NGUYỄN VĂN BA

Phát triển hệ thống hướng đối tượng với UML 2.0 và C++

(*In lần thứ hai*)

Sách dùng cho:

- Sinh viên các trường Đại học, Cao đẳng
- Các nhà xây dựng hệ thống chuyên nghiệp
- Các kỹ sư phân tích và thiết kế

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI
16 Hàng Chuối - Hai Bà Trưng - Hà Nội
Điện thoại: (04) 9724852; (04) 9724770. Fax: (04) 9714899

Chịu trách nhiệm xuất bản:

Giám đốc: PHÙNG QUỐC BẢO

Tổng biên tập: NGUYỄN BÁ THÀNH

Biên tập: HỒ ĐỒNG

LAN HƯƠNG

Trình bày bìa: HẢI ĐỒNG

HÁT TRIỂN HỆ THỐNG HƯỚNG ĐỐI TƯỢNG VỚI UML 2.0 VÀ G++

Mã số: 1L-01 ĐH2008
In 1000 cuốn, khổ 16 x 24 cm tại Xưởng in Tạp chí tin học & đời sống
Số xuất bản: 136 - 2007/GXB/03 - 13/ĐHQGHN, ngày 13/2/2007
Quyết định xuất bản số: 01 LK/XB
In xong và nộp lưu chiểu quý I năm 2008.

LỜI NÓI ĐẦU

Mục tiêu hướng tới của cuốn sách này đã được bao hàm đầy đủ trong tựa đề của nó. Đó là: "Phát triển hệ thống hướng đối tượng với UML 2.0 và C++". Để bạn đọc có thể có ngay một cái nhìn bao quát về nội dung cuốn sách, xin hãy lần lượt xét các vế trong tựa đề này.

Phát triển hệ thống

Hệ thống được đề cập ở đây là hệ thống phần mềm, hay nói rộng ra một chút là hệ thống tin học (bao gồm cả phần mềm và phần cứng). Vậy phát triển hệ thống được hiểu là quá trình xây dựng một hệ thống tin học, tính từ A đến Z, kể từ lúc manh nha ý đồ, đến khảo sát để tìm hiểu môi trường và nhu cầu, rồi phân tích để đi sâu vào chi tiết, thiết kế để làm cho nó thích ứng với các điều kiện kỹ thuật sẵn có, cài đặt để thực thi nó trong một ngôn ngữ lập trình và trên một nền tảng kỹ thuật, và cuối cùng là kiểm chứng và chuyển giao. Tuy nhiên, vì sự hạn chế về số trang, nên cuốn sách cũng chỉ có thể dành nhiều chú ý vào một số khâu chính trong quá trình phát triển hệ thống. Đó là: tìm hiểu nhu cầu, phân tích, thiết kế và cài đặt trên một ngôn ngữ lập trình là C++.

Định hướng cho sự phát triển hệ thống

Tồn tại khá nhiều phương pháp để tiến hành việc phát triển hệ thống. Để tránh một sự dàn trải theo lối "cưỡi ngựa xem hoa", cuốn sách buộc phải chọn một phương pháp để có thể trình bày sâu và kỹ. Tuy có nhiều phương pháp, song một cách đại thể, có thể tách chúng thành hai nhóm lớn tuỳ thuộc vào hai định hướng khác biệt. Đó là các phương pháp hướng chức năng và các phương pháp hướng đối tượng.

Các phương pháp hướng chức năng, nở rộ vào những năm 70, 80 của thế kỷ trước, lấy chức năng làm đơn vị phân rã khi tiến hành phân tích hệ thống. Câu hỏi về hệ thống thường được đặt ra sớm nhất cho người dùng, cũng như cho người thiết kế là câu hỏi: "Hệ thống phải làm gì?". Bởi vậy nghiên cứu hệ thống dựa vào các chức năng (tức là

việc phải làm) là một cách làm tự nhiên và dễ hiểu. Phương pháp hướng chức năng sẽ dẫn tới việc cài đặt hệ thống bằng các ngôn ngữ lập trình theo thủ tục (như Pascal, C...) [1]. Dù là dễ làm, dễ hiểu, thì dần dà theo năm tháng, các phương pháp hướng chức năng đã để lộ ra các nhược điểm khó chấp nhận: đó là các hệ thống được xây dựng theo cách này là khó sửa chữa, khó nâng cấp và ít có khả năng tái sử dụng vào các hoàn cảnh khác.

Các phương pháp hướng đối tượng khắc phục các nhược điểm trên và ra đời từ đầu các năm 90 đến nay, lại lấy đối tượng làm đơn nguyên cơ bản của hệ thống. Đối tượng là một sự kết hợp giữa chức năng và dữ liệu. Đó là một sự kết hợp hợp lý, vì mỗi chức năng chỉ thao tác trên một số dữ liệu nhất định và ngược lại mỗi dữ liệu chỉ được xử lý bởi một số chức năng nhất định. Không những hợp lý mà lại còn rất tự nhiên và dễ hiểu, vì các đối tượng tin học thường dùng để phản ánh hay mô phỏng các đối tượng trong thế giới thực (tức là các sự hay vật). Sự thành đạt ngày nay của các ngôn ngữ lập trình hướng đối tượng (như C++, Java...) đã khẳng định vị thế áp đảo của các phương pháp phân tích và thiết kế hướng đối tượng trước các phương pháp phân tích và thiết kế truyền thống. Cho nên, khỏi phải cân nhắc nhiều, cuốn sách này sẽ trình bày với bạn đọc các tri thức và công nghệ phát triển hệ thống theo định hướng đối tượng.

Ngôn ngữ mô hình hoá UML

Dù là dùng phương pháp nào, thì người phân tích và thiết kế luôn phải dùng một hình thức hiểu được nào đó để diễn tả các sắc thái khác nhau của hệ thống. Hình thức diễn tả đó có thể ở dạng văn tự, phương trình toán học, các bảng hoặc các đồ thị. Người ta gọi đó là các mô hình, và việc sử dụng mô hình để diễn tả hệ thống được gọi là mô hình hoá.

Trước đây mỗi phương pháp phát triển hệ thống đề nghị một loại mô hình riêng. Sự khác biệt trong ngôn ngữ diễn tả hệ thống đó, giống như người nói tiếng Anh, kẻ nói tiếng Việt về cùng một vấn đề, đã gây ra những khó khăn không cần thiết. Xu hướng phát triển tất yếu là phải

---

[1] Bạn đọc muốn tìm hiểu về các phương pháp hướng chức năng xin tham khảo cuốn "Phân tích và thiết kế hệ thống thông tin - Các phương pháp cấu trúc", xuất bản năm 2003 của cùng tác giả.

đi đến thống nhất ngôn ngữ. Vì vậy mà vào năm 1997 đã ra đời ngôn ngữ mô hình hoá thống nhất UML. Ngôn ngữ mô hình hoá UML được công nhận là chuẩn, nhưng vẫn được tiếp tục nâng cấp. Đến nay (cuối 2004) đã có phiên bản UML 2.0. UML dùng các mô hình ở dạng biểu đồ. Phiên bản UML 2.0 đưa ra 13 loại biểu đồ, thay cho 9 loại biểu đồ dùng trong các phiên bản 1.x trước đó. Cuốn sách này sẽ trình bày với bạn đọc về UML 2.0 và cách vận dụng nó vào tiến trình phát triển hệ thống hướng đối tượng.

Cài đặt với C++

Việc cài đặt hệ thống luôn luôn phải sử dụng một ngôn ngữ lập trình nào đó, mà các ngôn ngữ lập trình lại thường khá khác biệt về cú pháp và ngữ nghĩa với nhau. Cuốn sách này sẽ trình bày việc cài đặt hệ thống trong ngôn ngữ lập trình C++.

Khi ta đã tiến hành phân tích và thiết kế hướng đối tượng, thì chuyển qua cài đặt trên một ngôn ngữ lập trình hướng đối tượng, như C++, là một sự chọn lựa tự nhiên hơn cả. Nhưng tại sao lại chọn lựa C++, mà không phải là một ngôn ngữ lập trình hướng đối tượng khác? Không phải vì C++ đặc sắc hơn hay mới hơn, mà chẳng qua là vì, theo sự kỳ vọng của tác giả, thì C++ là quen thuộc với đông đảo bạn đọc hơn cả (phần lớn các khoa công nghệ thông tin ở các trường đại học đều đã có giáo trình "Lập trình hướng đối tượng với C++"). Và một khi đã là quen thuộc, thì bạn đọc khi nghiên cứu các chương về cài đặt ở cuối cuốn sách này sẽ khỏi bận tâm về việc học thêm một ngôn ngữ lập trình mới, và sẽ tập trung chú ý vào các ý tưởng chủ đạo trong cài đặt. Nếu đã nắm bắt được các ý tưởng chủ đạo đó rồi, thì cho dù sau này bạn đọc sử dụng một ngôn ngữ lập trình hướng đối tượng khác, việc vận dụng các ý tưởng đó cũng không còn mấy khó khăn.

