đó các tên và kiểu tham số, kiểu trả về là không bắt buộc:

OperationName (paramName1 : paramType1, paramName2 : paramType2) : ReturnType
e

Gán phương thức cho lớp dựa trên trách nhiệm (Responsibility)

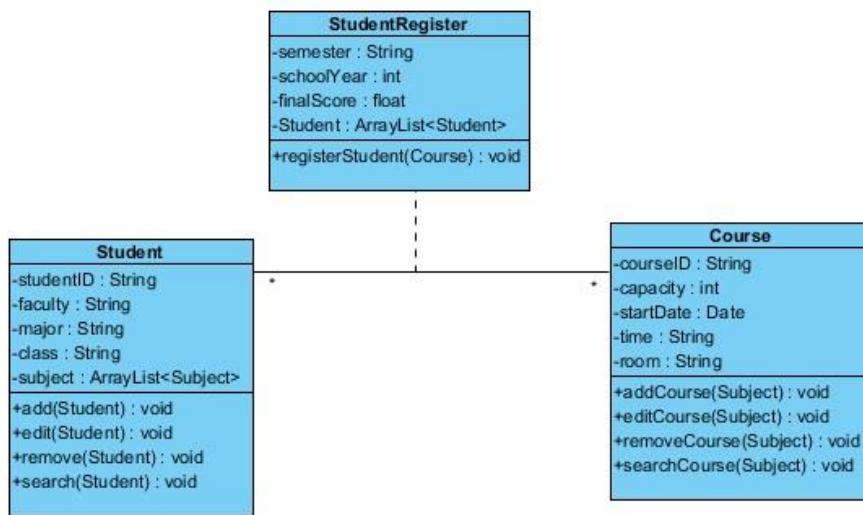
Câu hỏi đặt ra là một phương thức nào đó nên được gán cho lớp nào? Bất cứ khi nào thêm chi tiết vào đối tượng, ta cần xem xét đến trách nhiệm của đối tượng đó với hệ thống. Điều này giúp ta tìm ra và gán một cách đúng đắn các phương thức và thuộc tính cho các lớp. Mặc dù nhiều thuộc tính được phát hiện trong giai đoạn phân tích tĩnh, một số thuộc tính có thể được phát hiện thêm trong giai đoạn hiện thực hóa ca sử dụng. Khi phát hiện thông tin hoặc hành động nào đó cần cho hệ thống hãy tự hỏi rằng “đối tượng nào chịu trách nhiệm thực hiện hành động này?” *Thông tin cho ta các thuộc tính, hành động cho ta các thao tác.*

Trong việc gán phương thức, nguyên lý *kết hợp mạnh* (strong cohesion) cho rằng một đối tượng không nên chịu trách nhiệm nhiều hơn một thao tác trong một luồng công việc.

Ví dụ, nếu phải chịu trách nhiệm lấy các thông tin chi tiết từ lớp *Student* và xử lý nó, chúng ta nên chọn hai đối tượng để thực hiện hai công việc này, một là đối tượng biên và một đối tượng điều khiển. Như vậy, hai đối tượng này phải kết hợp chặt chẽ với nhau để tạo ra được một luồng xử lý công việc.

Thuật ngữ *kết dính* (coupling) nhằm chỉ mức độ phụ thuộc giữa các lớp với nhau và hai lớp phụ thuộc mạnh vào nhau được gọi là có *kết dính chặt* (tightly/highly coupled), ngược lại gọi là *kết dính lỏng lẻo* (loosely coupled). Nguyên lý *kết dính lỏng lẻo* (loosely coupled) cho rằng nên xem các đối tượng như là một khách hàng (yêu cầu các câu hỏi và đưa ra câu lệnh) hoặc nhà cung cấp (trả lời các câu hỏi và tiến hành các dịch vụ) hơn là một đối tượng cộng tác cả hai chiều. Khi đó, nhà cung cấp gọi là kết dính lỏng lẻo với các khách hàng của nó. Cơ sở cho nguyên lý này là sự cộng tác hai chiều ở đây có thể làm cho hệ thống trở nên phức tạp và khó bảo trì hơn sau này.

Ví dụ: Hệ thống Quản lý học tập theo tín chỉ

**Hình 4.16: Trách nhiệm phương thức**

Phương thức `registerStudent(Course)` thuộc lớp `RegisterStudent` là lớp liên kết giữa `Student` và `Course`. Mỗi sinh viên sẽ đăng ký `Course` thuộc một `Subject` nào đó, điều kiện đăng ký khóa học dựa trên các ràng buộc về môn điều kiện và có thể số trình còn nợ của sinh viên. Phương thức `registerStudent(Course)` thuộc lớp `RegisterStudent` có thể đảm bảo tính kết dính lồng léo trong thiết kế quan hệ giữa các lớp. Các phương thức thuộc lớp `Student`, `Course` đều gọi đến các thuộc tính đã được khai báo trước đó trong các lớp tương ứng nên không ảnh hưởng đến lớp `RegisterStudent`.

Tóm lại, dựa trên biểu đồ giao tiếp (Hình 4.13), chúng ta có thể gán phương thức cho các lớp tương ứng: `RegisterStudent`, `ManageStudent`, `W_RegisterStudent`, `ManageCourse`, `Course`. Chú ý rằng, mặc dù các phương thức có cùng tên (ta cũng có thể đặt tên khác tùy ý) gán cho các lớp khác nhau nhưng ý nghĩa các phương thức này là hoàn toàn khác nhau nên việc cài đặt sau này cũng khác nhau. Ví dụ phương thức `registerStudent(course)` thuộc lớp biên `W_RegisterStudent` nghĩa là gửi yêu cầu đăng ký cho lớp `ManageRegister`, `registerStudent(course)` thuộc lớp `ManageRegister` tiếp tục gửi yêu cầu đăng ký cho lớp `RegisterStudent`, `registerStudent(course)` trong lớp `RegisterStudent` sẽ thực thi việc lưu đăng ký vào cơ sở dữ liệu.

Ví dụ Gán phương thức cho các lớp dựa trên Biểu đồ trạng thái 4.13

RegisterStudent

- `registerStudent(course)`
- `forward()`

ManageRegister

- `displayMessage()`
- `registerStudent(course)`

W_RegisterStudent

- `registerStudent(course)`
- `search(course)`

ManageCourse

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

- forward()
- search(course)

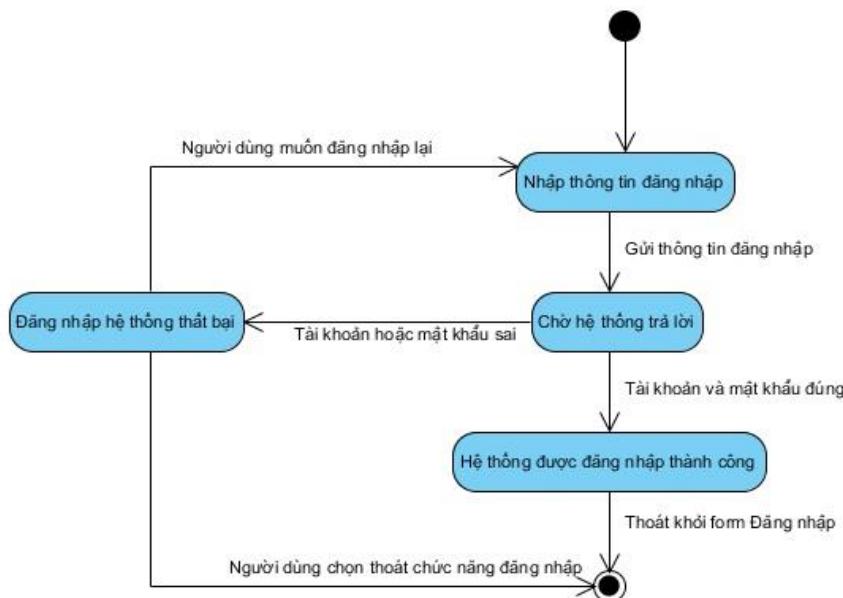
Course

- search(course)
- forward()

Mô hình hóa trạng thái

Đôi khi, một thực thể có một vòng đời phức tạp nên cần biểu diễn bằng biểu đồ máy trạng thái. Biểu đồ này hỗ trợ cho chúng ta khi xác định các phương thức với các tham số cho hành vi của chính đối tượng đó.

Ví dụ, hình 4.17 chỉ ra một mô hình vòng đời trạng thái của hệ thống Quản lý học tập theo tín chỉ với ca sử dụng Đăng nhập. Trong biểu đồ này, một hộp với các góc tròn chỉ ra một trạng thái, với một nhãn gắn với tên của nó. Một mũi tên chỉ ra một sự chuyển tiếp tới trạng thái khác – nhãn trên mũi tên chỉ ra nguyên nhân gây nên sự chuyển tiếp. Một vòng tròn đen với một mũi tên từ ngoài các điểm của nó vào một trạng thái ban đầu – một trạng thái mà trong đó một đối tượng có thể được sinh ra. Một mũi tên chỉ đến một đường tròn đen chỉ ra nguồn là một trạng thái cuối cùng – một trạng thái mà một đối tượng có thể kết thúc vòng đời.



Hình 4.17: Biểu đồ trạng thái Đăng nhập hệ thống

4.5 CASE STUDY: HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Phân tích là công việc tiếp theo của tiến trình phát triển phần mềm sau khi chúng ta hoàn thành quá trình Xác định yêu cầu. Để tiến hành phân tích hệ thống chúng ta cần nắm rõ *Đầu vào, Đầu ra* của quá trình này. Trong đó, đầu vào của quá trình Phân tích là sản phẩm của pha Xác định yêu cầu bao gồm:

- Danh sách các tác nhân
- Danh sách các ca sử dụng
- Các kịch bản

- Biểu đồ ca sử dụng
- Các phác họa giao diện

Đầu ra của pha Phân tích là biểu đồ lớp, biểu đồ giao tiếp và biểu đồ trạng thái. Quá trình phân tích bao gồm hai giai đoạn: phân tích tĩnh và phân tích động. Trong phân tích tĩnh, chúng ta tiến hành thực hiện những công việc sau:

1. Xác định các lớp.
2. Xét mối quan hệ giữa các lớp.
3. Tìm các thuộc tính của các lớp.
4. Đưa ra biểu đồ lớp.

Đối với phân tích động là xây dựng biểu đồ giao tiếp và gán phương thức cho các lớp.

Trong phần này, chúng ta sẽ trình bày một số bước của pha phân tích yêu cầu Hệ thống đăng ký học theo tín chỉ. Bạn đọc có thể xem đầy đủ hơn trong phần Phụ lục B.

4.5.1 Phân tích tĩnh

Xác định lớp

Để xác định các lớp thực thể ta dùng kỹ thuật trích danh từ trong các ca sử dụng và kịch bản. Các danh từ thu được từ các kịch bản là:

Hệ thống học tập tín chỉ, Khoa, Mã Khoa, Tên Khoa, Trưởng Khoa, Chuyên Ngành, Mã chuyên ngành, Tên Chuyên ngành, Trưởng bộ môn, Khoa phụ trách, Môn Học, Mã môn, Tên môn, Số tín chỉ, Số tiết Lý thuyết, Số tiết Thực hành, Số tiết bài tập, Môn điều kiện, Sinh viên, Mã SV, Họ tên, Ngày sinh, Giới tính, Khóa, Lớp, Giảng viên, Mã GV, Họ tên, Ngày sinh, Giới tính, Bộ Môn, Học vị, Mã môn, Tên môn, Mã lớp, Ngày bắt đầu, Buổi học, Giờ học, Phòng học, Sĩ Số, Số đăng ký, thời khóa biểu, lịch thi, kết quả học tập.

Loại bỏ các danh từ nằm ngoài phạm vi mục đích của hệ thống và các danh từ hoặc cụm từ trùng lắp và các danh từ làm thuộc tính của lớp như:

Tên, mã, ngày sinh, địa chỉ, giới tính... là các thuộc tính của các lớp Người, Sinh viên, Giáo viên, Người quản lý.

Ngành, khoa là thuộc tính của Sinh viên, Giáo viên.

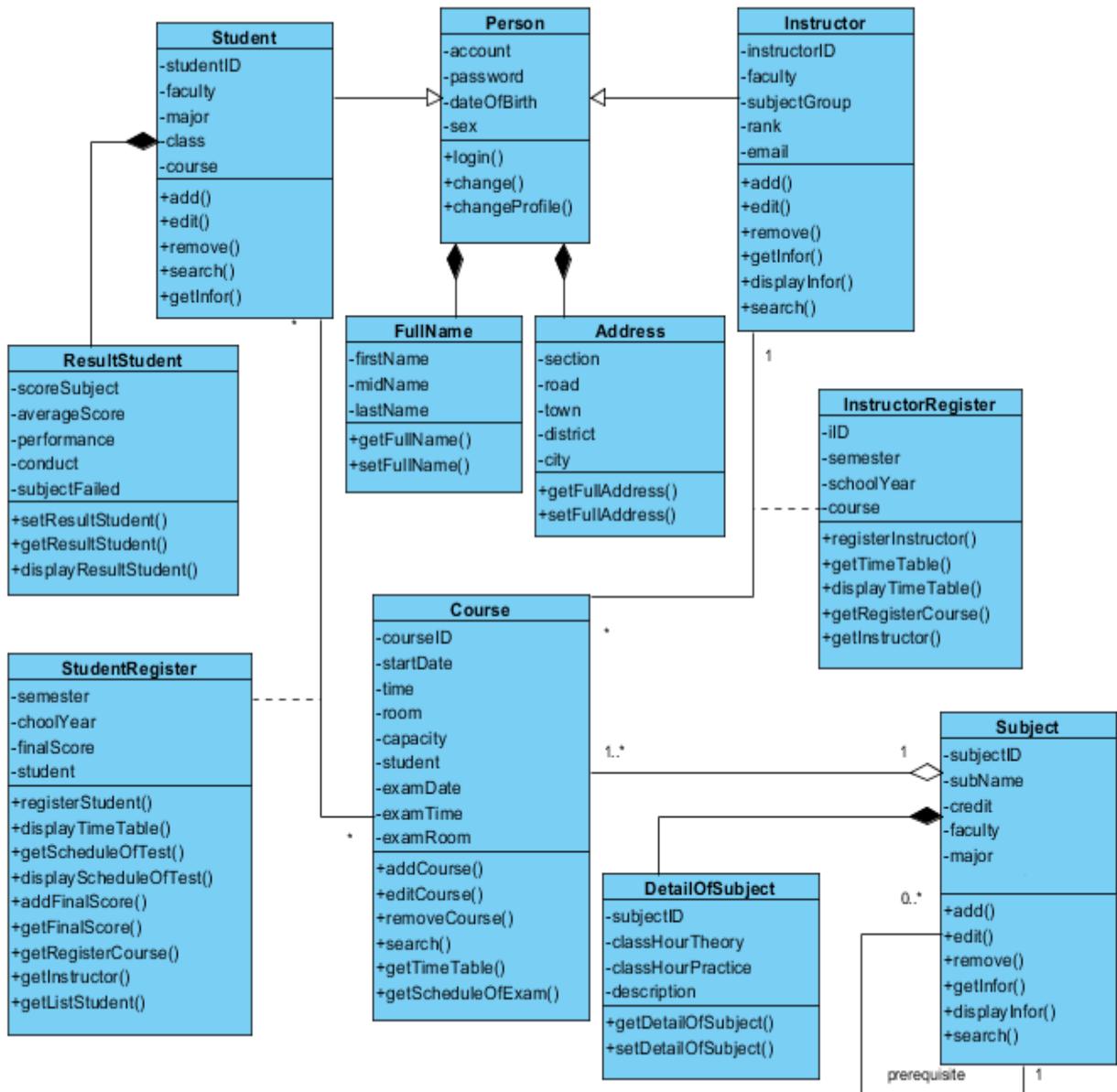
Tên môn học, số tiết, học kỳ, số tín chỉ là thuộc tính của lớp Môn học.

Vậy các danh từ sau có thể là ứng viên cho các lớp thực thể: *People* (Người), *Name* (Họ tên), *Address* (Địa chỉ), *Student* (Sinh viên), *Instructor* (Giảng viên), *Subject* (Môn học), *StudentRegister* (Sinh viên Đăng ký), *InstructorRegister* (Giảng viên đăng ký), *Course* (Lớp học phần), *ResultStudent* (kết quả học tập).

Quan hệ giữa các lớp

Bảng quan hệ giữa các lớp được biểu diễn trong Hình 4.18

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU



Hình 4.18: Quan hệ giữa các lớp

Xác định thuộc tính và gán phương thức cho các lớp

Để dễ theo dõi, chúng ta có thể liệt kê các thuộc tính và phương thức cho các lớp thực thể dưới dạng bảng sau đây:

	Định nghĩa	Chứa các thuộc tính cơ bản của các đối tượng là Người trong hệ thống
	Thuộc tính	<ul style="list-style-type: none"> ❖ account: tên tài khoản đăng nhập vào hệ thống. ❖ pasword: mật khẩu đăng nhập hệ thống ❖ dateOfBirth: ngày tháng năm sinh ❖ sex: giới tính: Nam = 1 ; Nữ = 0
	Phương thức	<ul style="list-style-type: none"> ❖ login(account,password): mỗi người sử dụng hệ thống đều phải đăng nhập tài khoản và mật khẩu riêng của mình. Phương thức

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

<pre> Person -account : String -password : String -dateOfBirth : Date -sex : boolean +login(account, password) : boolean +change(password) : void +changeProfile(account) : void </pre>	<p>này trả về giá trị True nếu đăng nhập thành công, False nếu đăng nhập không thành công.</p> <ul style="list-style-type: none"> ❖ change(password): người dùng sau khi đăng nhập thành công có thể thực hiện đổi mật khẩu. ❖ changeProfile(account): người dùng sau khi đăng nhập thành công có thể thay đổi thông tin cá nhân của mình
<pre> FullName -firstName : String -midName : String -lastName : String +getFullName() : void +setFullName() : void </pre>	<p>Định nghĩa Có quan hệ kiểu hợp thành (composition) với lớp Người. Việc tách thành lớp Họ tên phục vụ cho việc quản lý và tìm kiếm dễ dàng hơn.</p> <p>Thuộc tính</p> <ul style="list-style-type: none"> ❖ firstName, midName, lastName: ba thuộc tính Họ, Đệm, Tên tương ứng với từng trường trong họ tên đầy đủ <p>Phương thức</p> <ul style="list-style-type: none"> ❖ getFullName(): lấy ra Họ tên đầy đủ ❖ setFullName(): gán các trường thành Họ tên đầy đủ
<pre> Address -section : String -road : String -town : String -district : String -city : String +getFullAddress() : void +setFullAddress() : void </pre>	<p>Định nghĩa Có quan hệ kiểu hợp thành (composition) với lớp Người, , việc tách thành lớp Địa chỉ phục vụ cho việc quản lý và tìm kiếm dễ dàng hơn</p> <p>Thuộc tính</p> <ul style="list-style-type: none"> ❖ section: số nhà ❖ road: đường/phố ❖ town: phường ❖ district: quận ❖ city: thành phố <p>Phương thức</p> <ul style="list-style-type: none"> ❖ getFullAddress(): lấy ra địa chỉ đầy đủ ❖ setFullAddress(): gán các trường thành địa chỉ đầy đủ
<pre> Student -studentID : String -faculty : String -major : String -class : String -course : ArrayList<Course> +add(Student) : void +remove(Student) : void +edit(Student) : void +getInfor(Student) : void +displayInfor(Student) : void +search(Student) : void </pre>	<p>Định nghĩa Lớp Sinh Viên sẽ kế thừa từ lớp Người và mang đầy đủ thuộc tính của lớp Người</p> <p>Thuộc tính</p> <ul style="list-style-type: none"> ❖ studentID: mã sinh viên ❖ faculty: khoa sinh viên theo học ❖ major: chuyên ngành học tập của sinh viên ❖ class: mỗi sinh viên sau khi nhập học sẽ được xếp vào một lớp để quản lý <p>Phương thức</p> <ul style="list-style-type: none"> ❖ add(Student): thêm sinh viên vào trong cơ sở dữ liệu ❖ edit(Student): sửa thông tin sinh viên trong cơ sở dữ liệu ❖ remove(Student): xóa sinh viên khỏi cơ sở dữ liệu ❖ getInfor(Student): lấy thông tin của sinh viên trong cơ sở dữ liệu ❖ displayInfor(Student): hiển thị thông tin sinh viên trên giao diện. ❖ search(Student): tìm kiếm sinh viên

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

Instructor -instructorID : String -faculty : String -subjectGroup : String -rank : String -email : String +add(Instructor) : void +edit(Instructor) : void +remove(Instructor) : void +getInfor(Instructor) : void +displayInfor(Instructor) : void +search(Instructor) : void	Định nghĩa Lớp Giảng Viên kế thừa từ lớp Người và mang đầy đủ thuộc tính của lớp Người
	Thuộc tính <ul style="list-style-type: none"> ❖ iID: mã Giảng Viên ❖ faculty: giảng viên thuộc khoa nào ❖ subjectGroup: Bộ môn ❖ rank: học vị của giảng viên ❖ email: địa chỉ email giảng viên
Course -courseID : String -startDate : Date -time : String -room : String -capacity : int -student : ArrayList<Student> -examDate : Date -examTime : String -examRoom : String +addCourse(Subject) : void +editCourse(Subject) : void +removeCourse(Subject) : void +search(Course) : void +getTimeTable(Course) : void +getScheduleOfExam(Course) : void	Định nghĩa Mỗi sinh viên khi đăng ký học một môn sẽ tương ứng với một Lớp giảng . Tương tự mỗi Giảng Viên cũng có thể đăng ký dạy theo các Lớp giảng phù hợp tương ứng với môn giảng viên đăng ký dạy.
	Thuộc tính <ul style="list-style-type: none"> ❖ courseID: mã Lớp giảng tương ứng ❖ startDate: ngày bắt đầu học ❖ time: thời gian học ❖ room: phòng học ❖ capacity: sĩ số lớp học ❖ Student: mảng danh sách các sinh viên tham gia lớp giảng ❖ examDate: ngày thi kết thúc môn học ❖ examTime: giờ thi kết thúc môn học ❖ examRoom: phòng thi kết thúc môn học
	Phương thức <ul style="list-style-type: none"> ❖ addCourse(Subject): thêm một lớp giảng mới tương ứng với một Môn học. ❖ editCourse(Subject): sửa thông tin một lớp giảng ❖ removeCourse(Subject): xóa một lớp giảng trong cơ sở dữ liệu ❖ getInfor(Course): Lấy thông tin chi tiết của toàn bộ lớp giảng bao gồm môn học, ngày học, giờ học, phòng học ❖ search(Course): tìm kiếm thông tin lớp giảng ❖ getTimeTable(Course): lấy thông tin về thời gian và phòng học của lớp học ❖ getScheduleOfExam(Course): lấy thông tin về thời gian và phòng thi cuối kỳ
	Định nghĩa Sinh viên sẽ phải theo học các môn học theo thứ tự trong mỗi học kỳ để tích lũy tín chỉ. Thông tin chi tiết về các môn được thể hiện

CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

Subject <pre>-subjectID : String -subName : String -credit : int -faculty : String -major : String +addSubject() : void +editSubject() : void +removeSubject() : void +getInfor(Subject) : void +displayInfor(Subject) : void +search(Subject) : void</pre>		qua các thuộc tính và phương thức của lớp Môn Học
	Thuộc tính	❖ subjectID: mã môn học ❖ subName: tên môn học ❖ credit: số tín chỉ của môn học ❖ faculty: môn học thuộc khoa nào ❖ major: môn học thuộc chuyên ngành nào
	Phương thức	❖ add(Subject): thêm môn học vào cơ sở dữ liệu ❖ edit(Subject): sửa lại các thông tin về môn học ❖ remove(Subject): xóa môn học khỏi cơ sở dữ liệu ❖ getInfor(Subject): lấy thông tin chi tiết về môn học. ❖ displayInfor(Subject): hiển thị thông tin môn học trên giao diện ❖ search(Subject): tìm kiếm thông tin Môn học
DetailOfSubject <pre>-subjectID : String -classHourTheory : String -classHourPractice : String -description : String +getDetailSubject() : void +setDetailSubject() : void</pre>	Định nghĩa	Vì thuộc tính của Môn học nhiều, trong thực tế không phải lúc nào cũng cần truy cập đến, các thuộc tính như hình thức thi, số tiết lý thuyết, bài tập... sinh viên chỉ quan tâm khi đã đăng ký học khóa học môn đó trong học kỳ, nên tách thêm lớp ChiTietMonHoc
	Thuộc tính	❖ subjectID: mã môn học ❖ classHourTheory: số tiết lý thuyết ❖ classHourPractice: số tiết thực hành
	Phương thức	❖ getDetailSubject(): xem thông tin Chi tiết về Môn học ❖ setDetailOfSubject(): sửa thông tin Chi tiết về Môn học
StudentRegister <pre>-semester : String -schoolYear : int -finalScore : float +registerStudent(Course) : void +displayTimeTable(Student) : void +getScheduleOfTest(Subject) : void +displayScheduleOfTest(Subject) : void +addFinalScore(studentID, subjectID) : void +getFinalScore(studentID, subjectID) : void +getRegisterCourse(Student) : void +getInstructor(Course) : void +getListStudent(Course) : void</pre>	Định nghĩa	Là lớp liên kết giữa lớp Sinh Viên và Khóa học, chứa các phương thức gọi tới thuộc tính của cả hai lớp Sinh Viên và Khóa học.
	Thuộc tính	❖ semester: học kỳ ❖ schoolYear: năm học ❖ finalScore: điểm tổng kết cuối cùng của môn học sinh viên đăng ký
	Phương thức	❖ registerStudent(Course): đăng ký học khóa học của một môn nào đó. Tên sinh viên sẽ được thêm vào danh sách sinh viên của khóa học đó và cập nhật vào cơ sở dữ liệu ❖ displayTimeTable(Student): hiển thị thời khóa biểu học các môn của sinh viên trên giao diện ❖ getScheduleOfTest(Subject): lấy lịch thi của các môn học mà sinh viên đăng ký học trong học kỳ

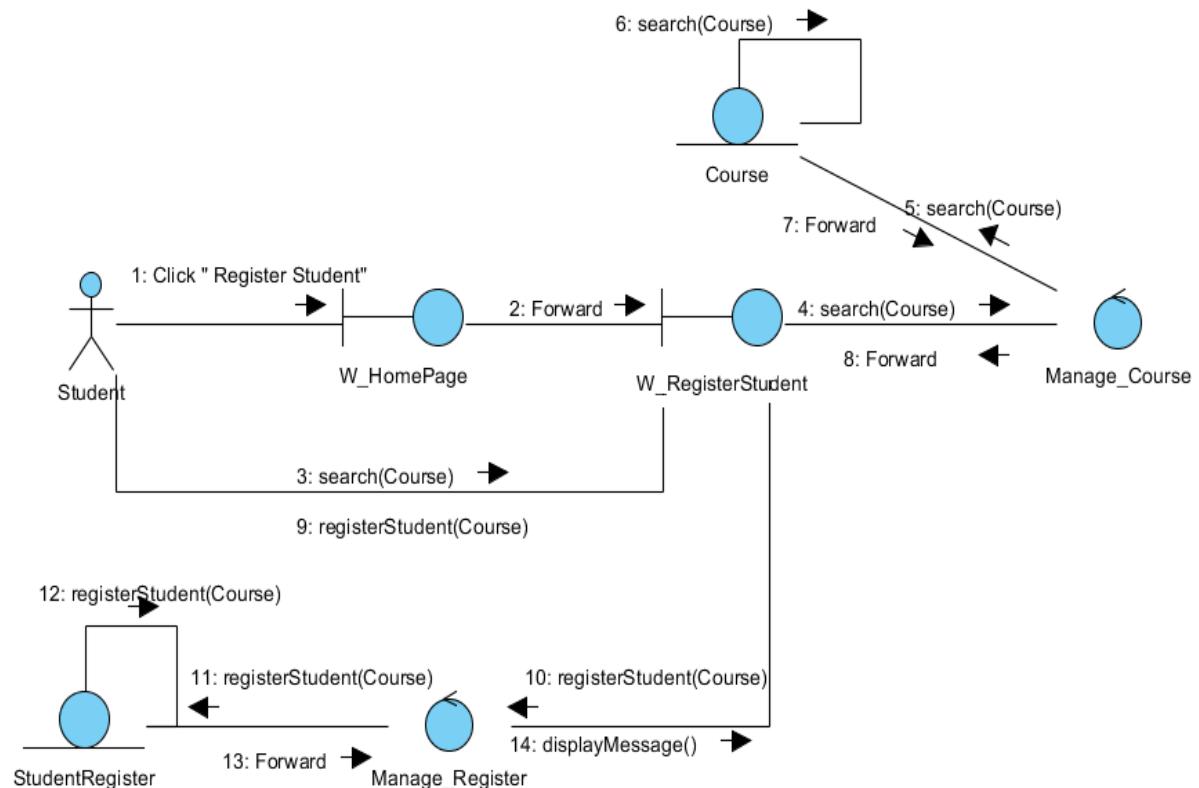
CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

		<ul style="list-style-type: none"> ❖ displayScheduleOfTest(Subject): hiển thị lịch thi học kỳ trên giao diện ❖ addFinalScore(sID,subjected): thêm điểm tổng cuối kỳ của môn học Sinh viên đăng ký ❖ getFinalSocre(sID,subjectID): lấy điểm tổng kết cuối cùng của môn học tương ứng với lớp học Sinh viên đăng ký ❖ getRegisterCourse(Student): lấy những khóa học mà sinh viên đã từng đăng ký học ❖ getListStudent(Course): lấy danh sách những sinh viên trong một lớp học. 		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="background-color: #ADD8E6;">InstructorRegister</th> </tr> <tr> <td> -<i>iID</i> : String -<i>semester</i> : String -<i>schoolYear</i> : int -<i>course</i> : ArrayList<Course> +<i>registerInstructor(Course)</i> : void +<i>getTimeTable(Instructor)</i> : void +<i>displayTimeTable(Instructor)</i> : void +<i>getRegisterCourse(Instructor)</i> : void +<i>getInstructor(Course)</i> : void </td> </tr> </table>	InstructorRegister	- <i>iID</i> : String - <i>semester</i> : String - <i>schoolYear</i> : int - <i>course</i> : ArrayList<Course> + <i>registerInstructor(Course)</i> : void + <i>getTimeTable(Instructor)</i> : void + <i>displayTimeTable(Instructor)</i> : void + <i>getRegisterCourse(Instructor)</i> : void + <i>getInstructor(Course)</i> : void	Định nghĩa Là lớp liên kết giữa lớp Giảng Viên và Khóa học, chứa các phương thức gọi tới thuộc tính của cả hai lớp Giảng Viên và Khóa học	Thuộc tính <ul style="list-style-type: none"> ❖ iID: mã giảng viên ❖ semester: học kỳ ❖ schoolYear: năm học ❖ Course: danh sách các Lớp giảng mà giảng viên đăng ký dạy
InstructorRegister				
- <i>iID</i> : String - <i>semester</i> : String - <i>schoolYear</i> : int - <i>course</i> : ArrayList<Course> + <i>registerInstructor(Course)</i> : void + <i>getTimeTable(Instructor)</i> : void + <i>displayTimeTable(Instructor)</i> : void + <i>getRegisterCourse(Instructor)</i> : void + <i>getInstructor(Course)</i> : void				
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="background-color: #ADD8E6;">ResultStudent</th> </tr> <tr> <td> -<i>scoreSubject</i> : ArrayList<Subject> -<i>averageScore</i> : float -<i>performance</i> : String -<i>conduct</i> : String -<i>subjectFailed</i> : ArrayList<Subject> +<i>setResultStudent(studentID)</i> : void +<i>getResultStudent(studentID)</i> : void +<i>displayResultStudent(studentID)</i> : void </td> </tr> </table>	ResultStudent	- <i>scoreSubject</i> : ArrayList<Subject> - <i>averageScore</i> : float - <i>performance</i> : String - <i>conduct</i> : String - <i>subjectFailed</i> : ArrayList<Subject> + <i>setResultStudent(studentID)</i> : void + <i>getResultStudent(studentID)</i> : void + <i>displayResultStudent(studentID)</i> : void	Định nghĩa Vì lớp sinh viên có nhiều thuộc tính, trong khi kết quả học tập là những thuộc tính không phải lúc nào cũng cần truy cập đến trong hệ thống quản lý họa tập theo tín chỉ, do vậy tách ra thành một lớp riêng	Phương thức <ul style="list-style-type: none"> ❖ setInstructor(Course): giảng viên đăng ký dạy Lớp giảng của một môn nào đó. Phương thức này sẽ gán Mã giảng viên vào trong cơ sở dữ liệu tương ứng với trường Giảng viên của mỗi bản ghi về Lớp giảng ❖ getTimeTable(Instructor): lấy thông tin về lịch dạy của các Lớp giảng mà Giảng viên đã đăng ký ❖ displayTimeTable(Instructor): hiển thị lịch dạy của giảng viên trên giao diện ❖ getRegisterCourse(Instructor): lấy danh sách những lớp học mà giảng viên đã đăng ký dạy ❖ getInstructor(Course): lấy thông tin về giảng viên của một lớp học
ResultStudent				
- <i>scoreSubject</i> : ArrayList<Subject> - <i>averageScore</i> : float - <i>performance</i> : String - <i>conduct</i> : String - <i>subjectFailed</i> : ArrayList<Subject> + <i>setResultStudent(studentID)</i> : void + <i>getResultStudent(studentID)</i> : void + <i>displayResultStudent(studentID)</i> : void				

	Phương thức	<ul style="list-style-type: none"> ❖ getResultStudent(sID): thống kê kết quả học tập các môn của sinh viên ❖ displayResultStudent(sID): hiển thị bảng điểm học tập của sinh viên trên giao diện
--	------------------------	---

4.5.2 Phân tích động

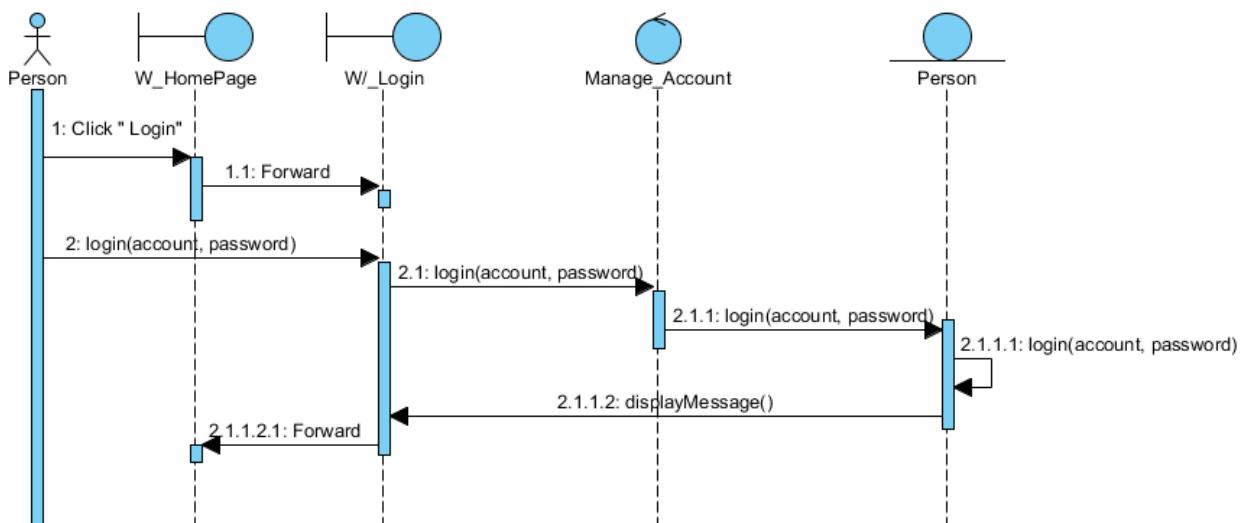
Biểu đồ giao tiếp



Hình 4.19: Biểu đồ giao tiếp Đăng ký học

Biểu đồ tuần tự

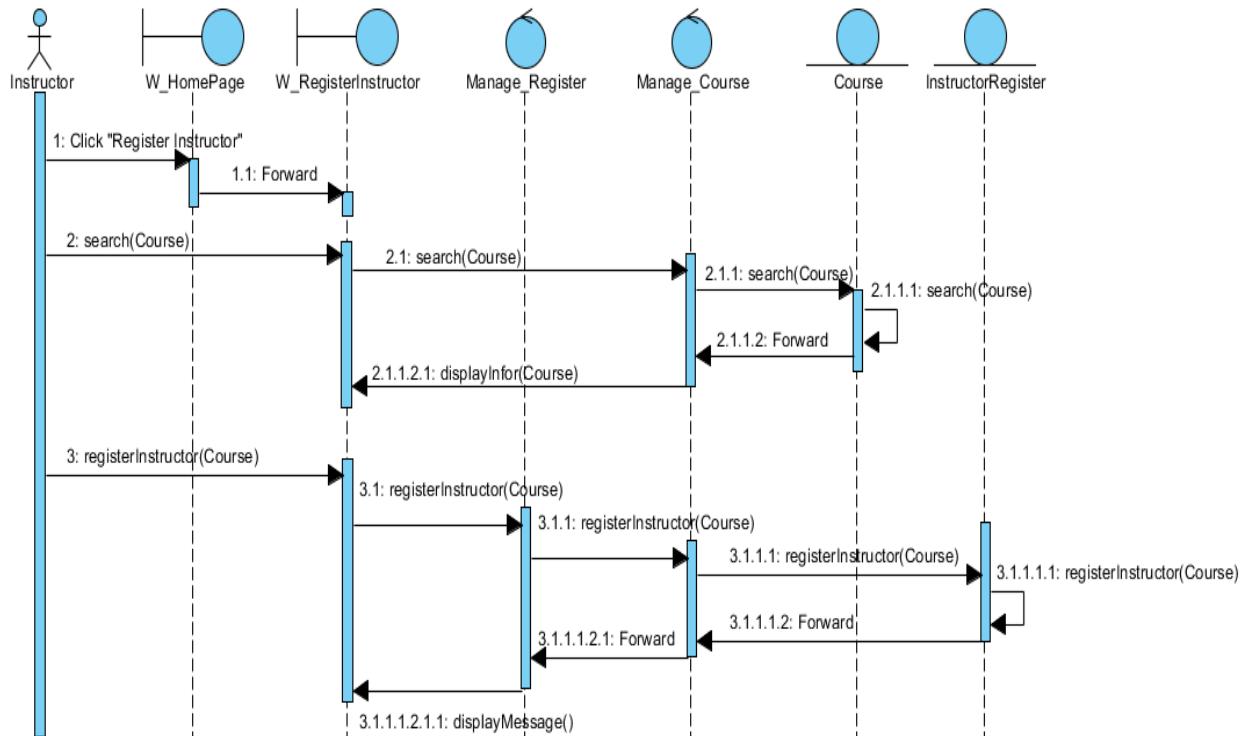
Đăng nhập



Hình 4.20: Biểu đồ tuần tự Đăng nhập

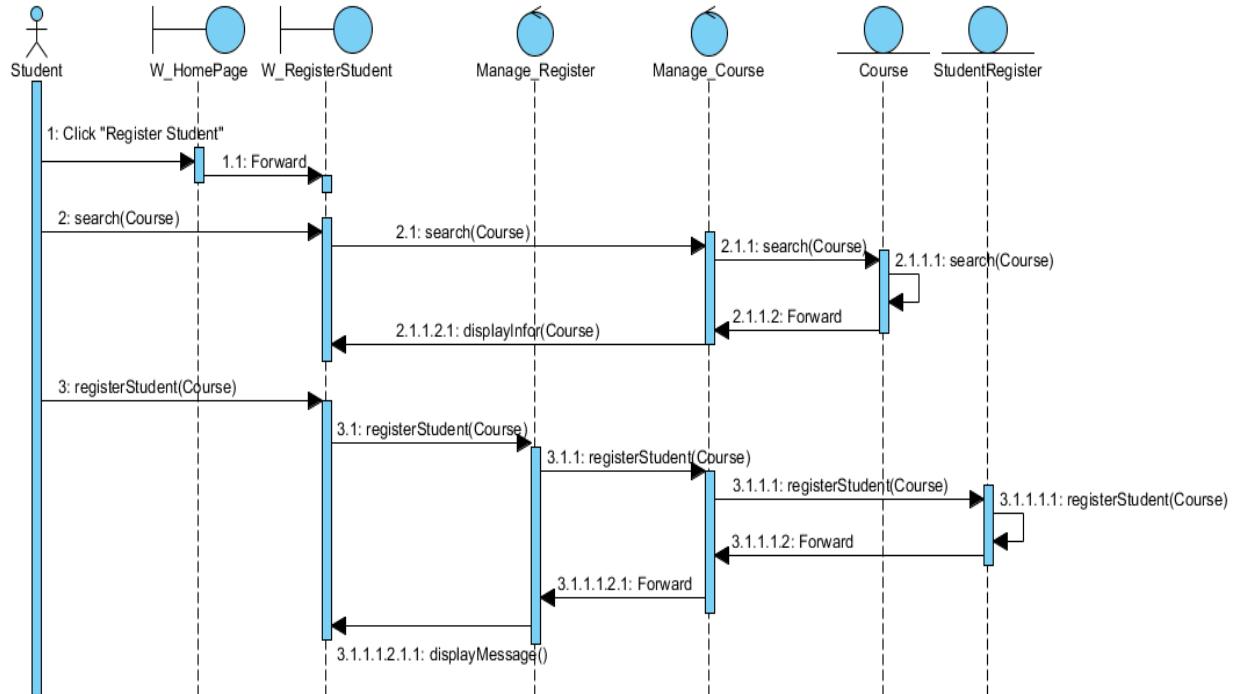
CHƯƠNG 4. PHÂN TÍCH YÊU CẦU

Giảng viên đăng ký môn dạy



Hình 4.21: Biểu đồ tuần tự giảng viên đăng ký dạy

Sinh viên đăng ký môn học



Hình 4.22: Biểu đồ tuần tự sinh viên đăng ký học

4.6 KẾT LUẬN

Chương này đã trình bày các bước trong phân tích yêu cầu theo cách tiếp cận hướng đối tượng. Nội dung bao gồm hai giai đoạn: phân tích động và phân tích tĩnh. Một số khái niệm cũng như biểu đồ UML liên quan các bước đã được trình bày đầy đủ và chi tiết hơn so với chương 2. Nhiều ví dụ minh họa đã được trình bày để bạn đọc hiểu rõ hơn các khái niệm, các biểu đồ. Case study Hệ quản lý đăng ký học theo tín chỉ đã được trình bày chi tiết theo từng bước trong pha phân tích để bạn đọc dễ theo dõi và hiểu đầy đủ hơn nữa các pha này.

BÀI TẬP

1. Trong Hệ thống quản lý bán hàng Siêu thị, lôp hóa đơn *Hoadon*, đơn đặt hàng *DonDathang* qua mạng của siêu thị máy tính có thể xem là lớp liên kết giữa khách hàng *Khachhang* và máy tính *Maytinh*. Hãy xác định các thuộc tính của các lớp này.
2. Trong hệ quản lý bán hàng, xây dựng các ca sử dụng nhập hàng dành cho nhân viên quản lý kho, nhập hóa đơn dành cho nhân viên bán hàng và đặt hàng dành cho khách hàng. Hãy xác định các lớp tương ứng, xây dựng biểu đồ lớp và các biểu đồ giao tiếp của các ca sử dụng này.
3. Dựa trên các biểu đồ giao tiếp trong câu 2, hãy xác định các phương thức và gán phương thức cho các lớp tương ứng. Sử dụng VP để sinh mã nguồn Java.
4. Viết chương trình với Java cho các chức năng dành cho các nhân viên và khách hàng ở Câu 2 dựa trên mã sinh ra ở Câu 3.
5. Xây dựng các bước phân tích cho Hệ quản lý đặt chỗ thuê xe và sử dụng Tool VP để hoàn thiện pha phân tích.
6. Viết chương trình để khách hàng đặt chỗ thuê xe trong Hệ quản lý thuê xe ở Câu 5.
7. Hoàn thiện pha phân tích của Hệ quản lý học theo tín chỉ.
8. Hãy xây dựng các bước phân tích yêu cầu cho Hệ thống Quản lý Thư viện điện tử và sử dụng Tool VP để hoàn thiện bài tập của mình.
9. Hãy xây dựng các bước phân tích yêu cầu cho Hệ thống Quản lý Siêu thị sách và sử dụng Tool VP để hoàn thiện bài tập của mình.
10. Hãy xây dựng các bước phân tích yêu cầu cho Hệ thống Quản lý thanh toán hóa đơn tiền điện và sử dụng Tool VP để hoàn thiện bài tập của mình.

CHƯƠNG 5

THIẾT KẾ KIẾN TRÚC HỆ THỐNG

5.1 GIỚI THIỆU

Trong Chương 4, chúng ta đã xem xét các bước trong pha phân tích và như vậy phân tích giúp ta hình dung ra nghiệp vụ cần gì. Trong chương này chúng ta sẽ xem xét các bước của pha thiết kế nghĩa là đưa ra quyết định về cách xây dựng hệ thống như thế nào. Nói cách khác, phân tích là nhằm trả lời câu hỏi “cái gì”, còn thiết kế là để trả lời câu hỏi “như thế nào”.

Hoạt động chính của pha thiết kế là chuyển đổi các biểu diễn lớp phân tích thành các biểu diễn lớp thiết kế. Trong pha này, nhóm dự án sẽ xem xét cần thận hệ thống mới sẽ hoạt động và được tích hợp như thế nào trong môi trường và hệ thống hiện thời. Nhóm cần phải khảo sát nhiều chiến lược thiết kế và quyết định chiến lược nào sẽ được sử dụng. Ví dụ, xây dựng từ đầu, mua một số môđun và địa phương hóa hay để bên thứ ba xây dựng một phần hệ thống.

Mặc dù ranh giới giữa hai pha phân tích và thiết kế là không rõ ràng, nhưng các tiến trình hoạt động của những pha này cần đến những ý tưởng hoàn toàn khác nhau. Sự mờ nhạt của ranh giới này có thể là cố ý như trong phương pháp luận phát triển lặp và tăng dần RUP, hoặc là ngẫu nhiên do nhóm phát triển phần mềm chưa có kinh nghiệm. Nhiều kinh nghiệm và nghiên cứu đã chỉ ra rằng việc tách biệt rõ ràng giữa pha phân tích và thiết kế là rất hữu ích vì nó đảm bảo chắc chắn rằng các yêu cầu đã được hiểu rõ ràng trước khi xem xét các giải pháp.

Không có một quy tắc nào cho việc chuyển từ mô hình phân tích sang mô hình thiết kế. Vì nó là quá trình có tính sáng tạo liên quan đến kinh nghiệm của nhóm phát triển, công nghệ sử dụng cho phát triển, sở thích cá nhân... Một khi đã hiểu rõ được sản phẩm của các pha xác định yêu cầu và phân tích, người thiết kế có thể hoặc sử dụng những kết quả có được từ pha phân tích hoặc bắt đầu với một tờ giấy trắng để tiến hành pha thiết kế. Nghĩa là chúng ta có thể không quan tâm về việc có một sự tương thích giữa những đối tượng phân tích và những đối tượng thiết kế, mà chú trọng hơn vào vấn đề thiết kế có dẫn tới một giải pháp hiệu quả không. Trong tài liệu này, chúng ta tiếp tục sử dụng các kết quả của pha phân tích cho tiến trình thiết kế.

Trong pha thiết kế, chúng ta không những cần đàm xuất công nghệ lựa chọn cho thiết kế và còn phải xem xét mức độ tác động của các lựa chọn này như thế nào tới các thư viện, các mẫu (pattern) và các khuôn dạng (framework) có sẵn và thậm chí tới những ghi chú UML chi tiết mà chúng ta sử dụng. Rõ ràng rằng thiết kế càng được tổng quát hơn thì càng ít gắn bó với một công nghệ cụ thể hơn. Điều này sẽ có tiện lợi là giảm bớt yêu cầu thành viên nhóm phát triển phải thành thạo nhiều công nghệ và đảm bảo chúng ta tránh được công nghệ lỗi thời. Tuy nhiên, mặt trái của việc thiết kế tổng quát là chúng ta có thể không lợi dụng được những lợi ích từ công nghệ cụ thể mà nó đem lại.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Lịch sử đã chứng minh rằng nhiều công nghệ xuất hiện và biến mất thường xuyên hơn những lý thuyết làm cơ sở cho các công nghệ đó. Ví dụ, nhiều công nghệ ra đời dựa trên các ngôn ngữ lập trình (COBOL, Fortran, Pascal, Ada, Modula, PL/I, C, C++, Smalltalk, Eiffel, C# và Java...) bị chi phối chỉ bởi hai lý thuyết cơ sở là lập trình hướng cấu trúc và lập trình hướng đối tượng. Vì vậy, rất hợp lý khi cho rằng “tổng quát thì an toàn hơn cụ thể”.