Cấu trúc của cuốn sách

Nội dung của cuốn sách được trình bày theo hai tuyến chính:

- Một mặt là sự trình bày lần lượt các khái niệm và mô hình của UML,
- Mặt khác là sự trình bày lần lượt các bước triển khai của tiến trình phát triển hệ thống.

Hai tuyến này được trình bày song song và hỗ trợ cho nhau dọc theo các chương của cuốn sách. Các mô hình của UML được giới thiệu dần dần theo trật tự vận dụng vào các bước của tiến trình phát triển. Còn các chương thì lại được phân theo các chủ đề lớn của việc phát triển hệ thống.

Hai chương đầu chỉ có tính chất dẫn nhập. Chương I nhằm giới thiệu lại cho bạn đọc những ý tưởng chủ đạo trong lập trình hướng đối tượng, để từ đó bạn đọc thấy trước cái đích của phân tích và thiết kế (vì phân tích và thiết kế là để đi đến lập trình), và sẽ dễ lý giải hơn về các cung cách phân tích thiết kế sẽ được trình bày dọc theo cuốn sách. Chương II giới thiệu một cách khái quát về mô hình, ngôn ngữ UML và tiến trình phát triển hệ thống được dùng trong cuốn sách.

Chương III đề cập hai bước khởi đầu của tiến trình phát triển hệ thống. Đó là bước 1 (Nghiên cứu sơ bộ) và bước 2 (Nhận định và đặc tả các ca sử dụng). Mặt khác, Chương III giới thiệu biểu đồ ca sử dụng của UML.

Đi vào phân tích, thì hai mục tiêu lớn là phân tích về cấu trúc của hệ thống và phân tích về hành vi của hệ thống. Chương IV đề cập việc mô hình hoá cấu trúc. Nó trình bày các khái niệm cơ bản là : đối tượng, lớp, gói và loài. Qua đó nó giới thiệu bốn biểu đồ dùng để diễn tả cấu trúc tĩnh trong UML, đó là: biểu đồ lớp, biểu đồ đối tượng, biểu đồ gói và biểu đồ cấu trúc da hợp. Mặt khác thì Chương IV sẽ trình bày hai bước của tiến trình phát triển hệ thống, nhằm vào việc mô hình hoá cấu trúc. Đó là bước 3 (Mô hình hoá lĩnh vực ứng dụng) và bước 4 (Xác định các đối tượng và lớp tham gia các ca sử dụng).

Chương V đề cập về thứ hai của phân tích hệ thống, đó là mô hình hoá hành vi. Nó trình bày hai bước của tiến trình phát triển hệ thống, nhằm vào việc mô hình hoá hành vi. Đó là bước 5 (Mô hình hoá sự tương tác) và bước 6 (Mô hình hoá sự ứng xử). Lồng vào hai bước đó, thì ở đây cũng được giới thiệu hai biểu đồ của UML dùng để diễn tả sự tương tác là biểu đồ trình tự và biểu đồ giao tiếp, và một biểu đồ dùng để diễn tả sự ứng xử, đó là biểu đồ máy trạng thái. Ngoài ra Chương V cũng giới thiệu thêm ba biểu đồ khác về hành vi, đó là biểu đồ hoạt động, biểu đồ bao quát tương tác và biểu đồ thời khắc.

Chương VI đề cập việc thiết kế. Có lẽ đây không những là chương dài nhất, mà còn là chương phức tạp nhất. Điều đó cũng là tự nhiên, vì

trong tiến trình phát triển của hệ thống, thì việc thiết kế thường chiếm nhiều thời gian và công sức nhất. Chương này sẽ trình bày ba bước trong tiến trình phát triển hệ thống nhằm mục đích thiết kế. Đó là bước 7 (Làm nguyên mẫu giao diện người dùng), bước 8 (Thiết kế hệ thống), bước 9 (Thiết kế chi tiết). Mặt khác thì Chương VI cũng kết hợp trình bày các biểu đồ của UML dùng cho việc thiết kế, như là biểu đồ thành phần, biểu đồ bố trí.

Ba chương cuối cùng của cuốn sách đề cập việc cài đặt hệ thống (bước 10). Chương VII giới thiệu về các cơ sở của C++. Mặc dù cuốn sách hoàn toàn không có mục đích trình bày C++ một cách hoàn chỉnh, song nó cũng giới thiệu qua về các cơ sở của C++ ở đây, nhằm giúp các bạn đọc chưa rành lắm về C++ có một số ít kiến thức sơ bộ về C++ để có thể đọc tiếp các chương sau về cài đặt hệ thống. Chương VIII đề cập việc cài đặt các lớp trong C++, và chương cuối cùng, Chương IX, đề cập việc cài đặt các mối liên quan giữa các lớp trong C++.

Cuốn sách này có ích cho những ai

Cuốn sách này không thuộc loại sách nhập môn. Vì vậy bạn đọc, dù là ở ngoài ngành hay trong ngành công nghệ thông tin, nếu chỉ muốn tìm hiểu một cách khái lược và nhẹ nhàng về UML và tiến trình phát triển hệ thống hướng đối tượng, thì nội dung có phần chuyên sâu và nâng cao của cuốn sách này sẽ có thể gây cho bạn ít nhiều những khó khăn không cần thiết. Các cuốn sách như là [12], [23] và [10] có lẽ sẽ đáp ứng tốt hơn cho yêu cầu của bạn.

Cuốn sách này cũng không phải là một cuốn sách tra cứu về UML và C++, hoặc là sách dạy lập trình với C++. Các yếu tố của UML và C++ trình bày trong sách này đã được chọn lọc theo mục đích khiêm tốn của sách, và do đó đã không được trình bày vét cạn. Để tra cứu về UML, bạn đọc có thể sử dụng các tài liệu gốc của ba tác giả đầu tiên của UML là các cuốn [6] và [28], hoặc xem trên website [36]. Để tìm hiểu thực sự về C++, thì đã có rất nhiều sách thích hợp, chẳng hạn [31] hay [18].

Vậy rốt cục thì cuốn sách này chọn đối tượng phục vụ cho mình là những người đang hành nghề hay sẽ hành nghề phát triển hệ thống và đang có ý định bổ sung và cập nhật các kiến thức và công nghệ để vận

dụng thực sự vào nghề nghiệp của mình. Đó có thể là các nhà phân tích và thiết kế chuyên nghiệp, cũng có thể là người lập trình, người kiểm định hệ thống, và cũng có thể là các sinh viên đang theo học ở các khoa Công nghệ thông tin, ở bậc đại học hay ở bậc cao học. Nó cũng có thể giúp ích cho các thầy giáo về Công nghệ phần mềm khi soạn thảo giáo trình của mình.

Cuối cùng, tác giả bày tỏ sự biết ơn đối với các đồng nghiệp trong khoa CNTT của trường ĐHBK Hà Nội đã giúp đỡ động viên tác giả nhiều trong khi soạn thảo cuốn sách. Cuốn sách cũng khó tránh khỏi những sai sót, tác giả rất mong nhận được ý kiến phản hồi từ các độc giả, qua e-mail:

banv@it-hut.edu.vn

Nguyễn Văn Ba

Chương I
CÁC CƠ SỞ CỦA LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Phát triển hệ thống hướng đối tượng chính là quá trình thiết lập một hệ thống tin học qua các bước phân tích và thiết kế, để tiến tới lập trình trên một ngôn ngữ hướng đối tượng. Như vậy muốn hiểu việc phân tích và thiết kế hướng đối tượng (mà ta thường gọi là mô hình hoá hướng đối tượng), thì ta cần hiểu trước hết cái đích mà nó hướng tới, tức là lập trình hướng đối tượng. Chương này giúp bạn đọc có một cái nhìn khái quát (không đi vào chi tiết) về các cơ sở của lập trình hướng đối tượng, trên hai phương diện:

- Khuôn phép lập trình hướng đối tượng, tức là các đặc thù trong việc *vận dụng* một ngôn ngữ lập trình hướng đối tượng.
- Máy đối tượng, tức là các nguyên tắc cho việc *cài đặt* một ngôn ngữ lập trình hướng đối tượng.

§1. KHUÔN PHÉP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

1. CÁC KHUÔN PHÉP LẬP TRÌNH

a) Khuôn phép là gì?

Gần đây trong các tài liệu nước ngoài thường xuất hiện thuật ngữ "paradigm", mà ở đây ta diễn tả qua tiếng Việt bởi từ "khuôn phép"[1].

Paradigm bắt nguồn từ thuật ngữ Hy Lạp "paradeigma" có nghĩa là mô hình (model) hay mẫu (pattern). Thomas Kuhn, trong tập sách The Structure of Scientific Revolution, đã đưa ra định nghĩa chi tiết hơn:

---

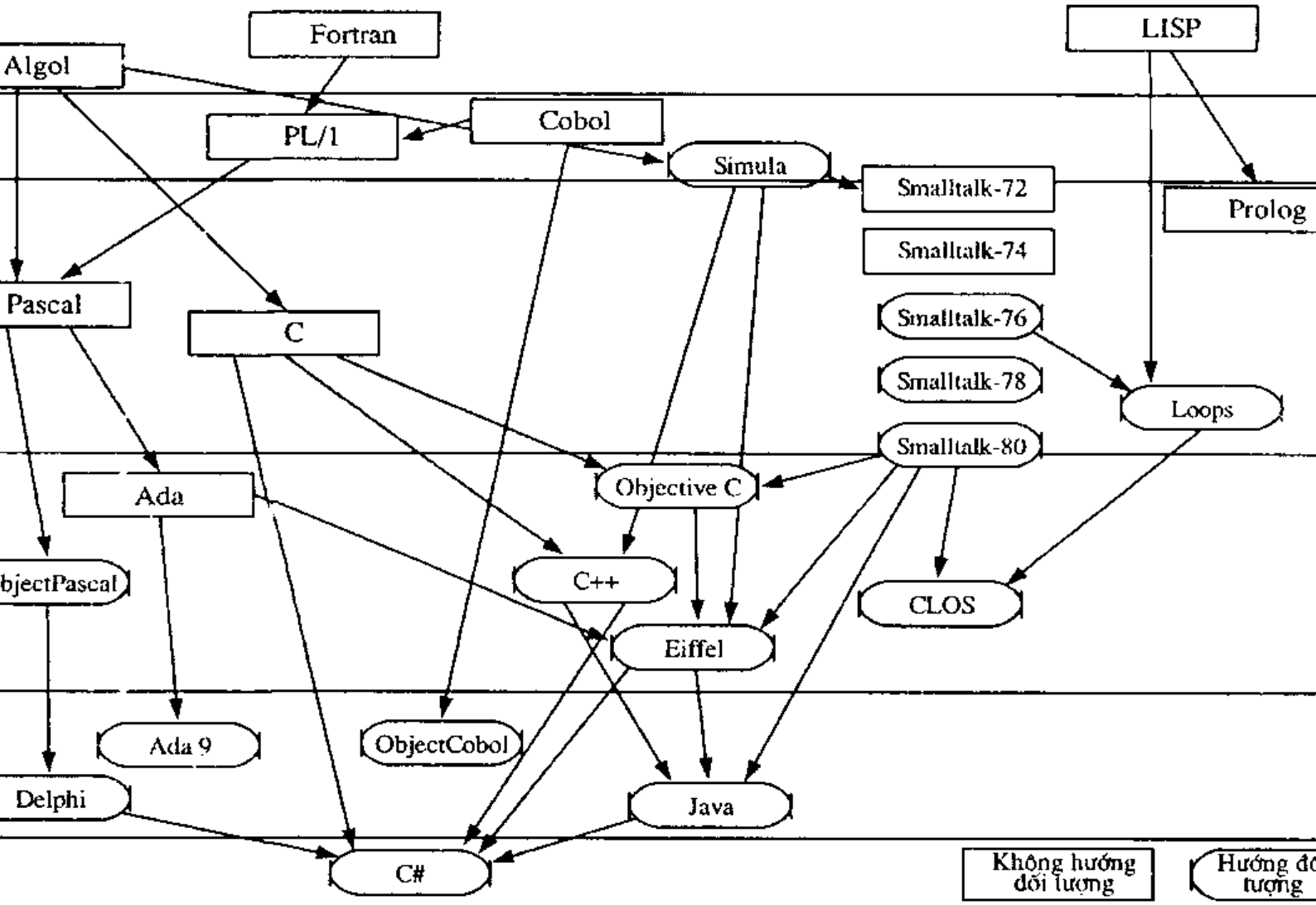
[1] Nếu bạn không thích từ "khuôn phép", thì xin thay bằng từ "mô thức".

"Paradigm là một tập hợp các lý thuyết, tiêu chuẩn và phương pháp được quy tụ lại nhằm biểu diễn cho một cách tổ chức tri thức".

b) Các khuôn phép lập trình truyền thống và hiện đại

Lịch sử phát triển gần 50 năm của các ngôn ngữ lập trình (xem Hình 1.1) đã đề xuất nhiều khuôn phép lập trình khác nhau. Sau đây là một số khuôn phép chính:

- *Lập trình thủ tục* (procedural programming) hay *lập trình chỉ thị* (imperative programming): Là loại lập trình theo cách thành lập một dãy các chỉ thị (lệnh) cho máy tính thực hiện để chạy giải thuật giải quyết bài toán. Đây là cách lập trình truyền thống, với các ngôn ngữ lập trình thế hệ hai và ba, như Pascal, C, Cobol, Ada.
- *Lập trình lôgic* (logical programming): Là loại lập trình trong đó người ta dùng các biểu thức lôgic vị từ để diễn tả các tính chất mà các biến của bài toán phải tuân thủ. Vậy đây là loại lập trình phi thủ tục hay còn gọi là lập trình khai báo (declarative programming), vì nó chỉ khai báo cái mà ta muốn có, mà không cho thủ tục để đạt tới cái đó. Bộ diễn giải (interpreter) của ngôn ngữ lập trình lôgic sẽ thực hiện các bước suy diễn loogic cần thiết để đạt đến yêu cầu. Trong số các ngôn ngữ lập trình lôgic, có thể kể Prolog, C5.
- *Lập trình hàm* (functional programming): là loại lập trình dựa trên các hàm. Các giai đoạn của chương trình được dẫn xuất trực tiếp từ lý thuyết tính toán hàm, do Church đề xuất, gọi là *λ-tính toán*. Trong số các ngôn ngữ lập trình hàm, có thể kể ML, CAML, Scheme và xa hơn một chút là Lisp.
- *Lập trình hướng đối tượng* (object oriented programming): Là loại lập trình sử dụng các đối tượng tin học, như là một sự bọc gói cả dữ liệu và thủ tục vào một, để *mô phỏng* các sự vật thực tế hay lôgic, qua đó mà tạo ra hiệu quả xử lý thông tin. Nhờ khả năng bám sắt thực tế, mà lập trình hướng đối tượng đã tạo ra một phong cách tư duy hoàn toàn mới, rời bỏ cách tư duy thủ tục, cũng như cách tư duy "chuồng bồ câu" (pigeon hole), xem dữ liệu như là các con chim bồ câu chui ra chui vào các ngăn chuồng của mình. Nhờ có sự giấu kín dữ liệu và thủ tục bên trong đối tượng đến mức tối đa, cho nên ta có thể làm *giảm thiểu được sự phức tạp*; chương trình là *dễ*



Hình 1.1. Lịch sử phát triển của các ngôn ngữ lập trình hướng đối tượng

phát triển, dễ chỉnh sửa. Đặc biệt là các đối tượng tin học lại thường gắn với lĩnh vực bài toán nhiều hơn là với chính bài toán (tức là từng ứng dụng cụ thể), bởi vậy chúng có tính ổn định cao, do đó có nhiều khả năng được *sử dụng lại*. Trong số các ngôn ngữ lập trình hướng đối tượng, ta có thể kể Smalltalk, Eiffel, C++, Java.

Các mục tiếp sau đây sẽ lần lượt trình bày một số nét chính trong khuôn phép lập trình hướng đối tượng, mà không đi sâu vào một ngôn ngữ lập trình hướng đối tượng cụ thể nào (trong các Chương VII, VIII, IX ở cuối sách, khi nghiên cứu bước cài đặt hệ thống, ta sẽ đề cập cụ thể các đặc điểm của C++).

2. ĐỐI TƯỢNG VÀ LỚP

a) Môđun hoá và định kiểu

Môđun cơ bản là *đối tượng*. Một đối tượng có một *căn cước* (tên hay địa chỉ) và bao gồm:

- Các *thuộc tính* (hay trường), là các dữ liệu mà nó lưu giữ;
- Các *thao tác* (hay phương thức), là các dịch vụ mà nó có thể cung ứng.