Vì phát triển phần mềm hướng đối tượng được tiến hành theo kiểu tăng dần, nên chúng ta không hy vọng có thể có một thiết kế đầy đủ cả hệ thống ngay từ đầu. Vì thế, lúc bắt đầu pha thiết kế, chúng ta cần lên kế hoạch cho những phần của hệ thống mà chúng ta sẽ thiết kế. Những mức ưu tiên ca sử dụng sẽ giúp chúng ta đánh dấu những ca sử dụng nào là cần thiết nhất. Như trình bày trong Chương 3, các ca sử dụng màu xanh phải được thiết kế đầy đủ; các ca sử dụng màu vàng thì chưa cần thiết kế ngay nhưng cần phải có một giải pháp dự phòng; ca sử dụng màu đỏ không buộc phải thiết kế, nhưng chúng vẫn nên có một giải pháp dự phòng.

Trong thực tiễn, chúng ta sẽ tìm một kiến trúc hệ thống có khả năng hỗ trợ một giải pháp hiệu quả, khả thi cho tất cả ca sử dụng. Sau đó, chúng ta sẽ tiến hành thiết kế chi tiết cho hầu hết các ca sử dụng quan trọng và thiết kế một phần cho các ca sử dụng ít quan trọng hơn. Giữa những tiến trình lặp lại, chúng ta sẽ điều chỉnh các mức ưu tiên sao cho hợp lý. Phần còn lại trong chương này sẽ trình bày các bước trong pha thiết kế và sau đó tập trung xem xét cách tiến hành các bước trong giai đoạn thiết kế hệ thống thế nào. Giai đoạn thiết kế các hệ thống con sẽ được trình bày trong chương tiếp theo.

5.2 CÁC BƯỚC TRONG PHA THIẾT KẾ

Pha thiết kế có thể được chia làm hai giai đoạn: *Thiết kế kiến trúc hệ thống* (hay thiết kế hệ thống, Thiết kế tổng thể) và *thiết kế chi tiết* (hay thiết kế hệ thống con). Thiết kế hệ thống yêu cầu chúng ta phải có cái nhìn tổng quát về các tác vụ trước khi đi sâu vào thiết kế chi tiết các hệ thống con (sẽ được giới thiệu trong Chương 6). Thật ra, trong phương pháp hướng đối tượng, không có ranh giới giữa thiết kế hệ thống và thiết kế chi tiết nhưng việc phân biệt hai giai đoạn này là cần thiết. Thiết kế bao gồm các hoạt động sau đây:

Thiết kế kiến trúc hệ thống

1. Lựa chọn công nghệ mạng cho hệ thống
2. Thiết kế tương tranh và an toàn-bảo mật
3. Phân rã hệ thống thành các hệ thống con
4. Xây dựng biểu đồ gói

Thiết kế chi tiết

1. Xây dựng biểu đồ lớp thiết kế
2. Xây dựng biểu đồ tuần tự
3. Xây dựng lược đồ cơ sở dữ liệu

Phần còn lại của chương này tập trung trình bày các bước trong giai đoạn thiết kế hệ thống. Giai đoạn thiết kế chi tiết các hệ thống con sẽ được xem xét trong Chương 6.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

5.3 LỰA CHỌN CÔNG NGHỆ MẠNG CHO HỆ THỐNG

Việc lựa chọn công nghệ mạng sẽ chỉ ra cách hệ thống được phân rã thành các thành phần logic và vật lý riêng biệt như thế nào. Kết quả của việc phân rã đó được gọi là *hình trạng hệ thống* (system topology). Để hiểu rõ hình trạng hệ thống, trước hết chúng ta sẽ khảo sát sơ lược lịch sử của các cấu trúc mạng và sau đó, xem xét các cấu trúc mạng hiện nay cũng như sự khác nhau giữa các kiểu máy khách client, giữa ứng dụng client-server với các ứng dụng phân tán khác. Cuối cùng sẽ xem xét biểu diễn hình trạng hệ thống bằng biểu đồ triển khai trong UML.

5.3.1 Kiến trúc mạng đơn tầng và hai tầng

Những năm 1940, máy tính là một thiết bị nguyên khối lớn, chỉ có khả năng chạy một chương trình tại một thời điểm. Sau đó, nó được cải tiến thành *Mainframe* – có khả năng chạy đồng thời nhiều chương trình, điển hình là chương trình một người dùng hoặc một chương trình cho một đợt (một đợt bao gồm một tập các dữ liệu giống nhau chạy qua một chương trình theo tuần tự). Mainframe có khả năng xử lý nhiều chương trình cùng một lúc vì nó chia thời gian cho CPU chạy các chương trình và mỗi chương trình khi tới lượt sẽ được thực thi.

Mô hình mainframe được xem là mô hình *kiến trúc một tầng* (one-tier Architecture). Nghĩa là, bất cứ chương trình nào được đưa vào, chỉ có duy nhất một mức hoạt động tính toán chạy trên một máy. Kiến trúc một tầng không có mạng, mặc dù có một sợi dây kim loại nối từ mỗi thiết bị tới mainframe, nhưng không tạo thành một mạng theo nghĩa hiện nay. Ưu điểm chính của mainframe là cài đặt đơn giản, nhưng nhược điểm chính của chúng là chỉ có thể tăng khả năng tính toán bằng cách mua thêm mainframe mới, hoặc nâng cấp cái cũ. Ngày nay, kiến trúc mạng này vẫn còn được sử dụng cho các hệ thống nghiệp vụ cỡ lớn.

Kiến trúc hai tầng (two-tier architecture) là thế hệ kiến trúc tiếp theo được phổ biến vào những năm 1970. Ý tưởng của kiến trúc này là nhằm tăng cường khả năng xử lý trên mỗi máy khách để cho các máy tính trung tâm không phải thực hiện toàn bộ các tiến trình. Như vậy, chúng ta có thể thêm hay thay thế các máy khách rẻ hơn các máy tính trung tâm. Các máy khách thường là các máy tính nhỏ hay các máy trạm, có thể truy nhập vào các máy tính trung tâm hay các server.

Với kiến trúc hai tầng, chương trình và dữ liệu phải được chuyển từ máy tính trung tâm sang máy khách. Điều này đòi hỏi phải có sự kết nối nhanh giữa hai bên và điều này dẫn đến ý tưởng hiện đại về mạng. Mạng là một tập hợp bất kỳ các máy chủ (host) được kết nối với nhau bằng các đường giao tiếp tốc độ cao.

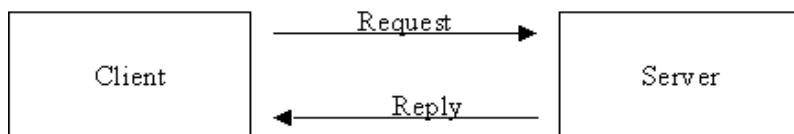
Khi có mạng máy tính, chúng ta có thể thêm các máy trung tâm cũng như các máy khách một cách dễ dàng nhằm nâng cao năng lực tính toán. Các máy khách cũng có thể quản lý chương trình và dữ liệu của nó một cách linh hoạt, mà không cần nhờ vào người quản trị hệ thống. Nếu chúng ta cho phép máy khách lưu trữ chương trình và dữ liệu, thì chúng ta cũng cần giải quyết một số vấn đề là khi dữ liệu thay đổi và chương trình được nâng cấp, máy khách sẽ bị lỗi. Vì vậy, nên hạn chế lưu trữ chương trình và dữ liệu trên máy khách.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Kiến trúc hai tầng đã mang lại khả năng đồ họa tinh vi và hệ thống cửa sổ cho máy khách để thay thế cho mô hình chỉ có văn bản của máy điện báo đánh chữ. Kiến trúc hai tầng vẫn còn được sử dụng rộng rãi hiện nay chủ yếu cho hệ điều hành Unix và máy chủ quản lý tệp (file server).

Hoạt động của mô hình kiến trúc hai tầng

Máy khách (Client) và máy chủ (Server) trao đổi thông tin với nhau dưới dạng các thông điệp (Message). Thông điệp gửi từ máy khách sang máy chủ gọi là các thông điệp yêu cầu (Request Message) mô tả công việc mà phần mềm khách muốn máy chủ thực hiện.



Hình 5.1: Kiến trúc chương trình Client-Server

Mỗi khi máy chủ nhận được một thông điệp yêu cầu, máy chủ sẽ phân tích yêu cầu, thực thi công việc theo yêu cầu và gửi kết quả về máy khách (nếu có) trong một thông điệp trả lời (Reply Message). Khi vận hành, một máy chủ sẽ phục vụ cho nhiều máy khách. Mỗi một ứng dụng trên mô hình máy khách - máy chủ (client-server) phải có một *Giao thức* (Protocol) riêng để trao đổi thông tin, phối hợp công việc giữa máy khách và máy chủ. Giao thức qui định một số vấn đề cơ bản sau:

- Khuôn dạng loại thông điệp.
- Số lượng và ý nghĩa của từng loại thông điệp.
- Cách thức bắt tay, đồng bộ hóa tiến trình truyền nhận giữa máy chủ và máy khách...

Thông thường phần mềm máy khách đảm nhận các chức năng về giao diện người dùng, như tạo các form nhập liệu, các thông báo, các biểu báo giao tiếp với người dùng; phần mềm máy chủ đảm nhận các chức năng về xử lý và lưu trữ dữ liệu. Mô hình kiến trúc hai tầng tạo điều kiện dễ dàng cho việc bảo trì, chia sẻ, tổng hợp dữ liệu trong toàn bộ công ty hoặc tổ chức. Các chức năng về hoạt động nghiệp vụ có thể được cài đặt ở phần mềm máy khách hoặc ở phần mềm máy chủ và khi đó sẽ tạo ra hai loại kiến trúc máy khách-máy chủ là máy khách nặng (Fat Client) và máy khách nhẹ (Thin Client).

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Máy khách nặng (Fat Client)

Theo mô hình này, các hoạt động nghiệp vụ được cài đặt bên phía máy khách và phần máy chủ thực hiện chức năng chủ yếu về truy vấn và lưu trữ thông tin. Mô hình này có ưu điểm là tạo ra ít giao tiếp trên mạng do dữ liệu tạm thời trong quá trình tính toán được lưu trên máy khách nên có thể giảm tắc nghẽn. Tuy nhiên, nó có một số nhược điểm là chi phí đầu tư phần cứng cho máy khách cao và khó nâng cấp vì phải cài đặt lại toàn bộ máy khách khi muốn nâng cấp hệ thống.

Máy khách nhẹ (Thin Client)

Ngày nay, các doanh nghiệp thường sử dụng rộng rãi mô hình Máy khách – máy chủ (client-server). Trong mô hình này, tất cả máy tính để bàn của các nhân viên (máy khách) sẽ được kết nối vào một máy tính trung tâm (máy chủ). Máy chủ sẽ đóng vai trò như một kho tài nguyên (dữ liệu, các thiết bị phần cứng và một số dịch vụ mạng khác...) phục vụ cho tất cả các máy khách nối với nó. Bên cạnh khả năng xử lý của máy chủ, các máy khách đều có khả năng xử lý (thường là yếu hơn máy chủ) và lưu trữ riêng để phục vụ cho các tác vụ không liên quan đến mạng như soạn thảo văn bản, bảng tính, đồ họa...

Tuy nhiên, trong một doanh nghiệp, các ứng dụng phi mạng của các máy khách thường là giống nhau như cài đặt Microsoft Windows, Microsoft Word, Microsoft Excel, các chương trình nghe nhạc, xem phim...

Do đó, công nghệ máy khách nhẹ đã được đưa ra nhằm đáp ứng nhu cầu “máy tính của người dùng có cấu hình tối thiểu”. Công nghệ máy khách nhẹ tương tự như công nghệ client/server, điểm khác duy nhất và cũng quan trọng nhất là tất cả năng lực xử lý lẫn lưu trữ của toàn bộ mạng máy tính đều tập trung vào máy chủ. Mọi máy khách đều chỉ còn một màn hình, bàn phím và chuột. Mọi chương trình hay phần mềm của người dùng sẽ được lưu trữ trên máy chủ và cũng sẽ được thi hành bởi CPU của máy chủ. Mỗi khi có một người sử dụng mới đăng ký vào hệ thống. Người đó sẽ được “cấp” cho một máy tính ảo để người dùng có thể chọn những phần mềm hoặc hệ điều hành (đã có sẵn trên máy chủ) mà họ ưa thích. Lợi điểm của công nghệ này là vừa an toàn cả vừa kinh tế nên càng ngày càng được khẳng định.

Tuy nhiên, mô hình này cũng có một số nhược điểm là tạo ra nhiều thông điệp trao đổi giữa máy khách và chủ và do đó, yêu cầu máy chủ phải có cấu hình cao để tránh dẫn đến khả năng tắc nghẽn.

5.3.2 Kiến trúc ba tầng

Kiến trúc ba tầng trở nên khá phổ biến vào những năm 1990 bằng cách chia hệ thống thành ba phần: giao diện người dùng, logic chương trình và lưu trữ dữ liệu. Trong kiến trúc này, mỗi chương trình được phân rã trên ba tầng khác nhau:

- *Tầng giao diện người dùng* (User Interface) hay tầng máy khách thể hiện giao diện mà người sử dụng có thể nhập yêu cầu, dữ liệu và xem kết quả. Nghĩa là nó chỉ xử lý việc giao tiếp với người sử dụng, nhập xuất... mà không thực hiện việc tính toán, kiểm tra, xử lý, hay các thao tác liên quan đến cơ sở dữ liệu.
- *Tầng ứng dụng* (Application Server, Business Rule) được biết như tầng logic nghiệp vụ hay tầng dịch vụ để chạy chương trình đa luồng. Tầng này thực hiện xử lý các chức

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

năng chính, kiểm tra các ràng buộc... Việc thực hiện này độc lập với cách thiết kế cũng như cài đặt giao diện và thông tin lấy từ tầng giao diện.

- *Tầng dữ liệu* (Database Server, Data Storage) nhằm lưu trữ dữ liệu và cung cấp cơ chế an toàn cho việc truy nhập đồng thời với sự giúp đỡ của hệ quản trị cơ sở dữ liệu. Tầng này thực hiện các công việc liên quan đến dữ liệu mà phần mềm cần đến như đọc, ghi...

Các ưu điểm của kiến trúc 3 tầng:

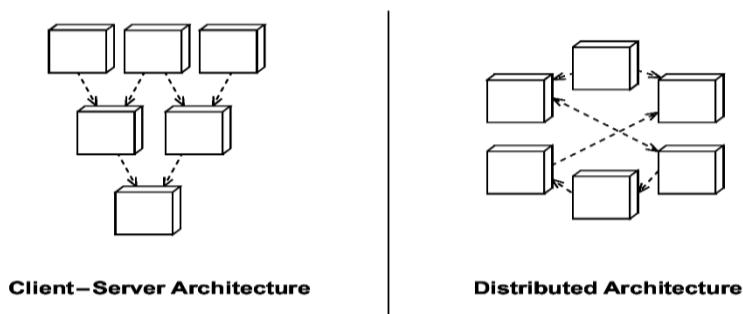
- *Tạo điều kiện dễ dàng khi phát triển*: Bất kỳ hệ thống lớn nào cũng đều bao gồm ba phần: logic chương trình, giao diện người dùng và cơ chế quản lý hiệu năng/bảo mật của dữ liệu. Việc phân chia hệ thống thành các phần như trên sẽ tạo điều kiện cho người lập trình thực hiện công việc một cách đơn giản hơn.
- *Sử dụng máy tính hiệu quả hơn*: Tùy theo từng tầng mà chúng ta sẽ sử dụng các máy tính sao cho phù hợp. Ví dụ, giao diện người dùng thực hiện nhiệm vụ đơn giản không đòi hỏi máy tính lớn; việc thực thi logic chương trình yêu cầu sử dụng CPU, bộ nhớ, nhưng không đòi hỏi dung lượng đĩa quá lớn, vì vậy có thể sử dụng máy tính server; quản lý dữ liệu yêu cầu nhiều về khả năng tính toán, và dung lượng đĩa, do đó có thể sử dụng máy chủ hay mainframe.
- *Cải tiến hiệu năng*: có thể nhân rộng các máy ở lớp dữ liệu và lớp giữa để lan truyền tính toán (cân bằng tải), mỗi tầng được chuyên môn hóa và như vậy sẽ dễ dàng tối ưu hóa.
- *Nâng cao tính bảo mật*: Thường thì hệ thống ba tầng được triển khai cho các máy client chạy trên mạng Internet. Vì vậy, chúng ta phải có cơ chế bảo mật nghiêm ngặt để bảo vệ máy chủ, chương trình và dữ liệu. Với kiến trúc ba tầng, chúng ta có thể đặt cơ chế bảo mật ở tầng giữa nhằm tránh những tấn công vô tình hay cố ý từ bên ngoài. Tầng dữ liệu ở sau tầng giữa, do đó chúng ta không cần phải bảo mật cho phần cứng hay sự giao tiếp của chúng. Điều này giúp tầng dữ liệu chạy với tốc độ cao và dễ thao tác.
- *Hạn chế đầu tư*: Đối với trường hợp chúng ta có một mainframe lưu trữ và xử lý dữ liệu trong nhiều năm, khi có sự cố chúng ta không muốn phải vứt bỏ tất cả và làm lại từ đầu. Kiến trúc ba tầng và mạng là phương án thích hợp nhất để giải quyết vấn đề này. Ta sử dụng tầng giữa làm trung gian khi client kết nối với mainframe, hoặc khi máy chủ kết nối tới máy khách.
- *Tính linh hoạt*: Được thể hiện rõ qua việc chúng ta có thể thêm hoặc bớt các máy tính trong hệ thống nếu hệ thống đó được thiết kế theo kiến trúc ba tầng. Ví dụ, khi phần logic được thiết kế đúng đắn, chúng ta có thể phát triển nó theo kiến trúc một tầng, sau đó phát triển lên thành hai tầng, ba tầng tùy theo yêu cầu.
- *Đa dạng kiểu dáng máy client*: máy tính ở tầng client chỉ thực hiện nhận đầu vào và hiển thị kết quả trên màn hình. Do đó, chúng ta có thể sử dụng các thiết bị với các giao diện khác nhau như máy tính cá nhân, PDAs, mobile-phone.... Khi đó, tầng giữa và tầng dữ liệu vẫn làm việc như nhau, không có thay đổi.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Với tất cả những ưu điểm trên, kiến trúc ba tầng nên được sử dụng khi thiết kế các hệ thống dù có kích cỡ nhỏ hay lớn.

5.3.3 Kiến trúc Client – Server và kiến trúc phân tán

Bất cứ khi nào cần kết nối với nhiều máy hoặc nhiều hệ thống phần mềm, chúng ta phải chọn giữa hai loại là client – server hoặc phân tán như được trình bày ở Hình 5.2. Mặc dù xuất phát từ thời mainframe, “client – server” có nghĩa là chúng ta có một lượng lớn các client, đơn giản gửi yêu cầu đến một máy chủ lớn xử lý các yêu cầu đó. Ngược lại, kiến trúc phân tán (hay ngang hàng) được đặc trưng bằng một tập các máy tự chủ, truyền thông nhau theo bất kỳ một hướng nào khi có nhu cầu phát sinh.



Hình 5.2: Kiến trúc Client – Server và phân tán [7]

Ví dụ quen thuộc nhất của kiến trúc client – server là mô hình các hệ thống thương mại điện tử. Trình duyệt web của các khách hàng phát đi các yêu cầu để kết nối với các máy chủ Web và khi đó máy chủ web phát đi các lệnh đến các hệ thống đầu cuối. Phần lớn các hệ thống hai tầng và ba tầng là mô hình client – server.

Kiến trúc phân tán thực hiện công việc tính toán lớn trải rộng ra trên nhiều máy tính trên mạng. Nghĩa là, khi có một lượng lớn dữ liệu cần thực hiện tính toán, chúng ta có thể chia dữ liệu hoặc tính toán đó thành nhiều bài toán tính toán trên nhiều máy độc lập.

Ví dụ, SETI@home (setiathome.ssl.berkeley.edu) - một tổ chức phi lợi nhuận có mục đích tìm kiếm và xử lý các tín hiệu radio lạ ngoài trái đất. Các dữ liệu radio được tự động thu thập và phân tán đến các máy trên mạng Internet. Mỗi máy riêng lẻ tiến hành phân tích dữ liệu được gửi đến và bắt đầu kết quả đáng chú ý nào cũng sẽ được gửi trở lại máy chủ trung tâm để phân tích kỹ hơn. Ý tưởng gộp một số lượng lớn các máy để giải quyết các bài toán phức tạp cũng được sử dụng để tìm các số nguyên tố, nghiên cứu bệnh ung thư... Điều này đã dẫn đến sự ra đời một lĩnh vực nghiên cứu mới là *tính toán lưới* (Grid Computing).

Các thuật ngữ “client – server” và “distributed” (hay “peer to peer”) cũng được sử dụng để mô tả các kiến trúc phần mềm, không phụ thuộc vào việc phần mềm được triển khai như thế nào trên các máy hoặc trên các mạng. Ví dụ dễ dàng thấy là các đối tượng chạy trong một chương trình thường được xem như các server có thể được sử dụng lại trong các ngữ cảnh khác nhau với những đối tượng client khác nhau. Trong những ứng dụng đặc biệt, chúng ta có thể viết một nhóm các đối tượng cùng cộng tác với nhau theo kiểu phân tán.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Các liên kết truyền thông mạng có khuynh hướng là trao đổi hai chiều, nghĩa là mặc dù các liên kết có thể ban đầu được mở bởi client nhưng server vẫn có thể gửi thông tin đến client (client có nhận hay không phụ thuộc người thiết kế của phần mềm client). Vì thế, sự khác nhau giữa client – server và phân tán chỉ là một mô hình nhân tạo, được sử dụng bởi những người thiết kế để cấu trúc giải pháp của họ theo cách này hay cách kia. Kiến trúc client –server dễ phát triển hơn, nhưng chúng không đem lại hiệu năng cao (ví dụ, client thường rỗi rải khi server đang xử lý một trong những yêu cầu của nó). Ngược lại, các kiến trúc phân tán thường khó phát triển hơn nhưng chúng có thể đem lại hiệu quả tốt hơn. Lựa chọn kiến trúc phù hợp là bước rất tự nhiên trong quá trình phát triển hệ thống.

Ví dụ, trong một phiên mua bán là một quá trình xử lý theo từng bước (khách hàng hỏi các chi tiết về sản phẩm, người bán cung cấp các chi tiết sản phẩm, khách hàng chọn mua sản phẩm, người bán đưa ra mẫu đơn để mua, khách hàng điền vào mẫu đơn và gửi đi...). Như vậy quá trình này là một tương tác kiểu client –server.

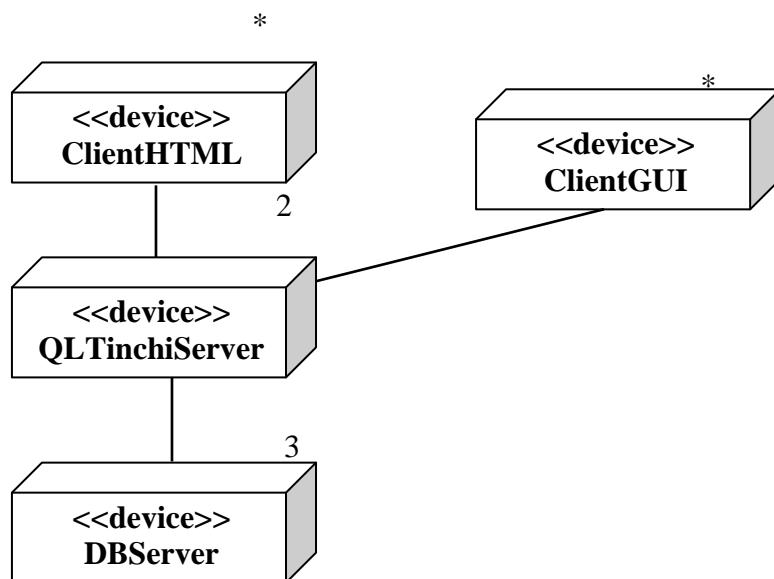
Ngược lại, hệ mô phỏng chuyến bay nhiều người dùng yêu cầu máy tính của phi công thực thi công việc theo thời gian thực biểu diễn qua buồng lái và truyền thông duy nhất cần thiết là phát tin định kì về vị trí hiện thời của máy bay. Vì thế đối với hệ thống mô phỏng chuyến bay nhiều người dùng thì kiến trúc phân tán là thích hợp.

5.3.4 Biểu diễn hình trạng mạng với UML

Các kiến trúc hệ thống có thể được mô tả bằng một biểu đồ triển khai trong UML (xem Hình 5.3). Biểu đồ triển khai đơn giản chỉ nêu ra các nút, đường truyền thông và vô số các điểm khác. Mỗi nút trong biểu đồ này thể hiện một máy trạm (thể hiện với từ khóa UML <<device>>). Một đường truyền thông chỉ ra rằng hai nút có thể liên lạc với nhau theo một cách nào đó. Các nút có dấu * thể hiện có rất nhiều nút có thể tồn tại trong thời gian chạy. Trong biểu đồ, chúng ta có các nút lặp lại (QLTinchiServer và DBServer) và các nút nhân lên nhiều lần (ClientHTML và ClientGUI). Các biểu đồ triển khai giống như các biểu đồ lớp và biểu đồ đối tượng, trong đó chúng có thể nêu ra các kiến trúc có thể (kiểu nút) và các kiến trúc thực sự (các thể hiện nút). Khi chỉ ra các thể hiện nút, giống như các đối tượng trong biểu đồ đối tượng, nhãn nút có dạng *tên:Kiểu* và nên được gạch chân.

Các biểu đồ triển khai thông thường có lời mô tả kèm theo. Trong Hình 5.3, tầng dữ liệu QLTinchi bao gồm hai server cơ sở dữ liệu (gọi là DBServer). Có ba nút như thế cung cấp lượng lớn dữ liệu. Tầng giữa, liên lạc với tầng dữ liệu bao gồm hai máy server (QLTinchiServer). Mỗi QLTinchiServer có thể được truy nhập đồng thời bởi bất cứ điểm HTMLClient hay GUIClient nào.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG



Hình 5.3: Biểu đồ triển khai cơ bản cho Hệ quản lý học tín chỉ

5.4 THIẾT KẾ TƯƠNG TRANH VÀ AN TOÀN-BẢO MẬT

5.4.1 Thiết kế tương tranh

Trong hệ thống mạng, nhiều hoạt động thường xảy ra cùng một lúc như trong một thẻ thông nhất và những tiến trình riêng lẻ thực thi các chức năng như một phần của hệ thống. Các hệ thống này gọi là các *hệ thống tương tranh* (concurrent system) hay đồng thời. Mặc dù phát triển hệ thống sẽ dễ dàng hơn nếu xác lập được một hàng đợi có thứ tự dựa vào tất cả người dùng và các tiến trình. Tuy nhiên, trong thực tế chúng ta phải biến hỗn độn này thành thứ tự bằng nỗ lực lập trình của mình. Tính toán tương tranh xem xét một số vấn đề:

- Làm sao đảm bảo được rằng thông tin được cập nhật đầy đủ trước khi có người thao tác trên cập nhật đó? Ví dụ, dừng mọi truy nhập vào đăng ký môn học cho tới khi tất cả các chi tiết môn học đã được thêm vào.
- Làm sao đảm bảo được rằng thông tin không được cập nhật trong khi nó đang được đọc? Ví dụ, không cập nhật môn học trong khi nó đang được bạn đọc xem.

Việc giải quyết các vấn đề này sẽ được xem xét ở hai mức độ khác nhau. Ở mức thấp, các giao dịch cơ sở dữ liệu và điều khiển luồng được sử dụng để bảo vệ dữ liệu bên trong các tiến trình riêng lẻ. Ở mức cao, chúng ta cần sử dụng các quy tắc của hệ thống và quy tắc nghiệp vụ để điều khiển hoạt động tương tranh.

Cách dễ dàng nhất để xử lý tương tranh là thêm ràng buộc vào hệ thống hay đưa thêm vào quy tắc nghiệp vụ. Để hiểu rõ hơn về thêm ràng buộc hệ thống, ta hãy xem ví dụ về Hệ quản lý học theo tín chỉ. Thay vì cố cập nhật môn học trong khi sinh viên đang xem, chúng ta có thể cập nhật trong một cơ sở dữ liệu riêng và chuyển lại sau đó. Cách làm này có nghĩa là người sử dụng truy nhập Internet chỉ được phép đọc thôi và điều này sẽ giúp chúng ta dễ dàng viết code hơn. Người ta cũng có thể sử dụng các quy tắc nghiệp vụ để giúp việc phát triển thuận lợi hơn.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Ví dụ, xem xét một hệ thống mua bán vé máy bay. Khách hàng An tới một văn phòng đặt vé tại Hà nội cùng thời điểm với khách hàng Ban tới một văn phòng đặt vé khác cũng tại Hà nội. Cả hai cũng định mua một vé cho cùng chuyến bay đi Đà Nẵng. Thật không may, chỉ còn lại một vé duy nhất. Cả hai cùng lúc hỏi “Có còn vé không?” Cả hai nhân viên bán vé sẽ cùng kiểm tra trên hệ thống và sẽ cùng trả lời “Còn vé”. Khi có sự tranh chấp như vậy thì người khách hàng đầu tiên nói “Đồng ý, tôi sẽ mua nó” sẽ thắng. Điều này cũng còn phụ thuộc vào khả năng của nhân viên và sự trễ mạng từ máy của văn phòng đặt vé tới nơi đặt máy chủ. Trong kịch bản bán vé nói trên, chúng ta phải đảm bảo, ít nhất là không tình cờ bán hai vé khi chỉ còn một ghép duy nhất. Đó là một vấn đề tương tranh mức tiến trình, bởi nó có thể được điều khiển bởi tiến trình máy chủ dựa trên phân loại yêu cầu: thứ nhất đến để mua vé, thứ hai đưa ra một thông điệp lỗi. Nhưng nó có thể không tốt về phía nghiệp vụ vì có thể khiến khách hàng bức tức khi trả lời rằng còn một vé mà chỉ ngay sau đó vài giây đã không còn gì cả.

Để tránh điều này, chúng ta có thể đưa ra thêm quy tắc nghiệp vụ như sau: khi một nhân viên truy vấn số vé còn lại, nếu chỉ còn một vé, nó tạm thời được giữ cho tới khi nhân viên hủy bỏ yêu cầu hoặc thời hạn đặt vé đã hết (ví dụ, việc đặt vé tạm thời được phép kéo dài 10 giây nếu nhân viên không hủy bỏ nó trước). Với quy tắc nghiệp vụ mới này, chúng ta có thể đảm bảo rằng chỉ khách hàng đầu tiên đưa ra yêu cầu tới máy chủ mới mua được vé (máy chủ có thể tích hợp truy vấn và lưu trữ bên trong một dịch vụ nghiệp vụ đơn).

Giả sử việc đặt vé cho An thành công. Nhân viên đang phục vụ An có 10 giây để yêu cầu anh ta chi trả cho vé đã giữ, hoặc nhân viên có thể hủy bỏ trong thời gian đó nếu An thay đổi ý định. Nhân viên cũng sẽ hủy bỏ nếu phương thức chi trả của An có lỗi, đó là một lý do cho việc có giữ tạm thời cho người đầu tiên. Ban lúc này nghĩ rằng vé đã được bán hết trước khi cô ấy tới cửa hàng đặt vé, vì thế cô ấy không có lý do gì để tức giận với nhà cung cấp. Nếu cuối cùng An không mua vé và sau đó Ban biết rằng vé chưa bán hết thì cô ấy có thể chỉ đơn giản nghĩ rằng “Ai đó đã hủy bỏ”.

5.4.2 Thiết kế an toàn-bảo mật

An toàn-bảo mật là một vấn đề rất quan trọng và luôn luôn được quan tâm một cách đặc biệt trong hầu hết các hệ thống. Do đó, việc xem xét đầy đủ về an toàn phải là chủ đề riêng và nằm ngoài phạm vi giáo trình này. Trong phần này, tài liệu chỉ trình bày một cách tổng quan để bạn đọc thấy được vai trò quan trọng của an toàn trong phát triển các hệ thống thông tin.

Các khía cạnh an toàn-bảo mật

Một hệ thống được gọi là an toàn khi hệ thống được bảo vệ khỏi sự lạm dụng dù là vô tình hay cố ý. An toàn là thuật ngữ rộng hơn rất nhiều, nó có thể được chia thành năm khía cạnh :

- *Sự riêng tư* (Privacy): Thông tin có thể được che dấu, và chỉ ở trạng thái sẵn sàng với những người dùng được phép tác động vào nó như xem hoặc chỉnh sửa. Điều này đảm bảo cho người sử dụng khai thác tài nguyên của hệ thống theo đúng chức năng, nhiệm vụ đã được phân cấp, ngăn chặn được sự truy nhập thông tin bất hợp pháp.
- *Xác thực* (Authentication): Cần biết nơi mà phần thông tin được gửi đến để quyết định xem phần thông tin đó có đáng tin cậy hay không

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

- **Tính không thể bác bỏ được** (Irrefutability): Ngược với xác thực, tính không thể bác bỏ được nhằm đảm bảo rằng người tạo ra thông tin không thể phủ nhận rằng họ chính là người tạo ra nó. Điều này hữu ích cho chúng ta khi có bất kì sai sót xảy ra.
- **Tính toàn vẹn** (Integrity): Đảm bảo rằng thông tin không bị mất mát trên đường đi, bảo đảm sự nhất quán của dữ liệu trong hệ thống. Có thể đưa ra các biện pháp để ngăn chặn được việc thay đổi bất hợp pháp hoặc phá hoại dữ liệu.
- **Tính an toàn** (Safety): phải có thể điều khiển việc truy cập tài nguyên (như máy móc, tiến trình, cơ sở dữ liệu và các tệp). Tính an toàn cũng được hiểu như là quyền hạn.

Bốn yêu cầu đầu tiên có thể được thực hiện bằng cách sử dụng mã hóa số. Riêng tính an toàn thì phức tạp hơn nhiều. Thông thường, khi một đoạn mã trình đang được thực thi, hệ điều hành sẽ áp dụng một kiểu điều khiển nào đó để đoạn mã trình có thể thực hiện được như điều khiển truy cập các tệp, các thư mục và các chương trình khác. Tuy nhiên, sự điều khiển của hệ điều hành là không phải khi nào cũng có thể thực hiện được.

Nếu hệ thống hoạt động trên mạng, thì vấn đề an toàn càng trở nên quan trọng hơn. Bởi vì, qua mạng, các hacker có thể cố gắng chiếm đoạt chương trình đang chạy trên máy của ta nhằm mục đích phá hoại.

Những quy luật chung về an toàn-bảo mật

Để thiết kế một hệ thống an toàn tốt, ta buộc phải suy nghĩ một cách nghiêm túc để chắc chắn rằng không ai có thể xâm nhập bất hợp pháp vào hệ thống. Sau đây là một vài điều đáng lưu ý để bảo vệ hệ thống của bạn:

- Ngăn chặn việc xâm nhập máy chủ khi không được phép.
- Các thông tin nhạy cảm như chi tiết về ý tưởng kinh doanh hay chiến lược kinh doanh, các hồ sơ cá nhân, chi tiết về số thẻ tín dụng mà bạn đang dùng, các thông tin liên quan đến an ninh quốc gia... phải được đảm bảo không bị truyền ra ngoài.
- Đảm bảo thông tin khi đi ra bên ngoài không bị “nghe lén” trong quá trình truyền và chỉ có đúng người nhận mới có thể đọc được.
- Bảo vệ mật mã (password) của khách hàng và nhân viên, nó không chỉ đơn thuần là chính sách bảo mật mà nó còn liên quan tính riêng tư cao.
- Bảo vệ tài nguyên hệ thống của client: bảo vệ client chống lại các truy cập tài nguyên bất hợp pháp và chống lại sự phá hoại từ bên ngoài. Vì chúng ta muốn cung cấp dịch vụ chất lượng tốt và không muốn bị kiện ra tòa do vài sai lầm.

Ngoài ra, chúng ta cũng có thể thuê các hacker có đạo đức hay những người có chuyên môn để xem xét, kiểm tra hệ thống. Điều này sẽ giúp ta không bị quá bất ngờ khi một lỗi nào đó xuất hiện. Trên đây chỉ là một số các lưu ý khi xây dựng các chính sách bảo mật cho hệ thống, khi nhắc đến khía cạnh an toàn thì ta nên biết một số vấn đề liên quan đến cách tấn công và hiểu cách tấn công vào hệ thống như thế nào để có thể đưa ra các chính sách hợp lý.

5.5 PHÂN RÃ HỆ THỐNG THÀNH CÁC HỆ THỐNG CON

Đối với các doanh nghiệp lớn, ta không thể gộp tất cả các thực thể nghiệp vụ và các tiến trình nghiệp vụ vào trong một hệ thống phần mềm đơn lẻ vì điều đó sẽ quá phức tạp, khó sử dụng

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

và khó bảo trì. Chúng ta nên chia nhỏ phần mềm thành các hệ thống, rồi chia các hệ thống thành các hệ thống con và sau đó chia thành các tầng.

5.5.1 Hệ thống và hệ thống con

Chúng ta hãy xem khái niệm đơn giản “khách hàng”. Khái niệm này được nhìn theo những cách khác nhau bởi các bộ phận khác nhau trong một tổ chức lớn như bộ phận bán hàng, bộ phận thanh toán, bộ phận nhập hàng, bộ phận phân phối... Nếu chúng ta cố gộp chúng lại với nhau thành một hệ phần mềm đơn lẻ hỗ trợ tất cả những bộ phận đó thì “khách hàng” sẽ có hàng trăm thuộc tính và hàng trăm các thao tác khác nhau.

Do đó, hệ thống nghiệp vụ như vậy nên có một số hệ thống tách biệt nhau, mỗi hệ thống được cài đặt bởi các nhóm phát triển khác nhau để việc sử dụng lại các đối tượng không tương thích được giảm bớt. Khi đó, thông tin nào cần được truyền giữa các hệ thống phải được truyền theo một cách xác định, được điều khiển qua những giao diện xác định trước. Để giảm độ phức tạp, mỗi hệ thống nên được chia nhỏ hơn nữa thành các hệ thống con riêng biệt.

5.5.2 Các cụm

Bên trong một hệ phần mềm, ta thường sử dụng nhiều cụm mã trình. *Cụm (layer) là một tập hợp các đối tượng cộng tác với nhau tùy theo những tiện ích được cung cấp bởi các cụm dưới*. Ví dụ, thư viện hệ thống Unix cung cấp truy cập tới các tiện ích của hệ điều hành mức thấp qua tầng các hàm C. Những tầng này làm giảm sự phức tạp bằng cách chia nhỏ phần cài đặt thành nhiều phần dễ quản lý hơn. Những tầng này cũng có khả năng đem lại cơ hội tái sử dụng, bởi mỗi tầng được viết một cách độc lập với các tầng ở trên. Bertrand Meyer từng nói [7]:

Một hệ phần mềm thực thụ, thậm chí một phần mềm nhỏ theo như chuẩn ngày nay, đều liên quan đến rất nhiều lĩnh vực mà không thể đảm bảo độ chính xác của nó nếu xử lý tất cả các thành phần và các tính chất trên cùng một mức. Thay vào đó, cách tiếp cận theo cụm là cần thiết trong đó mỗi cụm dựa vào các cụm thấp hơn.

Dù không quan tâm đến tổng số cụm, nhưng nói chung cụm trên cùng thường biểu diễn giao diện người dùng, cụm dưới cùng thường biểu diễn hệ điều hành, hoặc kết nối mạng. Các cụm có thể là “mở” (cụm trên có thể gọi một số đối tượng của cụm dưới và cụm dưới quản lý các đối tượng nhưng không hoàn toàn giàu chúng) hoặc “đóng” (đóng gói hoàn toàn các cụm dưới nghĩa là các đối tượng cụm dưới ẩn với cụm trên). Không có quy định cụm nào nên để mở hay đóng. Nói chung, cụm đóng yêu cầu lập trình nhiều hơn và chạy chậm hơn so với cụm mở vì cần thực hiện sao chép và chuyển đổi thông tin hơn. Ngược lại, cụm mở thì thường không an toàn vì cụm dưới không được bảo vệ và khó để bảo trì vì các cụm đều phụ thuộc nhiều hơn một cụm trên nó.

Ở mức độ nào đó, chúng ta có thể chuyển đổi giữa các cụm mà không ảnh hưởng đến các đoạn mã trình khác. Ví dụ, chúng ta có thể vứt bỏ cụm cao nhất và thay thế bằng cái khác. Chúng ta có thể thay đổi cụm trung gian bằng cụm khác có cùng giao diện mà không ảnh hưởng đến các cụm ở trên. Thường thì, các cụm được thực thi lại tại các lớp dưới khi ta thay đổi hệ thống, hoặc tại đỉnh khi ta thay đổi giao diện người dùng tới thiết bị mới.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

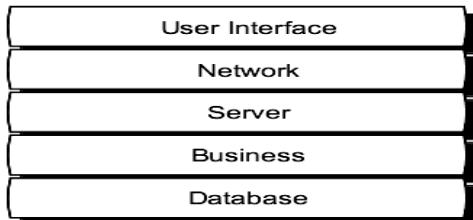
Các cụm cho hệ thống đơn tầng



Hình 5.4: Các tầng trong hệ thống đơn tầng [7]

Hình 5.4 biểu diễn một phân cụm đơn giản và có thể được thực thi trong hệ thống đơn tầng. Cụm đáy là cụm cơ sở dữ liệu (Database), công việc của nó là chuyên dữ liệu đi và về giữa hệ quản trị cơ sở dữ liệu và cụm nghiệp vụ (Business). Vì phần lớn các ứng dụng đều có yêu cầu lưu trữ dữ liệu, nên nếu hệ thống đơn giản lưu trữ dữ liệu dưới dạng tệp thì cụm cơ sở dữ liệu sẽ là một hệ thống tệp thay vì hệ quản trị cơ sở dữ liệu. Phía trên của cụm cơ sở dữ liệu là cụm nghiệp vụ bao gồm những đối tượng thực thể và những đối tượng hỗ trợ thực thi. Cuối cùng, phía trên của cụm nghiệp vụ là cụm giao diện người dùng (user interface) chứa các đối tượng mà công việc của nó là mô tả những lựa chọn có giá trị cho con người, đưa các lệnh và dữ liệu đến cụm nghiệp vụ và hiển thị dữ liệu được trả về từ cụm nghiệp vụ.

Các cụm cho hệ thống hai và ba tầng



Hình 5.5: Các cụm trong hệ thống hai hoặc ba tầng [7]

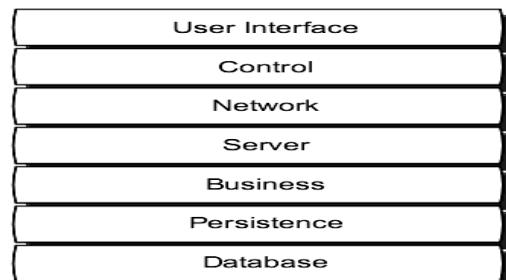
Các hệ thống hai hay ba tầng thực thi trên mạng lấy thông tin từ giao diện người dùng chuyển tới cụm nghiệp vụ chạy trên máy chủ. Vấn đề này có thể giải quyết bằng cách sử dụng nhiều hơn hai tầng (Hình 5.5). Cụm mạng (Network) chứa các đối tượng giúp cho mạng trong suốt với giao diện người dùng (user interface). Cụm giao diện người dùng có thể truy nhập trực tiếp đến các đối tượng cụm dịch vụ (server). Cụm dịch vụ chứa các đối tượng sử dụng tập các dịch vụ nghiệp vụ của cụm nghiệp vụ (Business). Trong hệ thống hai tầng thì cụm cơ sở dữ liệu ở trên cùng máy với cụm dịch vụ và cụm nghiệp vụ. Trong hệ thống ba tầng, lớp cơ sở dữ liệu trải rộng ra trên mạng nhưng chi tiết được ẩn bởi DBMS.

Nếu chúng ta dùng dạng HTML để lấy thông tin từ client tới tầng trung gian (hoặc server), thì giao diện người dùng hay vị trí mạng ít rõ ràng hơn. Trong trường hợp này thì giao diện người dùng là một phần trên client (HTML page và HTML form) và một phần trên server (như servlet và JSPs).

Các cụm phiên dịch

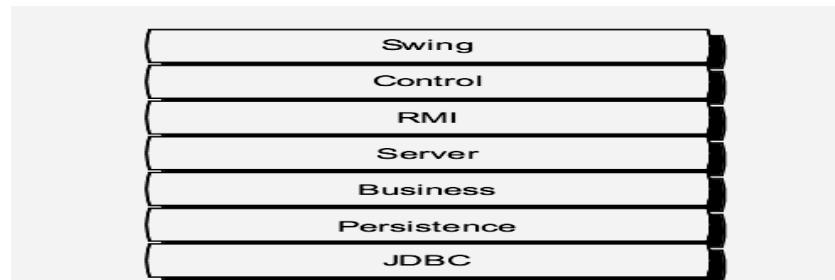
Thông thường các cụm khác nhau sẽ có các trọng tâm khác nhau. Ví dụ, khi thiết kế giao diện người dùng, chúng ta thường đề cập đến menu, dialog, notebook... Khi nói về cụm mạng ta thường quan tâm về giao thức (protocol), băng thông (bandwidth)... Trong cụm dịch vụ thì chúng ta liên quan tới tính bảo mật (security), đa luồng (multi-thread)... Trong cụm nghiệp vụ, chúng ta quan tâm nhất đến tính trừu tượng, các thuộc tính, các thao tác, đa hình, tái sử dụng và các nguyên tắc khác của mô hình hướng đối tượng. Cuối cùng, trong cụm cơ sở dữ liệu, thường ta phải giải quyết vấn đề khóa, bảng, các công nghệ cơ sở dữ liệu, các phụ thuộc hàm và các khía cạnh khác của cơ sở dữ liệu. Nếu có liên kết những phần này trực tiếp với nhau, sẽ dẫn tới công việc quá phức tạp và quá nhiều liên kết (liên kết mạnh, khi việc thực thi của một đối tượng quá phụ thuộc đối tượng khác sẽ làm mã nguồn trở nên khó bảo trì). Chúng ta có thể giảm sự phức tạp và mức độ liên kết bằng cách thêm các cụm hoạt động giống như một *cụm phiên dịch* (translation layer). Cụm phiên dịch có thể được sử dụng theo hai cách:

- Chuyển đổi cụm nghiệp vụ (trong trường hợp đơn tầng) hoặc cụm mạng (trong trường hợp đa tầng) thành các chức năng nhỏ được yêu cầu bởi người dùng đầu cuối như các *cụm điều khiển* (control). Cụm điều khiển quản lý việc giao tiếp của giao diện người dùng với các phần còn lại của hệ thống.
- Cụm phiên dịch thông thường khác được sử dụng là *cụm lưu trữ* (persistence layer) nằm giữa cụm nghiệp vụ và cụm cơ sở dữ liệu. Nó loại sự phụ thuộc của cụm nghiệp vụ vào các cơ chế lưu trữ thực sự đang sử dụng. Điều này sẽ làm cho các cơ chế lưu trữ sau này trở nên dễ dàng thay đổi hơn, ví dụ như từ tệp sang DBMS. Hình 5.6 mô tả hệ thống đa tầng với các cụm điều khiển và lưu trữ được thêm vào.



Hình 5.6: Các cụm phiên dịch trong một hệ thống đa tầng [7]

5.5.3 Ví dụ : Java Layers - Applet plus RMI



Hình 5.7: Các cụm Applet RMI [7]

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Đoạn này trình bày ví dụ minh họa một Java client với GUI. Hình 5.7 biểu diễn tập đầy đủ các cụm của hệ thống 3 tầng với giao thục mạng RMI của Java. Trong ví dụ này, cụm giao diện người dùng sử dụng thư viện Swing (thư viện GUI của Java). Phía dưới cụm giao diện người dùng là cụm điều khiển (control) chứa tất cả các mã trình dành cho việc truy cập các dịch vụ nghiệp vụ. Điều này có lợi hơn là phải được ẩn trong những đối tượng giao diện người dùng vì phải cài đặt lại khi thêm các giao diện cho thiết bị mới như iPhone. Do 95% cụm mạng trong Java được cung cấp qua RMI Framework, nên ta phải tuân theo một số luật đơn giản trong xây dựng cụm điều khiển và cụm dịch vụ để các đối tượng dịch vụ có khả năng truy cập từ bất kỳ client nào.

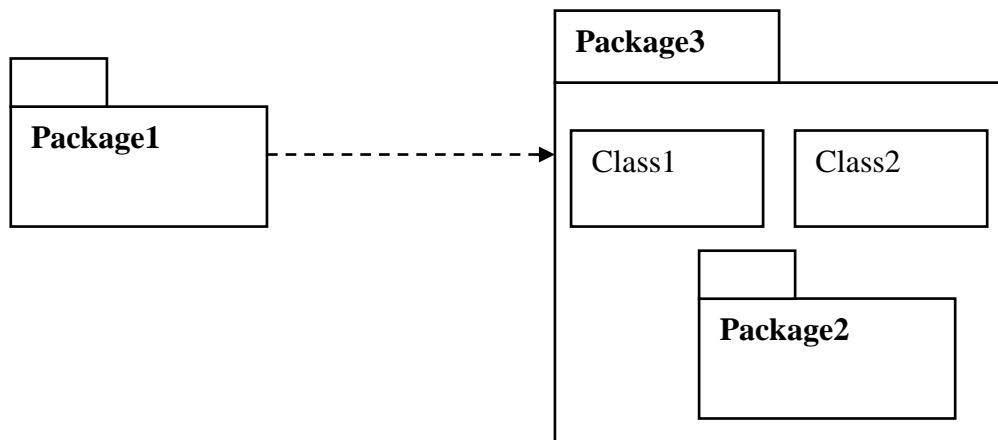
Cụm dịch vụ (Server), cụm nghiệp vụ (business) và cụm lưu trữ (persistence) giống hệt những cụm được mô tả trong những phần trước. Cụm cơ sở dữ liệu được cung cấp bởi thư viện JDBC (Java Database Connectivity). JDBC cho phép chúng ta truy cập vào bất kỳ cơ sở dữ liệu quan hệ nào. Vì bao gồm cả lớp lưu trữ, nên ta có thể thay thế JDBC bằng một cơ sở dữ liệu hướng đối tượng hoặc hệ thống file mà không ảnh hưởng đến các đối tượng nghiệp vụ.