Để dễ mô tả và kiểm soát việc sử dụng các đối tượng, người ta gom chúng theo các kiểu, gọi là các *lớp*. Vậy lớp là một mô hình (một kiểu) cho phép từ đó thành lập các đối tượng có chung cấu trúc và khả năng dịch vụ. Lớp có một cái tên và một đặc tả gồm hai phần: phần định nghĩa các thuộc tính và phần định nghĩa các thao tác. Đối tượng là *cá thể* (instance) của lớp. Vậy thực chất một đối tượng chỉ cần giữ các giá trị thuộc tính của riêng nó, còn định nghĩa thuộc tính và thao tác thì đã có ở lớp của nó rồi.

b) Trách nhiệm và phân loại

Mỗi lớp (và do đó các đối tượng của nó) được giao phó một trách nhiệm nào đó, ẩn chứa trong khả năng lưu giữ thông tin và cung ứng dịch vụ của nó. Nói chung thì một đối tượng/lớp thường có trách nhiệm phản ánh (mô phỏng hay điều khiển) một sự vật (vật chất hay lôgic) trong thế giới thực. Song cũng có những đối tượng/lớp làm những nhiệm vụ khác, có tính chất nội bộ hay trung gian. Người ta

thường phân loại đối tượng/lớp theo trách nhiệm của chúng. Sau đây là một số loại hay gặp:

+ *Thực thể* (entity), dùng để phản ánh một sự vật (sự kiện hay vật thể) trong thế giới thực;
+ *Điều khiển* (control), dùng để kiểm soát quá trình làm việc của các đối tượng khác;
+ *Biên* (boundary), dùng để chuyển đổi các thông tin khi đi qua biên giới của hệ thống, hoặc dùng để cung cấp một khung nhìn (đại lý) tới một số đối tượng khác;
+ *Nguyên thuỷ* (primitive), diễn tả các kiểu nguyên thuỷ thường dùng trong ngôn ngữ lập trình, như nguyên, ký tự, xâu...; các cá thể của chúng là không có định danh;
+ *Liệt kê* (enumeration), diễn tả các kiểu liệt kê định nghĩa sẵn (như Boolean), hoặc do người dùng đưa vào.

c) Bọc kín

Đối tượng giấu kín nhiều nội dung bên trong nó:

+ Chỉ một số thao tác là *công khai* (public), nghĩa là truy cập được từ các đối tượng khác; còn các thuộc tính và các thao tác khác là *riêng tư* (private), chỉ được dùng bên trong đối tượng.
+ Một thao tác công khai, thực ra cũng chỉ công khai phần khai báo (tên thao tác và các tham số hình thức), còn phần cài đặt thì vẫn giấu kín.
+ Đối tượng lại có thể chứa các đối tượng khác trong nó (như là thuộc tính). Như vậy có sự phân cấp quản lý: bên ngoài đối tượng không cần biết đến các đối tượng bên trong. Đối tượng tự nó sẽ quản lý các đối tượng bên trong này, và các đối tượng bên trong đến lượt chúng lại quản lý các thành phần của mình.

Sự bao gói còn có một khía cạnh nữa: ở bên trong đối tượng thì một thao tác có quyền truy cập thoải mái tới các thuộc tính (dữ liệu) của đối tượng, giống như đối với các biến tổng thể, mà chẳng cần một nghi thức đặc biệt gì (chẳng hạn nghi thức chuyển giao qua tham số).

Sự bao gói cho phép cục bộ hoá thông tin, nhờ đó mà chương trình là dễ xây dựng, dễ chỉnh sửa.

d) Thừa kế và đa hình

Một lớp có thể *thừa kế* (inherite) một lớp khác. Mối liên quan này, gọi là liên quan cha/con, có thể kéo dài lên trên hay xuống dưới, mở rộng thành mối liên quan tổ phụ/cháu chắt. Một sự thừa kế là *đơn* nếu lớp con chỉ có nhiều nhất một cha, và là *bội* nếu lớp con có thể có nhiều cha.

Nội dung của sự thừa kế là: một đối tượng tạo lập từ một lớp không những có các thuộc tính và thao tác định nghĩa trong lớp đó, mà còn có cả các thuộc tính và thao tác định nghĩa trong các tổ phụ của lớp đó. Như vậy xét về mặt thực hiện, thì sự thừa kế là một cơ chế ngầm cho phép sử dụng lại các phần chương trình đã có.

Tuy nhiên sự thừa kế không phải luôn luôn là cứng nhắc như trên. Thực vậy, nó còn cho phép một lớp cháu chắt định nghĩa lại một thao tác đã định nghĩa ở một lớp tổ phụ, mà vẫn giữ nguyên tên cũ. Nhờ vậy, cùng một lời gọi thao tác, khi vận dụng vào đối tượng này hay đối tượng khác, có thể kích hoạt những cách thực hiện khác nhau. Gọi đó là sự *đa hình* hay *đa xạ* (polymorphism).

3. SỰ HOẠT ĐỘNG CỦA ĐỐI TƯỢNG

a) Khai báo, tạo lập, khởi gán và loại bỏ đối tượng

Để có thể sử dụng một đối tượng, nghĩa là yêu cầu đối tượng thực hiện một dịch vụ mà nó có, thì trước hết phải *khai báo* nó. Có ba chỗ để khai báo đối tượng:

+ Đối tượng là một thuộc tính (khai báo trong một lớp);
+ Đối tượng là một tham số hình thức (khai báo trong danh sách tham số của một thao tác);
+ Đối tượng là một biến cục bộ của một thao tác (khai báo trong thân của thao tác).

Khai báo chỉ nhằm cho tên của đối tượng (và tên của lớp tương ứng) để rồi có thể dùng tên đó trong văn bản chương trình. Nhưng muốn có đối tượng hoạt động thực sự, phải *tạo lập* và *khởi gán* nó. Tạo lập đối tượng thường được thực hiện bằng một toán tử đặc biệt, chẳng hạn trong Java là toán tử new. Khởi gán cho đối tượng, có thể thực hiện độc lập với tạo lập, thông qua một thao tác bất kỳ nào đó, nhưng

cũng có thể thực hiện ngay sau khi tạo lập (một cách tự động) nhờ một thao tác đặc biệt gọi là constructor đã được định nghĩa sẵn trong lớp tương ứng. Đối tượng khi đã hết tác dụng, có thể loại bỏ. Sự ra đời của đối tượng vào lúc chương trình đang chạy, chứ không phải lúc chương trình được biên dịch, gọi là sự *kết nối muộn* (late binding).

Như vậy, thực hiện một ứng dụng đối tượng chính là định nghĩa một tập hợp các lớp, rồi tạo lập một hay nhiều đối tượng từ các lớp đó và khởi động một thao tác trên một đối tượng đó, để rồi thao tác này sẽ kéo theo các thao tác trên các đối tượng đã tạo lập, gây nên sự kích hoạt lan truyền, tạo ra một kịch bản cho phép thực hiện một yêu cầu xử lý nào đó.

b) Giao tiếp bằng thông điệp

Một đối tượng là đang hoạt động, nếu một thao tác của nó đang được thực hiện. Khi hoạt động, đối tượng có thể tương tác với đối tượng khác (tức là làm cho đối tượng khác đi vào hoạt động). Hình thức tương tác duy nhất giữa các đối tượng là chuyển giao *thông điệp*. Thông điệp (message) là một yêu cầu gửi tới một đối tượng nhằm huy động một thao tác vốn có ở đối tượng này. Thông điệp phải mang tên đối tượng nhận, tên thao tác và danh sách các tham số thực sự của thao tác.

Thoạt nhìn, thì thao tác chẳng khác mấy một lời gọi lập trình truyền thống. Tuy nhiên cần thấy rõ các khác biệt quan trọng:

+ Thông điệp được gửi tới đối tượng, chứ không gửi trực tiếp tới thao tác. Điều đó có nghĩa là thao tác phải được thực hiện trên các dữ liệu và bối cảnh của đối tượng nhận.
+ Chỉ có các thao tác công khai mới được gọi tường minh trong thông điệp. Tuy nhiên cũng có thể huy động các thao tác riêng tư, nếu có một thao tác công khai giữ vai trò một đại lý: bấy giờ tên thao tác trong thông điệp là tên đại lý, để rồi thao tác này sẽ gọi các thao tác riêng tư sau.
+ Các tham số của một thông điệp có thể là các giá trị nguyên thuỷ (như số, xâu...), nhưng cũng có thể là một đối tượng thuộc một lớp nào đó, kể cả một tín hiệu điều khiển (tín hiệu cũng là một loại đối tượng).
+ Thông thường thì thông điệp cũng giống lời gọi chương trình con trong lập trình truyền thống ở chỗ khi gửi thông điệp đi thì

đối tượng phát sẽ ngưng hoạt động để chờ trả lời: đối tượng nhận sau khi hoàn thành dịch vụ sẽ trả lại điều khiển cho đối tượng phát kèm với kết quả trả lại nếu có và đối tượng phát lại tiếp tục hoạt động. Gọi đó là sự chuyển giao thông điệp *đồng bộ*. Tuy nhiên trong một môi trường nhiều CPU (chẳng hạn trên mạng), có thể đối tượng phát và đối tượng nhận ở trên hai CPU khác nhau; bây giờ thông điệp có thể phát đi mà không cần trả lời, đối tượng phát sẽ tiếp tục làm việc ngay. Như thế hai đối tượng có thể làm việc đồng thời. Gọi đó là sự chuyển giao thông điệp *không đồng bộ*.