5.6 XÂY DỰNG BIỂU ĐỒ GÓI

Khái niệm *gói* (package) trong UML cho phép chúng ta gom các lớp có liên quan với nhau (xem Chương 2). Biểu đồ gói có thể chứa các lớp hay các gói khác. Hình 5.8 chỉ ra *phụ thuộc* thể hiện bởi mũi tên không liền nét từ một gói đến một gói khác và ám chỉ rằng gói nguồn *Package1* có thể sử dụng một số các dịch vụ bên trong gói đích *Package3*. Một gói có thể sử dụng để biểu diễn:

- Một cụm (layer)
- Một hệ thống con (subsystem)
- Thư viện dùng lại (resusable library)
- Khuôn dạng (framework)
- Các lớp cần triển khai cùng nhau

Từ quan điểm lập trình, các gói có thể ánh xạ thành các cấu trúc như package trong Java hoặc namespace trong C++. Chú ý rằng các gói là khái niệm trong thời gian biên dịch và vì vậy nó tiện lợi trong tổ chức mã cho phát triển, triển khai và bảo trì.

**Hình 5.8: Biểu đồ gói trong UML**

Một số cá nhân phát triển cũng hay sử dụng biểu đồ gói để chỉ các cụm. Tuy nhiên, cách tiếp cận này được cho là có một số vấn đề:

- Một là các cụm thường được chọn trước khi quyết định cách tổ chức mã nguồn thành các gói và như vậy tổ chức các gói sau đó có thể lại khác ý định ban đầu.
- Hai là biểu đồ gói không cho phép chúng ta chỉ ra các thông tin quan trọng.

Ví dụ, sự tồn tại của cụm CGI Layer trong Hệ quản lý học theo tín chỉ là quan trọng nhưng không ánh xạ đến gói nào mà chúng ta có thể cài đặt hay mượn từ thư viện. Biểu đồ gói cũng không tốt khi thể hiện phân rã hệ thống thành các hệ thống con và khi đó biểu đồ triển khai nên được sử dụng.

5.7 KẾT LUẬN

Trong chương này chúng ta đã trình bày:

- Các bước trong thiết kế hệ thống, xem xét vấn đề lựa chọn công nghệ và cách phân rã hệ thống thành các thành phần logic và vật lý. Đặc biệt xem xét hình trạng mạng và cách phân rã phần mềm thành nhiều hệ thống, hệ thống con, cụm và các tầng. Cách biểu diễn kiến trúc bằng biểu đồ triển khai và biểu đồ gói trong UML.
- Những vấn đề đồng thời và an toàn – bảo mật nảy sinh trong mạng. Làm thế nào đảm bảo rằng thông tin được cập nhật hoàn toàn trước khi người sử dụng tác động đến nó và làm thế nào đảm bảo thông tin không được cập nhật khi đang đọc.

BÀI TẬP

1. Dựa vào kết quả có được trong Bài tập 1-3 trong Chương 4, hãy trình bày kiến trúc phân cụm và các gói cho hệ quản lý bán hàng của siêu thị máy tính.
2. Thảo luận về vấn đề tương tranh, an toàn trong hệ quản lý siêu thị này và đề xuất các quy tắc để xử lý những vấn đề này.
3. Hãy trình bày kiến trúc phân cụm và các gói cho hệ quản lý học theo tín chỉ và chỉ ra cách chọn lựa của mình.

CHƯƠNG 5. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

4. Thảo luận về vấn đề tương tranh, an toàn trong hệ quản lý học theo tín chỉ và đề xuất các quy tắc để xử lý những vấn đề này.
5. Hãy trình bày kiến trúc phân cụm và các gói cho hệ quản lý thư viện và chỉ ra cách chọn lựa của mình.
6. Thảo luận về vấn đề tương tranh, an toàn trong hệ quản lý thư viện và đề xuất các quy tắc để xử lý những vấn đề này.
7. Hãy trình bày kiến trúc phân cụm và các gói cho hệ quản lý đặt thuê xe và chỉ ra cách chọn lựa của mình.
8. Thảo luận về vấn đề tương tranh, an toàn trong hệ quản lý đặt thuê xe và đề xuất các quy tắc để xử lý những vấn đề này.
9. Hãy trình bày kiến trúc phân cụm và các gói cho hệ quản lý siêu thị mua bán sách và chỉ ra cách chọn lựa của mình.
10. Thảo luận về vấn đề tương tranh, an toàn trong hệ quản lý siêu thị mua bán sách và đề xuất các quy tắc để xử lý những vấn đề này.

CHƯƠNG 6

THIẾT KẾ CHI TIẾT

6.1 GIỚI THIỆU

Do sự khác biệt về quy mô của công việc thiết kế các hệ thống con hay thiết kế chi tiết và sự sáng tạo của từng nhóm đối với mỗi dự án, nên chúng ta không đưa ra đầy đủ hướng dẫn từng bước cho thiết kế các hệ đa tầng hướng đối tượng. Vì vậy, chương này chỉ tập trung trình bày những hoạt động chủ yếu với hy vọng tuân theo các bước này sẽ dẫn đến thiết kế các hệ phần mềm tốt hơn.

Trước hết, chúng ta cần tập trung xem xét cách thiết kế *cụm nghiệp vụ* (business layer). Các hoạt động trong thiết kế cụm này thường liên quan đến việc phải quyết định xem các đối tượng nào sẽ được đặt ở cụm này; chúng được kết nối như thế nào và giao diện của chúng sẽ là gì. Chúng ta phải biến đổi mô hình lớp hướng nghiệp vụ mà chúng ta đã phát triển trong pha phân tích thành các lớp cài đặt được theo ngôn ngữ lập trình mà chúng ta đã chọn trong pha thiết kế hệ thống. Trong tài liệu này, thiết kế chi tiết được trình bày ít nhiều độc lập với ngôn ngữ và các công nghệ được chọn. Tuy nhiên, để tiện minh họa, tài liệu này nghiêng về công nghệ Java và cơ sở dữ liệu quan hệ mà bạn đọc đã quen thuộc.

Việc thiết kế chi tiết liên quan đến việc chuyển đổi mô hình phân tích theo khái niệm thành các lớp cài đặt được theo chiến lược đặt ra trong mô hình thiết kế hệ thống. Thiết kế chi tiết thường bao gồm các hoạt động sau đây:

1. *Xây dựng mô hình lớp thiết kế*: Bước này chúng ta tập trung vào thiết kế các lớp thuộc cụm nghiệp vụ bao gồm các lớp thực thể trong miền bài toán và các lớp hỗ trợ cần thiết khác. Các lớp và trường² của cụm nghiệp vụ có thể có được bằng cách sử dụng mô hình lớp phân tích.
2. *Xây dựng lược đồ cơ sở dữ liệu*: Quyết định cách lưu trữ dữ liệu và thiết kế khuôn dạng lưu trữ. Dữ liệu lưu trữ là dữ liệu không bị mất đi khi hệ thống dừng hoạt động.
3. *Thiết kế giao diện người sử dụng*: Tiến hành xây dựng giao diện người dùng dựa trên phác thảo giao diện ban đầu được đưa ra trong pha xác định yêu cầu.

6.2 XÂY DỰNG MÔ HÌNH LỚP THIẾT KẾ

Khi chuyển từ pha phân tích sang pha thiết kế, có thể một số lớp sẽ bị loại bỏ (ví dụ như một số lớp điều khiển) và một số lớp khác sẽ được thêm vào (như các lớp để thực thi đa nhiệm hay các lớp từ các mẫu thiết kế). Người thiết kế sẽ tự quyết định thiết kế các đối tượng nghiệp vụ, biến và điều khiển sao cho có thể biến thành mã trình thực thi được. Thật ra, điều này cũng tùy thuộc vào cách tiếp cận của mô hình tiến trình mà dự án sử dụng.

Đối với mỗi lớp thiết kế đưa ra, chúng ta cần chọn tên và kiểu của các trường. Thông thường, các trường được dẫn xuất từ các thuộc tính, các quan hệ liên kết cũng như từ quan hệ thừa được tìm thấy trong suốt quá trình phân tích. Quan hệ kế thừa không nhất thiết phải

² Trong ngôn ngữ lập trình hướng đối tượng, thuật ngữ trường thông dụng hơn là thuộc tính.

được ánh xạ thành cái gì mới mà cần quyết định xem là giữ hay không giữ chúng. Như vậy, có thể có hay không có tính kế thừa trong hệ thống nhưng thường nó được đưa vào trong pha thiết kế hơn là phân tích. Lý do là phân tích nhằm phát hiện ra các thực thể nghiệp vụ còn thiết kế nhằm hướng đến tính hiệu quả trong cài đặt sau này.

6.2.1 Ánh xạ các phương thức

Khi chuyển đến giai đoạn thiết kế, chúng ta thường sử dụng các thuật ngữ *phương thức* (method) và *thông điệp* (message) trong lập trình hơn là thuật ngữ *thao tác* (operation) trong UML.

Cho đến nay, phương thức được đưa ra chỉ đơn thuần như một cách ghi lại hiện thực hóa các ca sử dụng và có thể dùng để kiểm định các lớp phân tích có hỗ trợ cài đặt hay không. Như vậy, một số phương thức gán cho các lớp trong pha phân tích có thể bị bỏ qua và một số phương thức mới lại được đưa vào cùng với các lớp mới trong pha thiết kế. Trong nhiều trường hợp, các thông điệp sẽ được thêm vào bởi một số lý do sau đây:

- Để cho phép các đối tượng client đọc hoặc thay đổi các giá trị của các trường.
- Để cho phép đối tượng client truy nhập dữ liệu dẫn xuất. Ví dụ, như một thông điệp để đọc hóa đơn các mặt hàng đăng ký mua, chúng ta cũng muốn có thể đọc được tổng hóa đơn.
- Nhờ kinh nghiệm hay trực giác nói rằng thông điệp thêm vào là có ích.
- Bởi vì khuôn dạng hay mẫu thiết kế mà chúng ta định sử dụng đòi hỏi phải có các thông điệp nào đó.

Khi thiết kế dịch vụ nghiệp vụ cho tầng giữa, chúng ta khám phá những thông điệp cho các đối tượng dịch vụ. Khi chỉ ra được cách đáp ứng dịch vụ của các đối tượng nghiệp vụ, chúng ta có khả năng thu được nhiều thông điệp hơn. Tóm lại, khi ánh xạ các lớp, các thuộc tính và các quan hệ trong pha phân tích sang pha thiết kế, các thông điệp sẽ bắt đầu xuất hiện nhiều hơn.

6.2.2 Các kiểu biến

Khi đưa vào một trường, chúng ta cần xem nó nên thuộc kiểu nào: kiểu cơ bản hay kiểu lớp. Thông thường, chúng ta hạn chế các kiểu sau đây:

- Kiểu cơ bản và kiểu lớp đơn giản mà chúng ta hy vọng tìm được trong mọi ngôn ngữ lập trình hướng đối tượng (ví dụ: int, float, boolean, String, List.....).
- Kiểu là những lớp mà chính chúng ta xây dựng.
- Những lớp có được từ các mẫu thiết kế và khuôn dạng mà chúng ta chọn sử dụng.

6.2.3 Phạm vi của các trường

Đồng thời với tên và kiểu của trường, chúng ta phải khai báo phạm vi truy nhập của chúng. Phạm vi chỉ rõ các đoạn mã trình nào cho phép đọc hay thay đổi giá trị của các trường. Có bốn cách truy nhập sau đây:

- *private* (ký hiệu dấu -) chỉ sử dụng trong phạm vi của lớp xác định.

- *package* (ký hiệu dấu ~): dùng trong phạm vi lớp xác định và tất cả những lớp cùng gói.
- *protected* (ký hiệu dấu #): dùng trong phạm vi lớp xác định, tất cả những lớp cùng gói và tất cả những lớp con của lớp định nghĩa (dù trong hay ngoài gói).
- *public* (ký hiệu dấu +): dùng bất cứ ở đâu.

Thông thường, một số ngôn ngữ cho phép người phát triển tạo ra các trường private. Ngoài lợi ích của việc đóng gói, nó sẽ làm cho chương trình biên dịch có các cơ hội tối ưu hơn. Đôi khi người phát triển sẽ tạo ra trường có phạm vi protected để người khác phát triển những lớp con có nhiều cơ hội để sửa đổi các hành vi của lớp cha (mặc dù việc làm này làm tăng sự gắn bó giữa lớp cha và lớp con). Các trường với phạm vi package là không tốt vì chúng có thể được truy nhập bởi đoạn mã trình trong gói mà không biết gì về đối tượng sở hữu nó. Đôi khi vì lý do thực tế như hiệu năng, người phát triển sẽ cho truy nhập kiểu package nhưng chỉ với ngôn ngữ lập trình nào có cách chỉ cho đọc giá trị (ví dụ, từ khóa final trong java).

Phạm vi cũng có thể áp dụng với thông điệp. Khi đó, thông điệp có thể là:

- *public*: nếu nó là phần của giao diện của package.
- *package*: nếu nó là mã cài đặt được sử dụng bởi chính lớp đó và bởi lớp trong cùng package.
- *protected*: nếu nó là mã cài đặt được sử dụng bởi chính lớp đó, lớp con và các lớp trong cùng package.
- *private*: nếu nó là mã cài đặt để sử dụng chỉ bởi chính lớp đó.

Ngoại trừ Java, không phải ngôn ngữ nào cũng hỗ trợ các kiểu truy nhập trên.

6.2.4 Các toán tử truy nhập

Có hai toán tử truy nhập: *get* trả về giá trị của trường và *set* thiết lập giá trị mới của trường. Các toán tử này làm dễ dàng cho việc bảo trì hơn và trình biên dịch dễ tối ưu hơn.

6.2.5 Ánh xạ các lớp, thuộc tính và kiểu quan hệ hợp thành

Trong UML, một số ký hiệu giống nhau được sử dụng cả cho biểu đồ lớp phân tích và biểu đồ lớp thiết kế. Khi ánh xạ sang biểu đồ lớp thiết kế, ba vấn đề chính cần phải quan tâm:

- Làm sao các lớp thiết kế phải thể hiện cài đặt được
- Thêm vào các thuộc tính và các kiểu thuộc tính
- Ánh xạ các quan hệ lớp để thể hiện được trong ngôn ngữ lập trình

Ví dụ, Hình 6.1 chỉ ra biểu đồ lớp phân tích với quan hệ hợp thành được ánh xạ thành biểu đồ lớp thiết kế. Trong trường hợp này, ta quyết định giữ lại cả hai lớp phân tích mà không hỗ trợ các lớp phụ thuộc (ngoài các lớp String). Các thuộc tính truy nhập trường vẫn nên để là private.

Trong pha phân tích, quan hệ hợp thành là song hướng: bắt đầu từ một trong hai đối tượng, chúng ta có thể dễ dàng tìm thấy các đối tượng ở đầu kia. Tuy nhiên, khi chuyển sang thiết kế, chúng ta phải đổi phó với một thực tế là các trường chỉ có thể hướng theo một chiều

tại thời gian chạy. Nghĩa là, chúng ta có thể lấy trường từ một đối tượng sở hữu tới đối tượng ở đầu kia, nhưng không là ngược lại. Vì vậy, chúng ta cần quyết định xem chúng ta có muốn có một trường ở mỗi đầu của mối quan hệ hoặc chỉ ở một đầu và nếu vậy thì đầu nào.

Chúng ta mong quan hệ hợp thành ban đầu là từ cái hợp thành đến cái được hợp thành. Điều này phản ánh một thực tế là cái được hợp thành là chủ sở hữu của mối quan hệ và như vậy cái hợp thành tương tự như một thuộc tính. Nghĩa là, đối tượng được hợp thành sử dụng dịch vụ của các thuộc tính hoặc các đối tượng hợp thành, nhưng không theo chiều ngược lại. Điều này cũng không tuyệt đối, nếu cảm thấy phù hợp với hệ thống cụ thể thì chúng ta có thể đặt một trường bên trong đối tượng được hợp hay bên trong cả được hợp và hợp thành. Sau khi đã quyết định hướng của hợp thành, chúng ta có thể thêm một đầu mũi tên cho biểu đồ lớp để chỉ ra cách hợp thành ánh xạ tới một trường. Trong thời gian chạy, các thông điệp chỉ có thể theo hướng mũi tên. Như vậy, ta có thể bỏ hợp thành và biểu diễn hợp thành như thuộc tính bên trong được hợp thành.

Ví dụ: Quan hệ hợp thành của lớp người Person và lớp họ tên Name (Hình 6.1), ta thêm thuộc tính *name:Name* bên trong *Person*.

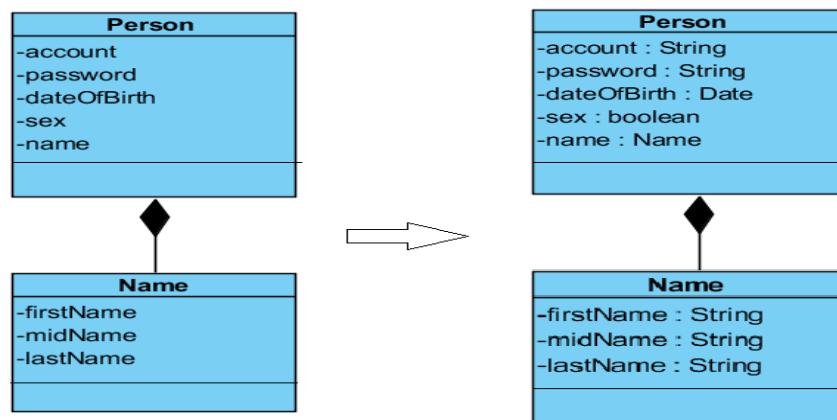


Figure 6.1: Ánh xạ quan hệ hợp thành

6.2.6 Ánh xạ các kiểu quan hệ khác

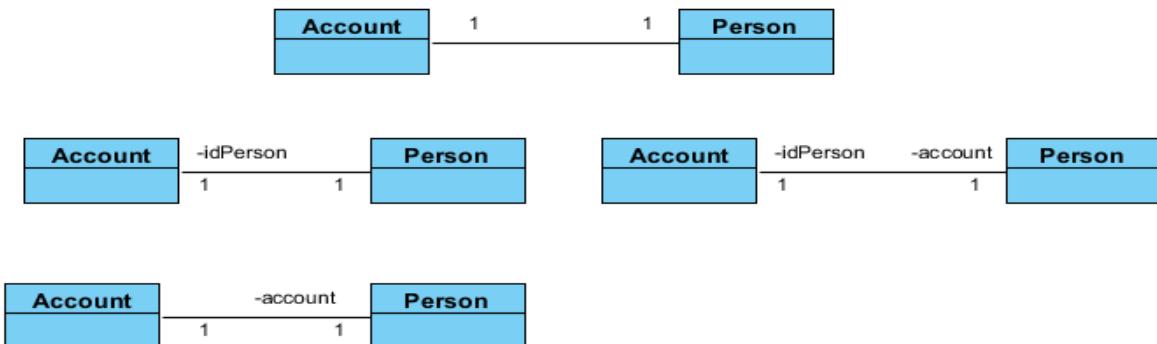
Như đã trình bày trong Chương 4, có ba loại quan hệ giữa các lớp chính: liên kết, kết hợp và hợp thành. Do mục đích là ánh xạ sang thiết kế, chúng ta không cần phân biệt giữa liên kết và kết hợp vì các ngôn ngữ lập trình hướng đối tượng không có sự phân biệt này. Vì vậy, thuật ngữ liên kết sẽ được sử dụng cho phần còn lại của phần này.

Đa số liên kết mà chúng ta quyết định giữ lại từ phân tích, cùng với bất kỳ liên kết nào mà chúng ta thêm vào khi thiết kế thì chỉ như các trường trong các lớp. Vì các trường chỉ cho phép chuyển theo một hướng nên chúng ta cần phải quyết định xem chúng ta muốn đi theo cả hai hướng hay một hướng. Cách chúng ta cài đặt liên kết phụ thuộc vào tính nhiều ở mỗi điểm cuối: 1-1, 1-n hay m-m.

Liên kết kiểu 1-1

CHƯƠNG 6. THIẾT KẾ CHI TIẾT

Hãy xem phần biểu đồ lớp phân tích ở Hình 6.2. Có ba cách cài đặt kiểu liên kết này: đặt trường *account* vào trong *Person*; đặt một trường gọi *idPerson* vào trong *Account* hoặc kết hợp cả hai. Như vậy, ta có thể ánh xạ liên kết thành một chiều hay hai chiều.

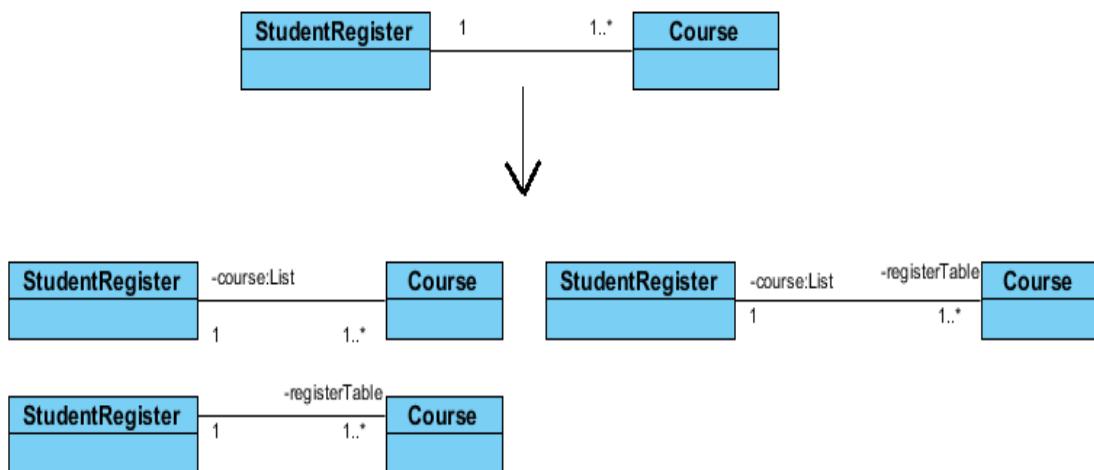


Hình 6.2: Ánh xạ liên kết 1-1

Nếu theo cách hai chiều (two-way), chúng ta cần thêm mã trình để đảm bảo rằng sự liên kết không làm thay đổi cách đi. Ví dụ, không có ý nghĩa khi thay đổi trường *idPerson* của *Account* để trả tới *Person* khác nếu chúng ta không thông báo cho *Person* ban đầu rằng *account* của nó đã thay đổi (có thể giá trị null). Tất nhiên có thể thực hiện sự đồng bộ hóa này bằng cách sử dụng các phương thức thích hợp trong *Person* và *Account* nhưng như thế thì hơi vung về. Vì vậy, phần lớn chúng ta nên chọn một chiều (one-way). Trong trường hợp này, nó có vẻ thích hợp để *Person* tham chiếu tới *Account* nhưng không phải là ngược lại.

Liên kết kiểu 1-n

Hình 6.3 biêt diêt một liên kết kiểu 1-n. Với trường hợp này, chúng ta phải quyết định có nên đặt một trường trong một đầu hoặc cả hai và những đối số tương tự cung cấp như trong trường hợp 1-1.



Hình 6.3: Ánh xạ liên kết kiểu 1-n

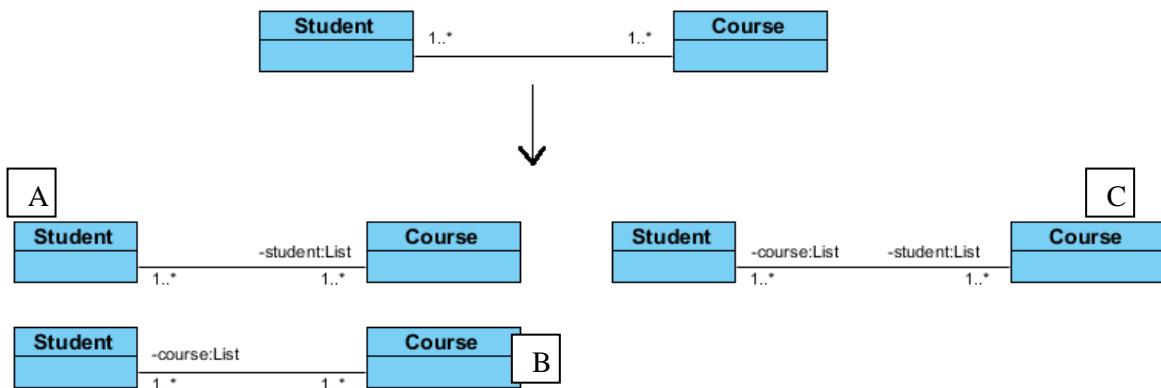
Tuy nhiên, không giống như trường hợp 1-1, nếu chúng ta quyết định đặt một trường vào mỗi đầu cuối, chúng ta phải chuẩn bị để lưu trữ nhiều hơn một đối tượng liên quan.

Ví dụ, nếu chúng ta thêm trường *course*: *Course* vào *StudentRegister*, thì trường này sẽ phải lưu một hoặc nhiều đối tượng *Course*. Như vậy *StudentRegister* cần phải sử dụng một số loại lớp List, Collection hay Array để tổ chức các đối tượng thành viên của nó.

Trong Hình 6.3, một List đã được sử dụng. Một vấn đề tinh tế cần nêu bàn ở đây là có thể lựa chọn Collection giống như Set. Tuy nhiên, khi thêm một đối tượng tới một Set, phần lớn chương trình sẽ kiểm tra xem các đối tượng đã có chưa để nó không xuất hiện hai lần và sẽ dẫn đến “chi phí” tính toán đắt. Vì vậy, nếu chắc rằng sẽ không phải thêm một đối tượng hai lần, ta nên sử dụng Bag. Nếu ngôn ngữ ta sử dụng (ví dụ như Java) không cung cấp kiểu như Bag thì ta sử dụng List.

Liên kết kiểu n-m

Đây là trường hợp phức tạp nhất. Hãy xem xét ví dụ biểu diễn trong Hình 6.4. Ở đây, chúng ta có một đối tượng *s:Student* có thể thuộc nhiều đối tượng *Course* và ngược lại. Xét ba trường hợp: A, B và C. Thường với liên kết nhiều-nhiều n-m thì thực sự không có chủ sở hữu của sự liên kết hợp trong trường hợp này. Nếu chúng ta quyết định bắt đầu với một *Student* và sau đó chuyển hướng tới các đối tượng *Course* của nó, lựa chọn A sẽ là tốt. Ngược lại, nếu chúng ta quyết định bắt đầu với một *Course*, B sẽ là lựa chọn tốt hơn. Nếu chúng ta quyết định rằng có vẻ phù hợp như bắt đầu với một *Course* hoặc một *Student*, chúng ta thực sự nên lựa chọn C.

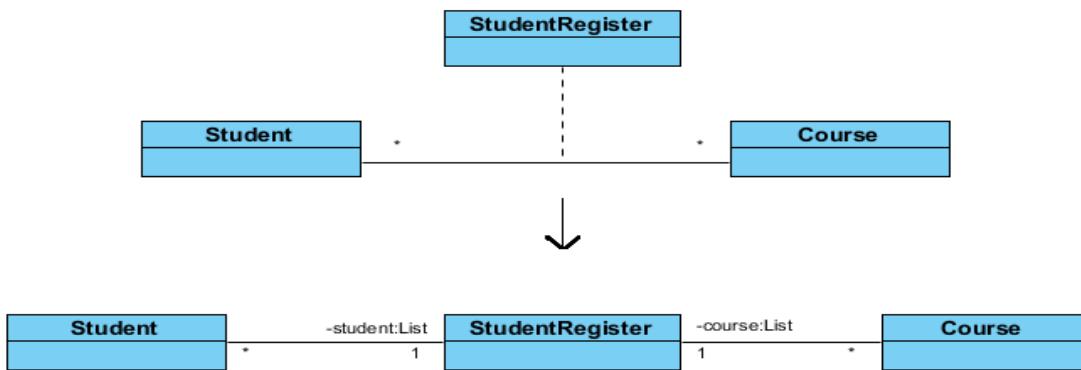


Hình 6.4: Ánh xạ liên kết n-m

Tuy nhiên với lựa chọn C, vấn đề đồng bộ hóa trở nên tồi tệ hơn so với các trường hợp 1 – 1 và 1 – n. Bởi vì chúng ta sẽ phải tìm các đối tượng thông qua List mà không truy nhập trực tiếp. Trong trường hợp như vậy, nên sử dụng lớp liên kết (association class) để giải quyết sự phức tạp này.

Các lớp liên kết

Chúng ta đã gặp các lớp liên kết trong phần phân tích. Một lớp liên kết là cần thiết nếu có dữ liệu liên quan tới một liên kết. Hình 6.5 biểu diễn một ví dụ của một lớp liên kết.

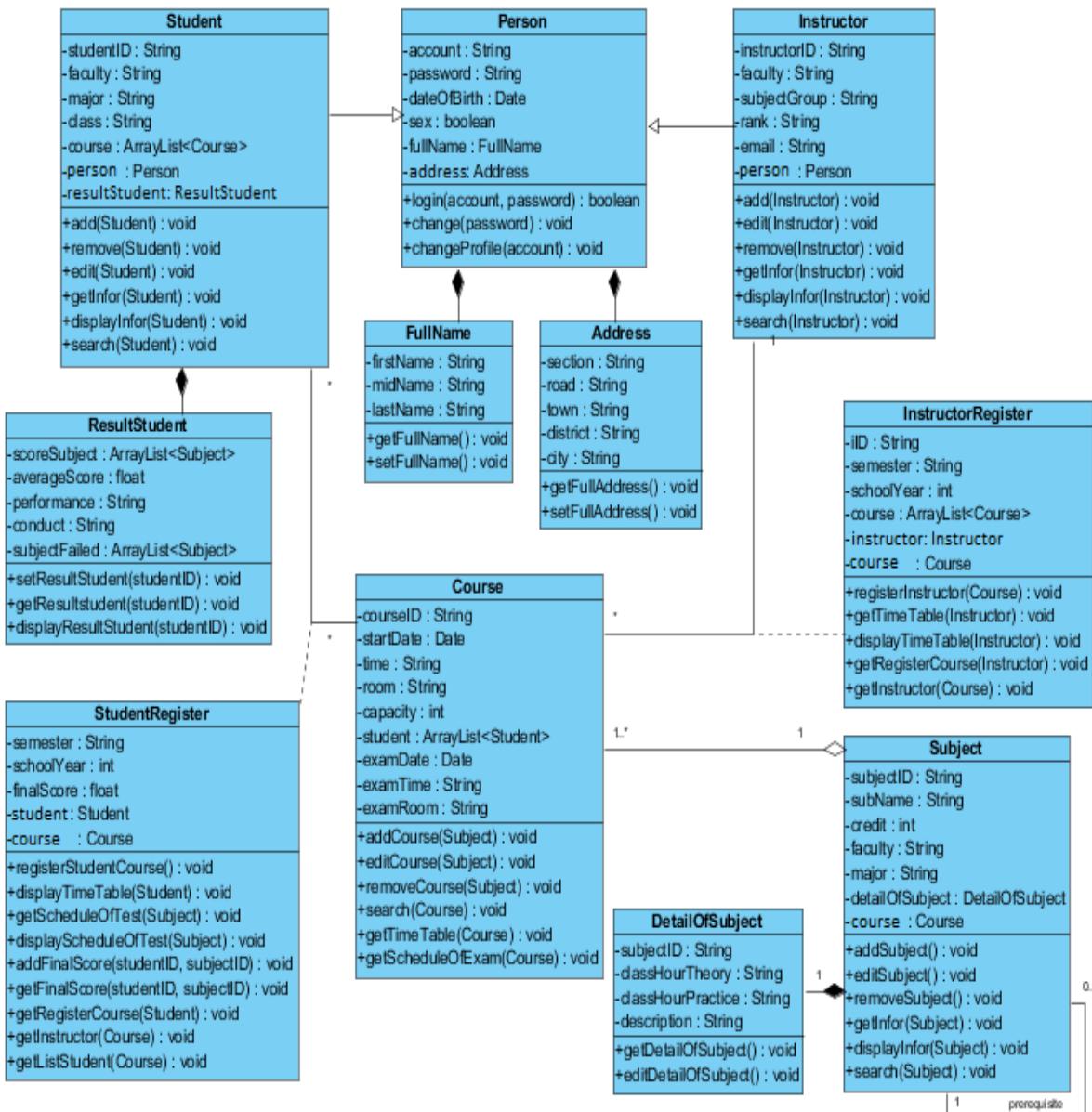
**Hình 6.5: Ánh xạ lớp liên kết**

Từ biểu đồ này, chúng ta biết các đối tượng sinh viên *Student* có thể đăng ký một số đối tượng môn học *Course* và các đối tượng môn học *Course* có thể được đăng ký bởi một số đối tượng *Student*. Nhưng với mỗi liên kết *StudentRegister*, ta cần ghi lại ngày *date* và *trạng thái* (đã được xác nhận hay chưa) đăng ký. Các dữ liệu này không thuộc về *Student* hoặc *Course*, mà thuộc về lớp liên kết. Do đó, chúng ta đưa ra *StudentRegister* như một lớp liên kết với các thuộc tính thích hợp.

Khi thiết kế một lớp liên kết, cách đơn giản nhất là tạo một lớp thiết kế với các trường cho tất cả các thuộc tính, và thêm hai trường để trả tới các đối tượng được biểu diễn ở giữa của Hình 6.5.

Nếu chúng ta muốn chuyển hướng từ *Student* tới các đối tượng *StudentRegister* của nó, chúng ta sẽ phải thêm một trường cho *Student*. Điều này sẽ tạo nên vấn đề đồng bộ hóa mà ta đã trình bày ở trên. Lớp liên kết là cách tổng quát nhất để ánh xạ liên kết từ phân tích tới thiết kế. Vì vậy chúng ta có thể đưa ra một lớp liên kết để biểu diễn mọi liên kết từ biểu đồ lớp phân tích. Điều này sẽ giảm được vấn đề đồng bộ hóa như chúng ta đã trình bày.

Ví dụ: Biểu đồ cụm lớp nghiệp vụ Hệ quản lý học tín chỉ đã được ánh xạ thành các lớp thiết kế để có thể code trong bất kỳ ngôn ngữ lập trình hướng đối tượng nào (Hình 6.6).



Hình 6.6: Các lớp cụm nghiệp vụ cho Hệ quản lý học tín chỉ

6.3 XÂY DỰNG LUẬQ C ĐỒ CƠ SỞ DỮ LIỆU

6.3.1 Các hệ quản trị cơ sở dữ liệu

Trong hơn bốn thập kỷ qua, nhiều Hệ quản trị cơ sở dữ liệu (DBMS: Database Management System) đã được phát triển như Hệ cơ sở dữ liệu quan hệ, Hệ cơ sở dữ liệu hướng đối tượng... Có một sự khác biệt giữa các ngôn ngữ lập trình mà chúng ta sử dụng để viết mã và cách chúng ta truy nhập cơ sở dữ liệu. Do đó, chúng ta cần phải thực hiện một vài kiểu ánh xạ theo thời gian chạy giữa các hệ thống phần mềm và hệ cơ sở dữ liệu. Một số công cụ như EJB của J2EE sẽ tạo ra mã ánh xạ cho chúng ta. Tuy nhiên, nếu muốn sử dụng hiệu quả các công cụ như vậy, chúng ta vẫn cần phải hiểu rõ các nguyên lý cơ bản. Cơ sở dữ liệu quan hệ thường được sử dụng để lưu trữ dữ liệu trong một hệ đa tầng trên Internet vì một số lý do sau đây:

- Cơ sở dữ liệu quan hệ là quen thuộc nhất. Mặc dù có nhiều cơ sở dữ liệu hướng đối tượng, nhưng chúng chưa được sử dụng rộng rãi, đặc biệt trong các hệ quản lý doanh nghiệp.
- Tất cả cơ sở dữ liệu quan hệ đều dựa trên mô hình toán học nên chúng xem là thống nhất trong nhiều dạng của DBMS.
- Tất cả các hệ cơ sở dữ liệu quan hệ trong thương mại đều hỗ trợ ngôn ngữ SQL (SQL: Structured Query Language).
- Ngôn ngữ lập trình Java cung cấp một thư viện hỗ trợ JDBC (Java Database Connectivity) kết nối tới cơ sở dữ liệu quan hệ.

6.3.2 Mô hình quan hệ

Mô hình quan hệ là một mô hình toán học có độ tin cậy cao và dễ dàng tối ưu. Tuy nhiên, không giống như mô hình mạng và mô hình phân cấp, mô hình quan hệ không có các liên kết vật lý. Tất cả các dữ liệu được chứa trong các bảng, các cột. Dữ liệu trong hai bảng được quan hệ với nhau thông qua các cột thay cho liên kết vật lý. Mặc dù mô hình hướng đối tượng có thể ánh xạ dễ dàng thành mô hình quan hệ nhưng sẽ khó để dự đoán ánh xạ nào là hiệu quả nhất cho hệ phần mềm của chúng ta. Do đó, cần phải có thử nghiệm trên nhiều ánh xạ để tìm ánh xạ thích hợp nhất cho hệ thống của chúng ta.

Trong phần này chúng ta xem xét một cách ánh xạ trực tiếp từ mô hình lớp thiết kế. Các ánh xạ này cũng đủ để nói rằng các kỹ thuật cơ bản liên quan đến sử dụng câu lệnh SQL để đọc dữ liệu từ CSDL thành đối tượng vào thời gian chạy và sau đó ghi dữ liệu mới vào CSDL.

Bảng

Mô hình quan hệ dựa trên các *bảng* dữ liệu chứa các *dòng* và các *cột*, mỗi cột chứa một kiểu riêng biệt. Chuẩn SQL định nghĩa một số kiểu mà ta có thể lựa chọn:

- VARCHAR(X): một chuỗi có thể nhiều nhất là X ký tự
- INTEGER: một kiểu số nguyên thường là 32 bit
- DATE: kiểu ngày tháng trong lịch
- TIMESTAMP: một sự kết hợp của ngày và thời gian trong ngày
- BOOLEAN: true hay false

Khóa

Một khóa là một giá trị hay bộ giá trị định danh duy nhất một dòng cho mỗi thực thể. Một số bảng chứa tập khóa tự nhiên, ví dụ Sinh viên có mã sinh viên. Một số bảng cần đưa ra khóa, ví dụ khách hàng cần có khóa để định danh người duy nhất mua số sản phẩm nào đó. Một khóa có thể kết hợp nhiều giá trị.

Ánh xạ mô hình đối tượng thành mô hình quan hệ bảng

Khi ánh xạ mô hình đối tượng thành Bảng, ta có thể bắt đầu hoặc với biểu đồ lớp phân tích hoặc biểu đồ lớp thiết kế. Biểu đồ lớp phân tích gần gũi với mô hình quan hệ Bảng hơn vì nó không chỉ ra hướng liên kết. Tuy nhiên, biểu đồ lớp thiết kế có kiểu gán cho các trường nên

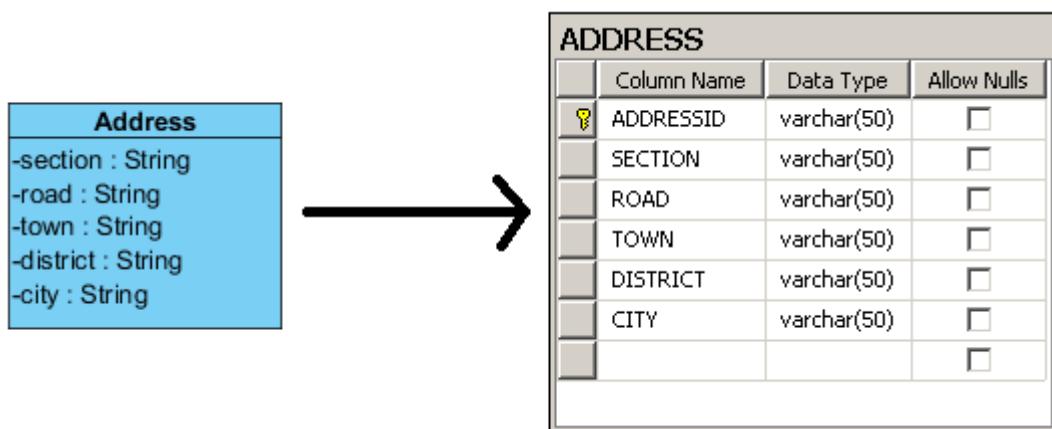
tiện lợi khi thiết kế bảng. Để giải quyết vấn đề này, chúng ta sẽ sử dụng mô hình phân tích để thiết kế các bảng và lấy kiểu từ mô hình thiết kế.

6.3.3 Ánh xạ các lớp thực thể

Để ánh xạ một thực thể (đối tượng nghiệp vụ) từ mô hình hướng đối tượng thành lược đồ quan hệ, ta đưa ra bảng có cùng tên như lớp thực thể, trong đó mỗi dòng biểu diễn một đối tượng duy nhất của miền nghiệp vụ.

Với trường đơn giản (kiểu cơ bản hay String), ta thêm một cột vào bảng với cùng tên như trường và kiểu dữ liệu thích hợp SQL. Các trường có kiểu đối tượng phải được xem xét theo cách khác mà ta sẽ trình bày trong mục tiếp theo.

Để tiện cho lập trình hướng đối tượng, ta đưa ra một thuộc tính ID có thẻ kiểu nguyên để làm khóa chính. Lợi ích của việc thêm ID là nó làm đơn giản việc code và làm cho việc đọc các đối tượng hoặc chuyển từ đối tượng này sang đối tượng khác dễ dàng và hiệu quả hơn. Do đó, chúng ta sử dụng cột ID cho các bảng được xây dựng trong chương này dù biểu đồ lớp tương ứng không có thuộc tính này (xem Hình 6.11).



Hình 6.11: Lược đồ quan hệ tương ứng lớp thực thể

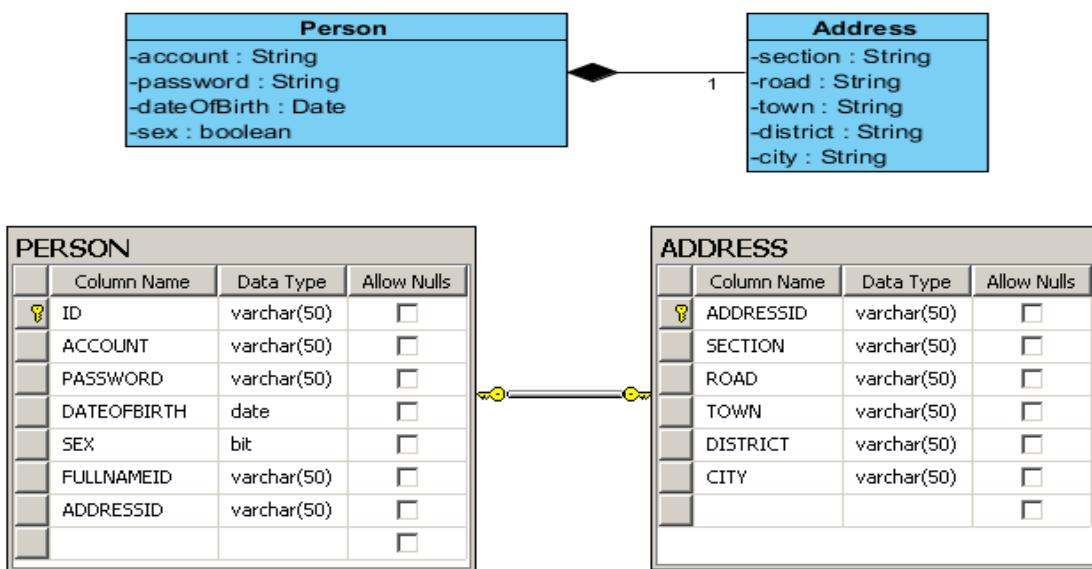
6.3.4 Ánh xạ các liên kết

Như trình bày trong phần trước, khi ánh xạ từ mô hình lớp phân tích thành mô hình lớp thiết kế, ta phải chuyển đổi hai chiều trong phân tích thành liên kết một chiều. Tuy nhiên, vì cơ sở dữ liệu quan hệ lưu trữ trực tiếp liên kết hai chiều nên chúng ta không gặp phải vấn đề như khi mô hình lớp thiết kế. Ví dụ, nếu lược đồ CSDL cho phép liên kết từ thực thể A đến thực thể B, thì DBMS và ngôn ngữ truy vấn sẽ cho phép ta liên kết từ B đến A. Do đó, lược đồ cơ sở dữ liệu của ta giống với mô hình lớp phân tích hơn là mô hình lớp thiết kế. Sau đây, chúng ta sẽ xem xét chi tiết cách ánh xạ các mô hình lớp thành các lược đồ cơ sở dữ liệu.

Liên kết 1-1

Với liên kết 1-1 chúng ta có thể thêm một khóa ngoại vào một trong các bảng thực thể. Một khóa ngoại là một thực thể trong một bảng tham chiếu đến khóa chính của một bảng khác. Nghĩa là, nó là một tham chiếu từ một dòng trong một bảng đến một dòng trong một bảng khác.

Trong Hình 6.12, Bảng PERSON chứa khóa ngoại ADDRESSID cho phép chúng ta tìm chi tiết của PERSON trong bảng ADDRESS. Chúng ta cũng có thể thêm khóa ngoại PERSONID vào bảng ADDRESS nhưng do tính chất hai chiều của cơ sở dữ liệu quan hệ, ta không cần đưa khóa ngoại vào hai bảng. Một hướng giải quyết khác là thay thế khóa ngoại bằng cách gộp hai bảng thành một bảng nhưng điều này chỉ nên thực hiện khi một bảng không được biểu diễn như một thực thể đúng nghĩa.

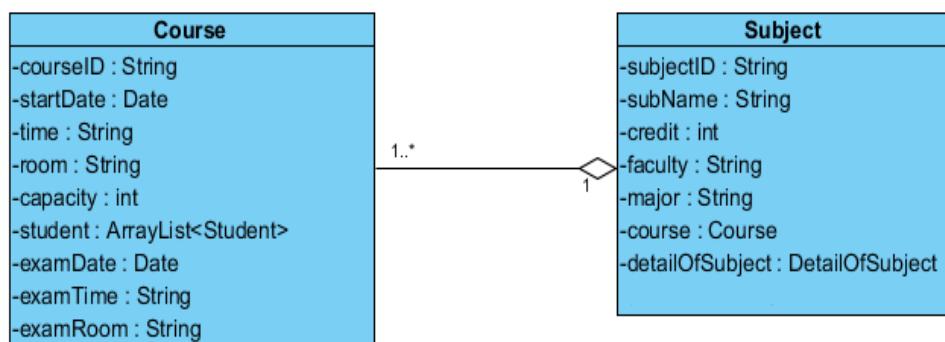


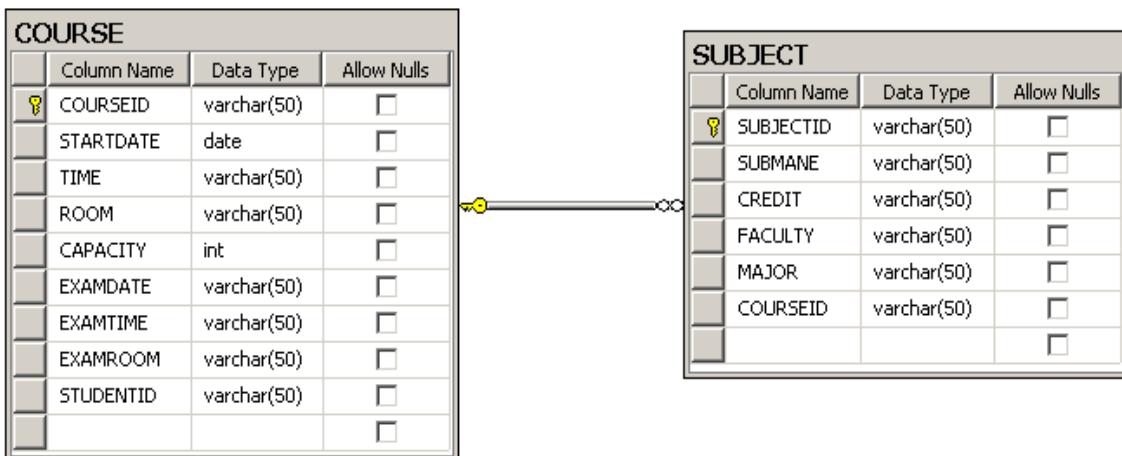
Hình 6.12: Ánh xạ liên kết 1-1 thành khóa ngoại

Trong trường hợp có một liên kết tùy chọn như tính nhiều, 1 và 0..1, chúng ta có thể thêm một khóa ngoại vào tùy chọn. Cơ sở dữ liệu quan hệ cũng hỗ trợ cột có thuộc tính null nhưng để đơn giản tốt hơn nên tránh dùng.

Liên kết 1-n

Với liên kết 1-n, chúng ta có thể thêm một khóa ngoại vào bảng “nhiều” như chỉ ra trong Hình 6.13. Vì một Subject có thể đảm nhiệm nhiều Course nên khóa courseID được thêm vào bảng Subject.





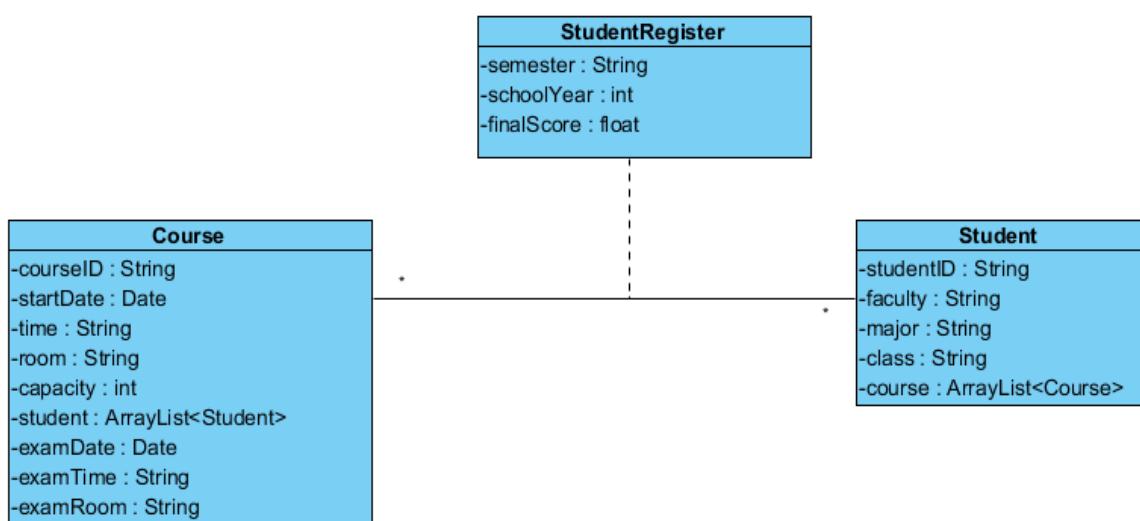
Hình 6.13: Ánh xạ quan hệ 1-n đến khóa ngoại

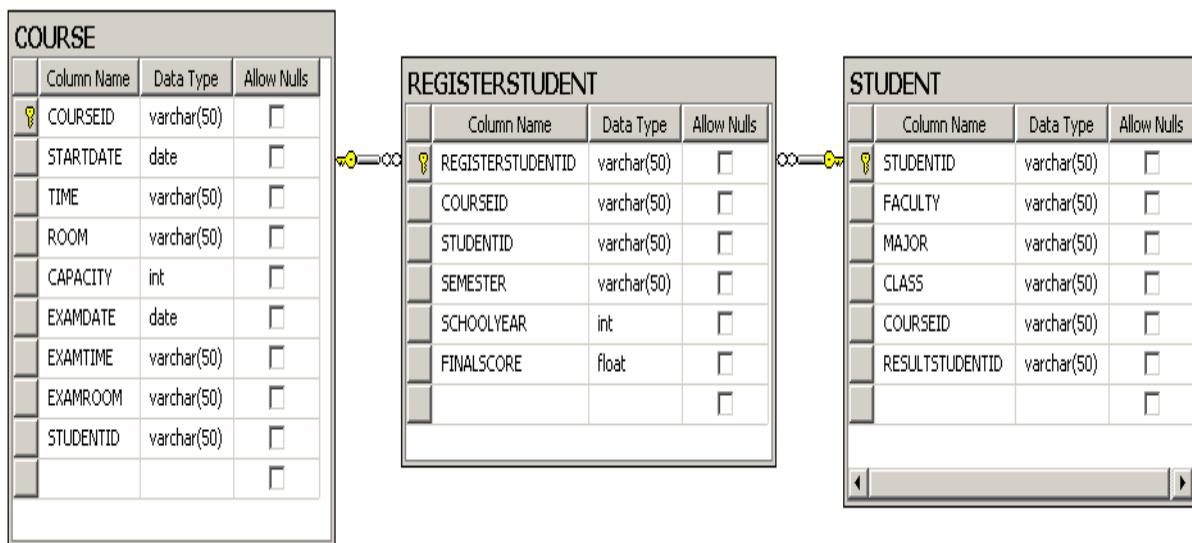
Liên kết n-m

Đối với quan hệ nhiều-nhiều, một khóa ngoại là chưa đủ để định danh nhiều thực thể ở mỗi đầu liên kết. Cách giải quyết là sử dụng một bảng liên kết, trong đó mỗi dòng trong bảng liên kết biểu diễn một kết nối giữa một thực thể trong một bảng và một thực thể trong bảng khác. Bảng liên kết có khoá chính là hợp của hai khoá ngoại ánh xạ đến bảng. Chúng ta có thể sử dụng bảng liên kết để lưu trữ quan hệ một - một, một - nhiều. Quan hệ n-m có thể xem là trường hợp đặc biệt của các lớp liên kết được trình bày tiếp theo sau.

Các lớp liên kết

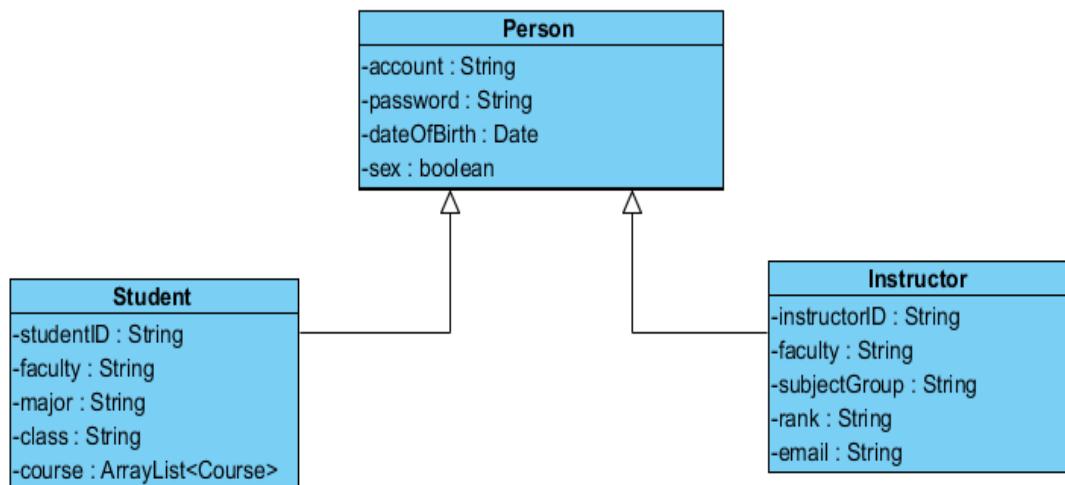
Các lớp liên kết có dữ liệu riêng nên cần được ánh xạ thành bảng liên kết. Bảng biểu diễn lớp liên kết có cột chứa thuộc tính của nó và có thể có một cột ID làm khoá chính nếu lớp liên kết biểu diễn một thực thể. Hình 6.14 biểu diễn bảng cho lớp liên kết StudentRegister. Bảng STUDENTREGISTER có khoá chính cho nó và hai khóa ngoại STUDENTID và COURSEID.





Hình 6.14: Ánh xạ lớp liên kết đến khóa ngoại

6.3.5 Ánh xạ kế thừa



Hình 6.15: Cây phân cấp lớp kế thừa

Có ba cách để ánh xạ quan hệ kế thừa sang cơ sở dữ liệu quan hệ.

Cách 1: Sử dụng một bảng cho lớp thực thể cha và tất cả các thuộc tính của lớp con được lưu trữ trong một bảng.

PERSON			
	Column Name	Data Type	Allow Nulls
KEY	ID	varchar(50)	<input type="checkbox"/>
	ACCOUNT	varchar(50)	<input type="checkbox"/>
	PASSWORD	varchar(50)	<input type="checkbox"/>
	DATEOFBIRTH	date	<input type="checkbox"/>
	SEX	bit	<input type="checkbox"/>
	FACULTY	varchar(50)	<input type="checkbox"/>
	MAJOR	varchar(50)	<input type="checkbox"/>
	CLASS	varchar(50)	<input type="checkbox"/>
	COURSEID	varchar(50)	<input type="checkbox"/>
	SUBJECTGROUP	varchar(50)	<input type="checkbox"/>
	RANK	varchar(50)	<input type="checkbox"/>
	EMAIL	varchar(50)	<input type="checkbox"/>
	FULLNAMEID	varchar(50)	<input type="checkbox"/>
	ADDRESSID	varchar(50)	<input type="checkbox"/>
			<input type="checkbox"/>
			<input type="checkbox"/>

Hình 6.16: Ánh xạ thành một lớp

Cách này có ưu điểm là đơn giản và tính đa hình được hỗ trợ khi một người có thể thay đổi vai trò hoặc có nhiều vai trò (ví dụ 1 người có thể vừa là giảng viên vừa là nghiên cứu sinh, một người có thể vừa là nhân viên vừa là khách hàng...). Tuy nhiên, nó có nhược điểm là mỗi lần thêm một thuộc tính mới vào bất kì bảng nào trong quan hệ kế thừa đều phải thêm một cột vào bảng và do đó dễ tăng sự trùng lặp trong phân cấp lớp.

Cách 2: Sử dụng một bảng cho một lớp con cụ thể, mỗi bảng chứa cả thuộc tính của lớp cha và của chính chúng (Hình 6.17)

STUDENT

	Column Name	Data Type	Allow Nulls
1	STUDENTID	varchar(50)	<input type="checkbox"/>
2	FACULTY	varchar(50)	<input type="checkbox"/>
3	MAJOR	varchar(50)	<input type="checkbox"/>
4	CLASS	varchar(50)	<input type="checkbox"/>
5	COURSEID	varchar(50)	<input type="checkbox"/>
6	RESULTSTUDENTID	varchar(50)	<input type="checkbox"/>
7	ACCOUNT	varchar(50)	<input type="checkbox"/>
8	PASSWORD	varchar(50)	<input type="checkbox"/>
9	DATEOFBIRTH	date	<input type="checkbox"/>
10	SEX	bit	<input type="checkbox"/>
11	FULLNAMEID	varchar(50)	<input type="checkbox"/>
12	ADDRESSID	varchar(50)	<input type="checkbox"/>
13			
14			
15			
16			
17			
18			
19			
20			

INSTRUCTOR

	Column Name	Data Type	Allow Nulls
1	INSTRUCTORID	varchar(50)	<input type="checkbox"/>
2	FACULTY	varchar(50)	<input type="checkbox"/>
3	SUBJECTGROUP	varchar(50)	<input type="checkbox"/>
4	RANK	varchar(50)	<input type="checkbox"/>
5	EMAIL	varchar(50)	<input type="checkbox"/>
6	ACCOUNT	varchar(50)	<input type="checkbox"/>
7	PASSWORD	varchar(50)	<input type="checkbox"/>
8	DATEOFBIRTH	date	<input type="checkbox"/>
9	SEX	bit	<input type="checkbox"/>
10	FULLNAMEID	varchar(50)	<input type="checkbox"/>
11	ADDRESSID	varchar(50)	<input type="checkbox"/>
12			
13			
14			
15			
16			
17			
18			
19			
20			

Hình 6.17: Ánh xạ thành 2 lớp

Cách này có lợi điểm là dữ liệu của đối tượng chỉ lưu trữ trong một bảng, không phải di chuyển qua các bảng để lấy dữ liệu. Tuy nhiên, có nhược điểm là khi thêm một thuộc tính vào lớp cha thì phải sửa ở tất cả các bảng là lớp con của nó.

Cách 3: Sử dụng mỗi bảng cho mỗi lớp (Hình 6.18). Khóa chính của lớp cha cũng sẽ là khóa chính cho cả hai lớp con. Nghĩa là những đối tượng vừa thuộc lớp cha Person và lớp Instructor sẽ có khóa giống nhau và vừa thuộc lớp cha Person và Student sẽ có khóa giống nhau. Lợi điểm của cách này là thoả mãn khái niệm hướng đối tượng nhất và hỗ trợ tính đa hình, hơn nữa dễ dàng sửa lớp cha và thêm các lớp con mới. Tuy nhiên, nhược điểm là có quá nhiều bảng và mất nhiều thời gian để đọc và ghi dữ liệu do phải xử lý trên nhiều bảng.

PERSON

	Column Name	Data Type	Allow Nulls
key	ID	varchar(50)	<input type="checkbox"/>
	ACCOUNT	varchar(50)	<input type="checkbox"/>
	PASSWORD	varchar(50)	<input type="checkbox"/>
	DATEOFBIRTH	date	<input type="checkbox"/>
	SEX	bit	<input type="checkbox"/>
	FULLNAMEID	varchar(50)	<input type="checkbox"/>
	ADDRESSID	varchar(50)	<input type="checkbox"/>

INSTRUCTOR

	Column Name	Data Type	Allow Nulls
key	INSTRUCTORID	varchar(50)	<input type="checkbox"/>
	FACULTY	varchar(50)	<input type="checkbox"/>
	SUBJECTGROUP	varchar(50)	<input type="checkbox"/>
	RANK	varchar(50)	<input type="checkbox"/>
	EMAIL	varchar(50)	<input type="checkbox"/>

STUDENT

	Column Name	Data Type	Allow Nulls
key	STUDENTID	varchar(50)	<input type="checkbox"/>
	FACULTY	varchar(50)	<input type="checkbox"/>
	MAJOR	varchar(50)	<input type="checkbox"/>
	CLASS	varchar(50)	<input type="checkbox"/>
	COURSEID	varchar(50)	<input type="checkbox"/>
	RESULTSTUDENTID	varchar(50)	<input type="checkbox"/>

Hình 6.18: Ánh xạ thành các bảng tương ứng

6.4 THIẾT KẾ GIAO DIỆN NGƯỜI SỬ DỤNG

Phần này dành xem xét việc thiết kế giao diện người sử dụng. Trong Chương xác định yêu cầu, chúng ta đã đề ra các phác thảo để làm sáng rõ thêm các yêu cầu hệ thống. Phần này chỉ trình bày một số gợi ý và lời khuyên để tạo ra các giao diện tốt và đơn giản cho khách hàng.

Ngay từ giai đoạn đầu của quá trình phát triển, khi tiến hành tìm hiểu yêu cầu và phân tích bài toán, việc xem xét các chức năng trong giao diện người dùng là rất hữu ích. Với một phương pháp luận hướng ca sử dụng và hầu hết các tác nhân đều là con người, thì cách thức các tác nhân tương tác với hệ thống là vấn đề quan trọng nhất. Cho đến bây giờ, chúng ta đã có những phác thảo giao diện người dùng và các đối tượng biên trong biểu đồ giao tiếp. Các phác thảo giao diện nhằm giúp chúng ta tạo ra các ca sử dụng hệ thống trong suốt quá trình xác định yêu cầu và ngược lại. Trong quá trình phân tích động, chúng ta đã sử dụng các biểu đồ giao tiếp để thể hiện cách thức thực hiện các ca sử dụng. Trong các biểu đồ này, các tác nhân được hiển thị tương tác với hệ thống thông qua các đối tượng biên.

Tuy nhiên, chúng ta vẫn cần thiết kế các giao diện người dùng. Chúng ta phải dựa vào các đối tượng biến, các phác thảo giao diện người dùng và các ca sử dụng để biến chúng thành các mô tả giao diện người dùng có thể cài đặt trực tiếp. Có hai lý do chính cho điều này:

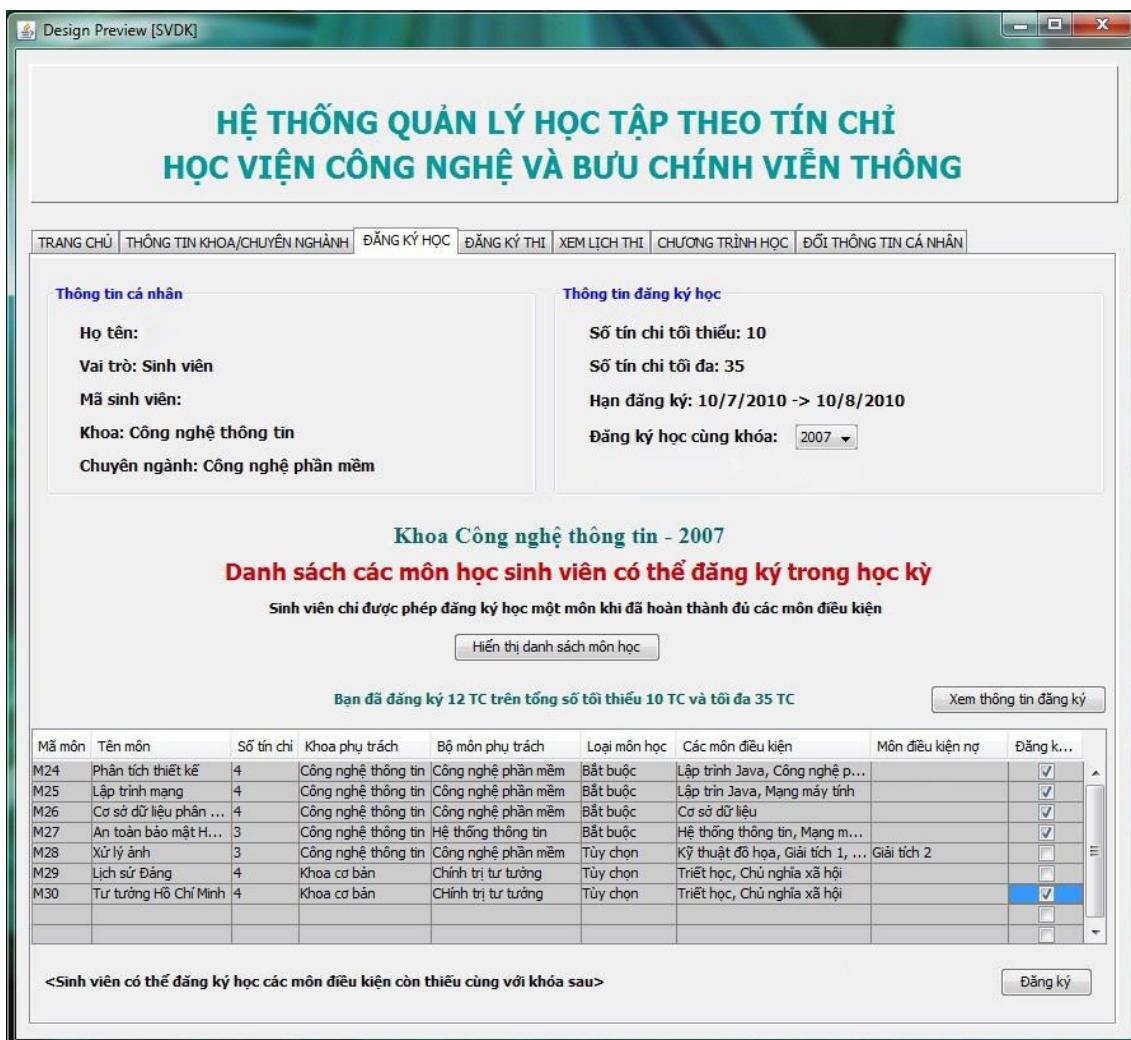
- Các hành vi đúng của hệ thống phụ thuộc vào cấu trúc bên trong của nó chứ không phải cách con người tương tác với nó. Tương tự như việc một chiếc xe bao gồm một động cơ, bốn bánh xe và một khung xe. Tùy thuộc vào giao diện và cách chúng ta sử dụng, chúng ta có thể có một chiếc xe chờ người, hoặc một chiếc xe kéo, hay một chiếc xe úi. Dù chọn cách sử dụng nào, nó vẫn là một chiếc xe.
- Chúng ta muốn viết những đoạn mã có thể tái sử dụng. Nếu chúng ta tập trung vào một tập các giao diện cụ thể, chúng ta có nguy cơ sẽ tạo ra một hệ thống chỉ làm việc với những giao diện này. Đây là một trong những vấn đề trong phát triển phần mềm truyền thống và hướng ca sử dụng cũng có vẻ như đi ngược lại nguyên tắc tái sử dụng. Thật sự các ca sử dụng đúng là một cách tốt để đảm bảo những người phát triển không đi vòng vo xem xét những vấn đề không liên quan.

Thay vì nghiên cứu sâu lý thuyết tương tác người – máy, trong phạm vi giáo trình này chúng ta chỉ xem xét một số nguyên lý cơ bản cho thiết kế giao diện người dùng.

Dựa vào ca sử dụng

Từ quan điểm hệ thống, ca sử dụng đơn giản giúp những người phát triển đi đúng đường. Đối với người dùng, ca sử dụng là tất cả mọi thứ. Vì vậy ca sử dụng nên được tận dụng để cung cấp cấu trúc giao diện người dùng. Nói chung, nên xây dựng các ca sử dụng đơn giản nghĩa chúng không bao gồm quá nhiều chức năng hoặc quá khó để quản lý. Vì vậy, mặc dù chúng ta mong mỗi giao diện người dùng sẽ biểu diễn cho một số ca sử dụng thì các giao diện vẫn phải đơn giản. Chúng ta cũng mong nhóm các ca sử dụng có liên quan được thể hiện trong cấu trúc giao diện người dùng.

Ví dụ, Hệ thống quản lý học theo tín chỉ được thể hiện với một giao diện người dùng đơn, bao gồm nhiều ca sử dụng (Hình 6.20). Với giao diện đơn, người sử dụng có thể thực hiện hàng loạt các chức năng như đăng ký học, đăng ký thi... Chúng ta nên tránh việc chia một ca sử dụng hoặc các ca sử dụng liên quan thành nhiều hơn một giao diện.

**Hình 6.19: Giao diện đăng ký học tín chỉ**

Đơn giản hóa sử dụng

Một nguyên lý đơn giản nhưng rất có ý nghĩa là làm mọi thứ trở nên đơn giản. Kể cả người dùng là mới hay là chuyên gia, nguyên lý đơn giản cũng đều quan trọng. Trong thương mại điện tử, chúng ta không muốn có một “rào cản” trước hàng hóa mà chúng ta muốn bán.

Một khách hàng vào trang web được trình bày với nhiều hướng dẫn sử dụng dài dòng thì họ sẽ rời đi ngay. Vì vậy, để thu hút khách hàng sử dụng trang web ngay lập tức, chúng ta phải xây dựng hướng dẫn từng bước một như: “click đây để mua” hay “nhập chi tiết tài khoản và click tiếp tục”. Hơn nữa, giao diện càng đơn giản thì càng dễ dàng mở rộng cho các thiết bị mới như điện thoại di động, PDA... Nhu cầu tạo ra các giao diện người dùng động cũng là một lí do quan trọng cho việc sử dụng cụm điều khiển (control layer) trong thiết kế.

Sử dụng giao diện kiểu Notebook

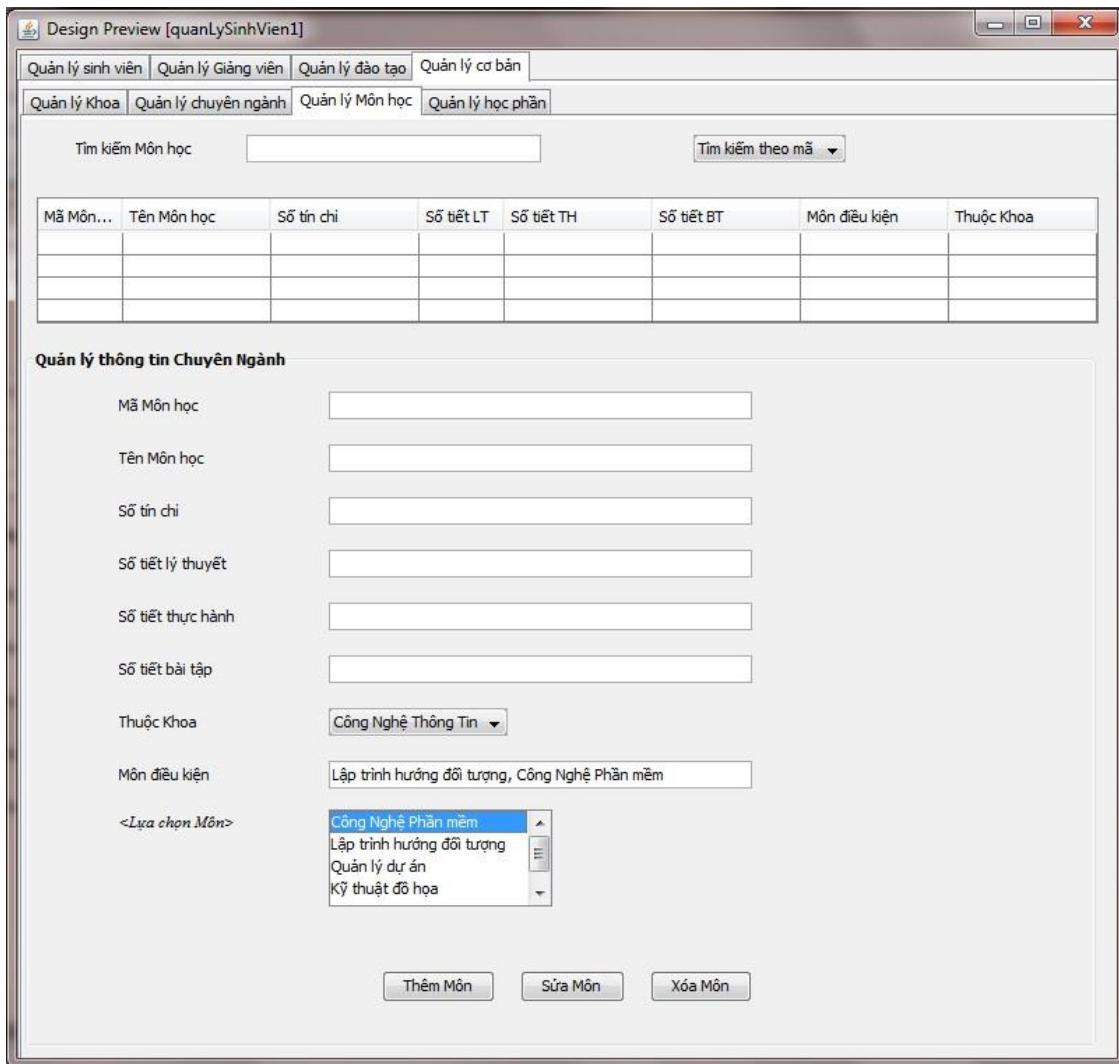
Giao diện kiểu Notebook là nhằm nhóm các ca sử dụng có liên quan với nhau. Notebook là một tập các trang mà chỉ có một trang hiển thị vào mỗi thời điểm, các trang khác sẵn sàng thông qua các tabs khác. Lợi ích của notebooks là kích thước và vị trí của nó như nhau, người

dùng có thể tập trung vào một vị trí trên màn hình, thậm chí với các tương tác lâu liên quan nhiều hơn một ca sử dụng. Điều này cải thiện kinh nghiệm và năng suất người dùng. Notebooks được hỗ trợ bởi hầu hết các thư viện giao diện GUI, như Swing của Java. Thông thường, một ca sử dụng sẽ ánh xạ chính xác tới một trang duy nhất. Thậm chí một ca sử dụng phức tạp cũng có thể được thể hiện trong một panel duy nhất (Hình 6.20).

Sử dụng wizards

Wizard là chuỗi các trang hướng dẫn người dùng từng bước một để thực hiện một hành động phức tạp. Khi người dùng chọn một hoạt động, chúng được hiển thị trong một trang biểu diễn bước 1, người dùng hoàn thành bước 1 và click chuột vào nút Next để tiếp tục bước 2... Trước khi hoàn tất một hoạt động, người dùng có thể vào lại các bước trước đó bằng cách sử dụng nút Quay lui. Mỗi trang của wizard bao gồm các trường, danh sách mà người dùng sử dụng để nhập dữ liệu cần thiết cho bước hiện tại. Bình thường, mỗi trang sẽ có hướng dẫn đơn giản, ví dụ như “nhập mật khẩu cũ của bạn vào ô dưới đây rồi chọn Tiếp tục”...

Việc thực hiện từng bước với những hướng dẫn đơn giản của wizard cho phép người dùng thực thi một hành động phức tạp mà không cần nhớ phải làm thế nào. Ngoài ra, một wizard, như một notebook, giúp người dùng tập trung vào một vị trí trên màn hình.



Hình 6.20: Giao diện quản lý môn học trong hệ đăng ký học tín chỉ

Tránh nhiều cửa sổ

Nhiều cửa sổ chồng nhau nên được tạo ra cho các chuyên gia máy tính hơn là cho những người dùng mới. thậm chí nếu người dùng đủ kiến thức máy tính để mở một trình duyệt web, chúng ta cũng không hy vọng họ có thể đối phó với nhiều cửa sổ trình duyệt hoặc các hộp thoại. Ta có thể tránh sử dụng nhiều cửa sổ bằng cách xếp chồng lên trang trước nội dung của hộp thoại. Notebook và wizard có thể được dùng cùng nhau để nhóm các tương tác phức tạp vào cùng một vị trí duy nhất trên màn hình. Do đó, người dùng không thấy mệt mỏi hay nhầm lẫn. Quy tắc “tại mỗi thời điểm chỉ làm một việc” cũng phù hợp với thiết kế các giao diện.

6.5 SỬ DỤNG FRAMEWORK, MẪU VÀ THƯ VIỆN

Một *mẫu* (pattern) là một giải pháp cho vấn đề lập trình trong mô hình hướng đối tượng. Mẫu cho phép người phát triển thực hiện công việc nhanh hơn, chất lượng mã nguồn tốt hơn và không mất sức nhiều. Việc xem xét một cách đầy đủ các mẫu thiết kế khác nhau nằm ngoại phạm vi của giáo trình này. Mục đích ở đây chỉ nhằm nhấn mạnh cho bạn đọc thấy rằng để trở thành nhà thiết kế tốt chúng ta cần phải tìm hiểu về các mẫu thiết kế. Nhiều mẫu đã được phổ

CHƯƠNG 6. THIẾT KẾ CHI TIẾT

bien trong nhiều tài liệu. Ví dụ, *singletons* (mỗi một lớp chỉ có một thể hiện); *factories* (để tạo những đối tượng); *homes* (tạo các đối tượng và tìm các đối tượng hiện có) và *states* (biểu diễn vòng đời của một đối tượng). Mỗi một mẫu đều có tên, một vài dòng miêu tả và ví dụ để sử dụng nó.

Khuôn dạng (framework) hay *khung làm việc* giống như là một mẫu, nghĩa là một cách để kết hợp các thành phần của một hệ thống lại với nhau. Tuy nhiên, nó có hai khác biệt lớn: một là nó có vẻ lớn hơn nhiều, hai là một số mã nguồn đã được viết sẵn cho người sử dụng. Người thiết kế cần phải tìm kiếm khuôn dạng nào phù hợp để giải quyết vấn đề của mình để tránh viết lại mã nguồn không thật sự cần thiết. Một trong những khuôn dạng đầy đủ và phổ biến đó là EJB (Enterprise JavaBeans). EJB cho phép người phát triển xây dựng logic nghiệp vụ tàng giữa mà không phải viết một dòng mã nào để xử lý lưu trữ, giao dịch, an toàn-bảo mật, tương tranh, phân tán và đa luồng.

Thư viện (library) là một tập các lớp đã được viết sẵn cho người sử dụng. Ta cần biết các thư viện có sẵn để có thể áp dụng cho bài toán của mình mà không phải viết lại các lớp, các phương thức đã được xây dựng. Thư viện của Java J2EE là một ví dụ. Nó có ba phiên bản cho doanh nghiệp, chuẩn và Micro dành cho những nghiệp vụ khác nhau như hệ thống lớn (thương mại điện tử và hệ thống phân tán...) hay những hệ thống vừa phải (desktop) hay là những hệ thống nhỏ như điện thoại di động hay PDA.

6.6 KẾT LUẬN

Trong chương này, chúng ta đã xem xét các bước thiết kế hệ thống con, một quá trình quyết định các đối tượng nào dự định cài đặt và các giao diện nào cần có:

- Xây dựng mô hình lớp thiết kế từ mô hình lớp phân tích.
- Cách ánh xạ mô hình lớp thành lược đồ cơ sở dữ liệu quan hệ.
- Trình bày một vài hướng dẫn cho thiết kế giao diện người sử dụng.

BÀI TẬP

1. Dựa vào kết quả có được trong Bài tập Chương 5, hãy trình bày thiết kế chi tiết cho hệ quản lý bán hàng của siêu thị máy tính.
2. Dựa vào kết quả có được trong Bài tập Chương 5, hãy trình bày thiết kế chi tiết cho hệ quản lý học theo tín chỉ.
3. Dựa vào kết quả có được trong Bài tập Chương 5, hãy trình bày thiết kế chi tiết cho hệ quản lý thư viện.
4. Dựa vào kết quả có được trong Bài tập Chương 5, hãy trình bày thiết kế chi tiết cho hệ quản lý đặt thuê xe.
5. Dựa vào kết quả có được trong Bài tập Chương 5, hãy trình bày thiết kế chi tiết cho hệ quản lý siêu thị mua bán sách.

PHỤ LỤC A

LỰA CHỌN CÔNG NGHỆ

A.1 GIỚI THIỆU

Mặc dù chúng ta có bộ tài liệu yêu cầu và phân tích đầy đủ và ngay cả biểu đồ kiến trúc ban đầu nhưng vẫn chưa đưa ra lựa chọn nào về công nghệ cài đặt mà chúng ta sẽ sử dụng. Việc lựa chọn công nghệ có vai trò vô cùng quan trọng trong một quy trình sản xuất phần mềm. Thời gian trì hoãn việc chọn công nghệ càng lâu, thì càng ít cơ hội để khai thác những điểm tốt của những công nghệ đó. Quyết định chọn lựa công nghệ trong giai đoạn này của tiến trình phát triển (trước thiết kế chi tiết) là một sự kết hợp tốt. Khi chúng ta đã quyết định chọn công nghệ rồi, chúng ta sẽ hiểu sâu sắc các chọn lựa để trở thành định hướng với mỗi chức năng duy nhất của một công nghệ cụ thể.

A.2 CÔNG NGHỆ TẦNG CLIENT

Chúng ta hãy nhìn phần mềm chạy trên các client trong hệ đa tầng. Chúng ta có hai sự chọn lựa chính: chúng ta có thể thực thi một chương trình ứng dụng hoặc một trình duyệt web. Các loại ứng dụng thực thi có thể bao gồm:

- Truyền thông giữa người với người: email, chat...
- Truyền file hoặc trao đổi file.
- Đăng nhập từ xa.
- Những ứng dụng đặc quyền

Client kết nối máy chủ bằng một trình duyệt web có thể sử dụng những công nghệ sau:

- Dạng HTML.
- JavaScript.
- Plug-ins
- Điều khiển ActiveX.
- Java Applets.

Tất cả các công nghệ trên sử dụng một vài loại giao thức, ví dụ IMAP cho email, AIM cho thư tín trực tiếp và HTTP/CGI cho dạng thức HTML, để truyền thông với ít nhất 1 máy khác (mail server, messaging server hoặc Web Server).

Client ứng dụng có nhược điểm là chúng yêu cầu cài đặt phần mềm trên máy client trước khi sử dụng. Tuy nhiên, với một số mục đích, chúng là một lựa chọn tốt. Một trình duyệt web hay một ứng dụng trên chính nó là đặc biệt hữu ích cho phần mềm client bởi vì nó có thể nâng cao bằng cách phát triển đa tầng. Ví dụ, một trình duyệt web có thể sử dụng để kiểm tra chi tiết tài khoản ngân hàng với một Java Applet. Mỗi công nghệ trình duyệt đều có ưu điểm và nhược điểm. Ví dụ:

- HTML thường hỗ trợ nhiều và rộng, nhưng dạng thức HTML còn nguyên thủy và chưa tự động xác nhận trên client.
- JavaScript cho phép lập trình trên client (ví dụ, xác nhận dữ liệu trên dạng thức HTML), nhưng JavaScript phải được biên dịch. Do đó, sẽ chậm hơn khi biên dịch code, không rõ ràng theo hướng đối tượng và với các trình duyệt khác nhau sẽ cung cấp các mức độ khác nhau.
- Java là một ngôn ngữ hướng đối tượng đơn giản, rõ ràng, và là giải pháp tiềm năng tốt nhất. Java cũng cung cấp một phương thức an toàn bởi vì nó ngăn cấm truy cập tài nguyên trên máy chủ bộ mà không có sự xác nhận rõ ràng và chi tiết.
- Về nguyên lý, Java là sự lựa chọn tốt nhất để thực thi client trên một hệ thống ba tầng. Bởi vì sự thiếu hỗ trợ trình duyệt web, hầu hết các nhà phát triển chọn lựa dạng thức HTML.
- Hầu hết các công nghệ quan trọng đều được chuyển cho các thiết bị mới hơn, như PDA hoặc mobile phone. TCP/IP được sử dụng rộng rãi, thậm chí trên cả những thiết bị client nhỏ nó cũng làm việc qua kết nối không dây. Do đó, chuyển một hệ thống ba tầng tới một loại thiết bị mới hơn là bình thường với việc thiết kế lại giao diện người dùng để cho chúng nhỏ hơn và riêng tư hơn.

A.3 GIAO THỨC GIỮA TẦNG CLIENT VÀ TẦNG GIỮA

Phần mềm client, khi chạy như một ứng dụng hoặc trong một trình duyệt web, phải truyền thông với một server sử dụng qua giao thức nào đó. Hầu hết các giao thức đều được phân lớp: tại đây, chúng ta có giao thức mức thấp như là TCP/IP và trên đỉnh chúng ta xây dựng một giao thức khác, đặc trưng với những nhiệm vụ cụ thể. Ví dụ, trên đỉnh của TCP/IP, ta có thể dùng giao thức SSL để mã hóa và giải mã thông tin với mục đích riêng tư và toàn vẹn thông tin. Trên đỉnh của SSL, chúng ta có thể chạy HTTPS, một giao thức bảo mật cho phép một client gọi yêu cầu một tài liệu bằng URI và trả lại nội dung của tài liệu. Các giao thức thường sử dụng được chia làm 2 loại: chuyên dụng và chung. Giao thức chuyên dụng gồm:

- IMAP (email).
- AIM (AOL instant messaging).
- NNTP (USENET news).
- HTTP/CGI (HTML form).
- FTP(truyền file)
- Telnet (đăng nhập từ xa).

Giao thức chung (có khả năng thực hiện nhiều nhiệm vụ) bao gồm:

- TCP/IP (mức thấp của tầng giao vận, cũng được hiểu như sockets).
- JRMP (để truyền thông Java- tới -Java).
- IIOP (dùng cho truyền thông với CORBA, tương tự như RMI nhưng là một ngôn ngữ đa thực thi).

A.4 CÔNG NGHỆ TẦNG GIỮA

Ứng dụng của Server là thường đa luồng, thiết kế cho một thông lượng lớn (có khả năng xử lý hàng nghìn, hoặc hàng triệu client đồng thời). Một ứng dụng Server lắng nghe trên một vài cổng (điểm kết nối) chờ client kết nối.

Với một số loại phần mềm, vị trí của server tương tự như client. Trên client, chúng ta thực thi ứng dụng riêng lẻ hoặc code với máy chủ dựa trên trình duyệt web. Trên tầng server (tầng giữa), chúng ta có thể chạy ứng dụng riêng lẻ hoặc chạy một trình duyệt web và đặt code vào trong đó. Các ứng dụng riêng lẻ gồm:

- Thư, tin nhắn, tin tức và chat server.
- FTP server.
- Telnet server.
- RMI Registry (a look-up RMI objects).
- CORBA naming service (Một kỹ thuật tìm kiếm với đối tượng của CORBA).
- Java Naming and Directory Interface (JNDI) server (Một cách thức để anh xạ các tên chung để có thể sử dụng thay cho một RMI registry, một Thiết bị đặt tên CORBA, hay một đăng ký người dùng).
- Proprietary server (ví dụ, một tiến trình hosting đối tượng CORBA hoặc RMI, một EJB client, một .Net client).

Server code có thể được dùng trên máy chủ là một web server bao gồm:

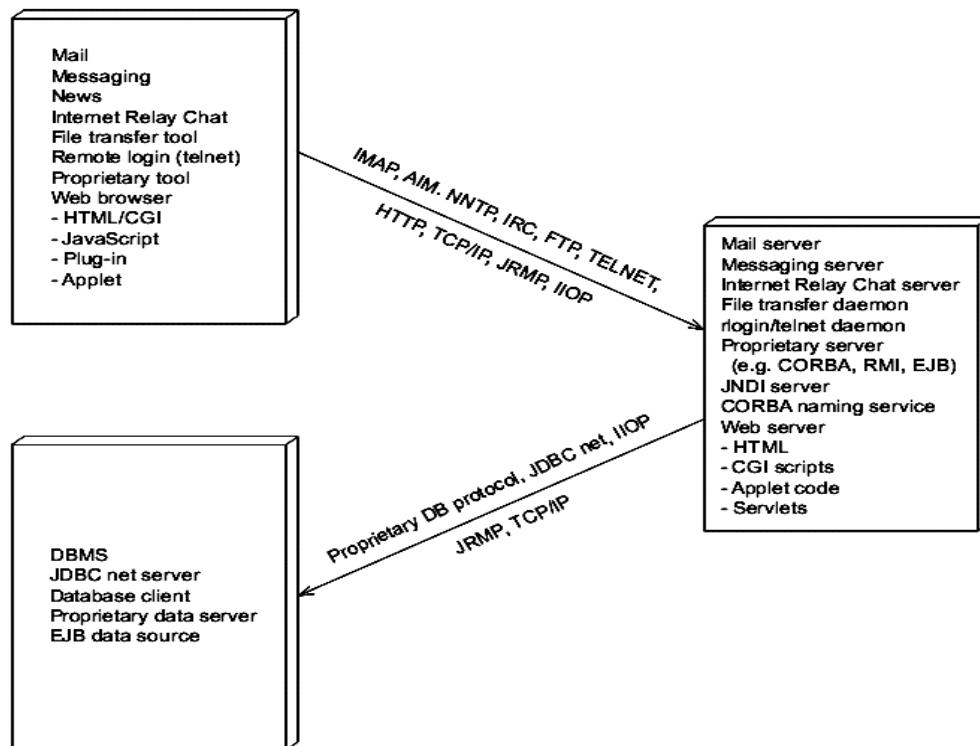
- Java Server Pages (JSPs), để xây dựng các trang Web.
- Active Server Pages (ASPs), tương tự như JSPs nhưng việc coding được thực hiện trên Visual Basic chứ ko phải là Java.
- CGI scripts (chúng là những file đã phiên dịch, được viết trên các ngôn ngữ như PERL, hoặc các chương trình có thể thực thi được).
- Servlets (các đối tượng của Java server có thể được truy cập bởi Java applets, JSPs hoặc dạng thức HTML).

RMI registries and CORBA naming services cho phép RMI và CORBA client tìm đối tượng server bằng tên. CGI Script là file nguyên bản được viết trong một vài ngôn ngữ lệnh như PERL hoặc có thể thực hiện được, biên dịch một ngôn ngữ lập trình theo cách thông thường. JSP là những file text bao gồm HTML nguyên sơ đặt rải rác với java code. Yêu cầu đầu tiên của JSP được biên dịch tới servlet, HTML nguyên sơ sẽ được thay thế với các câu lệnh in và java code theo đúng nguyên bản. JSP thường được sử dụng để cá nhân hóa các trang web bằng cách chèn một dãy câu lệnh cho khách hàng hiện tại đang login. Ví dụ, họ có thể được yêu cầu chính xác hoặc qua servlet. ASP tương tự JSP nhưng sử dụng công nghệ của Microsoft, vì vậy chúng không linh động.

A.5 CÔNG NGHỆ TẦNG GIỮA ĐẾN TẦNG DỮ LIỆU

Có một số cách truy cập tầng dữ liệu:

- Bao gồm database-client code trên tầng giữa để ta có thể truy cập một DBMS chạy trên tầng dữ liệu. Với java, chúng ta có thể thực hiện điều này với sự giúp đỡ của kỹ thuật java database connectivaty (JDBC).
- Giao tiếp với tầng dữ liệu có sử dụng bất kỳ công nghệ tầng client tới tầng giữa nào như đã trình bày.
- Truy cập tầng dữ liệu có thể sử dụng một số giao thức không phải TCP/IP (chúng ta sẽ chỉ thường xuyên thực hiện để truy cập một hệ thống kế thừa).
- Đặt EJB client code trên server tầng giữa, sau đó truy cập tầng dữ liệu qua EJB.
- Đặt .Net client code trong server tầng giữa. .Net là đối thủ của EJB framework.



Hình A.1: Tổng quan về các công nghệ trên 3 tầng [7]

A.6 CÁC CÔNG NGHỆ KHÁC

Tổng quan về J2EE

J2EE là một platform để phát triển những ứng dụng phân tán. J2EE bao gồm những phần sau (được xem như là J2EE Framework):

- J2EE Platform - một platform chuẩn để hosting các ứng dụng J2EE.
- Reference Implementation - một application server hỗ trợ chuẩn mới nhất của J2EE, ngoại trừ những tiêu điểm của nó là những đặc tính mới trong phiên bản chuẩn của J2EE, đây chưa phải là một sản phẩm hoàn chỉnh.
- Compatibility Test Suite - một công cụ để kiểm tra xem một application server có tương thích với chuẩn J2EE hay không. Thiếu cái này thì mỗi nhà cung cấp có thể hiểu và từ đó phát triển chuẩn J2EE theo những hướng khác nhau mà như thế thì làm giảm đi thế mạnh của J2EE platform là “write once, run anywhere”.

- Application Programming Model (APM) Blueprint – một mô hình ứng dụng tham khảo, được cung cấp để minh họa cách phát triển ứng dụng dùng J2EE.

Phát triển chương trình với J2EE

J2EE Framework cho phép phát triển những ứng dụng phân tán bằng cách cung cấp một tập các dịch vụ cơ bản như quản lý giao dịch, kiểm tra an toàn, quản lý trạng thái, quản lý tài nguyên. Các máy chủ ứng dụng đều cung cấp những dịch vụ cơ bản này của J2EE Framework.

Các thành phần của J2EE

J2EE được xây dựng trên một mô hình thành phần. Bốn thành phần cốt lõi cung cấp môi trường cho các thành phần khác của J2EE thông qua các API. Những thành phần cốt lõi này liên quan đến bốn kiểu container được hỗ trợ trong J2EE bao gồm, Application Client, Applet, Web và EJB:

- Java Application: thành phần này là 1 chương trình chạy bên trong Application Client container. Application Client container cung cấp những API hỗ trợ cho messaging, gọi từ xa, kết nối dữ liệu và lookup service. Application Client container đòi hỏi những API sau: J2SE, JMS, JNDI, RIM-IIOP và JDBC. Container này được hỗ trợ bởi nhà cung cấp application server.
- Applet – Applet thành phần là java applet chạy bên trong Applet container, chính là web browser có hỗ trợ công nghệ Java. Applet phải hỗ trợ J2SE API.
- Servlet và JSP – đây là Web-based component chạy ở bên trong Web container, được hỗ trợ bởi Web Server. Web container là một môi trường run-time cho servlet và jsp. Web Container phải hỗ trợ những API sau: J2SE, JMS, JNDI, JTA, JavaMail, JAF, RIM-IIOP và JDBC. Serlet và JSP cung cấp một cơ chế cho việc chuẩn bị, xử lý, định dạng nội dung động.
- Enterprise JavaBean (EJB) – EJB là thành phần nghiệp vụ chạy bên trong EJB container. EJB là phần nhân, cốt lõi của ứng dụng J2EE. EJB container cung cấp các dịch vụ quản lý giao dịch, bảo mật, quản lý trạng thái, quay vòng tài nguyên. EJB container phải hỗ trợ những API sau: J2SE, JMS, JNDI, JTA, JavaMail, JAF, RIM-IIOP và JDBC.

Giao thức tầng Client và tầng giữa

Mô hình TCP/IP được sử dụng kết hợp với một số giao thức khác như FTP, Telnet, IMAP...TCP/IP là một tập các giao thức xác định các qui tắc cũng như định dạng cho việc truyền thông này. TCP/IP hiện là giao thức mạng được sử dụng rộng rãi nhất trên thế giới. Các lý do khiến giao thức mạng này trở nên ngày càng thông dụng là:

- Tính có thể định tuyến và mở rộng được.
- Tính mở.
- Là một chuẩn đã được kiểm nghiệm, mang tính ổn định.
- Đã trở thành bộ giao thức sử dụng cho mạng Internet.

Các giao thức TCP/IP và Mô hình OSI

Bộ giao thức TCP/IP xây dựng dựa trên mô hình mạng do DoD phát triển, được gọi là mô hình DoD.