4. SỰ TIẾP NỐI CÁC THÔNG ĐIỆP

Sự tiếp nối các thông điệp tạo thành một dãy gọi là một kịch bản. Trong kịch bản đó cũng có thể có sự rẽ nhánh hay lặp, như trong sự tiếp nối các câu lệnh ở lập trình truyền thống. Tuy nhiên việc sử dụng rẽ nhánh hay lặp ở đây cũng có sự khác biệt.

a) Phép chọn (rẽ nhánh)

Trong các ngôn ngữ lập trình hướng đối tượng cũng thường có các câu lệnh điều kiện thông thường (IF, CASE). Song các câu lệnh điều kiện đó không thể dùng để kiểm chứng bản chất của đối tượng để thực hiện các xử lý riêng biệt được. Bấy giờ người ta lợi dụng sự thừa kế: sử dụng một cái tên chung cho một thao tác có trong mọi lớp con của một lớp và đã được định nghĩa lại trong mỗi lớp con đó. Như thế một phép chọn trong nhiều trường hợp chỉ được thực hiện bởi một chuyển giao thông điệp duy nhất.

Chẳng hạn đáng lẽ phải viết:

```
Nếu một người là lao động tự do thì
  tính thuế thu nhập theo thủ tục 1
Không thì Nếu người đó là lao động hưởng lương thì
  tính thuế thu nhập theo thủ tục 2
Không thì
  tính thuế thu nhập theo thủ tục 3
Hết nếu
```

thì ta chỉ cần viết

```
motNguoi.tinhthueTN();
```

Không có một kiểm chứng nào được thực hiện cả. Thực ra sự lựa chọn xảy ra vào lúc đối tượng motNguoi được tạo lập. Nó sẽ được tạo lập theo một trong ba lớp con của lớp Nguoi là LaodongTudo, LaodongHuongluong, KhongAnluong mà thao tác tinhthueTN() trong mỗi lớp đó được thực hiện theo một cách khác nhau, vì đã được định nghĩa lại.

b) Phép lặp (chu trình)

Nhiều khi ta phải truy cập một lúc hay lần lượt tới nhiều đối tượng. Nếu các đối tượng đó là tách rời và mỗi đối tượng phải được truy cập bằng tên riêng của nó, thì ta phải đưa ra một loạt các thông điệp như sau:

```
dt1.thongdiep1();
dt2.thongdiep2();
........
dtn.thongdiepn();
```

Tuy nhiên nếu các đối tượng đó có cùng bản chất, ta có thể gom chúng vào một đối tượng gọi là *đối tượng bội*. Lớp của đối tượng bội được gọi là *lớp số nhiều*. Lớp này quản lý một cấu trúc dữ liệu như danh sách, tập hợp, vectơ, từ điển... (các lớp cho những cấu trúc dữ liệu này đã được định nghĩa sẵn trong ngôn ngữ lập trình hướng đối tượng). Và như vậy chỉ cần gửi một thông điệp duy nhất cho đối tượng bội, việc lặp sẽ được thực hiện bên trong đối tượng bội.

Chẳng hạn muốn đưa tất cả các ô tô ra khỏi một Gara, thay vì viết:

```
Với mọi ôtô trong gara, hãy làm
    oto[so].chora();
Hết làm
```

thì ta xem motGara là một đối tượng bội, trong đó có chứa một tập hợp ôtô và chỉ cần gửi một thông điệp cho nó:

```
motGara.choraCacOto();
```

Thao tác ChoraCacOto khắc sẽ huy động cấu trúc dữ liệu trong nó để lần lượt đưa các ôtô ra ngoài.

5. BỐI CẢNH CỦA ĐỐI TƯỢNG

Mỗi đối tượng có một bối cảnh (tầm nhìn) bao gồm mọi đối tượng mà nó biết và huy động được. Có hai cách huy động:

+ Đối tượng gửi thông điệp cho một đối tượng trong bối cảnh, và như vậy chuyển điều khiển cho đối tượng đó.
+ Đối tượng chuyển giao một đối tượng trong bối cảnh theo tham số, và như vậy tăng cường đối tượng đó vào bối cảnh của đối tượng nhận.

Có năm loại đối tượng thuộc vào bối cảnh của một đối tượng, được kể lần lượt sau đây.

a) Các đối tượng thuộc tính (thành phần) của lớp

Đây là những đối tượng bên trong của lớp. Giả sử chúng đã được tạo lập, hoặc là cùng lúc với chính đối tượng, hoặc nhờ một thao tác khởi gán được thực hiện trước khi dùng các thao tác khác tác động lên các đối tượng thuộc tính này.

Mọi thao tác của lớp đều có thể truy cập trực tiếp đến một thuộc tính của lớp, cũng như đến mọi thuộc tính thừa kế, miễn là các thuộc tính thừa kế này (trong các lớp tổ phụ) đã được khai báo ở dạng bảo hộ (protected). Bấy giờ ta nói đối tượng hợp thành chuyển điều khiển cho đối tượng thành phần.

b) Các đối tượng được truyền đến theo tham số

Nếu trong lập trình truyền thống, các tham số chỉ là các dữ liệu bất động, thì trong lập trình hướng đối tượng, tham số lại có thể là đối tượng.[1]

Trong một thao tác, thì các tham số hình thức được xem như các biến cục bộ. Các biến này đã được khởi gán khi thao tác này được gọi bởi một đối tượng gọi (vậy khi thao tác này thực hiện thì các đối tượng tham số đã bảo đảm tồn tại). Tiếp đó thì đối tượng tham số có thể nhận một thông điệp. Như thế, thực chất của việc truyền đối tượng theo

[1] Trong cuốn sách này chúng tôi xem hai thuật ngữ *tham số* và *tham đối* là đồng nghĩa, cùng dịch cho thuật ngữ tiếng Anh là *parameter*. Khi muốn gợi nhớ rằng đây không nhất thiết là một số, thì chúng tôi dùng từ tham đối.

tham số là sự chuyển giao một tập hợp các thao tác mà đối tượng nhận sẽ được phép sử dụng. •

c) Các đối tượng cục bộ

Các đối tượng là biến cục bộ được người lập trình đưa vào thân của một thao tác, nhằm làm một nhiệm vụ trung gian nào đó, chẳng hạn để tạm thời tiếp nhận một đối tượng trả về từ một lời gọi, hoặc để sử dụng những thao tác của nó. Khi thao tác kết thúc, nếu đối tượng cục bộ trước đó không được truyền đi theo tham số, thì nó sẽ tự động bị huỷ bỏ.

d) Các đối tượng trả lại từ một thao tác

Một thao tác thực hiện xong có thể trả về một đối tượng. Đối tượng này cũng được sử dụng như một biến cục bộ.

đ) Chính đối tượng đang xét

Một thao tác trong một lớp có thể huy động một thao tác khác trong lớp đó, cho dù đó là thao tác thừa kế hay không. Lời gọi ở đây không cần chỉ rõ đối tượng nhận (vì là chính nó rồi). Gọi đó là lời gọi cục bộ. Thông thường thì một thao tác sử dụng cục bộ được khai báo là riêng tư (private), và đo đó không thể gọi từ bên ngoài đối tượng.

e) Các lớp xem như đối tượng

Ngoài năm loại đối tượng mà một đối tượng có thể huy động nói trên, thì đối tượng còn có thể huy động một *thao tác lớp*. Các thao tác này tác dụng lên lớp, chứ không phải lèn một đối tượng của lớp và không đòi hỏi tạo lập một đối tượng để dùng chúng. Trong lời gọi của các thao tác này, nơi đến sẽ là lớp, thay vì đối tượng như thường lệ. Chẳng hạn

```
Nhithuc nhithuc = Nhithuc.taolap(a,b);
```

Thao tác taolap() là một thao tác lớp định nghĩa trên lớp Nhithuc. Nó trả về một đối tượng của lớp Nhithuc với các hệ số cho bởi các tham số thực sự.