Các lớp trong mô hình OSI và DoD

Thay vì một mô hình gồm bảy lớp, mô hình DoD chỉ có bốn lớp. Lớp Truy nhập mạng (Network Access) dùng để miêu tả khuôn thước vật lý của mạng cũng như các thông điệp sẽ được định dạng như thế nào trong quá trình truyền dữ liệu. Lớp Internet có nhiệm vụ chuyển phát các gói dữ liệu trong liên mạng và Lớp vận CHUYỀN (Transport) sẽ thực hiện các công việc kiểm tra lỗi để đảm bảo sự chuyển phát tin cậy. Tất cả các chức năng mạng khác được thực hiện trong lớp ứng dụng (Application). Các giao thức trong bộ giao thức TCP/IP sẽ tương ứng với một lớp cụ thể của mô hình DoD và cũng có thể ánh xạ sang mô hình OSI. Dưới đây chúng ta sẽ tóm tắt mỗi giao thức TCP/IP và chỉ rõ mối quan hệ của nó với các lớp trong cả hai mô hình DoD và OSI.

Chức năng của các giao thức TCP/IP độc lập với kiến trúc mạng ở mức thấp. Dưới đây là tóm lược về chức năng cụ thể của các giao thức trong bộ giao thức TCP/IP.

Các giao thức lớp ứng dụng

- **FTP (File Transfer Protocol):** Giao thức truyền tệp
FTP cung cấp một phương thức chung để truyền tệp trong một liên mạng. Nó có thể bao gồm các tính năng bảo mật tệp thông qua sử dụng một cặp tên/mật khẩu để xác thực. Nó có thể cho phép chuyển tệp giữa các hệ thống máy tính khác nhau.
- **TFTP (Trivial File Transfer Protocol):** Giao thức truyền tệp đơn giản
Tương tự như FTP, cho phép truyền tệp giữa một host và một máy chủ FTP (FTP server). Tuy nhiên, giao thức này không bao gồm việc xác thực người sử dụng và dùng UDP chứ không phải là TCP làm giao thức giao vận.
- **HTTP (Hypertext Transport Protocol):** Giao thức truyền tệp siêu văn bản
Các trình duyệt Web và máy chủ Web sử dụng giao thức này để trao đổi các tệp (ví dụ các trang Web) qua mạng toàn cầu WWW hay intranet. Chúng ta có thể coi HTTP là giao thức yêu cầu và trả lời thông tin. Nó thường sử dụng để yêu cầu trả gửi trả các tài liệu Web. Ngoài ra, HTTP cũng được sử dụng để làm giao thức truyền thông giữa các tác tử (agent) sử dụng các giao thức TCP/IP khác.
- **SMTP (Simple Mail Transfer Protocol):** Giao thức chuyển thư đơn giản
Đây là giao thức được sử dụng để định tuyến các thư điện tử trong một liên mạng. Các ứng dụng thư điện tử sẽ cung cấp giao diện để truyền thông với SMTP và máy chủ thư điện tử.

Các giao thức khác: Bộ giao thức TCP/IP còn có nhiều giao thức khác ở lớp ứng dụng để đáp ứng cho một số dịch vụ cụ thể khác. Trong số những giao thức này có thể kể đến:

- **Telnet:** Giao thức điều khiển từ xa.
- **NFS (Network File System):** Hệ thống tệp tin máy chủ.
- **TCP (Transmission Control Protocol):** Giao thức kiểm soát truyền thông tin. Giao thức này cung cấp các dịch vụ hướng kết nối (connection-oriented) và thực hiện các công

việc như kiểm soát thứ tự của các gói tin ,việc đánh địa chỉ các dịch vụ cũng như các chức năng kiểm tra lỗi.

- UDP (User Data Protocol): Giao thức gói dữ liệu người dùng. Giao thức này cũng hoạt động ở lớp giao vận, giống như giao thức TCP. Tuy nhiên, đây không phải là giao thức hướng kết nối và làm phát sinh ít phụ phí hơn TCP. Do ít phụ phí hơn, nó truyền dữ liệu nhanh hơn, nhưng cũng ít tin cậy hơn.
- DNS (Domain Name System): Hệ thống tên miền. Đây là hệ thống được phân tán trong liên mạng để cung cấp việc phân giải tên/địa chỉ. Ví dụ, tên miền internet.vdc.com.vn sẽ được phân giải thành một địa chỉ cụ thể là 203.162.1.181.
- IP (Internet Protocol): Giao thức Internet. Đây là giao thức chính trong bộ giao thức TCP/IP. Đây là một giao thức phi kết nối (connectionless) có chức năng ra các quyết định trong việc định tuyến trong một liên mạng dựa vào các thông tin nó nhận được từ ARP. Sử dụng địa chỉ IP để xác định một mạng cụ thể trong một liên mạng, nó cũng kiểm soát các vấn đề liên quan tới đích chỉ IP khi thực hiện định tuyến.
- ICMP (Internet Control Message Protocol): Giao thức kiểm soát thông điệp Internet. Giao thức này kết hợp chặt chẽ với giao thức IP trong việc cung cấp các thông tin kiểm soát và báo lỗi trong quá trình di chuyển các gói dữ liệu trong một liên mạng.
- IGMP (Internet Group Multicast Protocol): Giao thức nhóm Multicast Internet. Giao thức này xác định các nhóm Multicast. Tất cả các thành viên của một nhóm có thể nhận các thông điệp phát quảng bá dành cho nhóm đó (gọi là các thông điệp multicast). Một nhóm Multicast có thể gồm các thiết bị trong cùng một mạng hay thuộc các mạng khác nhau.
- OSPF (Open Shortest Path First): Giao thức tìm đường ngắn nhất trước . Là giao thức tìm lùa chọn tuyến đường trong phương pháp định tuyến trạng thái liên kết. Giao thức này hiệu quả hơn RIP trong việc cập nhật thông tin cho bảng định tuyến, đặc biệt trong các mạng có qui mô lớn.
- ARP (Address Resolution Protocol): Giao thức phân giải địa chỉ. Giao thức này cung cấp địa chỉ Internet hoàn chỉnh bằng cách kết hợp một địa chỉ mạng (lôgíc) với một địa chỉ vật lý cụ thể. Nó làm việc cùng với các giao thức khác để cung cấp việc phân giải địa chỉ/tên logic.
- RARP và BOOTP. Cả hai giao thức này đều được sử dụng để xác định địa chỉ IP của một thiết bị từ địa chỉ MAC đã biết. BOOTP là một bước cải tiến đối với RARP và hiện được sử dụng nhiều hơn RARP. Các máy tính khi khởi động sẽ sử dụng giao thức này để lấy được địa chỉ IP của chúng từ một máy chủ BOOTP trên mạng. Giao thức BOOTP sẽ giám sát các gói tin yêu cầu địa chỉ do các host gửi tới máy chủ BOOTP sẽ được trả lời.
- DHCP (Dynamic Host Configuration Protocol): Giao thức cấu hình chủ động DHCP đơn giản hóa việc quản lý địa chỉ IP trong một mạng. Các máy chủ DHCP duy trì một danh sách gồm các địa chỉ chưa được sử dụng và các địa chỉ đã được gán và truyền thông tin về địa chỉ tới các host yêu cầu. DHCP gồm hai thành phần sau đây:
 - Một giao thức để phân phối các tham số cấu hình IP từ máy chủ DHCP tới host.
 - Một giao thức xác định các địa chỉ IP sẽ được gán cho các host như thế nào.

PHỤ LỤC B

HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Phụ lục này trình bày chi tiết hơn tài liệu pha xác định yêu cầu và phân tích của Hệ quản lý đăng ký học theo tín chỉ. Phân tích là công việc tiếp theo của tiến trình phát triển phần mềm sau khi chúng ta hoàn thành quá trình Thu thập yêu cầu. Để tiến hành phân tích hệ thống chúng ta cần nắm rõ *Đầu vào, Đầu ra* của quá trình này. Trong đó, đầu vào của quá trình Phân tích là sản phẩm của pha Xác định yêu cầu bao gồm:

- Danh sách các tác nhân
- Danh sách các ca sử dụng
- Các kịch bản
- Biểu đồ ca sử dụng
- Các phác họa giao diện

Đầu ra của pha Phân tích là biểu đồ lớp, biểu đồ giao tiếp và biểu đồ trạng thái. Quá trình phân tích bao gồm hai giai đoạn: phân tích tĩnh và phân tích động. Trong phân tích tĩnh, chúng ta tiến hành thực hiện những công việc sau:

- Xác định các lớp.
- Xét mối quan hệ giữa các lớp.
- Tìm các thuộc tính và phương thức của các lớp.
- Đưa ra biểu đồ lớp.

Đối với phân tích động là xây dựng biểu đồ giao tiếp và biểu đồ trạng thái.

Chúng ta sẽ trình bày chi tiết tiến trình xác định yêu cầu và phân tích yêu cầu Hệ thống đào tạo theo tín chỉ.

B.1 XÁC ĐỊNH YÊU CẦU

Danh sách tác nhân

Sinh viên: một người sử dụng có những thông tin được lưu trữ trong CSDL sinh viên, như: mã sinh viên, họ tên, lớp, khóa, Mỗi sinh viên phải được cung cấp mật khẩu truy nhập phù hợp với mã sinh viên để có thể truy cập vào hệ thống.

Giảng viên: một người sử dụng có những thông tin được lưu trữ trong CSDL giáo viên, như: mã giáo viên, họ tên, khóa, ... Mỗi giảng viên phải được cung cấp mật khẩu truy nhập phù hợp với mã giảng viên để có thể truy cập vào hệ thống.

Nhân viên phòng đào tạo: Một người chịu trách nhiệm điều hành hệ thống.

Danh sách Ca sử dụng

- U1: Đăng nhập: các tác nhân đăng nhập hệ thống.
- U2: Thoát: các actor thoát khỏi hệ thống.
- U3: Đăng ký môn học: sinh viên đăng ký các môn học trong kỳ tới.
- U4: Xem thời khóa biểu: sinh viên xem thời khóa biểu chi tiết các lớp học đã đăng ký của mình
- U5: Xem lịch thi học kỳ: sinh viên xem lịch thi học kỳ các môn đã đăng ký
- U6: Xem điểm học tập: sinh viên xem kết quả học tập của mình.
- U7: In bảng điểm: sinh viên in bảng điểm các khóa học của mình.
- U8: Tìm kiếm thông tin: sinh viên xem danh sách các môn học của từng khoa, chuyên ngành, hệ đào tạo
- U9: Đăng ký môn dạy: giảng viên đăng ký môn sẽ dạy trong kỳ tới.
- U10: Xem lịch giảng dạy: giảng viên xem lịch giảng dạy của mình trong kỳ tới.
- U11: Đổi mật khẩu: người dùng thay đổi mật khẩu truy cập hệ thống.
- U12: Thay đổi thông tin cá nhân: Người dùng thay đổi thông tin cá nhân sau khi đăng nhập hệ thống
- U13: Quản lý sinh viên: Nhân viên phòng đào tạo thực hiện chức năng quản lý sinh viên với các thao tác cơ bản: thêm sinh viên, Sửa thông tin sinh viên, Xóa sinh viên.
- U14: Quản lý giảng viên: Nhân viên phòng đào tạo thực hiện chức năng quản lý Giảng viên với các thao tác cơ bản: Thêm Giảng viên, Sửa thông tin giảng viên, Xóa giảng viên.
- U15: Quản lý cơ bản: Nhân viên phòng đào tạo thực hiện các chức năng quản lý cơ bản như Thêm Khoa, Thêm Chuyên Nghành...
- U16: Quản lý đào tạo: Nhân viên phòng đào tạo thực hiện các chức năng quản lý đào tạo như: Quản lý học phần, Quản lý đăng ký học và xếp lớp cho sinh viên, Quản lý đăng ký dạy của giảng viên, Quản lý kết quả học tập của sinh viên.
- U17: Gửi thông báo: phòng đào tạo gửi các thông báo tới sinh viên.
- U18: Xem thông báo: giảng viên, sinh viên xem thông báo của phòng đào tạo gửi đến

Khảo sát các ca sử dụng

Khi truy cập vào hệ thống đăng ký học theo tín chỉ, yêu cầu người dùng phải đăng nhập vào hệ thống (U1) trước khi có thể sử dụng các chức năng của hệ thống. Sau khi đăng nhập hệ thống, người dùng có thể sửa đổi mật khẩu (U11) tài khoản của mình Giảng viên sau khi đăng nhập hệ thống (U1) có thể đăng ký môn dạy và lớp dạy trong học kỳ (U9) và có thể xem lịch giảng dạy của mình (U10)

Sinh viên sau khi đăng nhập vào hệ thống (U1) có các tùy chọn sử dụng chức năng của hệ thống như Tìm kiếm thông tin (U8), Đăng ký môn học trong học kỳ (U3). Tùy vào các ràng buộc đối với môn học về các môn điều kiện, số tín chỉ tích lũy, cán bộ phòng đào tạo có thể xét sinh viên có đủ điều kiện học môn đó hay không và cập nhật lịch học cũng như lịch thi học kỳ (U16). Khi sinh viên không đủ điều kiện học môn nào đó, phòng đào tạo có thể gửi thông báo (U17) về các môn điều kiện còn thiếu tới sinh viên để sinh viên hoàn tất các môn học đó. Sinh viên có thể xem thông báo (U18), xem thời khóa biểu (U4) là lịch học các môn học đã đăng ký mà sinh viên có thể theo học cũng như lịch thi học kỳ các môn đó.

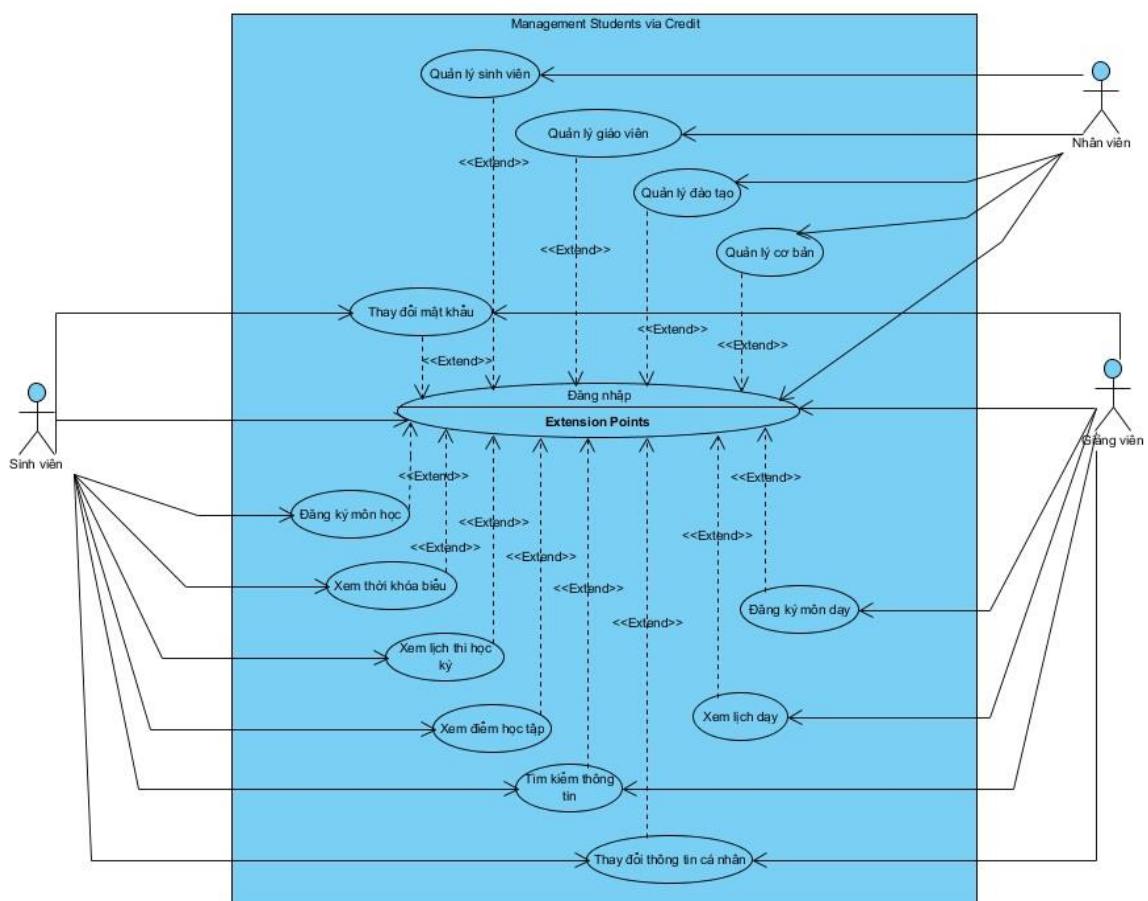
Phòng đào tạo có nhiệm vụ cập nhật bảng điểm (U16) đối với sinh viên, sinh viên có thể xem kết quả học tập của mình (U6) khi đăng nhập hệ thống (U1). Bên cạnh đó hệ thống còn có các tùy chọn cho sinh viên như In bảng điểm (U7) hay tìm kiếm thông tin (U8). Cán bộ

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Đào tạo sau khi đăng nhập vào hệ thống (U1) có thể cập nhật thông tin quản lý cơ bản (U15), thông tin về sinh viên (U13), thông tin về giảng viên (U14).

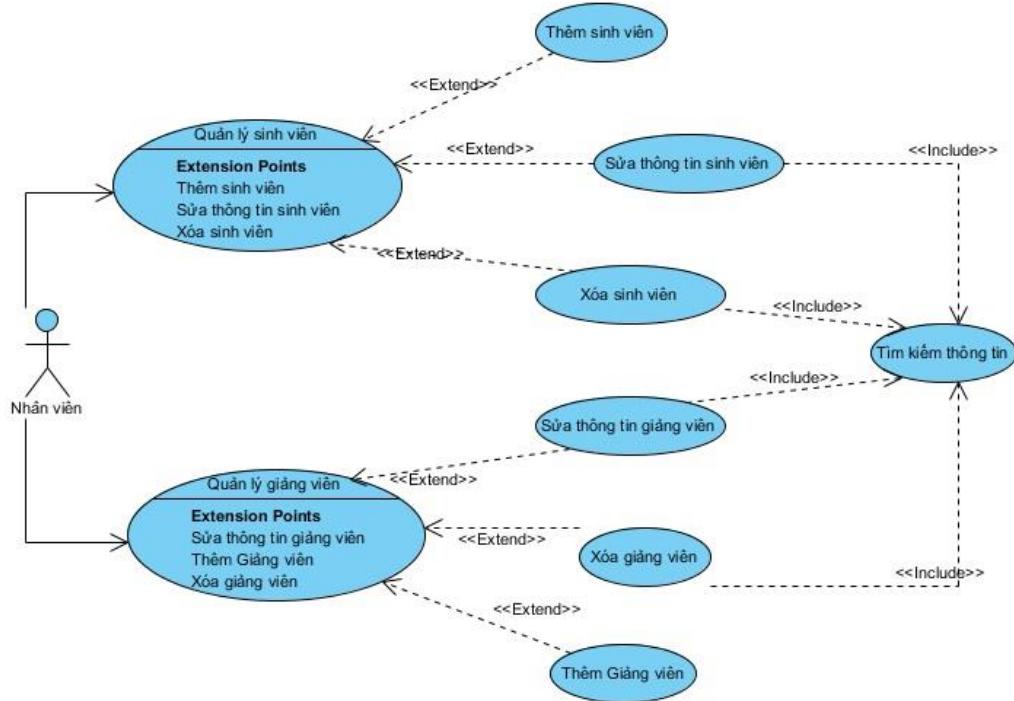
Qua chức năng quản lý đào tạo (U16), cán bộ đào tạo chia lớp học theo Học phần, chuyển đổi học phần cho sinh viên đã đăng kí vào những học phần mà có số lượng sinh viên đăng kí nhỏ hơn điều kiện tối thiểu.

Biểu đồ ca sử dụng

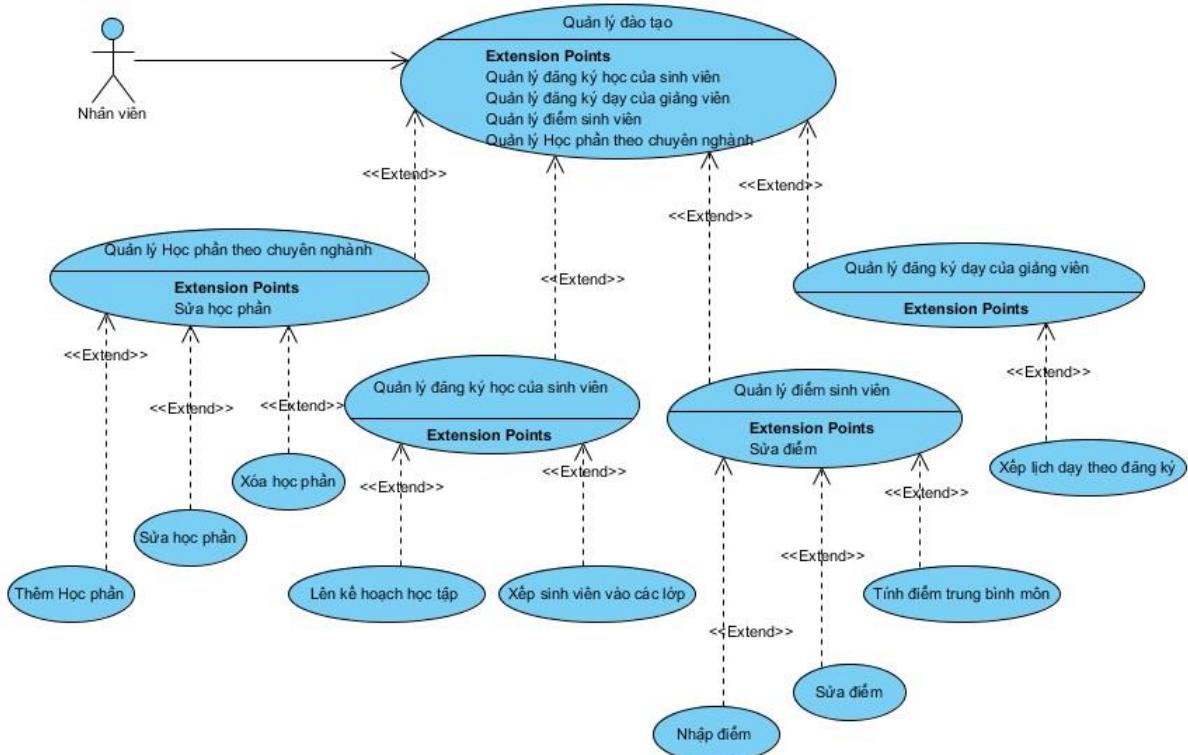


Hình B.1: Biểu đồ ca sử dụng

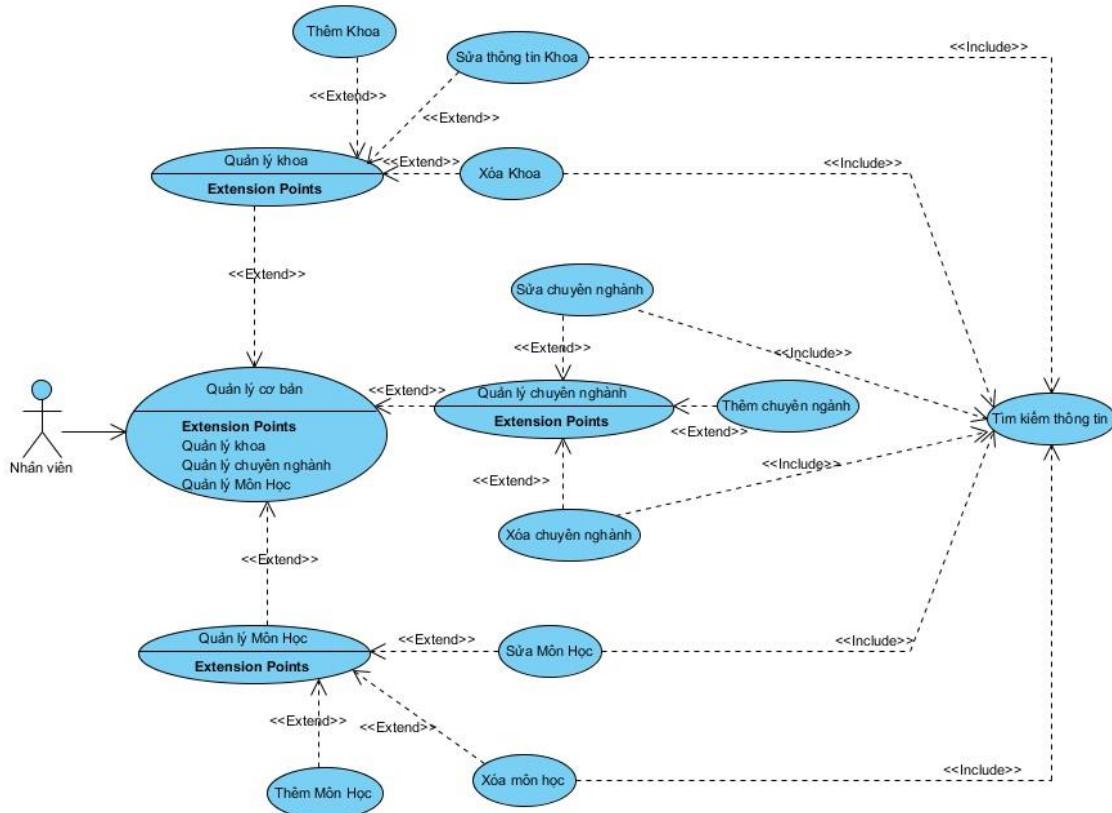
Biểu đồ Use case phân rã



Hình B.2: Phân rã ca sử dụng Quản lý sinh viên – Quản lý giảng viên



Hình B.3: Phân rã ca sử dụng Quản lý đào tạo



Hình B.4: Phân rã ca sử dụng Quản lý cơ bản

Kịch bản (Scenario) và phác thảo giao diện người dùng

Quản lý khoa

Thêm khoa

Tên Use Case	Thêm Khoa
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Khoa
Kích hoạt	Button “Thêm Khoa” trên Form Quản lý Khoa

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt form Quản lý cơ bản
2. Hệ thống hiện thị ba tùy chọn: Quản lý **Khoa**, Quản lý **Chuyên Nghành**, Quản lý **Môn học**.
3. **Cán bộ đào tạo** lựa chọn Quản lý **Khoa**
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về **Khoa** như **Mã Khoa**, **Tên Khoa**, **Trưởng Khoa**, **Mô tả chung**
5. **Cán bộ đào tạo** nhập thông tin về **Khoa** và chọn button “Thêm Khoa”
6. Hệ thống kiểm tra thông tin và lưu vào cơ sở dữ liệu
7. Hệ thống thông báo thêm khóa thành công

Ngoại lệ

- 7.1. Hệ thống thông báo nhập **Mã Khoa** có lỗi
 - 7.1.1. Hệ thống yêu cầu nhập lại **Mã Khoa**
 - 7.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau.

Phác thảo giao diện Quản lý Khoa

The screenshot shows a Windows application window titled "Design Preview [quanLySinhVien1]". At the top, there are tabs for "Quản lý sinh viên", "Quản lý Giảng viên", "Quản lý đào tạo", and "Quản lý cơ bản". Under "Quản lý cơ bản", there are sub-tabs: "Quản lý Khoa", "Quản lý chuyên ngành", "Quản lý Môn học", and "Quản lý học phần". Below the tabs is a search bar with fields "Tim kiem Khoa" and "Tim kiem theo mã". A large table below the search bar has columns: Mã Khoa, Tên Khoa, Trưởng Khoa, and Mô tả. To the right of the table is a section titled "Quản lý thông tin Khoa" containing four input fields: Mã Khoa, Tên Khoa, Trưởng Khoa, and Mô tả chung. At the bottom right are three buttons: "Thêm Khoa", "Sửa Khoa", and "Xóa Khoa".

B.5: Quản lý khoa

Quản lý Chuyên ngành

Thêm chuyên ngành

Tên Use Case	Thêm Chuyên Ngành
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Chuyên Ngành
Kích hoạt	Button “Thêm Chuyên Ngành” trên Form Quản lý Chuyên Ngành

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt form **Quản lý cơ bản**
2. Hệ thống hiện thị ba tùy chọn: Quản lý **Khoa**, Quản lý **Chuyên Ngành**, Quản lý **Môn học**, Quản lý **Học phần**
3. Cán bộ đào tạo lựa chọn Quản lý **Chuyên Ngành**
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Chuyên Ngành như **Mã chuyên ngành**, **Tên Chuyên ngành**, **Trưởng bộ môn**, **Khoa phụ trách**.
5. Cán bộ đào tạo nhập thông tin về Chuyên ngành và chọn button “Thêm Chuyên Ngành”
6. Hệ thống kiểm tra thông tin và lưu vào cơ sở dữ liệu
7. Hệ thống thông báo thêm **Chuyên Ngành** thành công

Ngoại lệ:

- 6.1. Hệ thống thông báo nhập **Mã Chuyên Ngành** có lỗi
 - 6.1.1. Hệ thống yêu cầu nhập lại **Mã Chuyên Ngành**.

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

6.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau

Sửa thông tin chuyên ngành

Tên Use Case	Sửa thông tin Chuyên Ngành
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Hiển thị thông tin đã thay đổi về Chuyên Ngành
Kích hoạt	Button “Sửa Chuyên Ngành” trên Form Quản lý Chuyên Ngành
Chuỗi sự kiện chính	
2. Cán bộ đào tạo kích hoạt form Quản lý cơ bản	
3. Hệ thống hiện thị ba tùy chọn: Quản lý Khoa , Quản lý Chuyên Ngành , Quản lý Môn học , Quản lý Học phần	
4. Cán bộ đào tạo lựa chọn Quản lý Chuyên Ngành	
5. Hệ thống hiển thị form để nhập các thông tin cần thiết về Chuyên Ngành như Mã chuyên ngành , Tên Chuyen ngành , Trưởng bộ môn , Khoa phụ trách .	
6. Cán bộ đào tạo Mã hoặc Tên Chuyên Ngành và chọn “Tìm kiếm”	
7. Hệ thống hiển thị thông tin về Chuyên Ngành tương ứng	
8. Cán bộ đào tạo lựa chọn Chuyên Ngành và thực hiện sửa thông tin về Chuyên Ngành và chọn button “Sửa Chuyên Ngành”	
9. Hệ thống kiểm tra thông tin và lưu vào cơ sở dữ liệu	
10. Hệ thống thông báo quá trình sửa thông tin về Chuyên Ngành thành công	
Ngoại lệ	
7.1. Hệ thống thông báo không tìm thấy Chuyên Ngành	
7.1.1. Hệ thống yêu cầu nhập lại Mã Chuyên Ngành .	
7.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau	

Xóa chuyên ngành

Tên Use Case	Xóa thông tin Chuyên Ngành
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống
Tiền điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống trả về trạng thái ban đầu
Đảm bảo thành công	Đã xóa thông tin
Kích hoạt	Button “Xóa Chuyên Ngành” trên Form Quản lý Chuyên Ngành
Chuỗi sự kiện chính	
2. Cán bộ đào tạo kích hoạt form Quản lý cơ bản	
3. Hệ thống hiện thị ba tùy chọn: Quản lý Khoa , Quản lý Chuyên Ngành , Quản lý Môn học , Quản lý Học phần	
4. Cán bộ đào tạo lựa chọn Quản lý Chuyên Ngành	
5. Hệ thống hiển thị form để nhập các thông tin cần thiết về Chuyên Ngành như Mã chuyên ngành , Tên Chuyen ngành , Trưởng bộ môn , Khoa phụ trách	
6. Cán bộ đào tạo nhập Mã hoặc Tên Chuyên Ngành và chọn “Tìm kiếm”	

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

- | |
|---|
| <p>7. Hệ thống hiển thị thông tin về Chuyên Ngành tương ứng</p> <p>8. Cán bộ đào tạo lựa chọn Chuyên ngành và chọn button “Xóa Chuyên Ngành”</p> <p>9. Hệ thống hiển thị thông báo khẳng định xóa Chuyên Ngành và thông báo xóa Chuyên Ngành thành công</p> |
|---|

Ngoại lệ

- | |
|--|
| <p>8.1. Hệ thống thông báo Mã Chuyên Ngành đã bị trùng</p> <p>8.1.1. Hệ thống yêu cầu nhập lại Mã Chuyên Ngành.</p> <p>8.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau</p> |
|--|

Phác thảo giao diện quản lý Chuyên Ngành

Hình B.6: Giao diện quản lý sinh viên

Quản lý Môn Học

Thêm Môn học

Tên Use Case	Thêm Môn Học
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Môn Học
Kích hoạt	Button “Thêm Môn Học” trên Form Quản lý Môn Học

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt form Quản lý cơ bản
2. Hệ thống hiển thị ba tùy chọn: Quản lý **Khoa**, Quản lý **Chuyên Ngành**, Quản lý **Môn học**. Quản lý **Học phần**.

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

3. Cán bộ đào tạo lựa chọn Quản lý Môn Học 4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Môn Học như Mã môn , Tên môn , Số tín chỉ , Số tiết Lý thuyết , Số tiết Thực hành , Số tiết bài tập , Khoa phụ trách , Môn điều kiện . 5. Cán bộ đào tạo nhập thông tin về Môn Học và chọn “Thêm Môn Học” 6. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo thêm Môn Học thành công
Ngoại lệ: <ul style="list-style-type: none"> 6.1. Hệ thống thông báo Mã Môn Học đã bị trùng <ul style="list-style-type: none"> 6.1.1. Hệ thống yêu cầu nhập lại Mã Môn Học 6.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau

Sửa thông tin Môn Học

Tên Use Case	Sửa thông tin Môn Học
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Hiển thị thông tin đã thay đổi về Môn Học
Kích hoạt	Button “Sửa Môn Học” trên Form Quản lý Môn Học
Chuỗi sự kiện chính	
1. Cán bộ đào tạo kích hoạt form Quản lý cơ bản 2. Hệ thống hiện thị ba tùy chọn: Quản lý Khoa , Quản lý Chuyên Ngành , Quản lý Môn học, Quản lý Học phần . 3. Cán bộ đào tạo lựa chọn Quản lý Môn học 4. Hệ thống hiển thị form tìm kiếm và để nhập các thông tin cần thiết về Môn học như Mã môn , Tên môn , Số tín chỉ , Số tiết Lý thuyết , Số tiết Thực hành , Số tiết bài tập , Khoa phụ trách , Môn điều kiện . 5. Cán bộ đào tạo nhập Tên Môn học và chọn “Tìm kiếm” 6. Hệ thống hiển thị thông tin về Môn học tương ứng 7. Cán bộ đào tạo lựa chọn môn học và thực hiện sửa thông tin về Môn học và chọn button “Sửa Môn học” 8. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo quá trình sửa thông tin về Môn học thành công	
Ngoại lệ	
8.1. Hệ thống thông báo không tìm thấy Mã Môn Học <ul style="list-style-type: none"> 8.1.1. Hệ thống yêu cầu nhập lại Mã Môn Học 8.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau 	

Xóa thông tin Môn học

Tên Use Case	Xóa thông tin Môn Học
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống
Tiền điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống trở về trạng thái ban đầu
Đảm bảo thành công	Đã xóa thông tin
Kích hoạt	Button “Xóa Môn Học” trên Form Quản lý Môn Học
Chuỗi sự kiện chính	

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

1. Cán bộ đào tạo kích hoạt form Quản lý cơ bản
2. Hệ thống hiện thị ba tùy chọn: Quản lý **Khoa**, Quản lý Chuyên Ngành, Quản lý Môn học, Quản lý Học phần.
3. Cán bộ đào tạo lựa chọn Quản lý Môn học
4. Hệ thống hiển thị form tìm kiếm và để nhập các thông tin cần thiết về Môn học như **Mã môn**, **Tên môn**, **Số tín chỉ**, **Số tiết Lý thuyết**, **Số tiết Thực hành**, **Số tiết bài tập**, **Khoa phụ trách**, **Môn điều kiện**.
5. Cán bộ đào tạo nhập **Mã** hoặc **Tên Môn học** và chọn “Tim kiếm”
6. Hệ thống hiển thị thông tin về **Môn học** tương ứng
7. Cán bộ đào tạo lựa chọn môn học muốn xóa và chọn button “Xóa Môn học”
8. Hệ thống hiển thị thông báo khẳng định sẽ xóa **Môn học** và thông báo xóa **Môn học** thành công

Ngoại lệ

- 8.1. Hệ thống thông báo không tìm thấy **Môn Học**.
 - 8.1.1. Hệ thống yêu cầu nhập lại **Môn Học**
 - 8.1.2. **Cán bộ đào tạo** nhập lại và tiếp tục các bước sau

Phác thảo giao diện Quản lý Môn học

The screenshot displays a Windows application window titled "Design Preview [quanLySinhVien1]". The window contains several tabs at the top: "Quản lý sinh viên", "Quản lý Giảng viên", "Quản lý đào tạo", "Quản lý cơ bản", "Quản lý Khoa", "Quản lý chuyên ngành", "Quản lý Môn học" (which is currently selected), and "Quản lý học phần". Below the tabs, there is a search bar with the placeholder "Tim kiem Mon hoc" and a dropdown menu labeled "Tim kiem theo mã". A large table follows, with columns: Mã Môn..., Tên Môn học, Số tín chỉ, Số tiết LT, Số tiết TH, Số tiết BT, Môn điều kiện, and Thuộc Khoa. To the right of the table, there is a section titled "Quản lý thông tin Chuyên Ngành" containing input fields for Mã Môn học, Tên Môn học, Số tín chỉ, Số tiết lý thuyết, Số tiết thực hành, Số tiết bài tập, Thuộc Khoa (with a dropdown menu showing "Công Nghệ Thông Tin"), and Môn điều kiện (with a dropdown menu showing "Lập trình hướng đối tượng, Công Nghệ Phần mềm"). A dropdown menu labeled "<Lựa chọn Môn>" is also present. At the bottom of the window, there are three buttons: Thêm Môn, Sửa Môn, and Xóa Môn.

Hình B.7: Giao diện quản lý môn học

Quản lý sinh viên

Thêm sinh viên

Tên Use Case	Thêm Sinh Viên
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Sinh Viên
Kích hoạt	Button “Thêm Sinh viên” trên Form Quản lý Sinh Viên
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý Sinh viên 2. Hệ thống hiển thị hai tùy chọn “Thêm Sinh Viên” và “Sửa/xóa sinh viên” 3. Cán bộ đào tạo lựa chọn form “Thêm Sinh viên” 4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Sinh viên và các tùy chọn: Tìm kiếm, Thêm Sinh viên, Sửa Sinh viên, Xóa Sinh viên. 5. Cán bộ đào tạo nhập thông tin về Sinh viên gồm có Mã SV, Họ tên, Ngày sinh, Giới tính, Khoa, Chuyên ngành, Khóa, Lớp và chọn button “Thêm Sinh viên” 6. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo thêm Sinh viên thành công 	
Ngoại lệ:	
6.2 Hệ thống thông báo Mã Sinh viên đã bị lỗi <ul style="list-style-type: none"> 6.2.1 Hệ thống yêu cầu nhập lại Mã Sinh viên. 6.2.2 Cán bộ đào tạo nhập lại và tiếp tục các bước sau 	

Sửa Sinh viên

Tên Use Case	Sửa thông tin Sinh viên
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Hiển thị thông tin đã thay đổi về Sinh Viên
Kích hoạt	Button “Sửa Sinh Viên” trên Form Quản lý Sinh Viên
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý Sinh viên 2. Hệ thống hiển thị hai tùy chọn “Thêm Sinh viên” và “Sửa/xóa Sinh viên” 3. Cán bộ đào tạo lựa chọn form “Sửa/xóa Sinh viên” 4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Sinh viên và các tùy chọn: Tìm kiếm, Thêm Sinh viên, Sửa Sinh viên, Xóa Sinh viên. 5. Cán bộ đào tạo nhập Mã Sinh viên hoặc Họ tên Sinh viên và chọn “Tìm kiếm” 6. Hệ thống kiểm tra thông tin và hiển thị thông tin về Sinh viên cần tìm kiếm gồm có Mã SV, Họ tên, Ngày sinh, Giới tính, Khoa, Chuyên ngành, Khóa, Lớp. 7. Cán bộ đào tạo lựa chọn một Sinh viên cần sửa thông tin 8. Hệ thống hiển thị các thông tin về Sinh viên trên form nhập liệu 9. Cán bộ đào tạo thay đổi thông tin và lựa chọn “Sửa Sinh viên” 10. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo sửa Sinh viên thành công 	
Ngoại lệ	
10.1. Hệ thống thông báo Mã Sinh viên không tồn tại. <ul style="list-style-type: none"> 10.1.1. Hệ thống yêu cầu nhập lại Mã Sinh viên. 	

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

10.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau.

Xóa thông tin sinh viên

Tên Use Case	Xóa thông tin Sinh viên
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống
Tiêu điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống trở về trạng thái ban đầu
Đảm bảo thành công	Đã xóa thông tin
Kích hoạt	Button “Xóa Sinh Viên” trên Form Quản lý Sinh Viên

Chuỗi sự kiện chính
1. Cán bộ đào tạo kích hoạt form Quản lý Sinh viên
2. Hệ thống hiển thị hai tùy chọn “Thêm Sinh viên” và “Sửa/xóa Sinh viên”
3. Cán bộ đào tạo lựa chọn form “Sửa/xóa Sinh viên”
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Sinh viên và các tùy chọn: Tìm kiếm, Thêm Sinh viên, Sửa Sinh viên, Xóa Sinh viên.
5. Cán bộ đào tạo nhập Mã Sinh viên hoặc Họ tên Sinh viên và chọn “Tìm kiếm”
6. Hệ thống kiểm tra thông tin và hiển thị thông tin về Sinh viên cần tìm kiếm gồm có Mã SV, Họ tên, Ngày sinh, Giới tính, Khoa, Chuyên ngành, Khóa, Lớp .
7. Cán bộ đào tạo lựa chọn Sinh viên cần xóa và lựa chọn “Xóa Sinh viên”
8. Hệ thống hiển thị thông báo khẳng định muốn xóa Sinh viên
9. Cán bộ đào tạo khẳng định xóa
10. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo xóa Sinh viên thành công

Ngoại lệ
10.1. Hệ thống thông báo Mã Sinh viên có lỗi.
10.1.1. Hệ thống yêu cầu nhập lại Mã Sinh viên
10.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau.

Phác thảo giao diện quản lý sinh viên:

Giao diện Thêm Sinh Viên

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Hình B.8: Giao diện Sửa/xóa Sinh viên

Quản lý Giảng viên vién

Thêm Giảng viên

Tên Use Case	Thêm Giảng Viên
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Giảng Viên
Kích hoạt	Button “Thêm Giảng Viên” trên Form Quản lý Giảng Viên

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt form **Quản lý Giảng viên**
2. Hệ thống hiển thị hai tùy chọn “Thêm Giảng Viên” và “Sửa/xóa Giảng viên”
3. Cán bộ đào tạo lựa chọn form **“Thêm Giảng viên”**
4. Hệ thống hiển thị form để nhập các thông tin cần thiết về **Giảng viên** và các tùy chọn: Tìm kiếm, Thêm Giảng viên, Sửa Giảng viên, Xóa Giảng viên.
5. Cán bộ đào tạo nhập thông tin về **Giảng viên** gồm có **Mã GV, Họ tên, Ngày sinh, Giới tính, Khoa, Bộ Môn, Học vị** và chọn button “Thêm Giảng viên”
6. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo thêm Giảng viên thành công

Ngoại lệ:

- 6.1. Hệ thống thông báo **Mã Giảng Viên** đã bị trùng
- 6.1.1. Hệ thống yêu cầu nhập lại **Mã Giảng viên**.
- 6.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau

Sửa Giảng viên

Tên Use Case	Sửa thông tin Giảng Viên
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Hiển thị thông tin đã thay đổi về Giảng Viên
Kích hoạt	Button “Sửa Giảng Viên” trên Form Quản lý Giảng Viên
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý Giảng viên 2. Hệ thống hiển thị hai tùy chọn “Thêm Giảng Viên” và “Sửa/xóa Giảng viên” 3. Cán bộ đào tạo lựa chọn form “Sửa/xóa Giảng viên” 4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Giảng viên và các tùy chọn: Tìm kiếm, Thêm Giảng viên, Sửa Giảng viên, Xóa Giảng viên. 5. Cán bộ đào tạo nhập Mã Giảng Viên hoặc Họ tên Giảng viên và chọn “Tìm kiếm” 6. Hệ thống kiểm tra thông tin và hiển thị thông tin về Giảng viên cần tìm kiếm gồm có Mã GV, Họ tên, Ngày sinh, Giới tính, Khoa, Bộ Môn, Học vị 7. Cán bộ đào tạo lựa chọn một Giảng viên cần sửa thông tin 8. Hệ thống hiển thị các thông tin về Giảng viên trên form nhập liệu 9. Cán bộ đào tạo thay đổi thông tin và lựa chọn “Sửa Giảng viên” 10. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo sửa Giảng viên thành công 	
Ngoại lệ	
<ol style="list-style-type: none"> 10.1. Hệ thống thông báo Mã Giảng Viên có lỗi <ol style="list-style-type: none"> 10.1.1. Hệ thống yêu cầu nhập lại Mã Giảng Viên. 10.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau. 	

Xóa thông tin Giảng viên

Tên Use Case	Xóa thông tin Giảng viên
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống
Tiền điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống trả về trạng thái ban đầu
Đảm bảo thành công	Đã xóa thông tin
Kích hoạt	Button “Xóa Giảng Viên” trên Form Quản lý Giảng Viên

Chuỗi sự kiện chính

Tên Use Case	Xóa thông tin Giảng viên
Tác nhân chính	Cán bộ đào tạo
Người chịu trách nhiệm	Người quản lý hệ thống
Tiền điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống trả về trạng thái ban đầu
Đảm bảo thành công	Đã xóa thông tin
Kích hoạt	Button “Xóa Giảng Viên” trên Form Quản lý Giảng Viên
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý Giảng viên 2. Hệ thống hiển thi hai tùy chọn “Thêm Giảng Viên” và “Sửa/xóa Giảng viên” 3. Cán bộ đào tạo lựa chọn form “Sửa/xóa Giảng viên” 4. Hệ thống hiển thị form để nhập các thông tin cần thiết về Giảng viên và các tùy chọn: Tìm kiếm, Thêm Giảng viên, Sửa Giảng viên, Xóa Giảng viên. 5. Cán bộ đào tạo nhập Mã Giảng Viên hoặc Họ tên Giảng viên và chọn “Tìm kiếm” 6. Hệ thống kiểm tra thông tin và hiển thị thông tin về Giảng viên cần tìm kiếm gồm có Mã GV, Họ tên, Ngày sinh, Giới tính, Khoa, Bộ Môn, Học vị. 7. Cán bộ đào tạo lựa chọn Giảng viên cần xóa và lựa chọn “Xóa Giảng viên” 	

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

- | | |
|--|--|
| | 8. Hệ thống hiển thi thông báo khăng định muốn xóa Giảng viên |
| | 9. Cán bộ đào tạo khăng định xóa |
| | 10. Hệ thống kiểm tra thông tin, lưu vào cơ sở dữ liệu và thông báo xóa Giảng viên thành công |

Ngoại lệ

- | |
|---|
| 10.1. Hệ thống thông báo Mã Giảng Viên bị lỗi. |
| 10.1.1. Hệ thống yêu cầu nhập lại Mã Giảng Viên |
| 10.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau. |

Phác thảo giao diện quản lý Giảng Viên

The screenshot shows a Windows application window titled "Design Preview [quanLySinhVien1]". The window has a tab bar at the top with four tabs: "Quản lý sinh viên", "Quản lý Giảng viên" (which is selected and highlighted in blue), "Quản lý đào tạo", and "Quản lý cơ bản". Below the tabs are two buttons: "Thêm Giảng viên" and "Sửa/xóa Giảng viên". The main content area is titled "Thông tin Giảng viên" and is divided into two sections: "Thông tin cá nhân" and "Thông tin giáo vụ". The "Thông tin cá nhân" section contains fields for Họ tên (Name), Ngày sinh (Date of birth), Quê quán (Hometown), Địa chỉ thường trú (Residence address), Số CMND (ID number), and Giới tính (Gender) with a dropdown menu showing "Nam". The "Thông tin giáo vụ" section contains fields for Mã GV (Teacher ID), Khoa (Faculty), Bộ môn (Department), and Học vị (Degree). At the bottom right of the form is a "Thêm GV" button.

Hình B.10: Giao diện thêm Giảng Viên

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Hình B.11: Giao diện Sửa/xóa Giảng viên

Quản lý đào tạo

Quản lý xếp lớp học

Thêm lớp học

Tên Use Case	Thêm Lớp học
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo 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	Đã thêm được Lớp học
Kích hoạt	Button “Thêm Lớp học” trên Form Quản lý xếp lớp học

Chuỗi sự kiện chính

1. Cán bộ đào tạo kích hoạt form **Quản lý Xếp lớp học**
2. Hệ thống hiển thị form với các thông tin tùy chọn cần thiết về **Khoa, Chuyên Ngành, Học kỳ**
3. Cán bộ đào tạo chọn các thông tin về **Khoa, Chuyên Ngành, Học kỳ** tương ứng và chọn “Hiển thị”
4. Hệ thống kiểm tra thông tin và hiển thị các **Môn học bắt buộc** và tự chọn mà **Sinh viên** cần học trong học kỳ tương ứng
5. Cán bộ đào tạo lựa chọn **Môn học** và điền các thông tin về lớp học tương ứng với môn học đó gồm có **Mã môn, Tên môn, Ngày bắt đầu, Buổi học, Giờ**

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

học, Phòng học, Sĩ số và chọn “Thêm lớp học”
6. Hệ thống kiểm tra thông tin và hiển thị Thêm lớp học thành công
Ngoại lệ:
10.1.Hệ thống thông báo Mã Lớp học đã bị trùng
10.1.1.Hệ thống yêu cầu nhập lại Mã Lớp học .
10.1.2.Cán bộ đào tạo nhập lại và tiếp tục các bước sau

Sửa lớp học

Tên Use Case	Sửa Lớp học
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống loại bỏ các thông tin đã thay đổi và quay lui lại bước trước
Đảm bảo thành công	Đã sửa được thông tin Lớp học
Kích hoạt	Button “Sửa Lớp học” trên Form Quản lý xếp lớp học
Chuỗi sự kiện chính	<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý Xếp lớp học 2. Hệ thống hiển thị form để tìm kiếm Lớp học theo Môn học tương ứng 3. Cán bộ đào tạo nhập Tên Môn học (hoặc Mã Môn học) và chọn tìm kiếm 4. Hệ thống kiểm tra thông tin và hiển thị thông tin về các lớp học tung ứng với môn học đó gồm có Mã môn, Tên môn, Mã lớp, Ngày bắt đầu, Buổi học, Giờ học, Phòng học, Sĩ số. 5. Cán bộ đào tạo lựa chọn lớp học muốn sửa 6. Thông tin về lớp học muốn sửa sẽ được hiển thị lên form nhập liệu 7. Các bộ đào tạo thay đổi thông tin về lớp học và lựa chọn “Sửa lớp học” 8. Hệ thống kiểm tra thông tin và hiển thị quá trình Sửa lớp học thành công
Ngoại lệ:	<p>4.1 Tên (hoặc Mã) Môn học không tìm thấy trong cơ sở dữ liệu</p> <p> 4.1.1 Hệ thống hiển thị thông báo không tìm thấy và yêu cầu người dùng nhập lại thông tin tìm kiếm</p> <p> 4.1.2 Cán bộ đào tạo nhập lại và tiếp tục các bước sau</p> <p>8.1. Hệ thống thông báo sửa thông tin bị lỗi</p> <p> 8.1.1. Hệ thống yêu cầu nhập lại thông tin.</p> <p> 8.1.2. Cán bộ đào tạo nhập lại và tiếp tục các bước sau</p>

Xóa lớp học

Tên Use Case	Xóa Lớp học
Tác nhân chính	Cán bộ đào tạo.
Người chịu trách nhiệm	Người quản lý hệ thống.
Tiền điều kiện	Cán bộ đào tạo đã Đăng nhập vào hệ thống
Đảm bảo tối thiểu	Hệ thống trả về trạng thái ban đầu
Đảm bảo thành công	Đã xóa được Lớp học
Kích hoạt	Button “Xóa Lớp học” trên Form Quản lý xếp lớp học
Chuỗi sự kiện chính	<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý Xếp lớp học 2. Hệ thống hiển thị form để tìm kiếm Lớp học theo Môn học tương ứng 3. Cán bộ đào tạo nhập Tên Môn học (hoặc Mã Môn học) và chọn tìm kiếm 4. Hệ thống kiểm tra thông tin và hiển thị thông tin về các lớp học tung ứng với

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

môn học đó gồm có Mã môn , Tên môn , Mã lớp , Ngày bắt đầu , Buổi học , Giờ học , Phòng học , Sĩ số . 5. Cán bộ đào tạo lựa chọn lớp học muốn xóa và lựa chọn “Xóa Lớp học” 6. Hệ thống hiển thị thông báo khẳng định muốn xóa Lớp học khỏi cơ sở dữ liệu hay không 7. Cán bộ đào tạo đồng ý 8. Hệ thống xóa lớp học khỏi cơ sở dữ liệu và hiển thị thông báo Xóa lớp học thành công.
Ngoại lệ: <ul style="list-style-type: none"> 4.1. Tên (hoặc Mã) Môn học có lỗi <ul style="list-style-type: none"> 4.1.1 Hệ thống hiển thị thông báo không tìm thấy và yêu cầu người dùng nhập lại thông tin tìm kiếm 4.1.2 Cán bộ đào tạo nhập lại và tiếp tục các bước sau

Phác thảo giao diện Quản lý xếp lớp:

Hình B.12: Giao diện quản lý đào tạo

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Quản lý lớp học

Tên use case:	Quản lý lớp học
Tác nhân chính	Cán bộ đào tạo
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Cán bộ đào tạo xem thông kê các lớp học thành công
Đảm bảo tối thiểu	Trở lại màn hình lựa chọn tùy chọn cơ bản
Kích hoạt	Người dùng chọn chức năng Quản lý lớp học
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý đăng ký học 2. Hệ thống hiển thị tùy chọn Quản lý lớp học và Quản lý danh sách lớp học 3. Cán bộ đào tạo lựa chọn Quản lý lớp học 4. Hệ thống hiển thị form Quản lý lớp học với các thông tin tùy chọn 5. Cán bộ đào tạo thay đổi các thông tin tùy chọn về Khoa, Chuyên Nghành, Học Kỳ và chọn “Xem thông tin lớp học” 6. Hệ thống hiển thị thông kê các lớp học tương ứng với các thông tin gồm có Mã môn, Tên môn, Số học phần, Mã lớp học, Ngày bắt đầu, Buổi học, Giờ học, Phòng học, Số đăng ký. 	
Ngoại lệ	
<ol style="list-style-type: none"> 6.1. Cán bộ đào tạo lựa chọn Khoa Quản trị kinh doanh với tùy chọn Học Kỳ IX (chỉ có với khoa Công nghệ thông tin) <ol style="list-style-type: none"> 6.1.1. Hệ thống hiển thị thông báo Chọn sai Học kỳ 6.1.2. Cán bộ đào tạo chọn lại và tiếp tục các bước sau 	

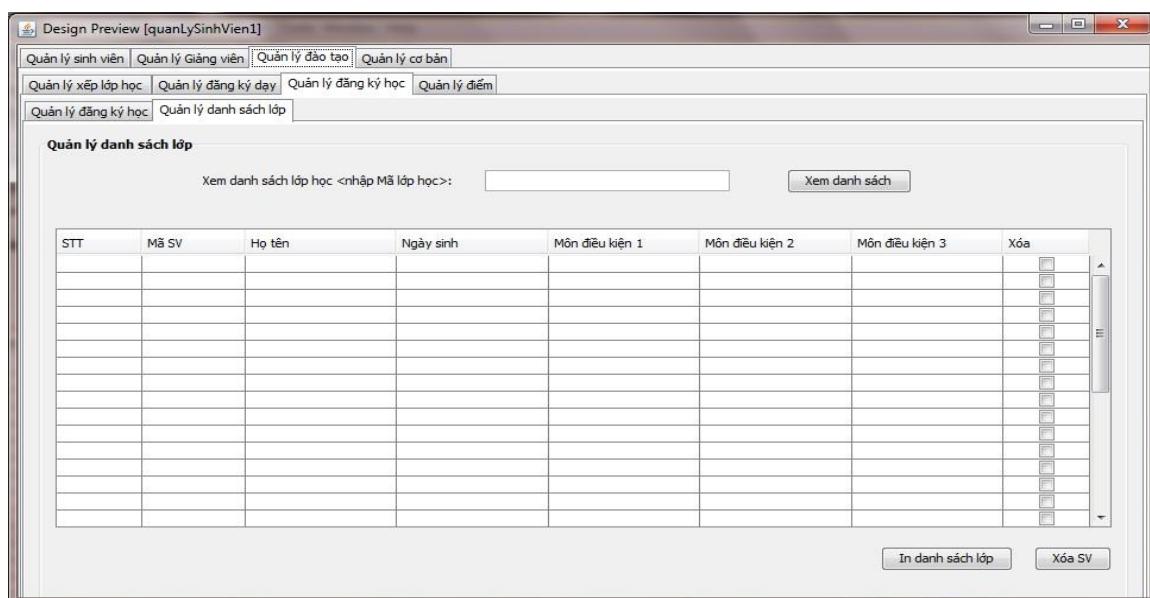
Phác thảo giao diện Quản lý lớp học

Hình B.13: Giao diện quản lý đào tạo

Quản lý danh sách lớp

Tên use case:	Quản lý lớp học
Tác nhân chính	Cán bộ đào tạo
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Cán bộ đào tạo xem thông kê các lớp học thành công
Đảm bảo tối thiểu	Trở lại màn hình lựa chọn tùy chọn cơ bản
Kích hoạt	Người dùng chọn chức năng Quản lý lớp học
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý đăng ký học 2. Hệ thống hiển thị tùy chọn Quản lý lớp học và Quản lý danh sách lớp học 3. Cán bộ đào tạo lựa chọn Quản lý danh sách lớp 4. Hệ thống hiển thị form Quản lý danh sách lớp 5. Cán bộ đào tạo tìm kiếm danh sách một lớp nào đó thông qua Mã lớp học 6. Hệ thống hiển thị thông kê danh sách các Sinh viên đã đăng ký học lớp đó cùng các thông tin liên quan về các môn điều kiện đi kèm 7. Cán bộ đào tạo lựa chọn xóa đối với những Sinh viên không đạt yêu cầu “Pass” các môn điều kiện và chọn Button “Xóa SV” 8. Hệ thống hiển thị tùy chọn có chắc chắn xóa Sinh viên khỏi danh sách lớp hay không 9. Cán bộ đào tạo khẳng định xóa 10. Hệ thống tự động gửi thông báo tới các Sinh viên bị xóa khỏi danh sách và thông báo đã xóa Sinh viên thành công 	
Ngoại lệ	
<ol style="list-style-type: none"> 6.1. Mã lớp học không tìm thấy trong cơ sở dữ liệu <ol style="list-style-type: none"> 6.1.1 Hệ thống hiển thị thông báo không tìm thấy và yêu cầu người dùng nhập lại thông tin tìm kiếm 6.1.2 Cán bộ đào tạo nhập lại và tiếp tục các bước sau 	

Phác thảo giao diện Quản lý danh sách lớp

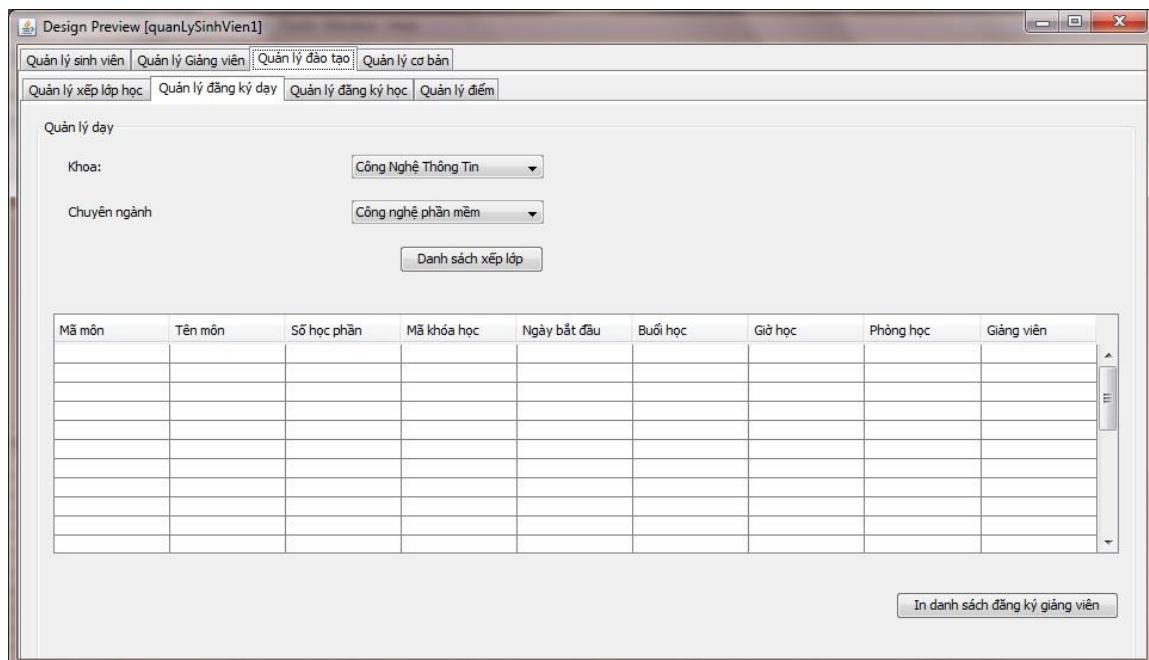


Hình B.14: giao diện Quản lý danh sách lớp

Quản lý đăng ký dạy

Tên use case:	Quản lý lớp dạy
Tác nhân chính	Cán bộ đào tạo
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Cán bộ đào tạo xem thống kê các lớp học thành công
Đảm bảo tối thiểu	Trở lại màn hình lựa chọn tùy chọn cơ bản
Kích hoạt	Người dùng chọn chức năng Quản lý lớp học
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Cán bộ đào tạo kích hoạt form Quản lý đăng ký dạy 2. Hệ thống hiển thị form Quản lý đăng ký dạy với các thông tin tùy chọn 3. Cán bộ đào tạo thay đổi các thông tin tùy chọn về Khoa, Chuyên Ngành và chọn “Danh sách xếp lớp” 4. Hệ thống hiển thị thống kê các lớp học tương ứng với các thông tin gồm có Mã môn, Tên môn, Số học phần, Mã Khóa học, Ngày bắt đầu, Buổi học, Giờ học, Phòng học, Giảng viên. 	
Ngoại lệ <ol style="list-style-type: none"> 4.1. Cán bộ đào tạo lựa chọn Khoa không phù hợp <ol style="list-style-type: none"> 4.1.1. Hệ thống hiển thị thông báo sai 4.1.2. Cán bộ đào tạo chọn lại và tiếp tục các bước sau 	

Phác thảo giao diện Quản lý đăng ký dạy



Hình B.15: Giao diện quản lý đăng ký dạy

Giảng viên đăng ký dạy

Tên use case:	Giảng viên đăng ký môn dạy
Ngữ cảnh	Giảng viên đăng ký môn dạy thành công
Tác nhân chính	Giảng viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Giảng viên đăng ký thành công môn dạy
Đảm bảo tối thiểu	Trở lại màn hình đăng ký để giảng viên có thể đăng ký lại

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Kích hoạt Chuỗi sự kiện chính	Người dùng chọn chức năng đăng ký môn dạy 6. Giảng viên kích hoạt form đăng ký môn dạy 7. Hệ thống hiển thị danh sách và thông tin chi tiết các môn và khóa học giảng viên có thể đăng ký dạy theo chuyên ngành 8. Giảng viên viên lựa chọn các môn dạy và khóa dạy tương ứng với mỗi môn đó 9. Hệ thống kiểm tra và cập nhật thông tin giảng viên đã đăng ký vào cơ sở dữ liệu và hiển thị đăng ký thành công 10. Hệ thống thông kê và hiển thị chi tiết thông tin các môn dạy và khóa dạy mà giảng viên đã đăng ký dưới dạng bảng
Ngoại lệ	9.1 Thông tin đăng ký lỗi: <ul style="list-style-type: none"> • Khóa học đã có giảng viên đăng ký trước đó • Thời gian một số môn dạy giảng viên đăng ký bị trùng nhau • Giảng viên (thỉnh giảng) không được dạy môn đăng ký 9.2 Hệ thống thông báo đăng ký không thành công và quay lại bước 2

Phác thảo giao diện Giáo viên đăng ký môn dạy

Mã môn	Tên môn	Số tín chỉ	Mã lớp	Ngày bắt đầu	Giờ học	Đăng ký dạy
M09	Lập trình hướng đối tượng	4	LTHDT-10	1/8/2010	Thứ 5 - Kíp 3 (12h30-15h00)	<input type="checkbox"/>
M09	Lập trình hướng đối tượng	4	LTHDT-02	1/8/2010	Thứ 5 - Kíp 4 (15h00-17h30)	<input checked="" type="checkbox"/>
M16	Công nghệ phần mềm	4	CNPM-01	1/8/2010	Thứ 2 - Kíp 2 (9h30-12h00)	<input type="checkbox"/>
M16	Công nghệ phần mềm	4	CNPM-02	1/8/2010	Thứ 3 - Kíp 3 (12h30-15h00)	<input checked="" type="checkbox"/>
M24	Phân tích thiết kế	4	PTTK-01	1/8/2010	Thứ 3 - Kíp 4 (15h00-17h30)	<input type="checkbox"/>
M24	Phân tích thiết kế	4	PTTK-02	1/8/2010	Thứ 6 - Kíp 3 (12h30-15h00)	<input checked="" type="checkbox"/>
M24	Phân tích thiết kế	4	PTTK-03	1/8/2010	Thứ 6 - Kíp 4 (15h00-17h30)	<input type="checkbox"/>

Hình B.15: Hình B.15: Giao diện đăng ký dạy

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Giảng viên xem lịch dạy

Tên use case:	Giảng viên xem lịch dạy
Ngữ cảnh	Giảng viên đăng ký môn dạy thành công
Tác nhân chính	Giảng viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Giảng viên đăng ký thành công môn dạy
Đảm bảo tối thiểu	Trở lại màn hình đăng ký để giảng viên có thể đăng ký lại
Kích hoạt	Người dùng chọn chức năng đăng ký môn dạy
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Giảng viên kích hoạt form Xem lịch dạy 2. Hệ thống hiển thị tùy chọn để Giảng viên lựa chọn Chuyên Ngành 3. Giảng viên lựa chọn Chuyên Ngành tương ứng 4. Hệ thống hiển thị danh sách và thông tin chi tiết các Môn học và Lớp học giảng viên đã đăng ký dạy theo chuyên ngành 5. Giảng viên có thể tùy chọn “Thay đổi” hoặc “In lịch giảng dạy” 	
Ngoại lệ	

The screenshot shows a Windows application window titled "Design Preview [SVDK]". The main title bar reads "HỆ THỐNG QUẢN LÝ HỌC TẬP THEO TÍN CHỈ HỌC VIỆN CÔNG NGHỆ VÀ BƯU CHÍNH VIỄN THÔNG". Below the title bar is a navigation menu with tabs: TRANG CHỦ, ĐĂNG KÝ DẠY, XEM LỊCH DẠY, and THAY ĐỔI THÔNG TIN CÁ NHÂN. The "XEM LỊCH DẠY" tab is selected.

The main content area displays a "Thông tin cá nhân" (Personal information) form. It includes fields for: Họ tên: (Name), Vai trò: (Role) - Giảng viên, Mã giảng viên: (Teacher ID) - GV10, Khoa: (Faculty) - Công nghệ thông tin, Chuyên ngành: (Specialty) - Công nghệ phần mềm, and Đăng ký dạy các môn khóa: (Courses taught) - 2007.

Below the form is a section titled "Thời khóa biểu các môn học và thông tin chi tiết lớp học" (Class schedule and detailed class information). It contains two buttons: "Xem thời khóa biểu" (View schedule) and "In thời khóa biểu" (Print schedule).

A table below lists the class schedule:

Mã môn	Tên môn	Số tín chỉ	Mã lớp	Ngày bắt đầu	Giờ học	Phòng học
M16	Công nghệ phần mềm	4	CNPM-01	1/8/2010	Thứ 2 - Kíp 2 (9h30-12h00)	305A3
M16	Công nghệ phần mềm	4	CNPM-02	1/8/2010	Thứ 3 - Kíp 3 (12h30-15h00)	311A3
M24	Phân tích thiết kế	4	PTTK-01	1/8/2010	Thứ 3 - Kíp 4 (15h00-17h30)	209A3
M24	Phân tích thiết kế	4	PTTK-02	1/8/2010	Thứ 6 - Kíp 3 (12h30-15h00)	309A3
M24	Phân tích thiết kế	4	PTTK-03	1/8/2010	Thứ 6 - Kíp 4 (15h00-17h30)	305A3
M32	Phát triển phần mềm hướng Agent	4	AGENT-01	1/8/2010	Thứ 2 - Kíp 3 (12h30-15h00)	211A3
M32	Phát triển phần mềm hướng Agent	4	AGENT-02	1/8/2010	Thứ 5 - Kíp 4 (15h00-17h30)	305A3

Phác thảo giao diện Giảng viên xem lịch giảng dạy

Sinh viên đăng ký môn học

Tên use case:	Sinh viên đăng ký môn học
Tác nhân chính	Sinh viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Sinh viên đăng ký thành công môn học
Đảm bảo tối thiểu	Trở lại màn hình đăng ký để sinh viên có thể đăng ký lại
Kích hoạt	Người dùng chọn chức năng đăng ký môn học
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Sinh viên kích hoạt form đăng ký môn học 2. Hệ thống hiển thị danh sách và thông tin chi tiết các khóa học Sinh viên có thể đăng ký học trong học kỳ gồm có thông tin về Mã môn, Tên môn, Số tín chỉ, Loại môn học, Môn điều kiện, Môn nợ. 3. Sinh viên lựa chọn các môn học và khóa học tương ứng với mỗi môn học 4. Hệ thống kiểm tra và cập nhật thông tin Sinh viên đã đăng ký vào cơ sở dữ liệu và hiển thị đăng ký thành công 5. Hệ thống thống kê và hiển thị chi tiết thông tin các môn học và khóa học mà Sinh viên đã đăng ký dưới dạng bảng 	
Ngoại lệ	
<p>4.1 Thông tin đăng ký không đủ điều kiện:</p> <ul style="list-style-type: none"> • Sinh viên chưa học đủ các môn điều kiện của môn học • Môn học đã đủ lớp và đủ số lượng Sinh viên trong 1 lớp. • Sinh viên đăng ký hai khóa học của cùng một môn học • Thời gian một số buổi học môn học Sinh viên đăng ký bị trùng nhau <p>4.2 Hệ thống thông báo đăng ký không thành công và quay lại bước 2</p>	

Phác thảo giao diện sinh viên đăng ký môn học

Sinh viên xem thời khóa biểu

Tên use case:	Sinh viên xem thời khóa biểu
Tác nhân chính	Sinh viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Sinh viên xem thời khóa biểu thành công
Đảm bảo tối thiểu	Trở lại màn hình chính
Kích hoạt	Người dùng chọn chức năng xem thời khóa biểu
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Sinh viên kích hoạt form Xem thời khóa biểu 2. Hệ thống hiển thị thời khóa biểu tương ứng với những môn học Sinh viên đã đăng ký học trong học kỳ với các thông tin Mã môn, Tên môn, Số tín chỉ, Mã lớp, Ngày bắt đầu, Giờ học, Phòng học, Giảng viên. 	
Ngoại lệ	
<p>2.1. Môn Sinh viên đăng ký không được xếp vào thời khóa biểu do Sinh viên chưa trả nợ đủ các môn điều kiện</p>	

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

**HỆ THỐNG QUẢN LÝ HỌC TẬP THEO TÍN CHỈ
HỌC VIỆN CÔNG NGHỆ VÀ BƯU CHÍNH VIỄN THÔNG**

TRANG CHỦ | THÔNG TIN KHOA/CHUYÊN NGHÀNH | ĐĂNG KÝ HỌC | ĐĂNG KÝ THI | XEM LỊCH THI | CHƯƠNG TRÌNH HỌC | ĐỔI THÔNG TIN CÁ NHÂN

Thông tin cá nhân <p>Họ tên: Sinh viên Vai trò: Sinh viên Mã sinh viên: Khoa: Công nghệ thông tin Chuyên ngành: Công nghệ phần mềm</p>	Thông tin đăng ký học <p>Số tín chỉ tối thiểu: 10 Số tín chỉ tối đa: 35 Hạn đăng ký: 10/7/2010 -> 10/8/2010 Đăng ký học cùng khóa: 2007</p>
---	---

Khoa Công nghệ thông tin - 2007

Danh sách các môn học sinh viên có thể đăng ký trong học kỳ

Sinh viên chỉ được phép đăng ký học một môn khi đã hoàn thành đủ các môn điều kiện

[Hiển thị danh sách môn học]

Bạn đã đăng ký 12 TC trên tổng số tối thiểu 10 TC và tối đa 35 TC

Xem thông tin đăng ký

Mã môn	Tên môn	Số tín chỉ	Khoa phụ trách	Bộ môn phụ trách	Loại môn học	Các môn điều kiện	Môn điều kiện nợ	Đăng k...
M24	Phân tích thiết kế	4	Công nghệ thông tin	Công nghệ phần mềm	Bắt buộc	Lập trình Java, Công nghệ p...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
M25	Lập trình mạng	4	Công nghệ thông tin	Công nghệ phần mềm	Bắt buộc	Lập trình Java, Mạng máy tính	<input checked="" type="checkbox"/>	<input type="checkbox"/>
M26	Cơ sở dữ liệu phân ...	4	Công nghệ thông tin	Công nghệ phần mềm	Bắt buộc	Cơ sở dữ liệu	<input checked="" type="checkbox"/>	<input type="checkbox"/>
M27	An toàn bảo mật H...	3	Công nghệ thông tin	Hệ thống thông tin	Bắt buộc	Hệ thống thông tin, Mạng m...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
M28	Xử lý ảnh	3	Công nghệ thông tin	Công nghệ phần mềm	Tùy chọn	Kỹ thuật đồ họa, Giải tích 1, ...	<input type="checkbox"/>	<input checked="" type="checkbox"/>
M29	Lịch sử Đảng	4	Khoa cơ bản	Chính trị tư tưởng	Tùy chọn	Triết học, Chủ nghĩa xã hội	<input type="checkbox"/>	<input type="checkbox"/>
M30	Tư tưởng Hồ Chí Minh	4	Khoa cơ bản	Chính trị tư tưởng	Tùy chọn	Triết học, Chủ nghĩa xã hội	<input checked="" type="checkbox"/>	<input type="checkbox"/>

<Sinh viên có thể đăng ký học các môn điều kiện còn thiếu cùng với khóa sau>

[Đăng ký]

B.16: Phác thảo giao diện Xem thời khóa biểu

**HỆ THỐNG QUẢN LÝ HỌC TẬP THEO TÍN CHỈ
HỌC VIỆN CÔNG NGHỆ VÀ BƯU CHÍNH VIỄN THÔNG**

TRANG CHỦ | ĐĂNG KÝ HỌC | XEM THỜI KHÓA BIỂU | XEM LỊCH THI | XEM ĐIỂM HỌC TẬP | THAY ĐỔI THÔNG TIN CÁ NHÂN

Thông tin cá nhân <p>Họ tên: Công nghệ thông tin Vai trò: Sinh viên Mã sinh viên:</p>	Khoa: Công nghệ thông tin Chuyên ngành: Công nghệ phần mềm Khóa học: 2007
--	--

Thời khóa biểu các môn học và thông tin chi tiết lớp học

[Xem thời khóa biểu] [In thời khóa biểu]

Mã môn	Tên môn	Số tín chỉ	Mã lớp	Ngày bắt đầu	Giờ học	Phòng học	Giảng viên
M24	Phân tích thiết kế	4	PTTK-02	12/08/2010	Kíp 4 (15h00-17h30)	305A3	Trần Đình Quế
M25	Cơ sở dữ liệu phân tán	4	CSDLPT-02	13/08/2010	Kíp 2 (9h30-12h00)	311A3	Phạm Thế Quế
M27	Quản lý dự án	4	QLDA-01	14/08/2010	Kíp 3 (12h30-15h00)	305A3	Nguyễn Quỳnh Chi
M28	An toàn bảo mật	3	ATBM-02	15/08/2010	Kíp 4 (15h00-17h30)	205A3	Dương Trần Đức
M30	Lập trình mạng	3	LTM-01	11/08/2010	Kíp 2 (9h30-12h00)	305A3	Hà Mạnh Đào
M31	Xử lý ảnh	3	XLA-01	11/08/2010	Kíp 3(12h30-15h00)	305A3	Đỗ Năng Toàn

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Sinh viên xem lịch thi học kỳ

Tên use case:	Sinh viên xem lịch thi học kỳ
Tác nhân chính	Sinh viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Sinh viên xem lịch thi thành công
Đảm bảo tối thiểu	Trở lại màn hình chính
Kích hoạt	Người dùng chọn chức năng xem lịch thi học kỳ
Chuỗi sự kiện chính	
1. Sinh viên kích hoạt form Xem lịch thi học kỳ 2. Hệ thống hiển thị lịch thi các môn tương ứng với những môn học Sinh viên đã đăng ký học trong học kỳ với các thông tin Mã môn, Tên môn, Ngày thi, Giờ thi, Phòng thi.	
Ngoại lệ	
2.1. Môn Sinh viên theo học không được xếp lịch thi do Sinh viên ko đủ điều kiện thi.	

Phác thảo giao diện Xem lịch thi



Sinh viên xem điểm học tập

Tên use case:	Sinh viên xem điểm học tập
Tác nhân chính	Sinh viên
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Sinh viên xem điểm học tập thành công
Đảm bảo tối thiểu	Trở lại màn hình chính
Kích hoạt	Người dùng chọn chức năng xem điểm

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Chuỗi sự kiện chính

1. Sinh viên kích hoạt form Xem điểm học tập
2. Hệ thống hiển thị các thông tin về kết quả học tập của Sinh viên với các thông tin **Tên môn**, **Số tín chỉ**, **Điểm lần 1**, **Điểm lần 2**, **Điểm Tổng kết**.

Ngoại lệ

Người dùng thay đổi thông tin cá nhân

Tên use case:	Người dùng thay đổi thông tin cá nhân
Tác nhân chính	Người dùng hệ thống
Tiền điều kiện	Đăng nhập hệ thống thành công
Đảm bảo thành công	Người dùng thay đổi thông tin thành công
Đảm bảo tối thiểu	Trở lại màn hình chính
Kích hoạt	Người dùng chọn chức năng thay đổi thông tin cá nhân

Chuỗi sự kiện chính

1. Sinh viên kích hoạt form Thay đổi thông tin cá nhân
2. Hệ thống hiển thị các thông tin cá nhân chi tiết của người dùng
3. Người dùng lựa chọn "Thay đổi thông tin"
4. Hệ thống hiển thị các thông tin chi tiết của người dùng trên form nhập liệu
5. Người dùng thay đổi thông tin và chọn "Lưu thông tin"
6. Hệ thống thông báo thông tin đã được thay đổi thành công

Ngoại lệ

Phác thảo giao diện thay đổi Thông tin cá nhân

B.2 PHÂN TÍCH TÍNH

Xác định lớp

Để xác định các lớp thực thể ta dùng kỹ thuật trích danh từ trong ca sử dụng và kịch bản. Các danh từ thu được từ các kịch bản là:

Hệ thống học tập tín chỉ, Khoa, Mã Khoa, Tên Khoa, Trưởng Khoa, Chuyên Ngành, Mã chuyên ngành, Tên Chuyên ngành, Trưởng bộ môn, Khoa phụ trách, Môn Học, Mã môn, Tên môn, Số tín chỉ, Số tiết Lý thuyết, Số tiết Thực hành, Số tiết bài tập, Môn điều kiện, Sinh viên, Mã SV, Họ tên, Ngày sinh, Giới tính, Khóa, Lớp, Giảng viên, Mã GV, Họ tên, Ngày sinh, Giới tính, Bộ Môn, Học vị, Mã môn, Tên môn, Mã lớp, Ngày bắt đầu, Buổi học, Giờ học, Phòng học, Sĩ Số, Số đăng ký, thời khóa biểu, lịch thi, kết quả học tập.

Loại bỏ các danh từ nằm ngoài phạm vi mục đích của hệ thống và các danh từ hoặc cụm từ trùng lặp và các danh từ làm thuộc tính của lớp như:

Tên, mã, ngày sinh, địa chỉ, giới tính: là thuộc tính của các lớp Người, Sinh viên, Giáo viên, Người quản lý.

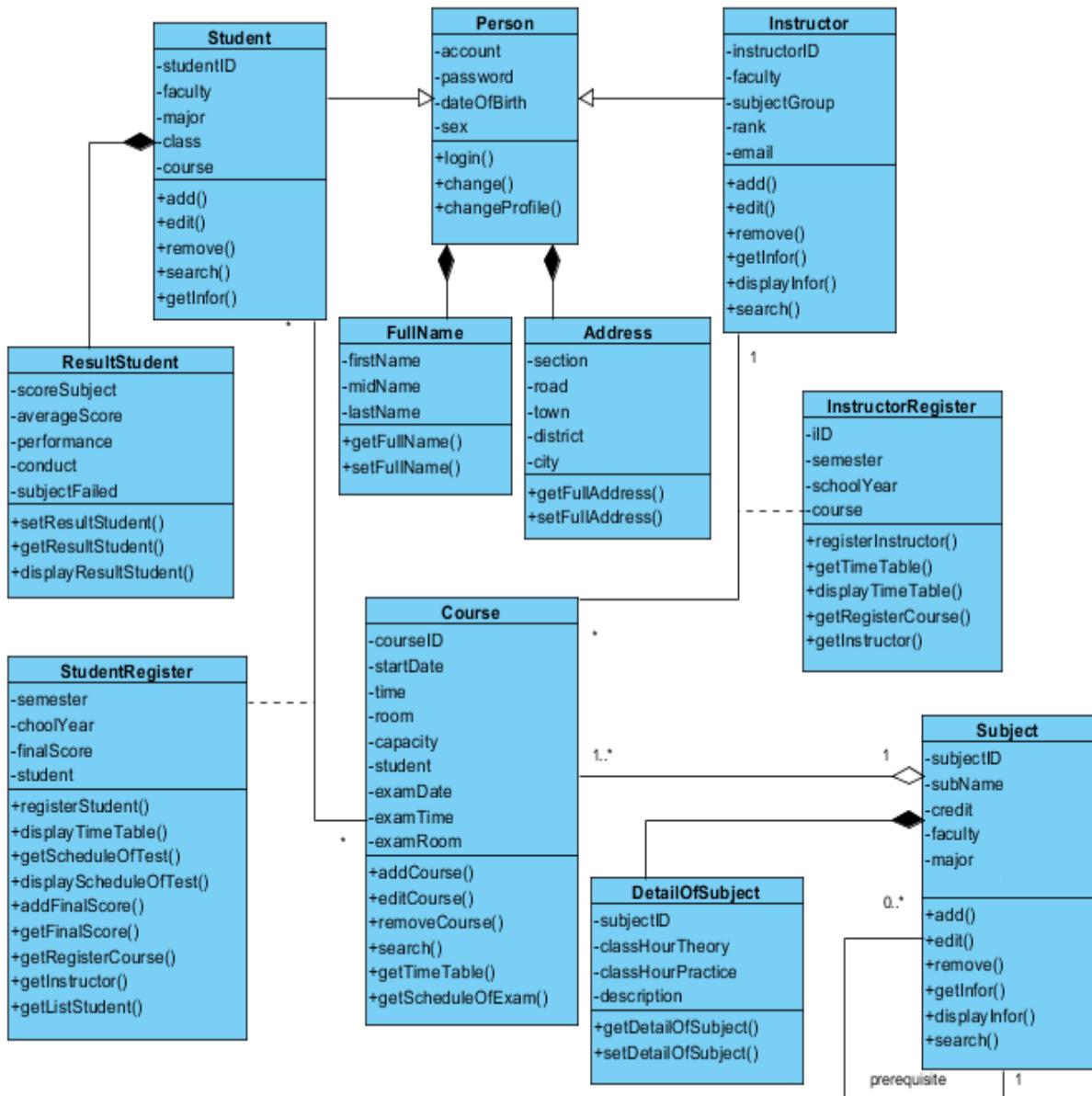
Ngành, khoa: là thuộc tính của Sinh viên, Giáo viên.

Loại môn học, số tiết, học kỳ, số tín chỉ: là thuộc tính của lớp Môn học.

Vậy các danh từ sau có thể là ứng viên các lớp thực thể: *People* (Người), *Name* (Họ tên), *Address* (Địa chỉ), *Student* (Sinh viên), *Instructor* (Giảng viên), *Subject* (Môn học), *RegisterStudent* (Môn học Đăng ký), *RegisterInstructor* (Môn dạy đăng ký), *Course* (Lớp học phần), *ResultStudent* (kết quả học tập).

Quan hệ giữa các lớp

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ



Trong phần này chúng ta chỉ liệt kê thuộc tính và phương thức cho các lớp thực thể

	Định nghĩa	Chứa các thuộc tính cơ bản của các đối tượng là Người trong hệ thống
Person	Thuộc tính	<ul style="list-style-type: none"> ❖ account: tên tài khoản đăng nhập vào hệ thống. ❖ password: mật khẩu đăng nhập hệ thống ❖ dateOfBirth: ngày tháng năm sinh ❖ sex: giới tính: Nam = 1 ; Nữ = 0
	Phương thức	<ul style="list-style-type: none"> ❖ login(account,password): mỗi người sử dụng hệ thống đều phải đăng nhập sử dụng tài khoản và mật khẩu riêng của mình. Phương thức này trả về giá trị True nếu đăng nhập thành công, False nếu đăng nhập không thành công

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

		<ul style="list-style-type: none"> ❖ change(password): người dùng sau khi đăng nhập thành công có thể thực hiện đổi mật khẩu ❖ changeProfile(account): người dùng sau khi đăng nhập thành công có thể thay đổi thông tin cá nhân của mình 		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="background-color: #ADD8E6;">FullName</th> </tr> <tr> <td> -firstName : String -midName : String -lastName : String +getFullName() : void +setFullName() : void </td> </tr> </table>	FullName	-firstName : String -midName : String -lastName : String +getFullName() : void +setFullName() : void	Định nghĩa Có quan hệ kiểu hợp thành (composition) với lớp Người, việc tách thành lớp Họ tên phục vụ cho việc quản lý và tìm kiếm dễ dàng hơn	Thuộc tính ❖ firstName, midName, lastName : ba thuộc tính Họ, Đệm, Tên tương ứng với từng trường trong họ tên đầy đủ
FullName				
-firstName : String -midName : String -lastName : String +getFullName() : void +setFullName() : void				
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="background-color: #ADD8E6;">Address</th> </tr> <tr> <td> -section : String -road : String -town : String -district : String -city : String +getFullAddress() : void +setFullAddress() : void </td> </tr> </table>	Address	-section : String -road : String -town : String -district : String -city : String +getFullAddress() : void +setFullAddress() : void	Định nghĩa Có quan hệ kiểu hợp thành (composition) với lớp Người, , việc tách thành lớp Địa chỉ phục vụ cho việc quản lý và tìm kiếm dễ dàng hơn	Phương thức ❖ getFullName() : lấy ra Họ tên đầy đủ ❖ setFullName() : gán các trường thành Họ tên đầy đủ
Address				
-section : String -road : String -town : String -district : String -city : String +getFullAddress() : void +setFullAddress() : void				
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="background-color: #ADD8E6;">Student</th> </tr> <tr> <td> -studentID : String -faculty : String -major : String -class : String -course : ArrayList<Course> +add(Student) : void +remove(Student) : void +edit(Student) : void +getInfor(Student) : void +displayInfor(Student) : void +search(Student) : void </td> </tr> </table>	Student	-studentID : String -faculty : String -major : String -class : String -course : ArrayList<Course> +add(Student) : void +remove(Student) : void +edit(Student) : void +getInfor(Student) : void +displayInfor(Student) : void +search(Student) : void	Định nghĩa Lớp Sinh Viên sẽ kế thừa từ lớp Người và mang đầy đủ thuộc tính của lớp Người	Thuộc tính ❖ sID : mã sinh viên ❖ faculty : khoa sinh viên theo học ❖ major : chuyên ngành học tập của sinh viên ❖ class : mỗi sinh viên sau khi nhập học sẽ được xếp vào một lớp để quản lý
Student				
-studentID : String -faculty : String -major : String -class : String -course : ArrayList<Course> +add(Student) : void +remove(Student) : void +edit(Student) : void +getInfor(Student) : void +displayInfor(Student) : void +search(Student) : void				
		Phương thức ❖ add(Student) : thêm sinh viên vào trong cơ sở dữ liệu ❖ edit(Student) : sửa thông tin sinh viên trong cơ sở dữ liệu ❖ remove(Student) : xóa sinh viên khỏi cơ sở dữ liệu ❖ getInfor(Student) : lấy thông tin của sinh viên trong cơ sở dữ liệu ❖ displayInfor(Student) : hiển thị thông tin sinh viên trên giao diện. ❖ search(Student) : tìm kiếm sinh viên		

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Instructor -instructorID : String -faculty : String -subjectGroup : String -rank : String -email : String +add(Instructor) : void +edit(Instructor) : void +remove(Instructor) : void +getInfor(Instructor) : void +displayInfor(Instructor) : void +search(Instructor) : void	Định nghĩa Lớp Giảng Viên sẽ kế thừa từ lớp Người và mang đầy đủ thuộc tính của lớp Người	Thuộc tính <ul style="list-style-type: none"> ❖ iID: mã Giảng Viên ❖ faculty: giảng viên thuộc khoa nào ❖ subjectGroup: Bộ môn ❖ rank: học vị của giảng viên ❖ email: địa chỉ email giảng viên 	Phương thức <ul style="list-style-type: none"> ❖ add(Instructor): thêm giảng viên vào trong cơ sở dữ liệu ❖ edit(Instructor): sửa thông tin giảng viên trong cơ sở dữ liệu ❖ remove(Instructor): xóa giảng viên khỏi cơ sở dữ liệu ❖ getInfor(Instructor): lấy thông tin của giảng viên trong cơ sở dữ liệu ❖ displayInfor(Instructor): Hiển thị thông tin Giảng viên trên giao diện ❖ search(Instructor): tìm kiếm Giảng Viên
	Định nghĩa Mỗi sinh viên khi đăng ký học một môn sẽ tương ứng với một Lớp giảng . Tương tự mỗi Giảng Viên cũng có thể đăng ký dạy theo các Lớp giảng phù hợp tương ứng với môn giảng viên đăng ký dạy.	Thuộc tính <ul style="list-style-type: none"> ❖ courseID: mã Lớp giảng tương ứng ❖ startDate: ngày bắt đầu học ❖ time: thời gian học ❖ room: phòng học ❖ capacity: sĩ số lớp học ❖ Student: mảng danh sách các sinh viên tham gia lớp giảng ❖ examDate: ngày thi kết thúc môn học ❖ examTime: giờ thi kết thúc môn học ❖ examRoom: phòng thi kết thúc môn học 	Phương thức <ul style="list-style-type: none"> ❖ addCourse(Subject): thêm một lớp giảng mới tương ứng với một Môn học. ❖ editCourse(Subject): sửa thông tin một lớp giảng ❖ removeCourse(Subject): xóa một lớp giảng trong cơ sở dữ liệu ❖ getInfor(Course): Lấy thông tin chi tiết của toàn bộ lớp giảng bao gồm môn học, ngày học, giờ học, phòng học ❖ search(Course): tìm kiếm thông tin lớp giảng ❖ getTimeTable(Course): lấy thông tin về thời gian và phòng học của lớp học ❖ getScheduleOfExam(Course): lấy thông tin về thời gian và phòng thi cuối kỳ
Course -courseID : String -startDate : Date -time : String -room : String -capacity : int -student : ArrayList<Student> -examDate : Date -examTime : String -examRoom : String +addCourse(Subject) : void +editCourse(Subject) : void +removeCourse(Subject) : void +search(Course) : void +getTimeTable(Course) : void +getScheduleOfExam(Course) : void			

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

<pre> Subject -subjectID : String -subName : String -credit : int -faculty : String -major : String +addSubject() : void +editSubject() : void +removeSubject() : void +getInfor(Subject) : void +displayInfor(Subject) : void +search(Subject) : void </pre>	Định nghĩa	<p>Sinh viên sẽ phải theo học các môn học trong mỗi theo thứ tự trong mỗi học kỳ để tích lũy tín chỉ. Thông tin chi tiết về các môn được thể hiện qua các thuộc tính và phương thức của lớp Môn Học</p>
	Thuộc tính	<ul style="list-style-type: none"> ❖ subjectID: mã môn học ❖ subName: tên môn học ❖ credit: số tín chỉ của môn học ❖ faculty: môn học thuộc khoa nào ❖ major: môn học thuộc chuyên ngành nào
	Phương thức	<ul style="list-style-type: none"> ❖ add(Subject): thêm môn học vào cơ sở dữ liệu ❖ edit(Subject): sửa lại các thông tin về môn học ❖ remove(Subject): xóa môn học khỏi cơ sở dữ liệu ❖ getInfor(Subject): lấy thông tin chi tiết về môn học. ❖ displayInfor(Subject): hiển thị thông tin môn học trên giao diện ❖ search(Subject): tìm kiếm thông tin Môn học
<pre> DetailOfSubject -subjectID : String -classHourTheory : String -classHourPractice : String -description : String +getDetailSubject() : void +setDetailSubject() : void </pre>	Định nghĩa	<p>Vì thuộc tính của Môn học nhiều, trong thực tế không phải lúc nào cũng cần truy cập đến, các thuộc tính như hình thức thi, số tiết lý thuyết, bài tập... sinh viên chỉ quan tâm khi đã đăng ký học khóa học môn đó trong học kỳ, nên tách thêm lớp ChiTietMonHoc</p>
	Thuộc tính	<ul style="list-style-type: none"> ❖ subjectID: mã môn học ❖ classHourTheory: số tiết lý thuyết ❖ classHourPractice: số tiết thực hành
	Phương thức	<ul style="list-style-type: none"> ❖ getDetailSubject(): xem thông tin Chi tiết về Môn học ❖ setDetailOfSubject(): sửa thông tin Chi tiết về Môn học.