§2. MÁY ĐỐI TƯỢNG

1. KHÁI NIỆM VỀ MÁY TRỪU TƯỢNG

Một chương trình viết trên ngôn ngữ máy sẽ được thực hiện trực tiếp bởi một máy tính thật. Trái lại một chương trình viết trên một ngôn ngữ cấp cao không thể thực hiện ngay trên máy tính thật, mà phải qua xử lý của một phần mềm (dịch hay diễn giải).

Tuy nhiên, để thấy rõ cơ chế hoạt động của một ngôn ngữ lập trình cấp cao, người ta hình dung ra một máy tính giả định có thể thực hiện trực tiếp một chương trình viết trên ngôn ngữ đó. Gọi đó là một máy trừu tượng. Chính bộ diễn giải (interpreter) đã mô phỏng sự hoạt động của máy trừu tượng này để thực hiện chương trình trên ngôn ngữ cấp cao (mà không dịch ra một chương trình đích). Một ngôn ngữ lập trình hướng đối tượng cũng tương ứng với một máy trừu tượng, gọi là *máy đối tượng*.

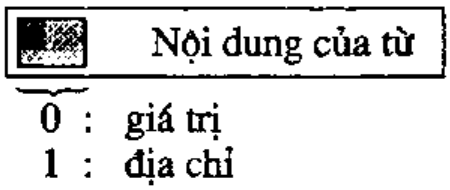
Ở đây ta cố gắng làm rõ, tuy không thể chi tiết lắm, cơ chế hoạt động của máy đối tượng, nhằm qua đó giải thích cơ chế hoạt động của một chương trình hướng đối tượng.

2. BỘ NHỚ CỦA MÁY ĐỐI TƯỢNG

Bộ nhớ của máy đối tượng là một dãy các từ máy, và thường tách thành ba phần:

- Một phần gọi là đống (tổ chức bảng băm) dùng để chứa các đối tượng, chúng xuất hiện dần dà trong bộ nhớ theo sự tạo lập của chúng.
- Một phần khác dùng để chứa các lớp, dưới một dạng thực hiện được.
- Phần còn lại dùng làm vùng làm việc cho các thao tác. Chỗ nhớ cho các biến cục bộ và các tham số của một thao tác sẽ được cấp phát động (cấp phát khi thao tác được huy động, và thu hồi khi thao tác kết thúc hoạt động).

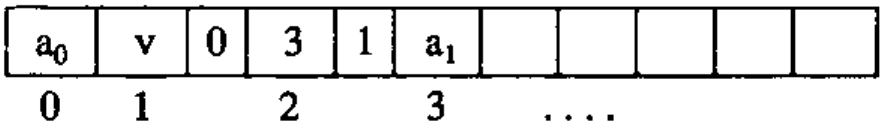
Mỗi từ của bộ nhớ gồm có hai phần. Phần đầu dùng để chỉ nội dung của từ là biểu diễn cho một giá trị hay một địa chỉ. Phần còn lại là nội dung của từ. Một giá trị biểu diễn nội dung này có thể là một số nguyên, một số thực, một ký tự hay một giá trị Boolean.



Hình I.2. Cấu trúc của một từ bộ nhớ

3. CẤU TRÚC CỦA MỘT ĐỐI TƯỢNG TRONG BỘ NHỚ

Một đối tượng chiếm một số từ liên tiếp trong bộ nhớ. Từ đầu tiên chứa địa chỉ cho phép truy cập vào lớp của đối tượng. Từ thứ hai chứa kích cỡ (số từ) của đối tượng tương ứng với lớp đó. Sau đó là phần chứa các dữ liệu của đối tượng. Một đối tượng có thể được truy cập bằng địa chỉ, ấy là địa chỉ của từ đầu tiên trong cấu trúc của đối tượng.



0 : địa chỉ của lớp

1 : kích cỡ của đối tượng

2 : thuộc tính nguyên, có giá trị 3

3 : thuộc tính đối tượng, có địa chỉ a_1

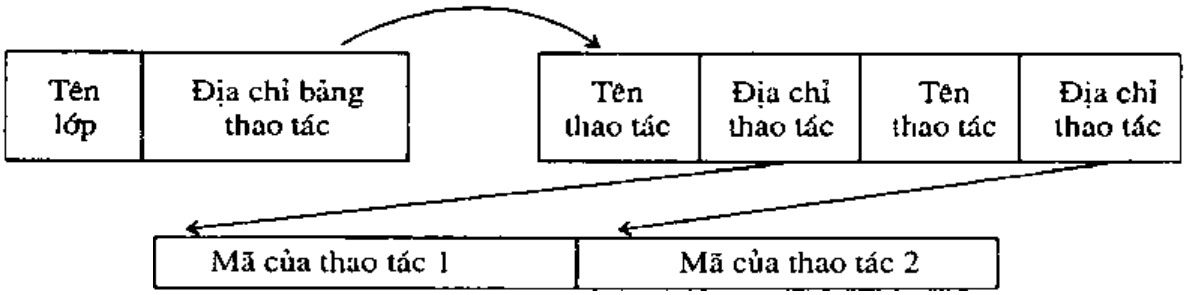
.....

Hình I.3. Cấu trúc của một đối tượng trong bộ nhớ

4. CẤU TRÚC CỦA MỘT LỚP TRONG BỘ NHỚ

Một lớp tạo thành bởi nhiều phần, liên kết với nhau bởi các địa chỉ:

- Phần đầu chứa tên lớp và một địa chỉ trỏ đến bảng các thao tác;
- Bảng các thao tác là một dãy các cặp: tên thao tác và địa chỉ phần mã của thao tác. Cuối bảng có thể thêm địa chỉ của lớp cha, nếu có;
- Phần mã thao tác gồm mã (chương trình chạy được) của các thao tác, đặt nối tiếp nhau.



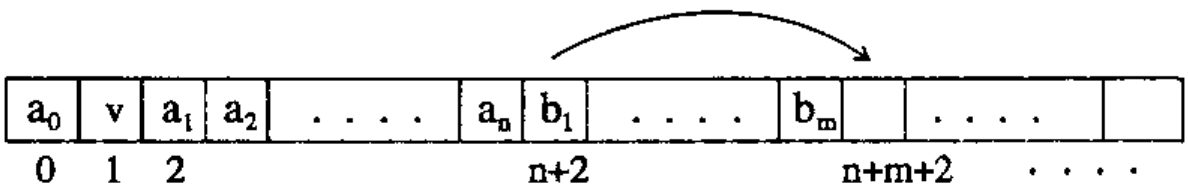
Hình I.4. Cấu trúc của một lớp trong bộ nhớ

Một lớp được truy cập bằng địa chỉ, ấy là địa chỉ của từ đầu tiên của nó (tên lớp). Địa chỉ được đặt trong mọi đối tượng được tạo lập từ lớp. Ta nói rằng đối tượng biết lớp của nó. Nhiều lớp con có thể định nghĩa các thao tác có cùng đặc tả: cùng tên, cùng tham số, mà không gây sự nhận biết nhầm lẫn khi dùng thao tác.

5. CẤU TRÚC VÙNG LÀM VIỆC CỦA MỘT THAO TÁC

Một thao tác khi được gọi, sẽ được cấp một vùng làm việc (cấp phát động) để lưu các tham số và các biến cục bộ của nó.

Hai từ đầu tiên của vùng làm việc được dành riêng để lưu địa chỉ vùng làm việc của thao tác gọi, và lưu kích cỡ của vùng làm việc của thao tác hiện thời (cho phép tính địa chỉ của từ đầu tiên của phần bộ nhớ chưa dùng tới). Tiếp theo là n từ dành để ghi địa chỉ của n tham số (như vậy các tham số là được chuyển giao theo địa chỉ, chứ không phải chuyển giao theo giá trị). Tiếp đến là m từ dành để ghi địa chỉ của m đối tượng là các biến cục bộ. Sau cùng thì chính là m cấu trúc của các đối tượng tương ứng với các biến cục bộ.



0 : địa chỉ vùng gọi
1 : kích cỡ vùng hiện tại
2 đến n+1 : địa chỉ của các tham số
n+2 đến n+m+1 : địa chỉ của đối tượng là biến cục bộ
từ n+m+2: các cấu trúc của các đối tượng biến cục bộ

Hình I.5. Cấu trúc của một vùng làm việc của một thao tác

6. CƠ CHẾ ĐỊA CHỈ

Một máy đối tượng muốn thực hiện một chương trình, phải di chuyển trong bộ nhớ để đọc và điều chỉnh các nội dung nhớ. Muốn thế nó phải lần theo địa chỉ. Có nhiều loại cơ chế địa chỉ. Ở đây, bởi sự cấp phát động, ta phải dùng các địa chỉ tương đối và theo chỉ số.