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

StudentRegister <pre>-semester : String -schoolYear : int -finalScore : float</pre> <pre>+registerStudent(Course) : void +displayTimeTable(Student) : void +getScheduleOfTest(Subject) : void +displayScheduleOfTest(Subject) : void +addFinalScore(studentID, subjectID) : void +getFinalScore(studentID, subjectID) : void +getRegisterCourse(Student) : void +getInstructor(Course) : void +getListStudent(Course) : void</pre>	Định nghĩa Là lớp liên kết giữa lớp Sinh Viên và Khóa học, chứa các phương thức gọi tới thuộc tính của cả hai lớp Sinh Viên và Khóa học.	
	Thuộc tính <ul style="list-style-type: none"> ❖ semester: học kỳ ❖ schoolYear: năm học ❖ finalScore: điểm tổng kết cuối cùng của môn học sinh viên đăng ký 	
	Phương thức <ul style="list-style-type: none"> ❖ enrollStudent(Course): đăng ký học khóa học của một môn nào đó. Tên sinh viên sẽ được thêm vào danh sách sinh viên của khóa học đó và cập nhật vào cơ sở dữ liệu ❖ displayTimeTable(Student): hiển thị thời khóa biểu học các môn của sinh viên trên giao diện ❖ getScheduleOfTest(Subject): lấy lịch thi của các môn học mà sinh viên đăng ký học trong học kỳ ❖ displayScheduleOfTest(Subject): hiển thị lịch thi học kỳ trên giao diện ❖ addFinalScore(sID,subjected): thêm điểm tổng cuối kỳ của môn học Sinh viên đăng ký ❖ getFinalSocre(sID,subjectID): lấy điểm tổng kết cuối cùng của môn học tương ứng với lớp học Sinh viên đăng ký ❖ getRegisterCourse(Student): lấy những khóa học mà sinh viên đã từng đăng ký học ❖ getListStudent(Course): lấy danh sách những sinh viên trong một lớp học. 	
InstructorRegister <pre>-iID : String -semester : String -schoolYear : int -course : ArrayList<Course></pre> <pre>+registerInstructor(Course) : void +getTimeTable(Instructor) : void +displayTimeTable(Instructor) : void +getRegisterCourse(Instructor) : void +getInstructor(Course) : void</pre>	Định nghĩa Là lớp liên kết giữa lớp Giảng Viên và Khóa học, chứa các phương thức gọi tới thuộc tính của cả hai lớp Giảng Viên và Khóa học	
	Thuộc tính <ul style="list-style-type: none"> ❖ iID: mã giảng viên ❖ semester: học kỳ ❖ schoolYear: năm học ❖ Course: danh sách các Lớp giảng mà giảng viên đăng ký dạy 	
	Phương thức <ul style="list-style-type: none"> ❖ setInstructor(Course): giảng viên đăng ký dạy Lớp giảng của một môn nào đó. Phương thức này sẽ gán Mã giảng viên vào trong cơ sở dữ liệu tương ứng với trường Giảng viên của mỗi bản ghi về Lớp giảng ❖ getTimeTable(Instructor): lấy thông tin về lịch dạy của các Lớp giảng mà Giảng viên đã đăng ký ❖ displayTimeTable(Instructor): hiển thị 	

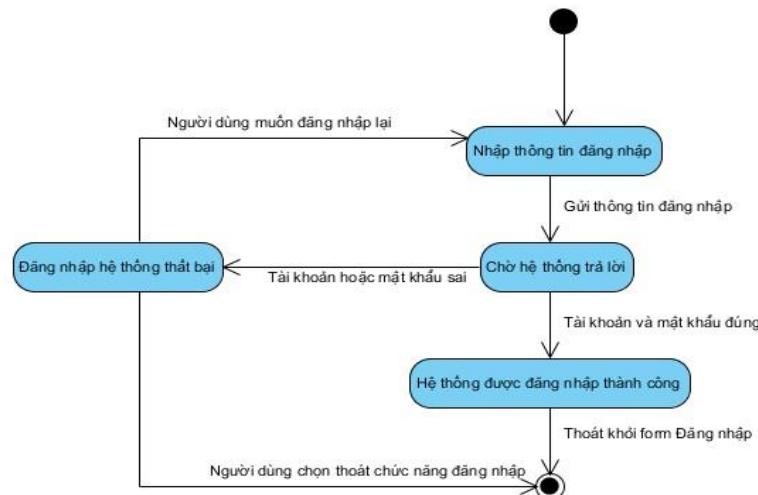
PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

		<p>lịch dạy của giảng viên trên giao diện</p> <ul style="list-style-type: none"> ❖ getRegisterCourse(Instructor): lấy danh sách những lớp học mà giảng viên đã đăng ký dạy ❖ getInstructor(Course): lấy thông tin về giảng viên của một lớp học
ResultStudent <pre>-scoreSubject : ArrayList<Subject> -averageScore : float -performance : String -conduct : String -subjectFailed : ArrayList<Subject> +setResultStudent(studentID) : void +getResultStudent(studentID) : void +displayResultStudent(studentID) : void</pre>	Định nghĩa Thuộc tính Phương thức	<p>Vì thuộc tính của sinh viên rất nhiều, trong khi kết quả họ tập là những thuộc tính không phải lúc nào cũng cần truy cập đến trong hệ thống quản lý họa tập theo tín chỉ, do vậy tách ra thành một lớp riêng KetQuaHocTap</p> <ul style="list-style-type: none"> ❖ scoreSubject: một mảng các môn học sinh viên đã pass và điểm tổng kết tương ứng các môn đó ❖ averageScore: điểm tổng kết trung bình của sinh viên tính đến thời điểm hiện tại ❖ performance: học lực ❖ conduct: hành kiểm ❖ SubjectFailed: danh sách các môn sinh viên đã học nhưng chưa pass <ul style="list-style-type: none"> ❖ getResultStudent(sID): thống kê kết quả học tập các môn của sinh viên ❖ displayResultStudent(sID): hiển thị bảng điểm học tập của sinh viên trên giao diện