Trong vùng nhớ cho các lớp và đối tượng, có thể sử dụng các con trỏ. Trong các vùng làm việc của các thao tác, thì việc truy cập vào các đối tượng là tham số hay biến cục bộ thực hiện bằng các địa chỉ tương đối (độ dời so với địa chỉ vùng làm việc).

7. HOẠT ĐỘNG CỦA MÁY ĐỐI TƯỢNG

Một thao tác đầu tiên cho phép tạo lập ít nhất một đối tượng. Tiếp đó thông điệp đầu tiên được gửi tới đối tượng này. Qua thông điệp, máy biết địa chỉ của đối tượng và tên của thao tác được gọi. Dựa theo địa chỉ lớp có trong cấu trúc của đối tượng, máy truy nhập vào lớp và thông qua bảng các thao tác, tìm thao tác cùng tên. Nếu tìm được, nó truy cập được vào mã của thao tác và thực hiện mã đó. Nếu không máy theo địa chỉ của đối tượng cha để tiếp tục tìm kiếm, nếu gặp thao tác thì thực hiện. Nếu không cứ thế lần ngược lên các tổ phụ. Rốt cục nếu không tìm được tên thao tác thì máy ngừng và một thông báo lỗi được đưa ra. Cứ mỗi một chuyển giao thông điệp (lời gọi một thao tác) thì máy thành lập một vùng làm việc mới (cấp phát động). Điều này cho phép thực hiện sự đệ quy. Khi một thao tác thực hiện xong, máy quay về vùng làm việc trước đó (và vùng làm việc vừa xong coi như bị loại bỏ).

Chương II
MÔ HÌNH HOÁ HƯỚNG ĐỐI TƯỢNG

Các việc phân tích và thiết kế hệ thống, tuy có khác nhau về nhiệm vụ và mục tiêu, về mức độ trừu tượng hoá, song chúng có chung những đặc điểm:

- đều phải đối đầu với sự phức tạp,
- đều là những quá trình *nhận thức* và *diễn tả* sự phức tạp thông qua các mô hình.

Chương này nhằm giới thiệu chung về khái niệm mô hình và mô hình hoá; tiếp theo sẽ đề cập sâu hơn về mô hình hoá hướng đối tượng, trong đó giới thiệu một cách khái quát ngôn ngữ mô hình hoá thống nhất UML và tiến trình mô hình hoá RUP. Một cái nhìn chung như vậy sẽ giúp bạn đọc dễ dàng theo dõi quá trình phát triển hệ thống, trình bày trong các chương sau của cuốn sách.

§1. ĐẠI CƯƠNG VỀ MÔ HÌNH HOÁ

1. MÔ HÌNH

Mô hình là một dạng trừu tượng hoá của một hệ thống thực. Nói chi tiết hơn, thì mô hình là một hình ảnh (một biểu diễn) của một hệ thống thực, được diễn tả:

- ở một mức độ trừu tượng hoá nào đó,
- theo một quan điểm (hay một góc nhìn) nào đó,
- bởi một hình thức diễn tả hiểu được (văn bản, phương trình, bảng, đồ thị v.v...) nào đó.

Việc dùng mô hình để nhận thức và diễn tả một hệ thống được gọi là *mô hình hoá*. Như vậy quá trình phân tích và thiết kế hệ thống cũng thường được gọi chung là quá trình mô hình hoá hệ thống.

2. MỤC ĐÍCH VÀ CHẤT LƯỢNG MÔ HÌNH HOÁ

Có ba mục đích của mô hình hoá:

- Mô hình hoá để hiểu: Nói cho cùng thì hiểu tức là hình thành được một hình ảnh xác thực và giản lược (ở trong đầu hay ở trên giấy) về đối tượng được tìm hiểu. Nói ngắn như gọn Jean Piaget thì "hiểu tức là mô hình hoá". Không thể nói rằng hiểu mà chưa có mô hình. Ngược lại, vận dụng tốt các loại mô hình, ta sẽ nhận thức vấn đề dễ dàng và nhanh chóng hơn.
- Mô hình hoá để trao đổi: Vì tính hiểu được của mô hình mà nó trở thành một thứ ngôn ngữ cho phép trao đổi giữa những người cùng quan tâm tới một vấn đề hay một hệ thống chung.
- Mô hình hoá để hoàn chỉnh: Nhờ sự minh bạch của mô hình mà ta dễ dàng nhận thấy hệ thống đã phù hợp với nhu cầu chưa, có chặt chẽ, có đầy đủ không, nhờ đó mà có thể hoàn thiện thêm. Hơn nữa mô hình còn giúp ta kiểm định, mô phỏng, thực hiện.

Từ ba mục đích trên, ta suy ra mô hình tốt phải đạt các yêu cầu sau:

+ dễ đọc,
+ dễ hiểu,
+ dễ trao đổi,
+ xác thực,
+ chặt chẽ,
+ đầy đủ,
+ dễ thực hiện.

3. CÁC PHƯƠNG PHÁP MÔ HÌNH HOÁ

Ngày nay tồn tại rất nhiều phương pháp mô hình hoá (phân tích và thiết kế) các hệ thống thông tin.

Ta hiểu phương pháp mô hình hoá là sự kết hợp của ba thành phần:

- Một *ký pháp* (notation) bao gồm một số khái niệm và mô hình. Mỗi phương pháp đều phải dựa trên một số không nhiều các khái niệm cơ bản và sử dụng một số mô hình diễn tả các khái niệm trên, kèm với các kỹ thuật triển khai hay biến đổi các mô hình đó.

- Một *tiến trình* (process) bao gồm các bước đi lần lượt, các hoạt động cần làm, các sản phẩm qua từng giai đoạn (như tư liệu, mô hình...), cách điều hành tiến trình đó và các đánh giá chất lượng của các kết quả thu được.
- Một (hay một số) *công cụ hỗ trợ* (CASE). Đó là các phần mềm hỗ trợ cho quá trình mô hình hoá, thường có các khả năng như sau:

  + sản sinh các mô hình và biểu đồ,

  + biến đổi và điều chỉnh nhanh các mô hình và biểu đồ,

  + kiểm tra cú pháp, sự chặt chẽ, sự đầy đủ,

  + kiểm thử và đánh giá,

  + mô phỏng và thực hiện mô hình.

4. HAI XU HƯỚNG CHÍNH CỦA MÔ HÌNH HOÁ

Dù đã có rất nhiều phương pháp mô hình hoá khác nhau, song ta có thể gom chúng theo hai xu hướng:

- Mô hình hoá *hướng chức năng* (từ 1970, với Youndon, Constantine, DeMarco v.v...), lấy chức năng làm đơn vị phân rã hệ thống;
- Mô hình hoá *hướng đối tượng* (từ 1990, với Booch, Rumbaugh, Jacobson, Yourdon & Coad, Shlaer & Mellor v.v...) lấy đối tượng làm đơn vị phân rã hệ thống.

Hai xu hướng mô hình hoá nói trên phản ánh hai khuôn phép lập trình khác biệt: lập trình theo thủ tục và lập trình theo đối tượng.

Cuốn sách này chỉ đề cập tới mô hình hoá hướng đối tượng. Nói cụ thể hơn, thì nó tập trung trình bày việc vận dụng một ngôn ngữ mô hình hoá hướng đối tượng là UML và tiếp đó là việc thục thi các kết quả mô hình hoá trên một ngôn ngữ lập trình hướng đối tượng là C++. Nhưng trước hết, để có một cái nhìn xuyên suốt, phần cuối của chương đại cương này sẽ cho một tổng quan về ngôn ngữ UML, tiến trình RUP và công cụ trợ giúp Rational Rose.

§2. NGÔN NGỮ UML

Ngôn ngữ UML (Unified Modeling Language) là một loại ký pháp mô hình hoá hướng đối tượng.

1. XUẤT XỨ

Ý tưởng về đối tượng bắt nguồn từ ngôn ngữ lập trình Simula, nhưng nó chỉ trụ vững được kể từ cuối các năm 80, với sự ra đời của ngôn ngữ lập trình Smalltalk và C++. Khi lập trình hướng đối tượng đã thành đạt, thì lập tức có nhu cầu về các phương pháp mô hình hoá hướng đối tượng. Từ đầu các năm 90, nhiều phương pháp mô hình hoá hướng đối tượng lần lượt ra đời, như OOAD của Grady Booch, OMT của Jim Rumbaugh (và các người khác), OOSE/Objectory của Ivar Jacobson (và các người khác), Pusion của Derek Coleman (và các người khác), OOA/OOD của Peter Coad và Edward Yourdon v.v... Các phương pháp đó sử dụng các ký pháp khác nhau, các bước đi khác nhau, tạo ra một sự phân tán, thậm chí là những cuộc chiến không cần thiết. Xu hướng tất yếu là phải quy tụ lại, phải có chuẩn.