B3. PHÂN TÍCH ĐỘNG

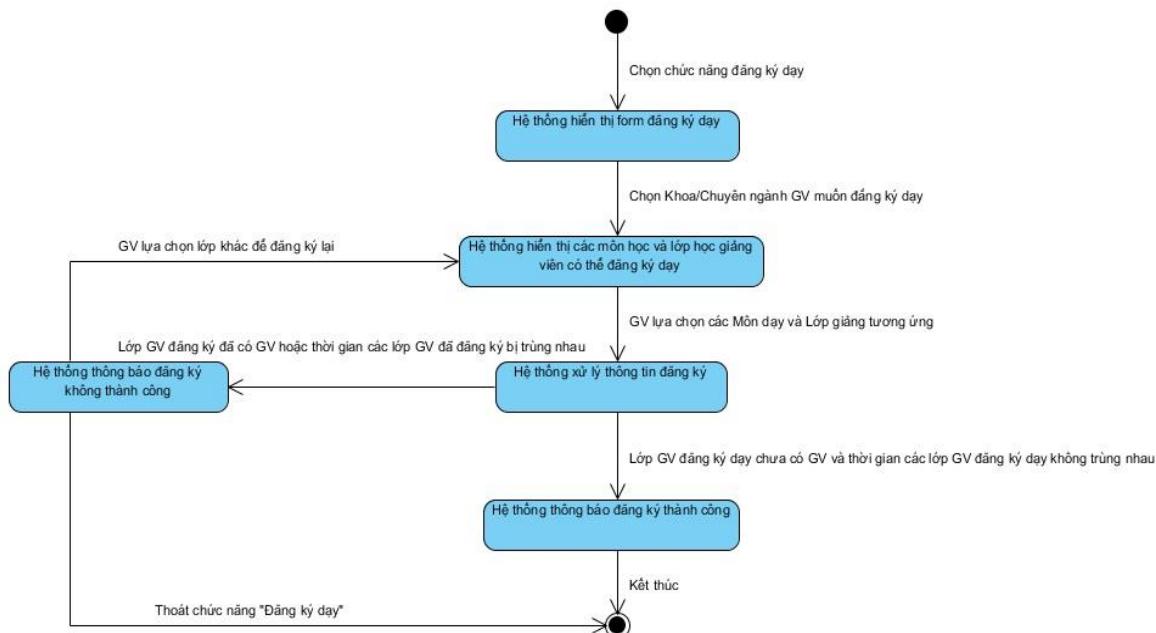
Biểu đồ trạng thái

Đăng nhập

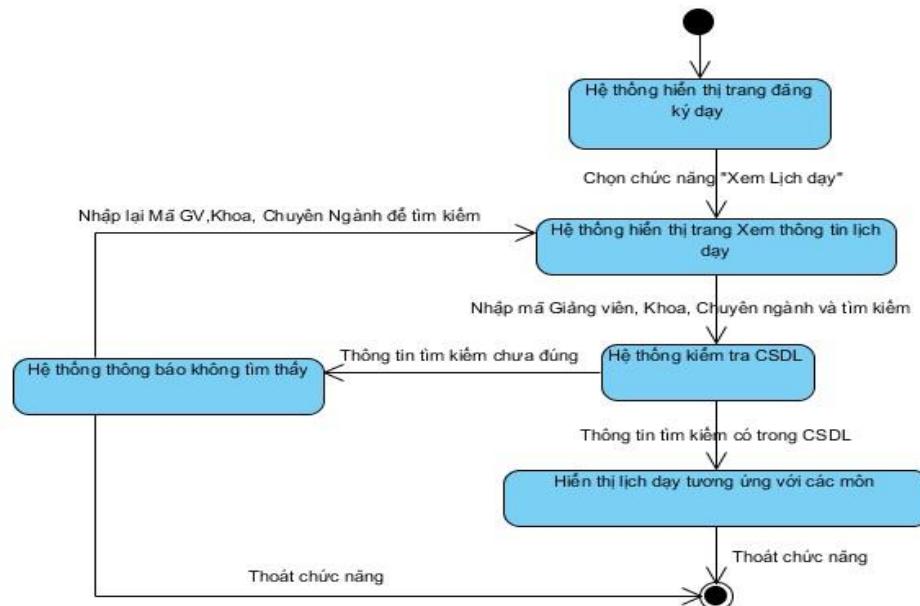


PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

Giảng viên đăng ký dạy

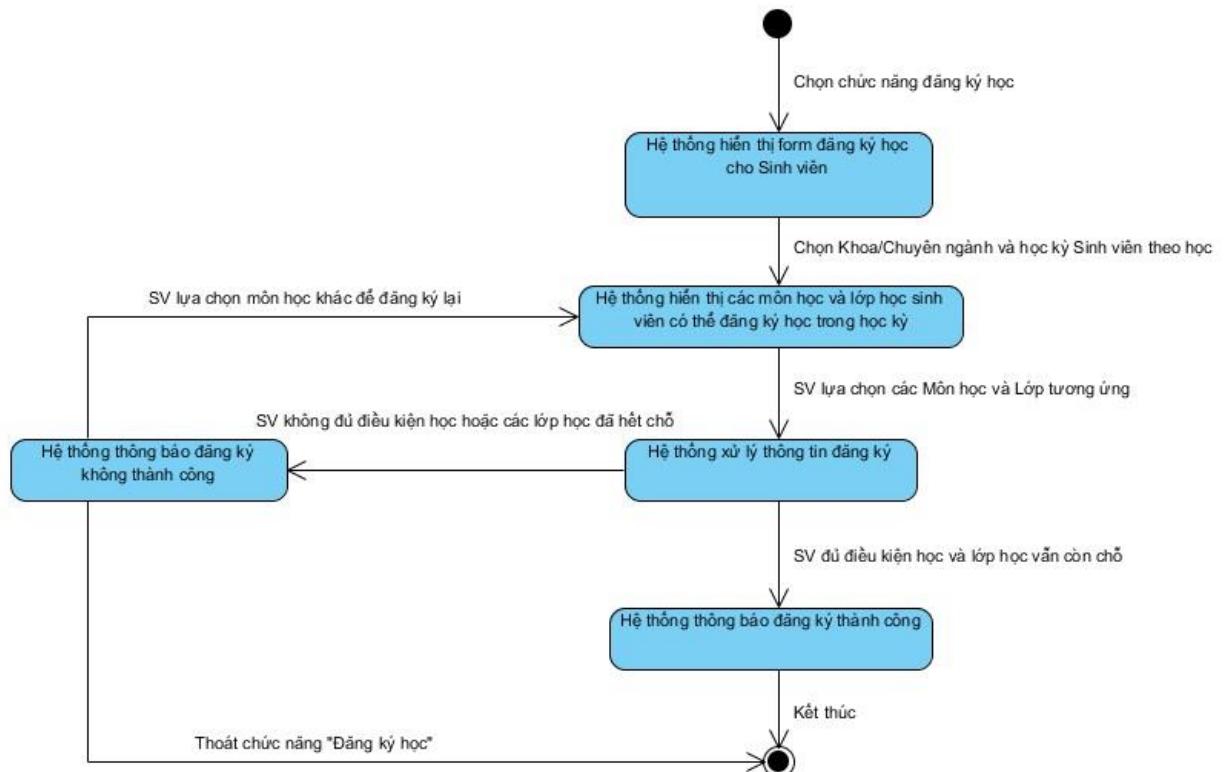


Giảng viên xem lịch dạy

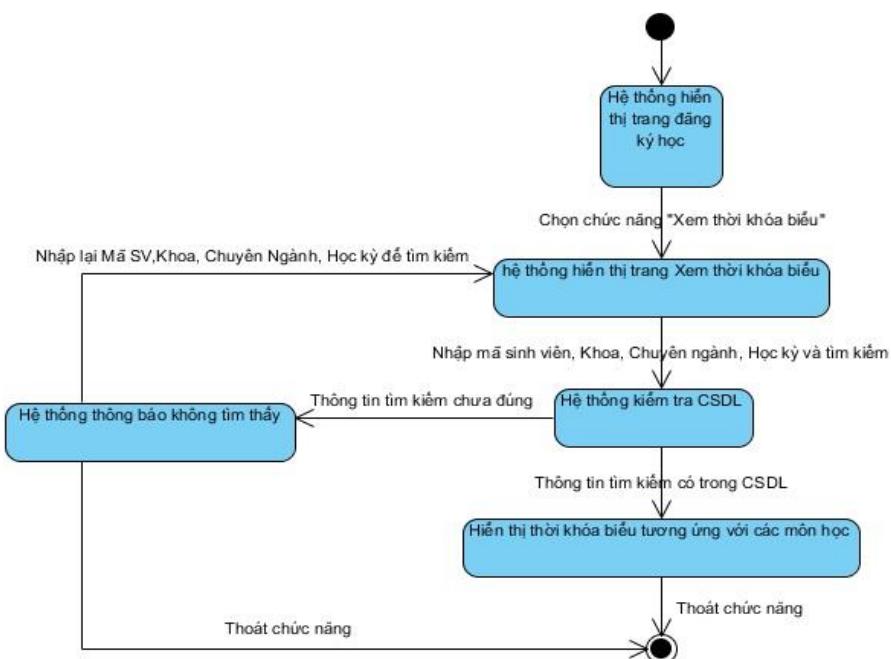


Sinh viên đăng ký học

PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ



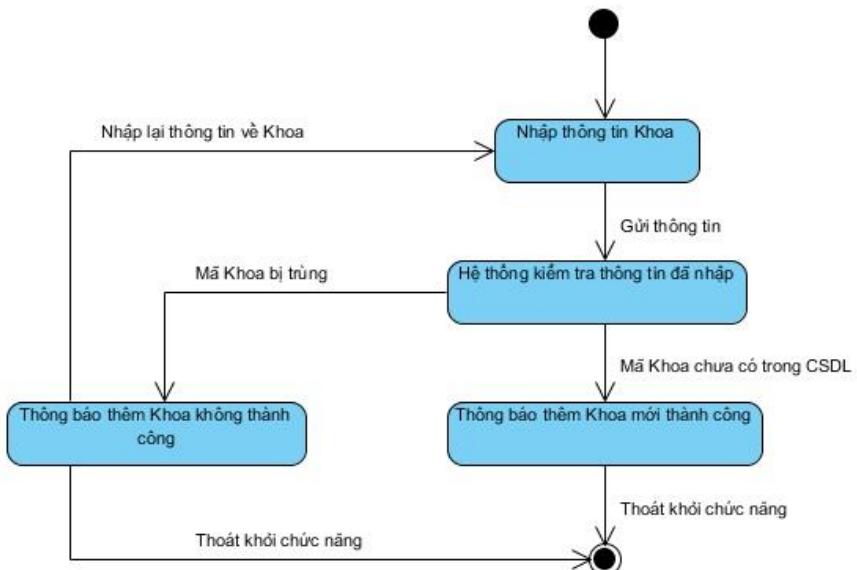
5.Sinh viên xem thời khóa biểu



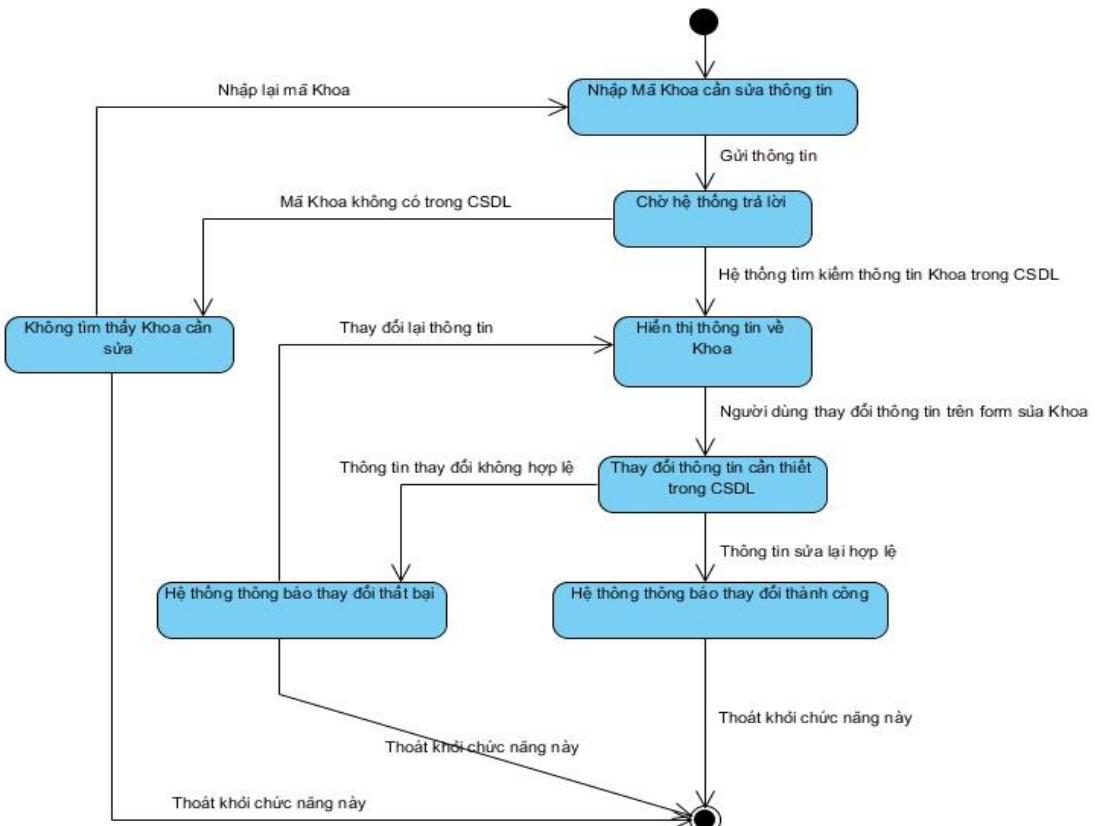
Nhân viên Quản lý cơ bản

Quản lý Khoa

Thêm Khoa

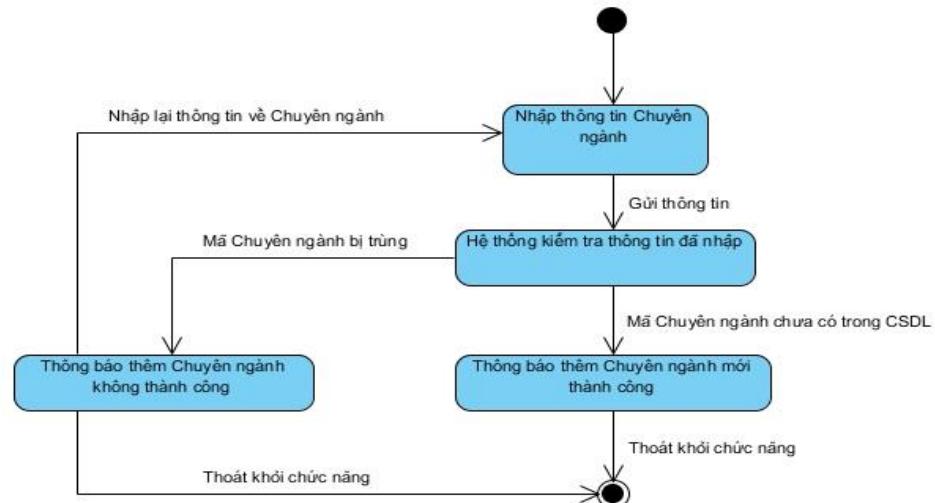


Sửa Khoa

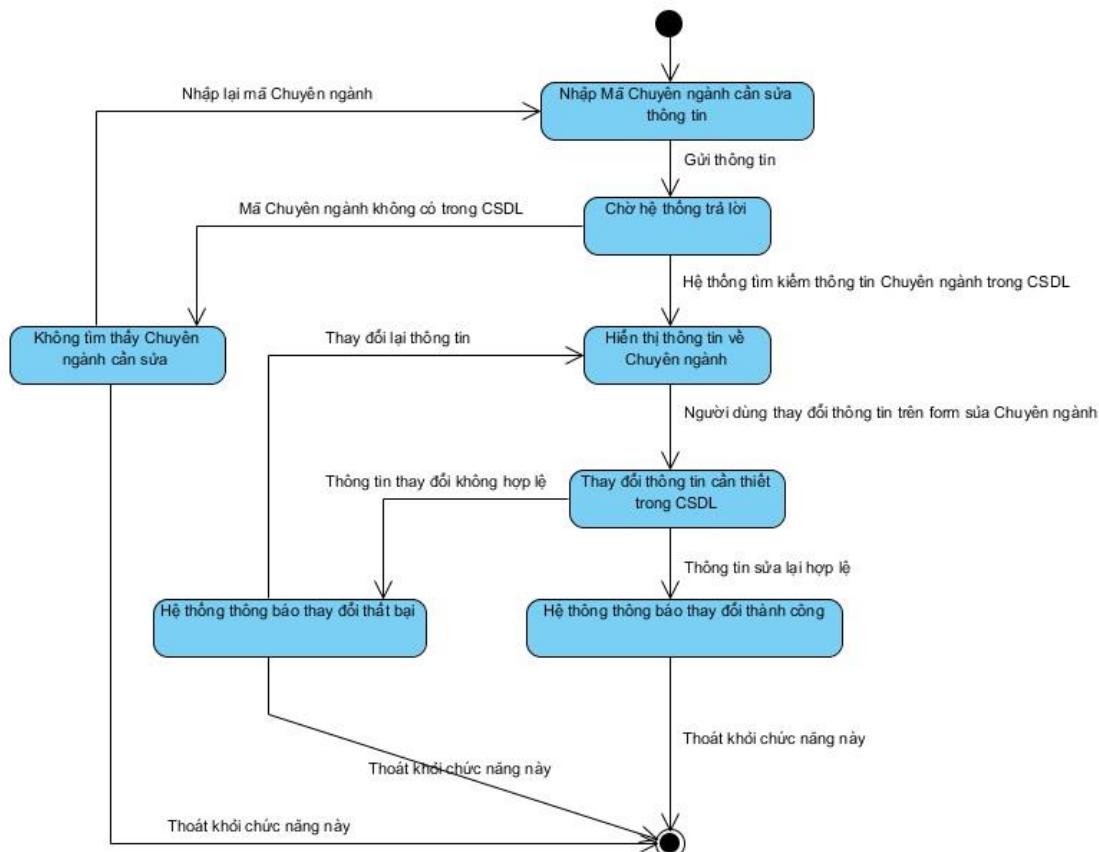


Quản lý chuyên ngành

Thêm Chuyên Ngành

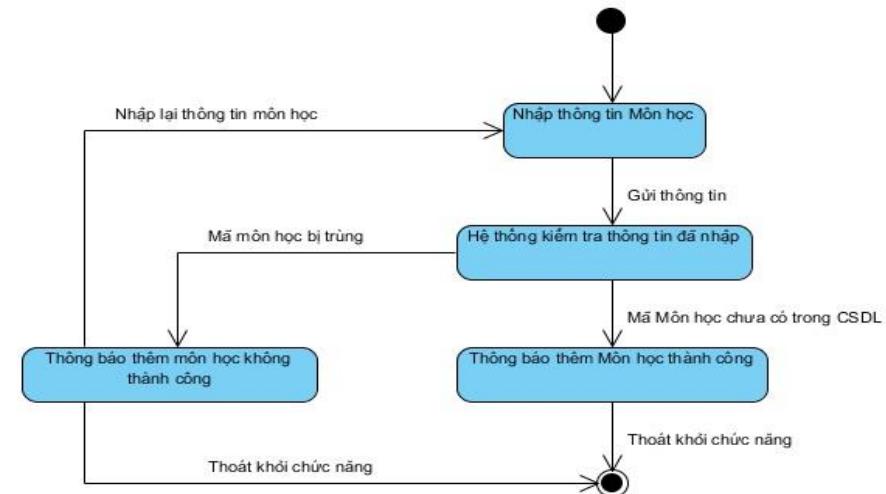


Sửa Chuyên Ngành

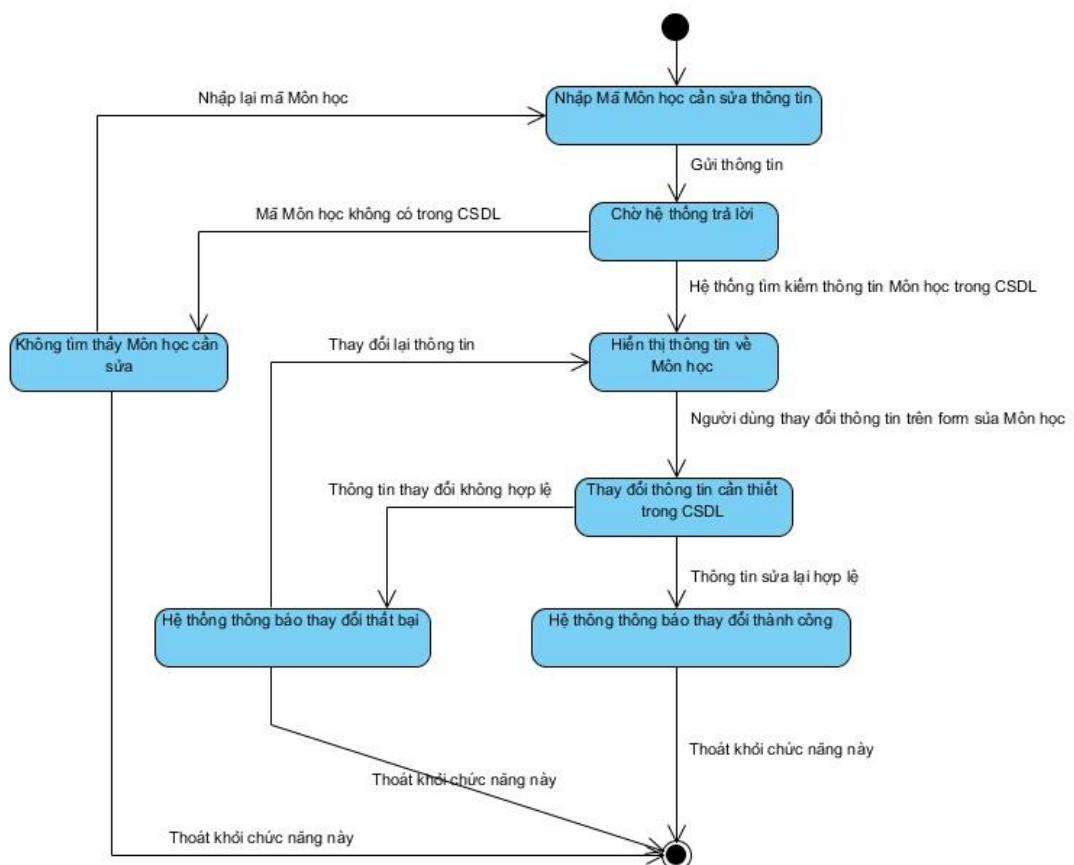


Quản lý môn học

Thêm Môn học

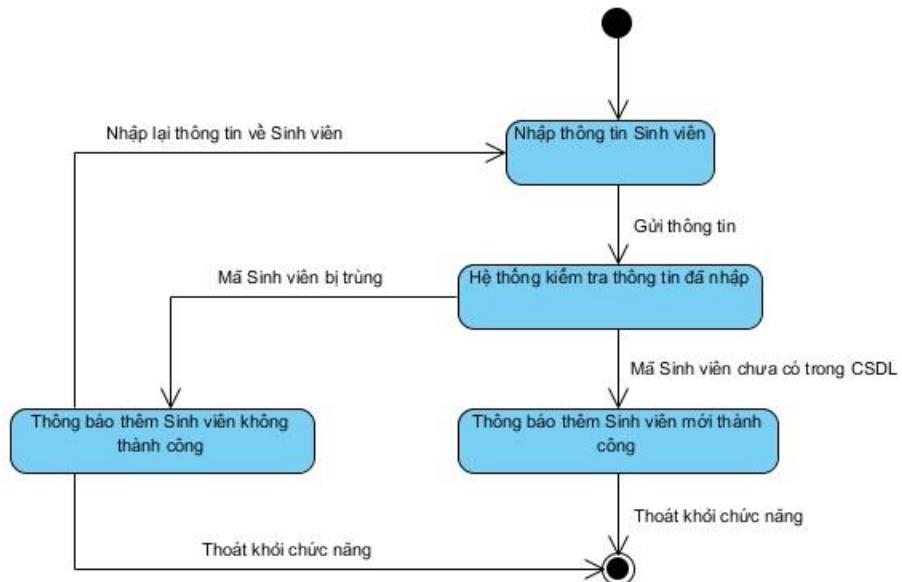


Sửa Môn học

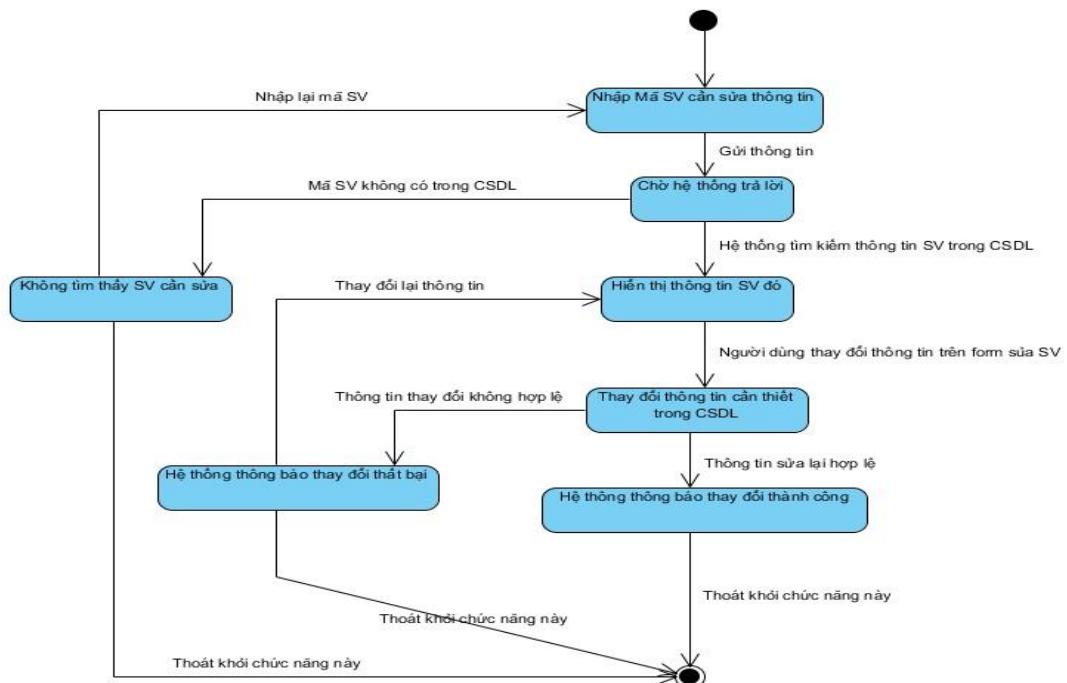


Quản lý sinh viên

Thêm Sinh viên

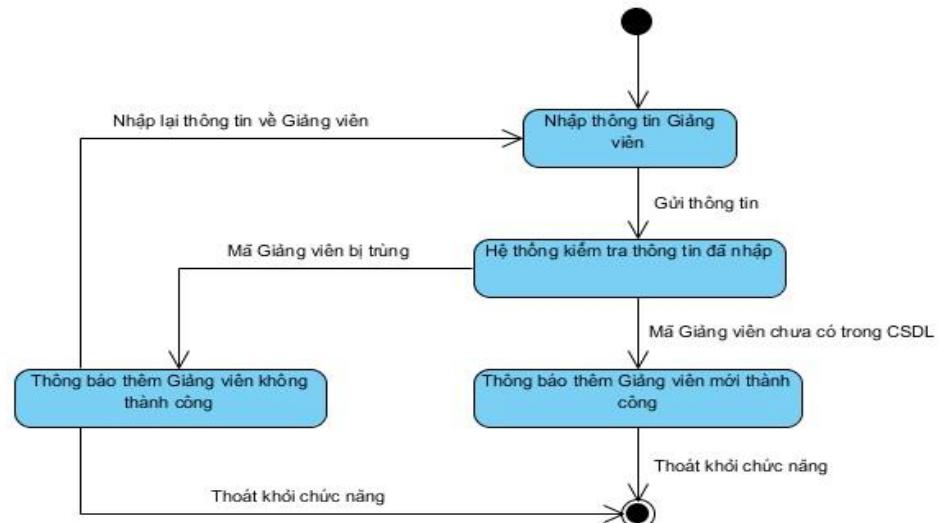


Sửa Sinh viên

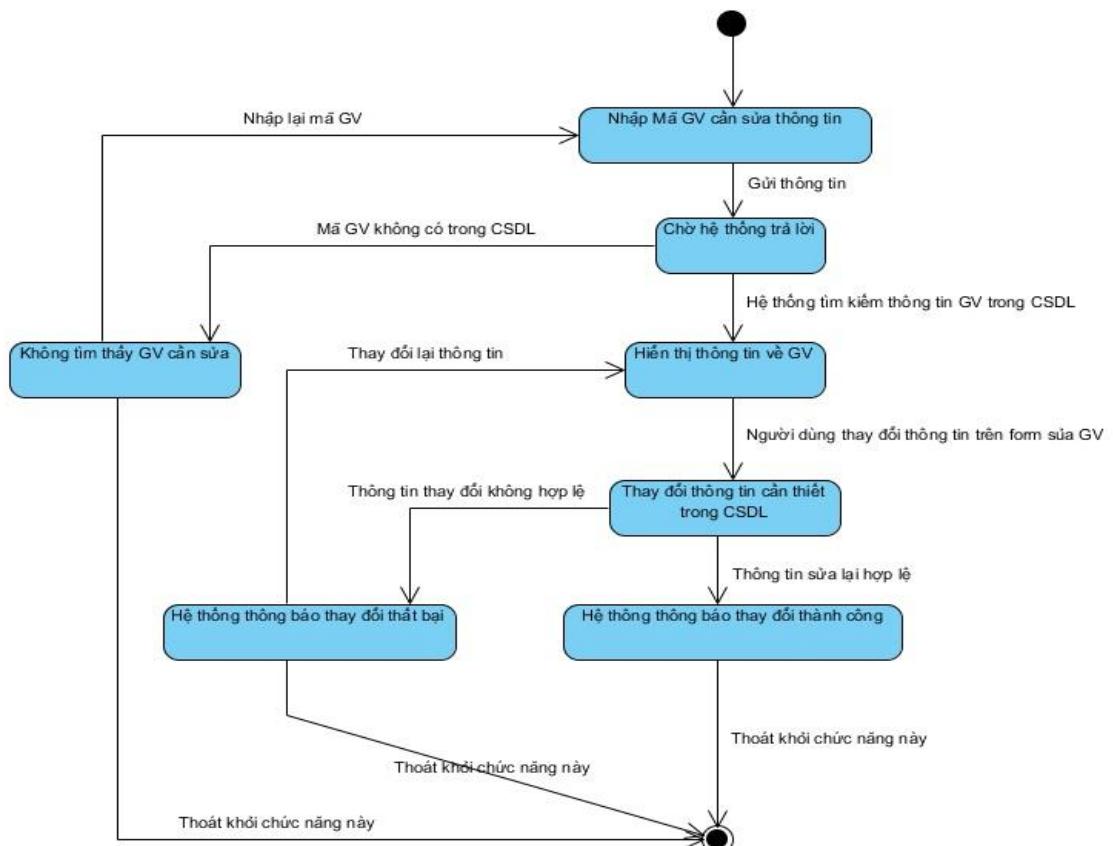


Quản lý giáo viên

Thêm Giảng viên

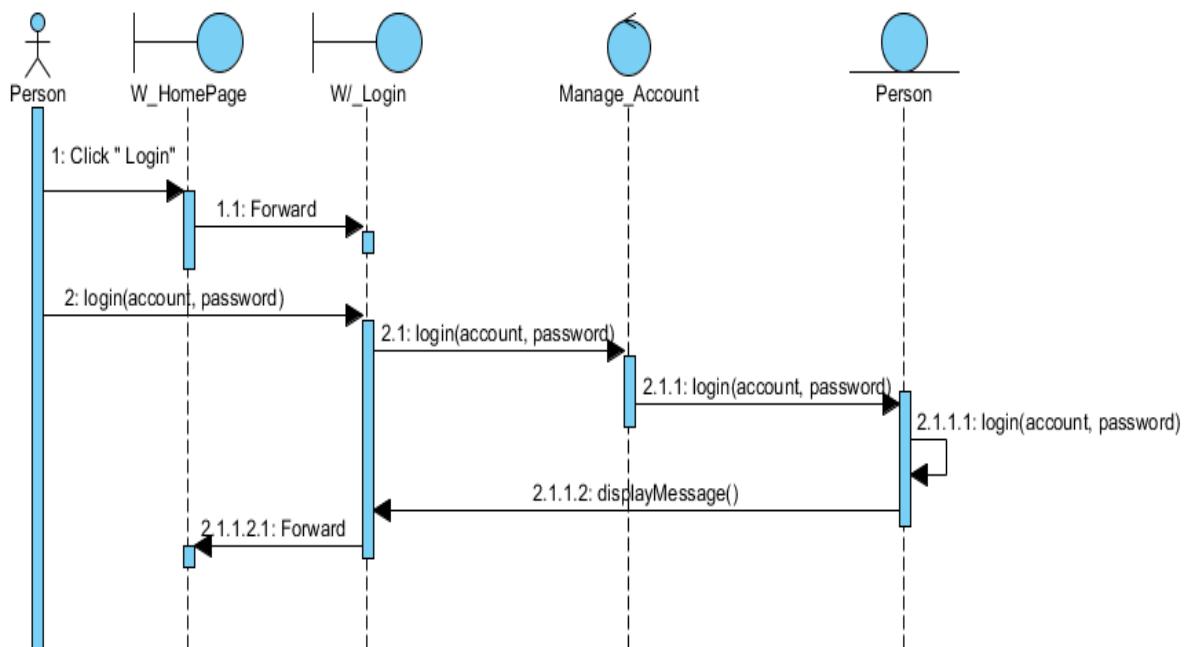


Sửa giảng viên

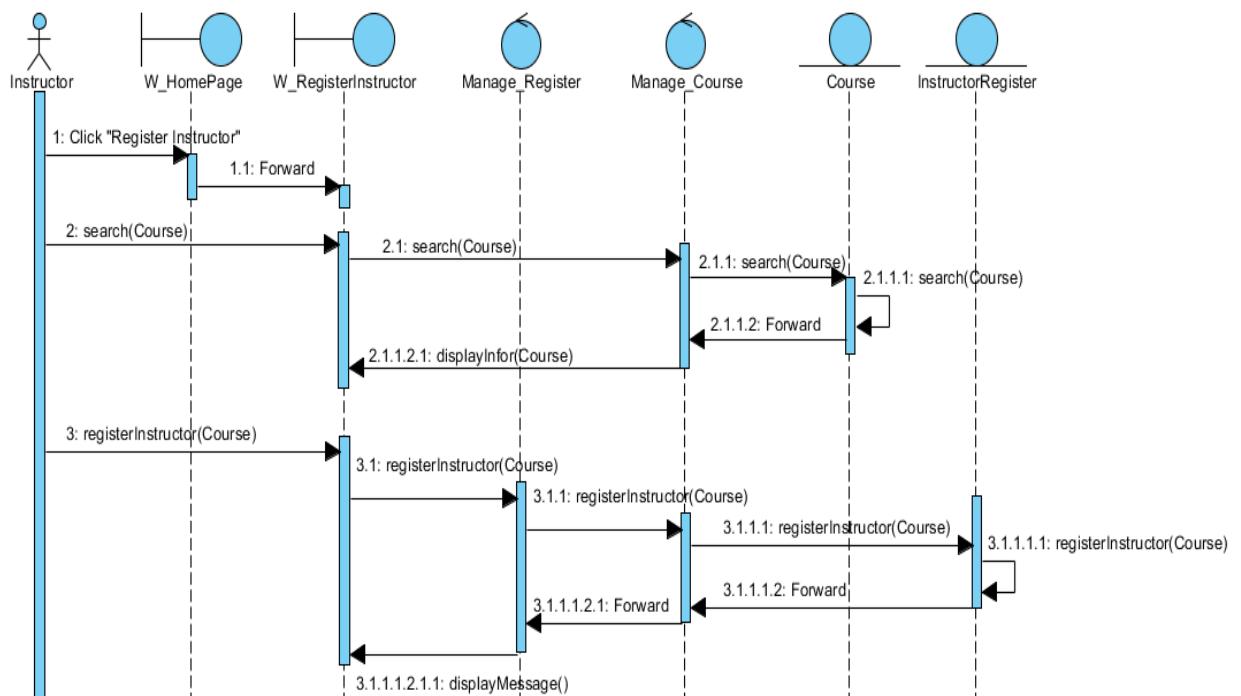


Biểu đồ tuần tự

Đăng nhập

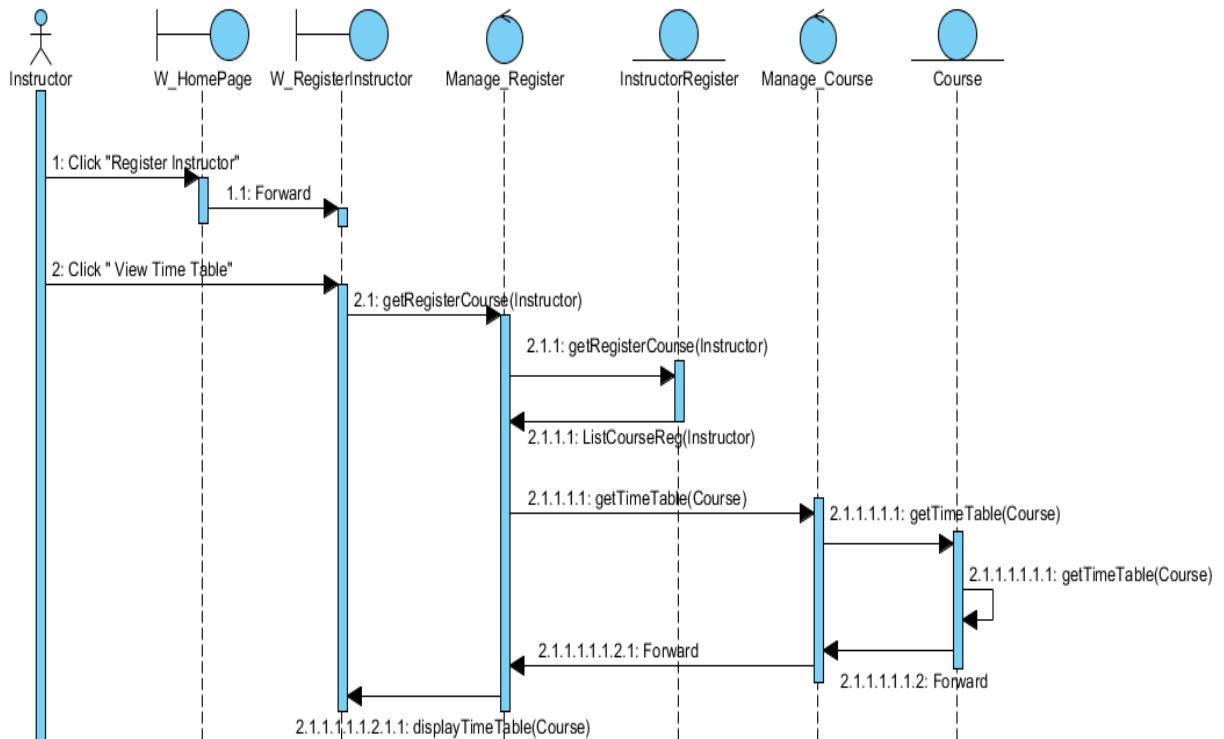


Giảng viên đăng ký môn dạy

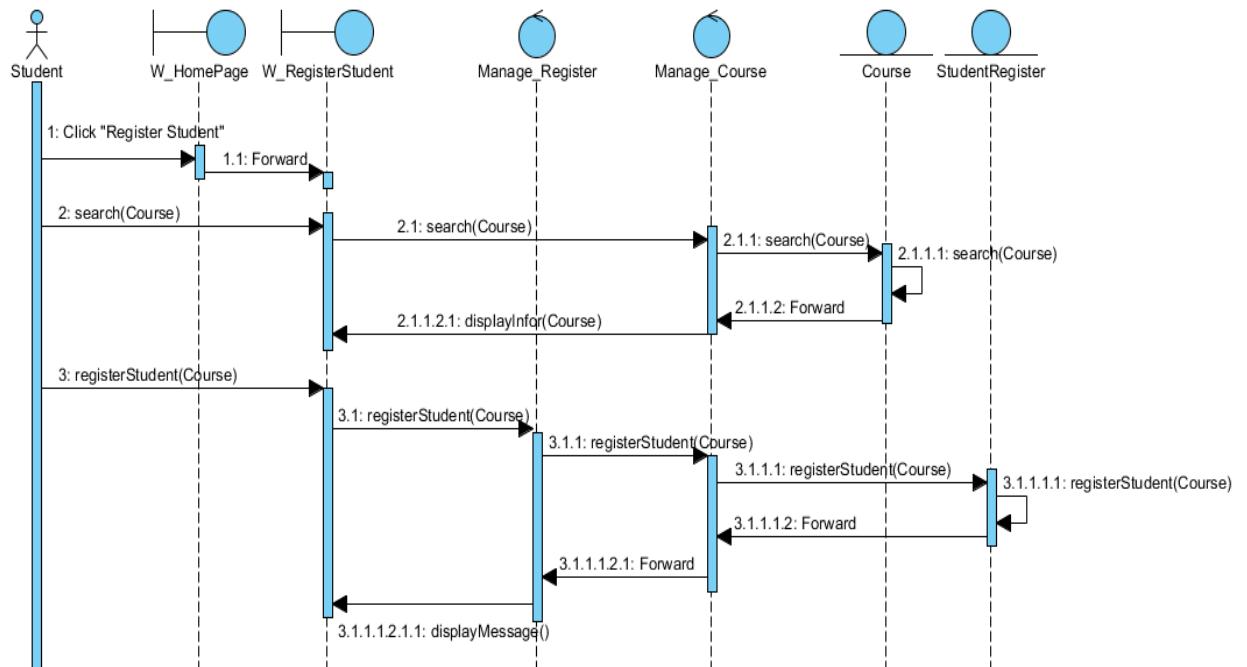


PHỤ LỤC B. HỆ QUẢN LÝ ĐĂNG KÝ HỌC THEO TÍN CHỈ

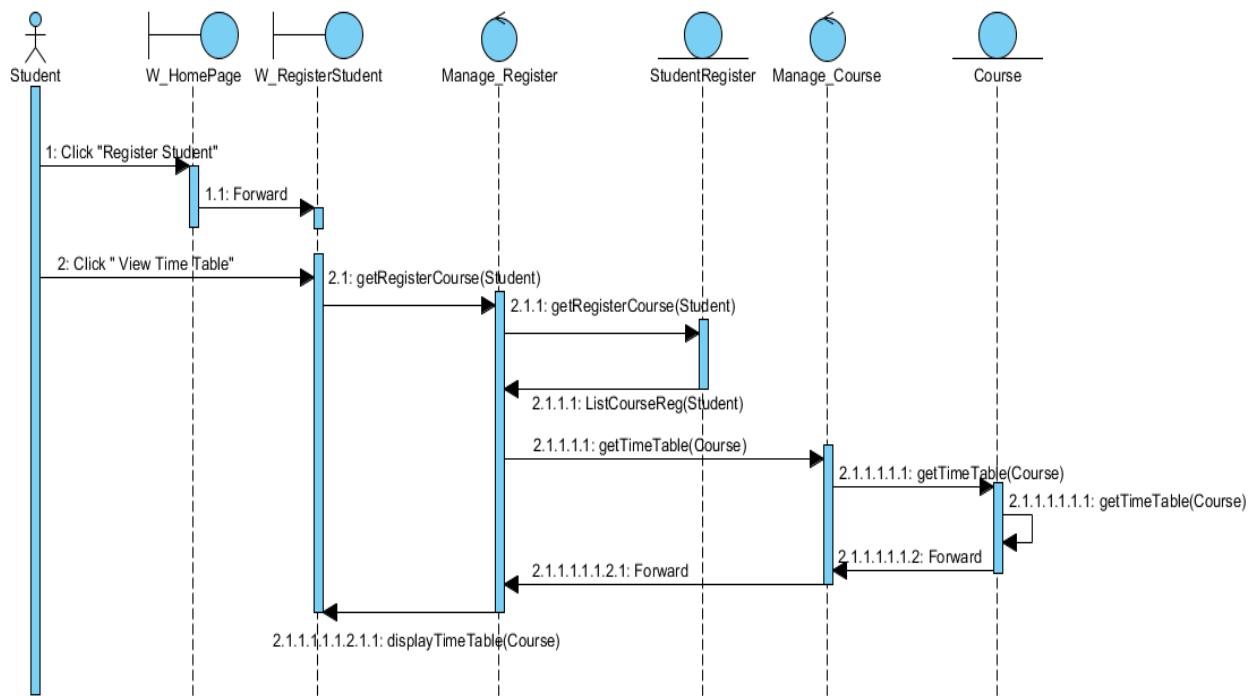
Giảng viên xem lịch dạy



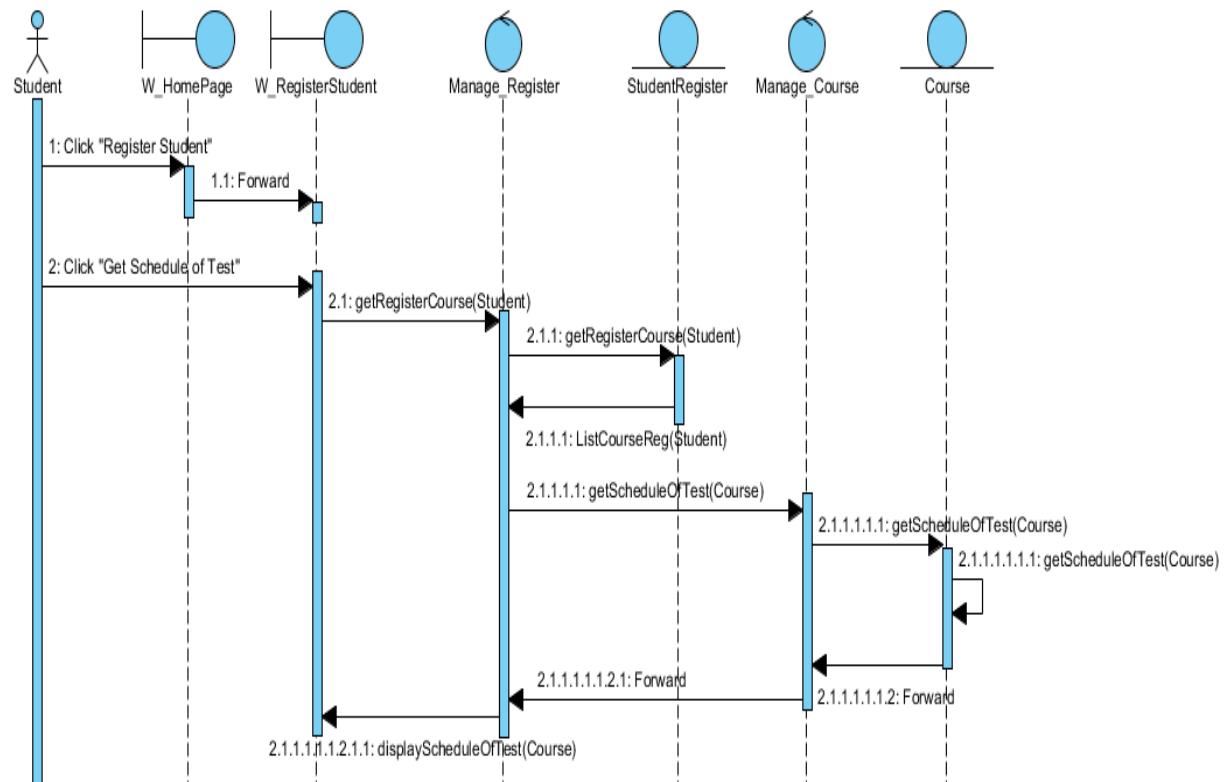
Sinh viên đăng ký môn học



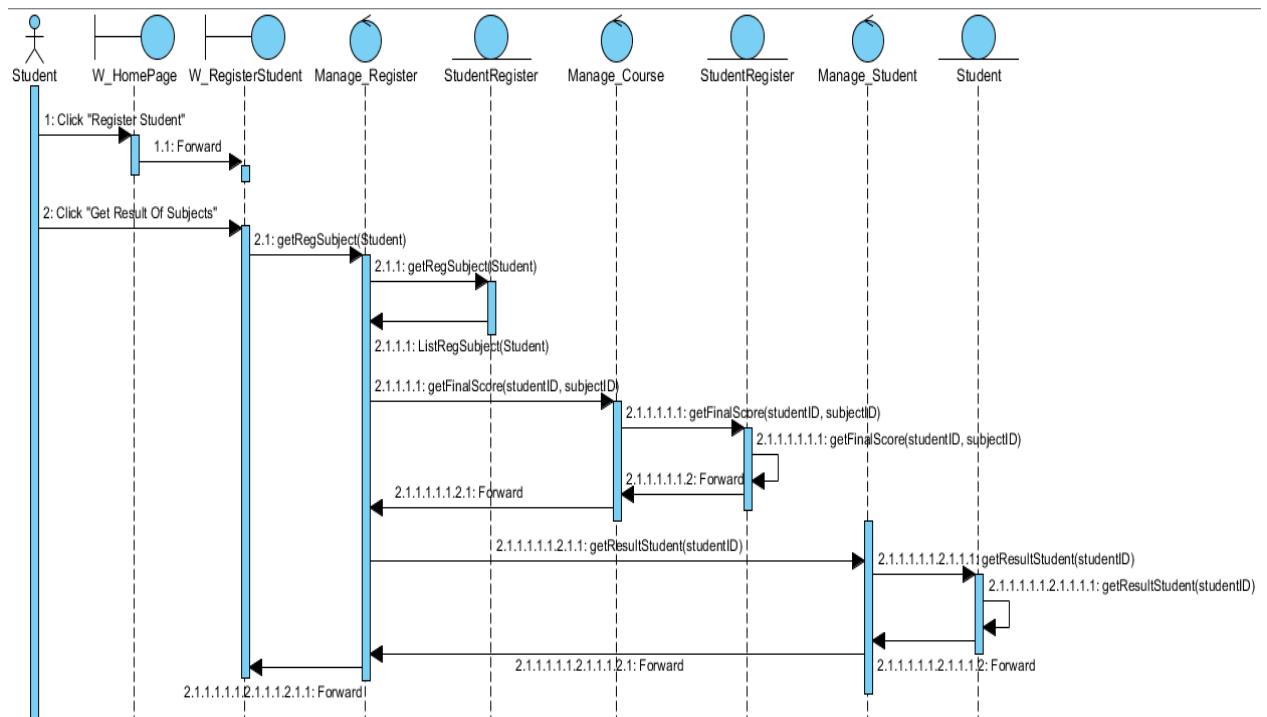
Sinh viên xem thời khóa biểu



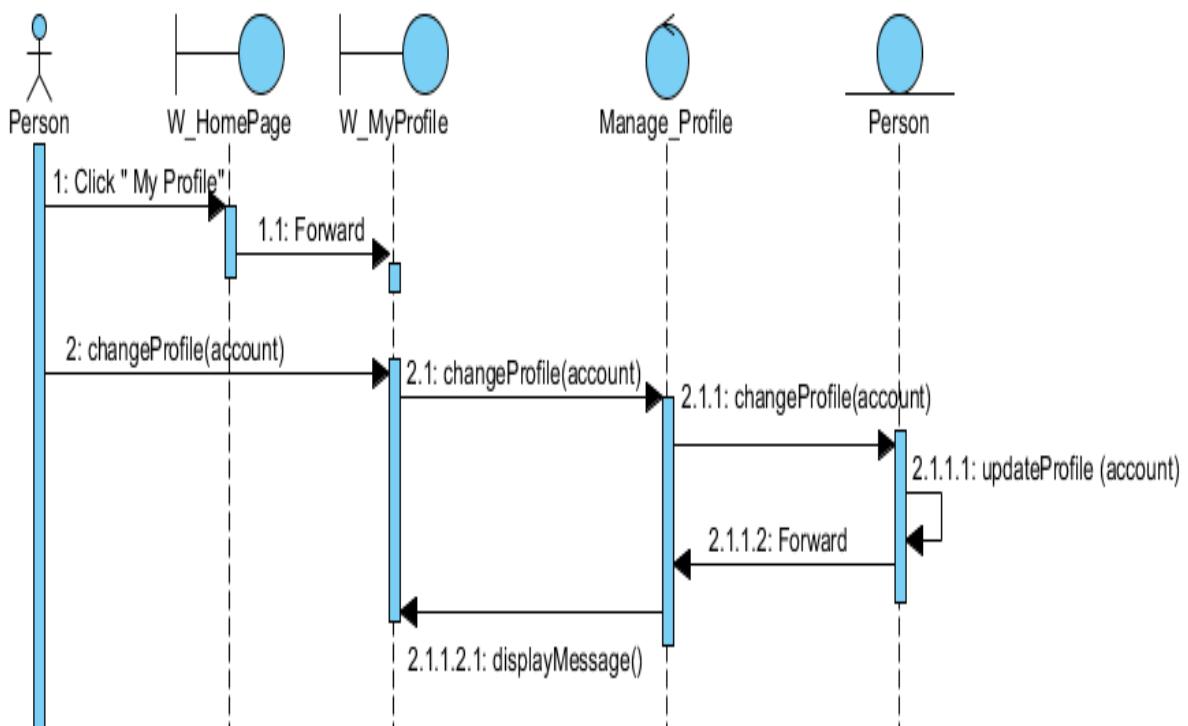
Sinh viên xem lịch thi học kỳ



Sinh viên xem điểm học tập

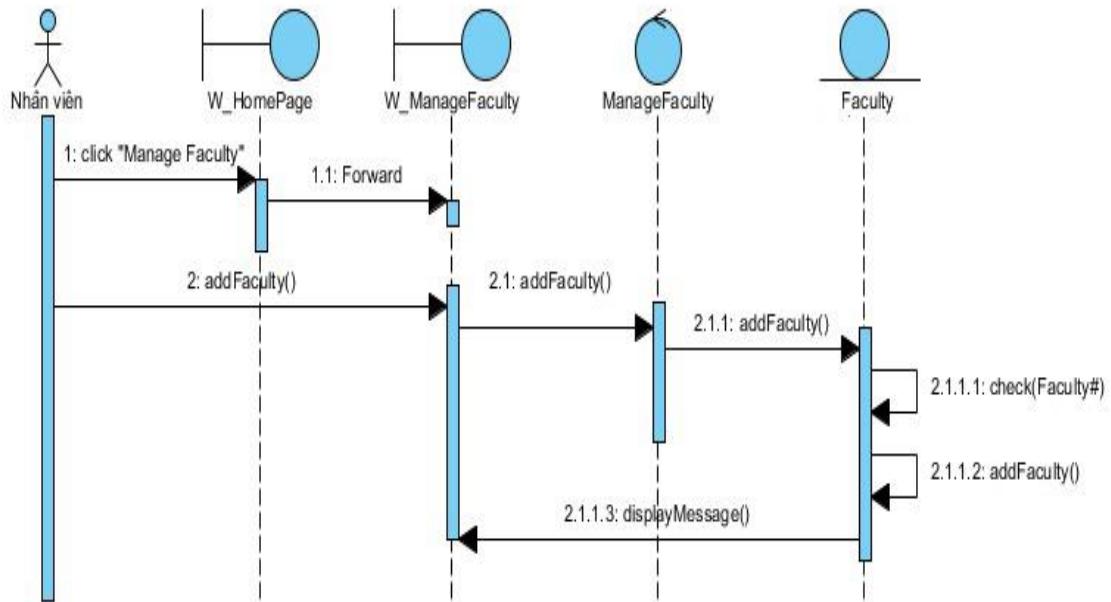


Người dùng thay đổi thông tin cá nhân

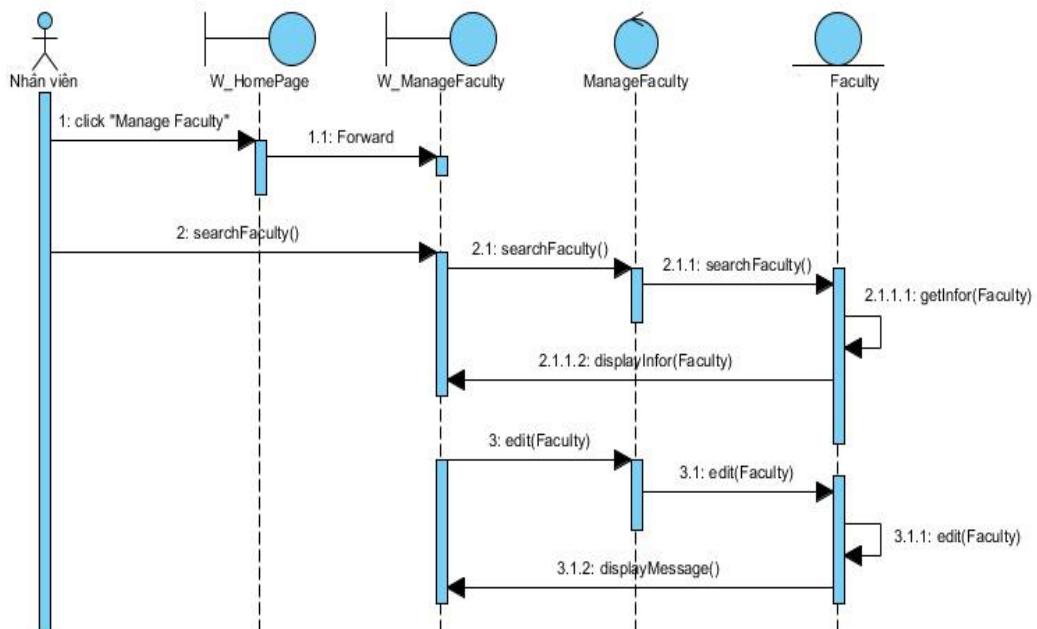


Nhân viên quản lý cơ bản

Quản lý khoa Thêm Khoa

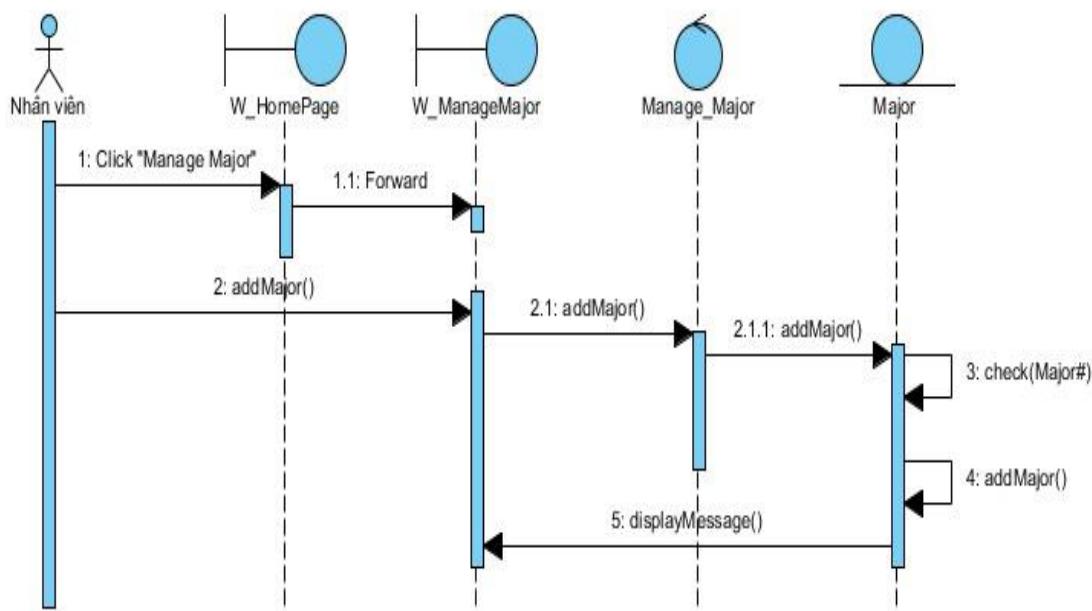


Sửa khoa

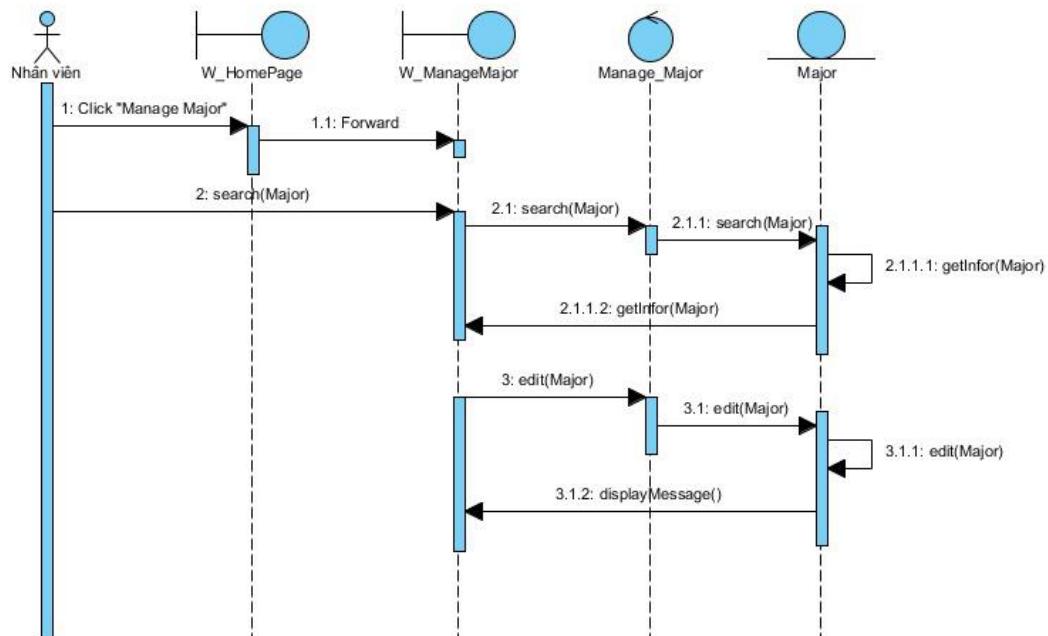


Quản lý chuyên ngành

Thêm Chuyên ngành

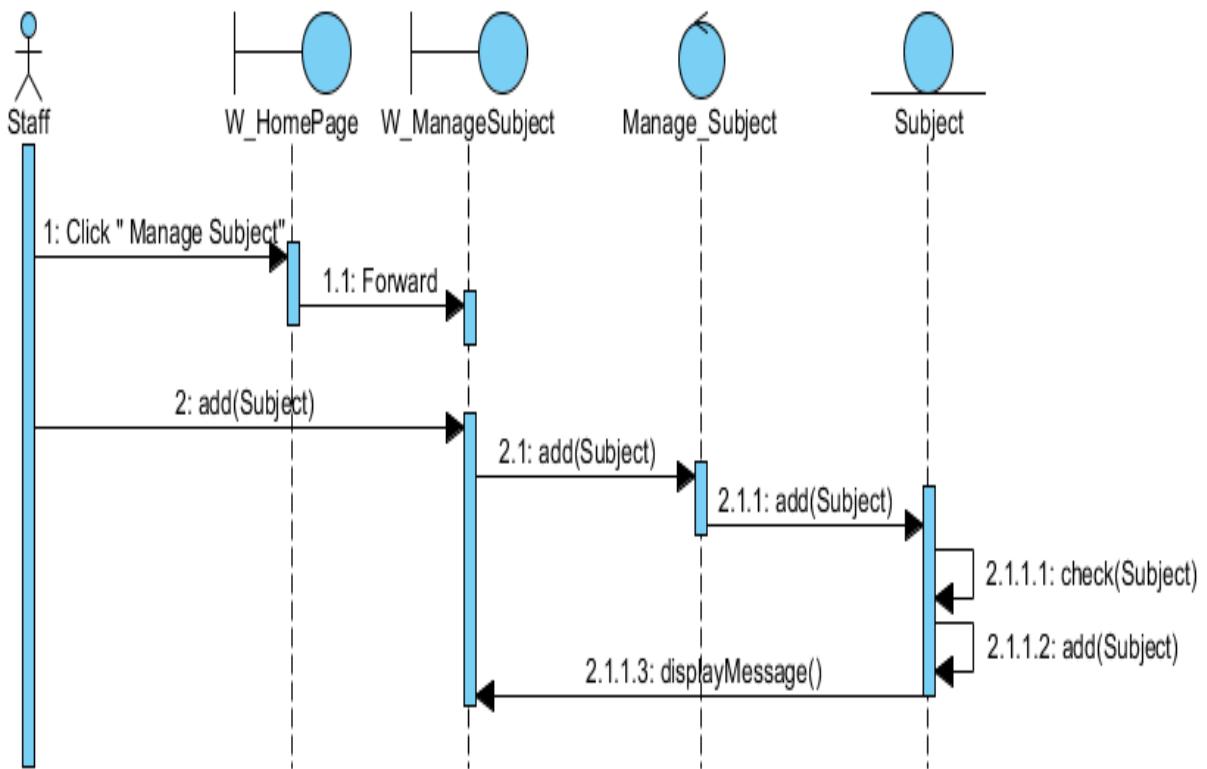


Sửa chuyên ngành

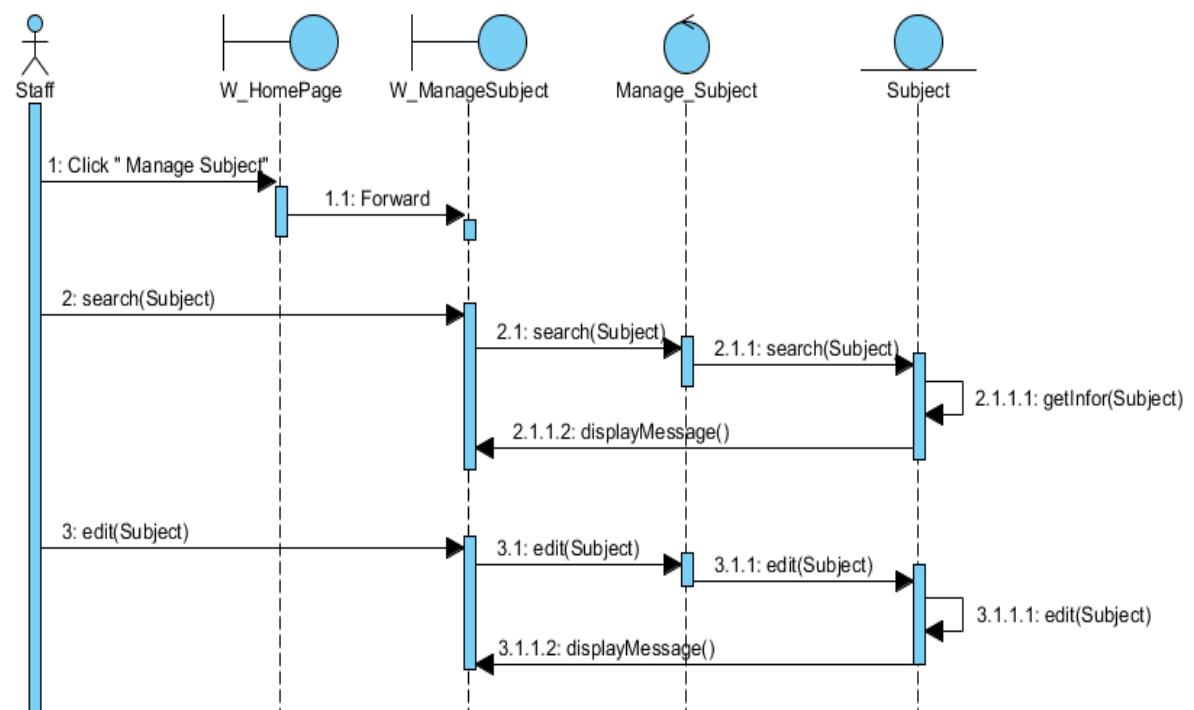


Quản lý môn học

Thêm Môn học



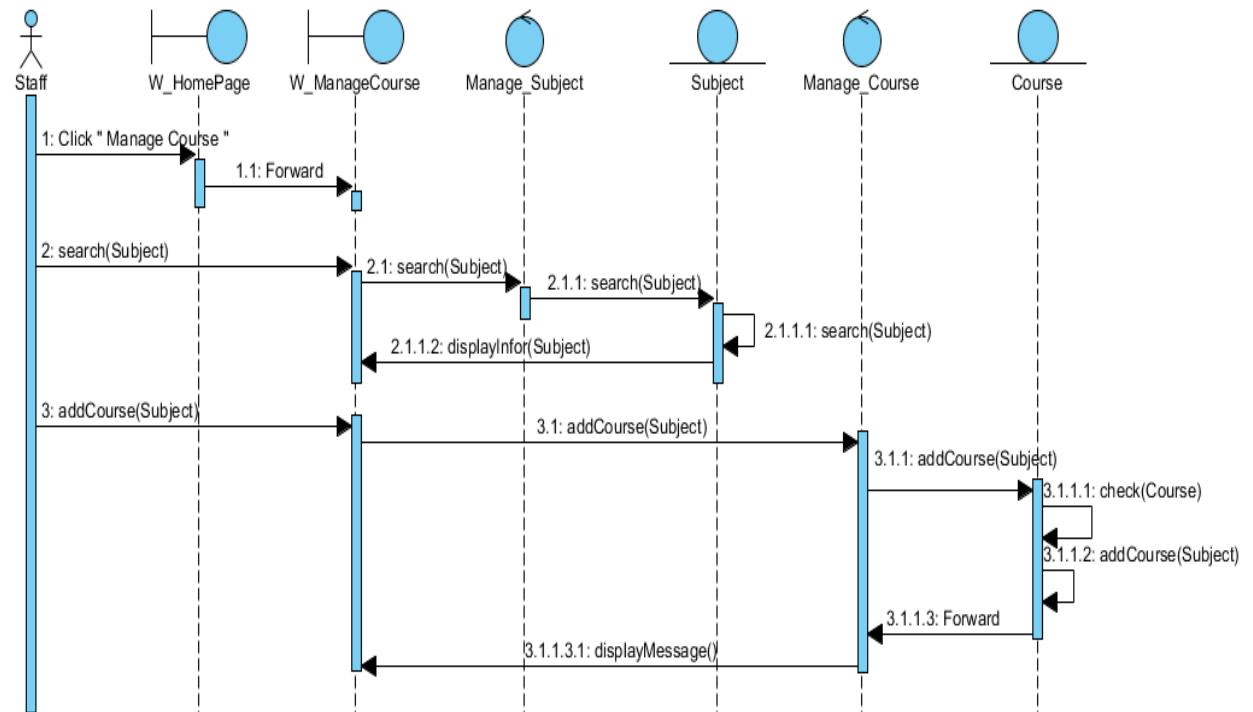
Sửa Môn học



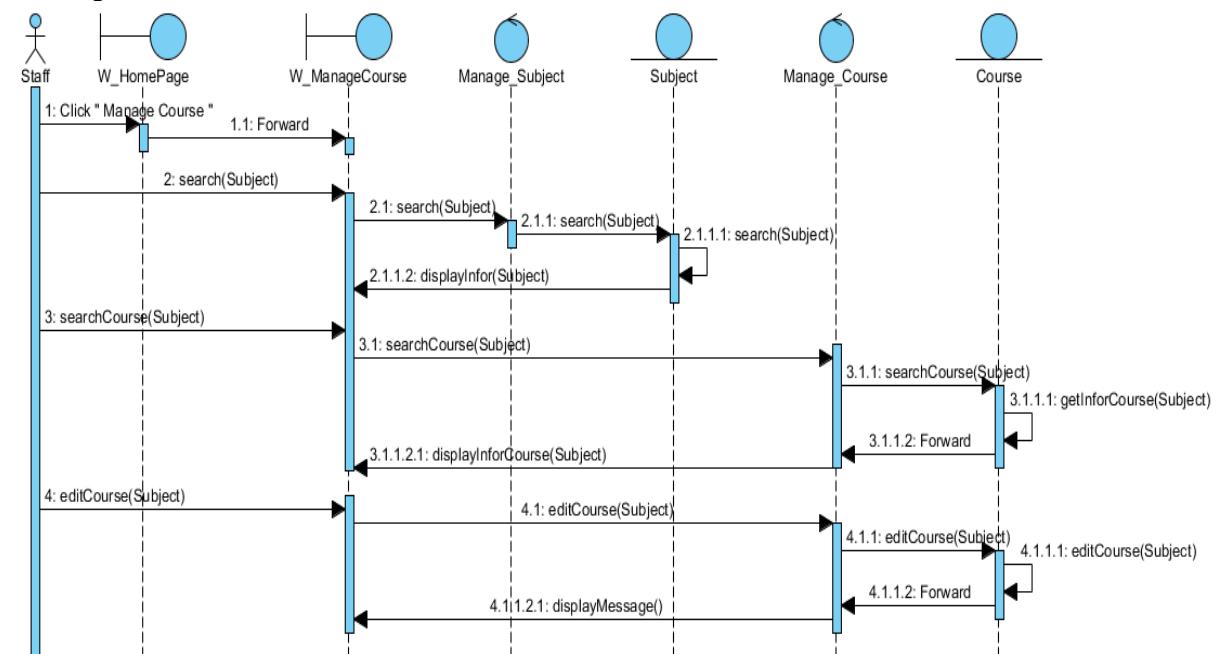
Nhân viên quản lý đào tạo

Quản lý xếp lớp học

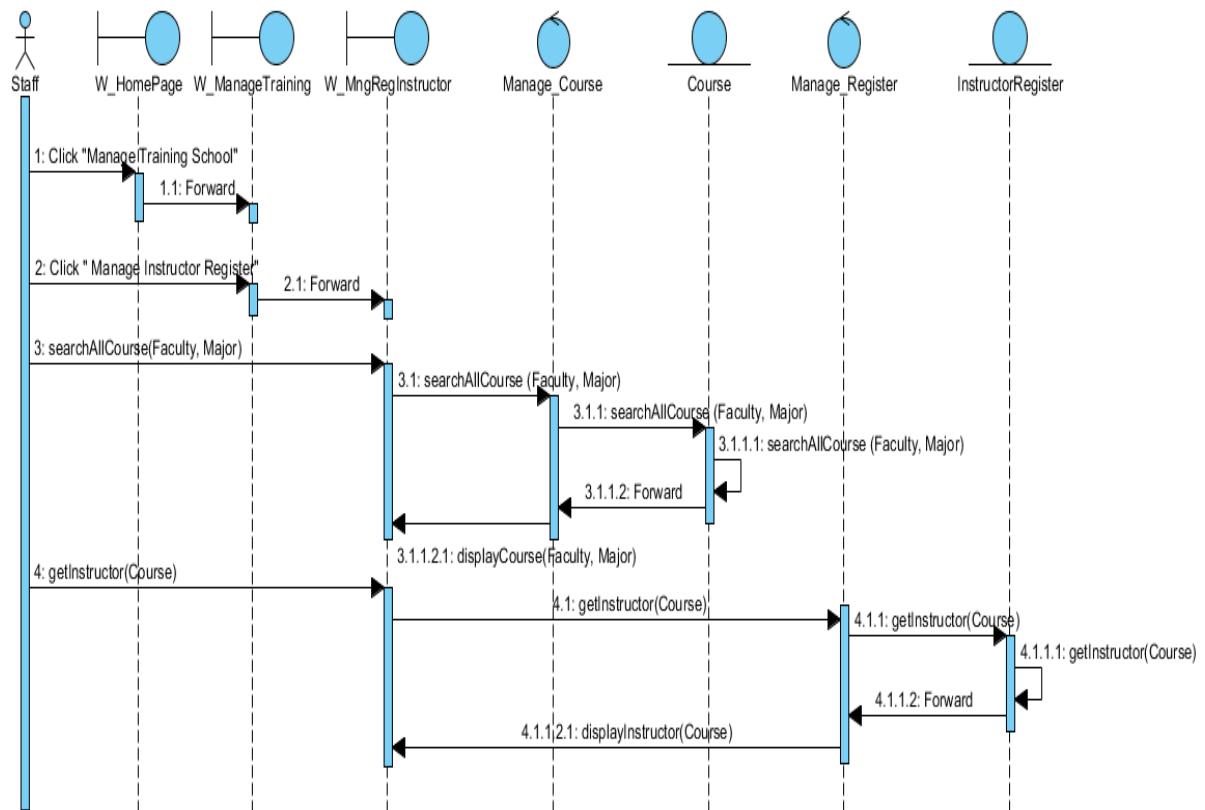
Thêm lớp học



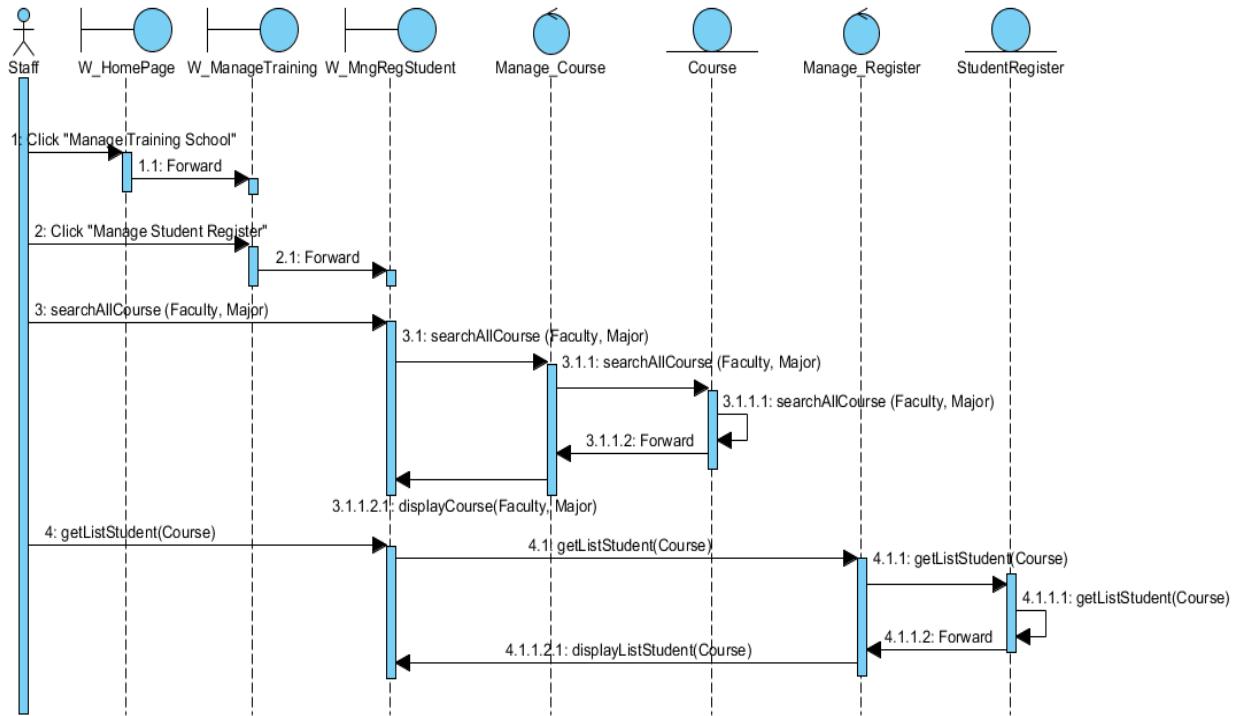
Sửa lớp học



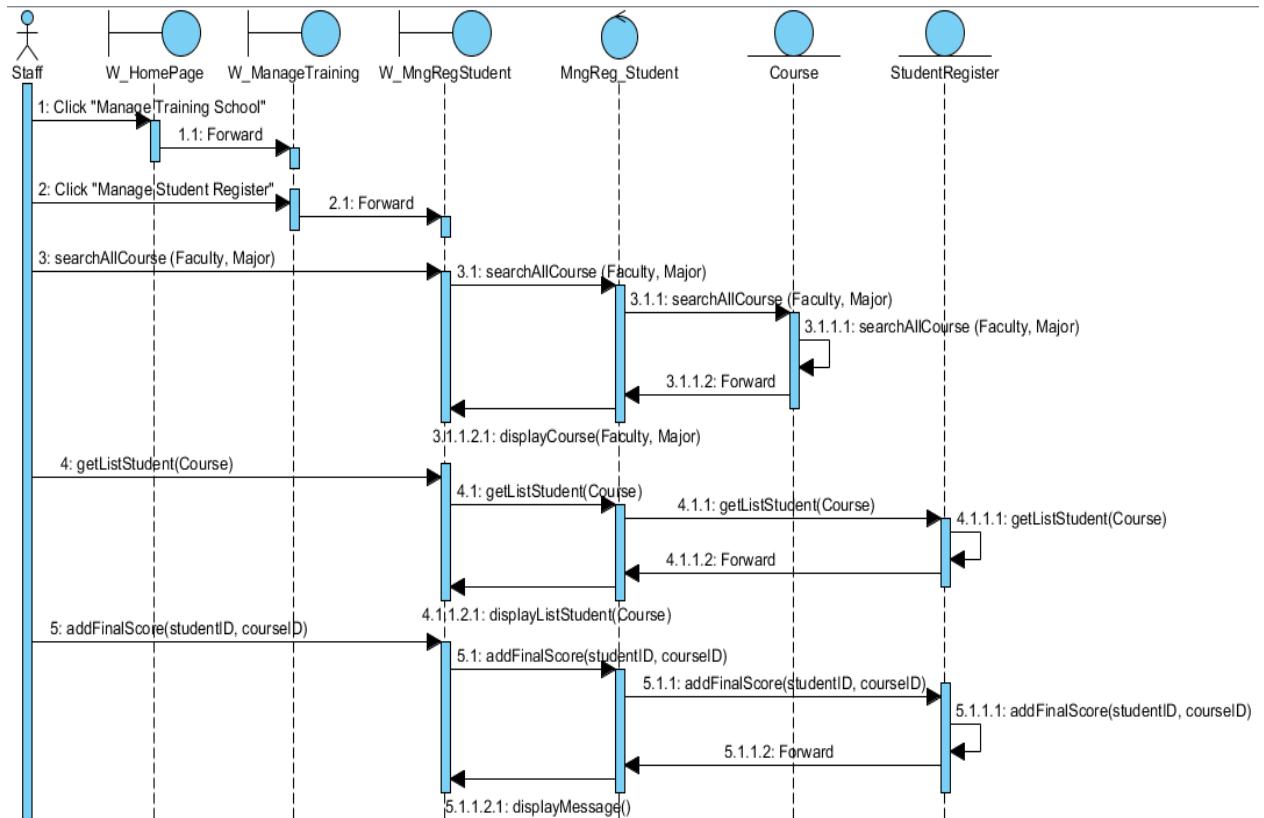
Quản lý đăng ký dạy của giảng viên



Quản lý đăng ký học sinh viên

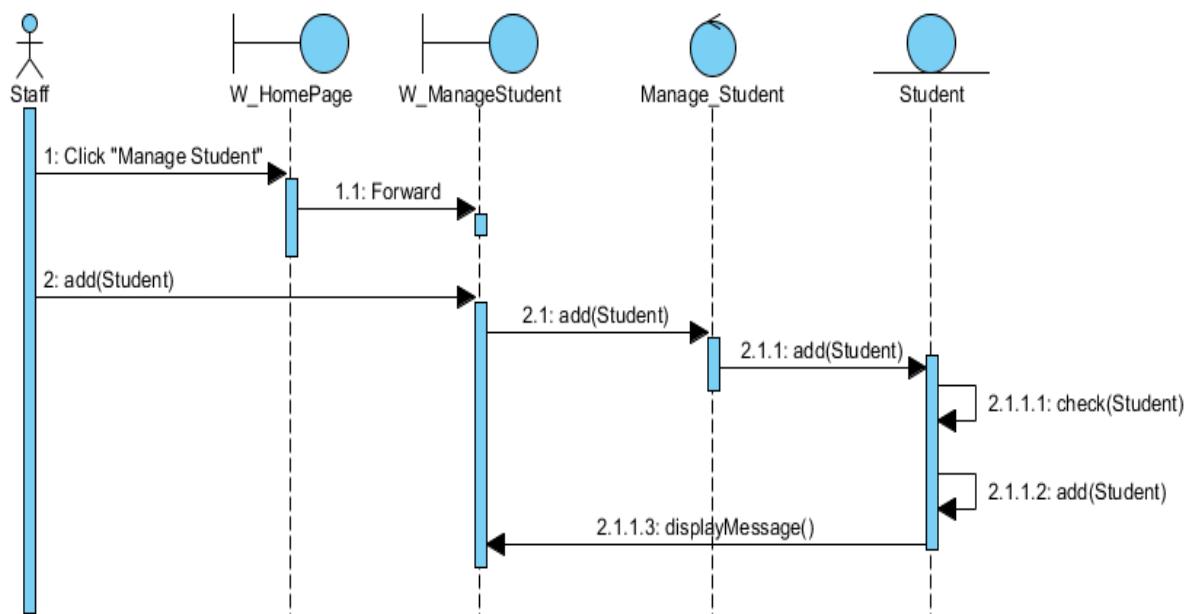


Quản lý điểm sinh viên

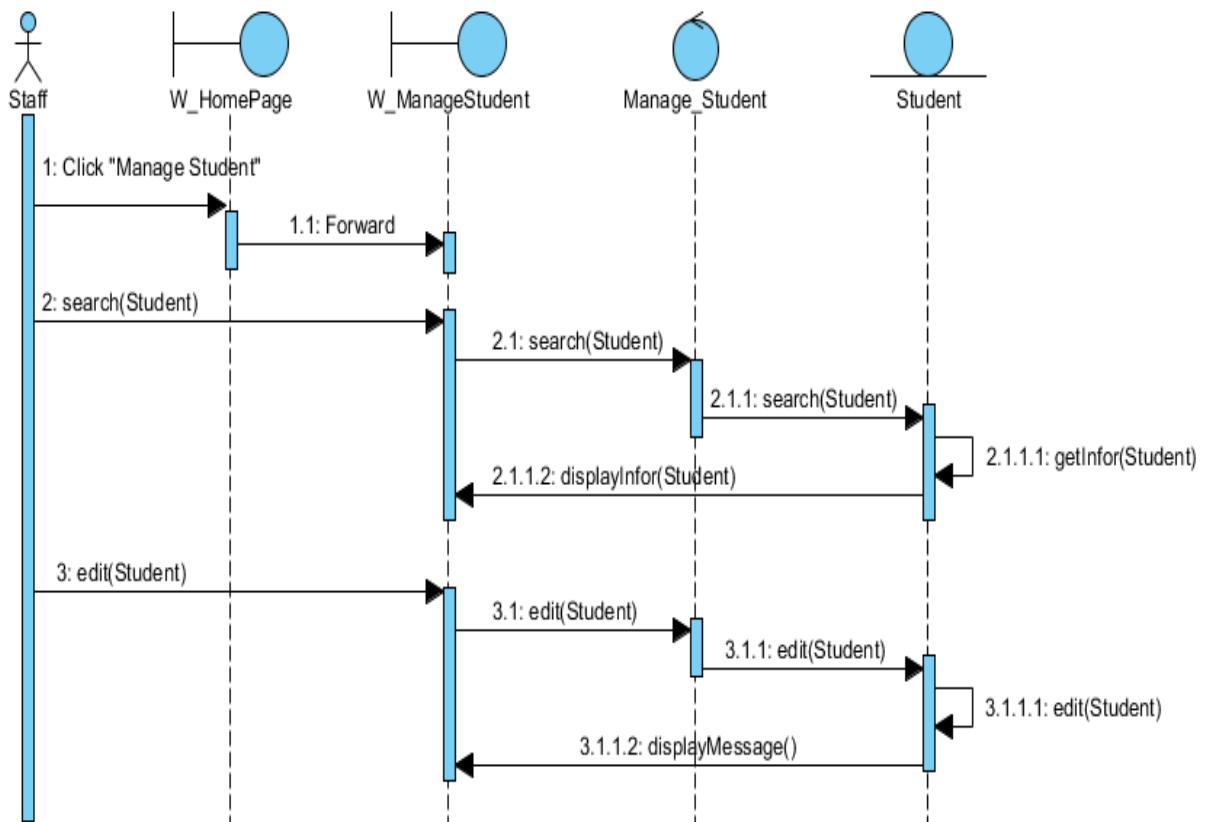


Quản lý sinh viên

Thêm sinh viên

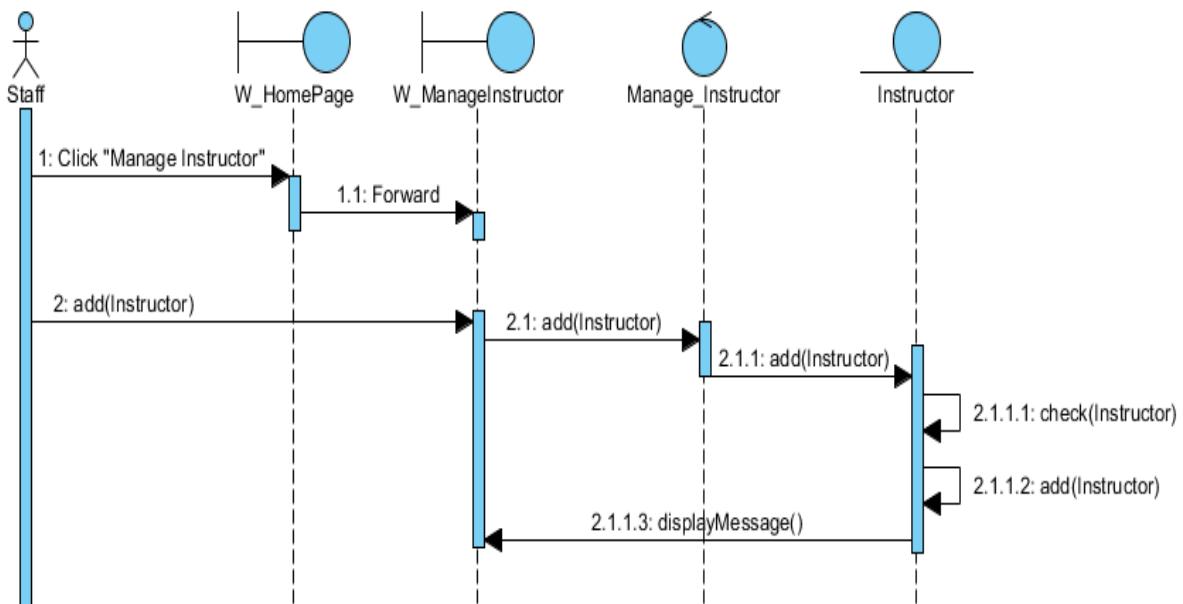


Sửa sinh viên

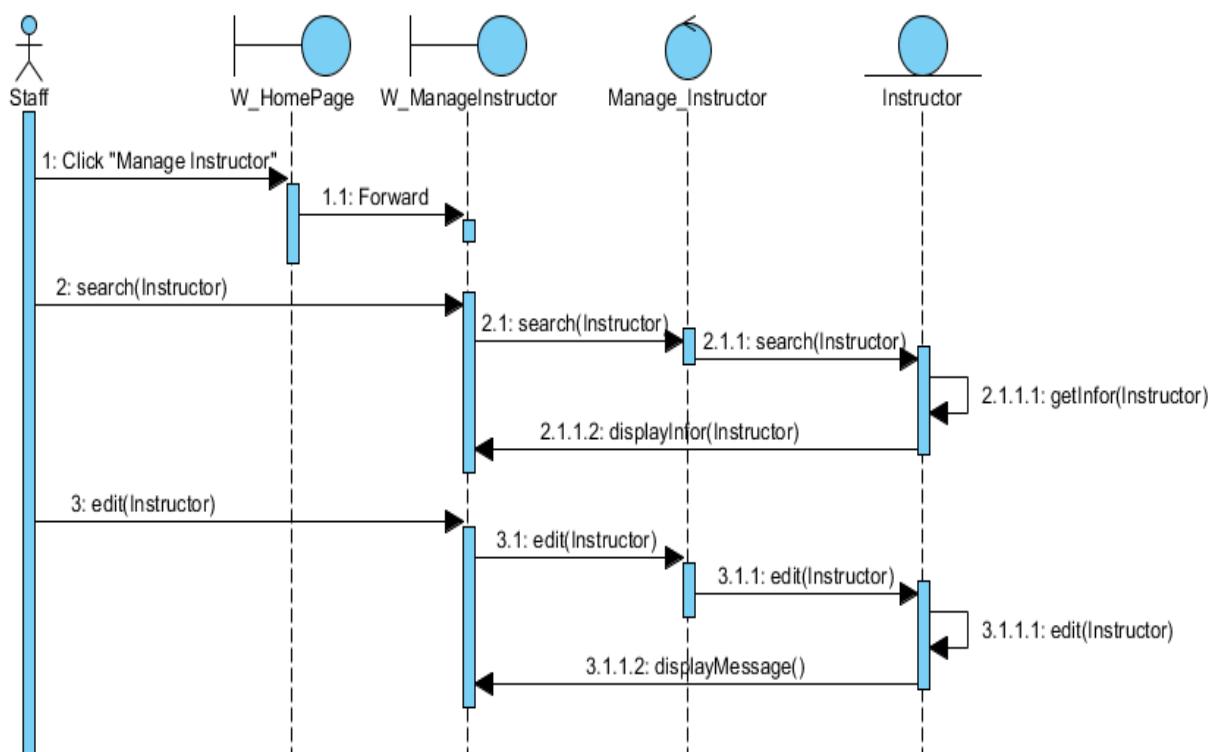


Quản lý giáo viên

Thêm giảng viên



Sửa giảng viên



TÀI LIỆU THAM KHẢO

- [1] Nguyễn Văn Ba, Phát triển hệ thống hướng đối tượng với UML 2.0 và C++, NXB Đại học Quốc gia Hà nội, 2005.
- [2] A. Dennis B. H. Wixom and David Tegarden, System Analysis and Design with UML version 2.0: An Object-Oriented Approach, Second Edition, John Wiley & Sons 2005.
- [3] Huỳnh Văn Đức, Đoàn Thiện Ngân, Giáo trình nhập môn UML, NXB Lao động Xã hội, 2003.
- [4] Đặng Văn Đức, Phân tích và thiết kế hướng đối tượng, NXB Giáo Dục, 2002
- [5] Hans-Erit, Magnus Penker, Brian Lyons, David Faado, UML2 Toolkit, Wiley Publishing, Inc, 2004
- [6] Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, Essentials of systems analysis and design, Fifth Edition, Pub. Pearson, 2011.
- [7] Mike O'Docherty, Object-Oriented Analysis and Design: Understanding System Development with UML 2.0, John Wiley & Sons, 2005.
- [8] Grady Booch, Object-oriented Analysis and Design, Second Edition, Addison Wesley, 1994.
- [9] R. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill, 2005
- [10] Trần Đình Quế và Nguyễn Mạnh Sơn, Phân tích và Thiết kế hướng đối tượng, Bài giảng dành cho Sinh viên Đại học Từ xa, Học viện CNBCVT, 2005
- [11] S. Schach, Object-oriented and classical software engineering, Sixth Edition, McGrawHill, 2006.
- [12] Brett Spell, Pro Java Programming, Second Edition, Apress 2006
- [13] Data Access Object Pattern. Tham khảo:
<http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>
- [14] Gregor Engels, Object-Oriented Modeling: A Roadmap. Tham khảo: <http://wwwcs.uni-paderborn.de/cs/ag-engels/Papers/2000/EG00objectorientedModelling.pdf>
- [15] R. Taylor, N. Medvidovic and E. Dashofy, Software Architecture: Foundations, Theory and Practice, Wiley Publisher, 2010.
- [16] David P. Tegarden et al., A Software Complexity Model of Object-Oriented Systems. Tham khảo <http://www.acis.pamplin.vt.edu/faculty/tegarden/wrk-pap/DSS.PDF>
- [17] Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, Essentials of systems analysis and design, Fifth Edition, Pub. Pearson, 2011.
- [18] A. J. A. Wang and K. Qian, Component Oriented Programming, Wiley, 2005
- [19] Microsoft, Microsoft application architecture guide, Second Edition, 2009
- [20] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns: Elements of Reusable

TÀI LIỆU THAM KHẢO

- Object Oriented Software, Addison Wesley, 1994
- [21] Partha Kuchana, Software architecture design patterns in Java, Auerbach Publications, 2004.