Tháng 1/1994 G. Booch và J. Rumbaugh bắt đầu hợp tác trong khuôn khổ công ty phần mềm Rational, nhằm xây dựng một "phương pháp hợp nhất" trên cơ sở hai phương pháp Booch 93 và OMT-2.

Năm 1995 I. Jacobsson, nhân vật chủ chốt của OOSE và các phương pháp Objectory gia nhập sự hợp tác, và từ đó họ được các đồng nghiệp gọi là "ba người bạn" (the three amigos). Họ quyết định thu hẹp mục tiêu, nhằm thành lập một ngôn ngữ mô hình hoá hợp nhất, hơn là một phương pháp hợp nhất.

Tháng 10/1995 họ đưa ra phác thảo UML, phiên bản 0.

Tháng 6/1996 ra đời phiên bản UML 0.9.

Tháng 1/1997 IBM và Softeam kết hợp với các thành viên UML để đưa ra phiên bản 1.1.

Ngày 14/11/1997 UML 1.1 được các thành viên OMG (Object Management Group) bỏ phiếu công nhận là chuẩn cho các ngôn ngữ mô hình hoá và trao đặc quyền xét lại (RTF - Revision Task Force) cho UML.

Tháng 6/1998 ra đời UML 1.2.

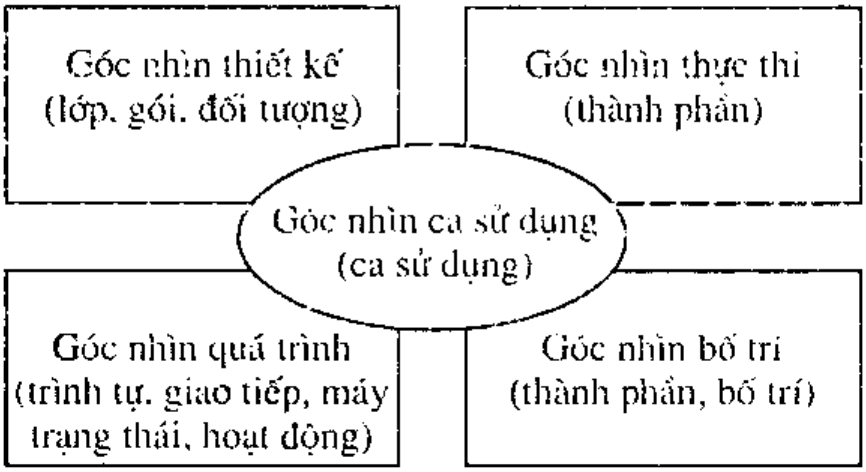
Tháng 10/1998 ra đời UML 1.3.

Tháng 5/2001 ra đời UML 1.4.

Tháng 6/2003 thông qua bản dự thảo UML 2.0 (dự kiến phê duyệt vào cuối năm 2004). Cuốn sách này trình bày UML theo phiên bản 2.0, dựa trên các đặc tả đã được chấp nhận cho đến nay.

2. CÁC GÓC NHÌN CỦA UML

UML cung cấp các mô hình để diễn tả hệ thống. Nhưng mỗi mô hình thì chỉ có thể diễn tả hệ thống theo một góc nhìn (view) nhất định. Tổng hợp lại thì UML cung cấp năm góc nhìn đối với hệ thống (Hình II.1). Mỗi góc nhìn thực hiện bởi một số biểu đồ (mô hình). Có thể có biểu đồ cùng thuộc vào các góc nhìn khác nhau.



Hình II.1. Các góc nhìn của UML

(1) Góc nhìn ca sử dụng: Là góc nhìn từ ngoài vào hệ thống. Đó là cách nhìn của các người dùng cuối, các người phân tích, người kiểm định. Nó không phản ánh tổ chức bên trong của phần mềm, mà chỉ làm rõ các chức năng lớn mà hệ thống phải đáp ứng cho người dùng. Với UML thì sắc thái tĩnh của góc nhìn này được thâu tóm trong các biểu đồ ca sử dụng. Còn sắc thái động của góc nhìn này được thâu tóm trong các biểu đồ tương tác, các biểu đồ máy trạng thái và các biểu đồ hoạt động.

(2) Góc nhìn thiết kế (còn gọi là góc nhìn lôgic): Là góc nhìn vào bên trong hệ thống, cho thấy các nhiệm vụ của hệ thống được thiết kế ra sao (thành các lớp, các giao diện, các hợp tác). Đó là cách nhìn của những người thiết kế hệ thống. Với UML, thì sắc thái tĩnh của góc nhìn này thể hiện trong các biểu đồ lớp, các biểu đồ đối tượng. Còn sắc thái động của góc nhìn này thể hiện trong các biểu đồ tương tác, các biểu đồ máy trạng thái, các biểu đồ hoạt động.

(3) Góc nhìn quá trình (còn gọi là góc nhìn song hành): Phản ánh các lộ trình điều khiển, các quá trình thực hiện, cho thấy sự hoạt động song hành hay đồng bộ của hệ thống. Với UML thì góc nhìn này

được thể hiện cùng với các biểu đồ như góc nhìn thiết kế, nhưng tập trung chú ý vào các lớp chủ động, là các lớp biểu diễn cho các lộ trình điều khiển và quá trình thực hiện.

(4) Góc nhìn thực thi (còn gọi là góc nhìn thành phần): Là góc nhìn đối với dạng phát hành của phần mềm (hệ thống vật lý) bao gồm các thành phần và tệp tương đối độc lập; có thể được lắp ráp theo nhiều cách để tạo ra hệ thống chạy được. Với UML, sắc thái tĩnh của góc nhìn này được thể hiện bởi các biểu đồ thành phần. Còn sắc thái động của góc nhìn này được thể hiện bởi các biểu đồ tương tác, các biểu đồ máy trạng thái, các biểu đồ hoạt động.

(5) Góc nhìn bố trí: Là góc nhìn về tôpô của phần cứng mà trên đó hệ thống được thực hiện. Nó chỉ rõ sự phân bố, sự sắp đặt các phần của hệ thống vật lý trên các đơn vị phần cứng. Với UML thì sắc thái tĩnh của góc nhìn này thể hiện qua các biểu đồ bố trí. Còn sắc thái động của góc nhìn này thể hiện qua các biểu đồ tương tác, các biểu đồ máy trạng thái, các biểu đồ hoạt động.

Có thể vận dụng các góc nhìn nói trên một cách tách biệt, vì với mỗi loại người quan tâm tới hệ thống, như người dùng cuối, người phân tích, người thiết kế, người tích hợp, người kiểm định, người quản lý dự án... thường chỉ tập trung chủ ý tới một phương diện nào đó của hệ thống mà thôi. Tuy nhiên năm góc nhìn trên phải có sự tương hợp lẫn nhau, chẳng hạn các nút trong góc nhìn bố trí phải tương ứng với các thành phần trong góc nhìn thực thi, và các thành phần này lại là sự thực hiện vật lý của các lớp, các giao diện, các hợp tác và các lớp chủ động của hai góc nhìn thiết kế và quá trình. Ngoài ra chẳng phải ngẫu nhiên mà trong Hình II.1, góc nhìn ca sử dụng lại được vẽ chính giữa, bởi vì góc nhìn ca sử dụng có một ảnh hưởng xuyên suốt đối với bốn góc nhìn còn lại. Thực vậy, thiết kế, thực thi, nghiên cứu quá trình hay bố trí đều phải nhằm đáp ứng các nhiệm vụ cơ bản của hệ thống, tức là các ca sử dụng.

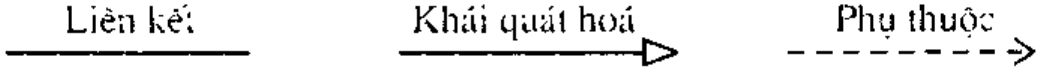
3. CÁC BIỂU ĐỒ CỦA UML

Các mô hình trong UML đều có dạng *biểu đồ* (điagram). Biểu đồ là một đồ thị, trong đó:

- Các *nút* (nodes) là các yếu tố của mô hình có dạng đồ hoạ hai chiều, như là lớp, trạng thái, gói v.v... (Hình II.2)

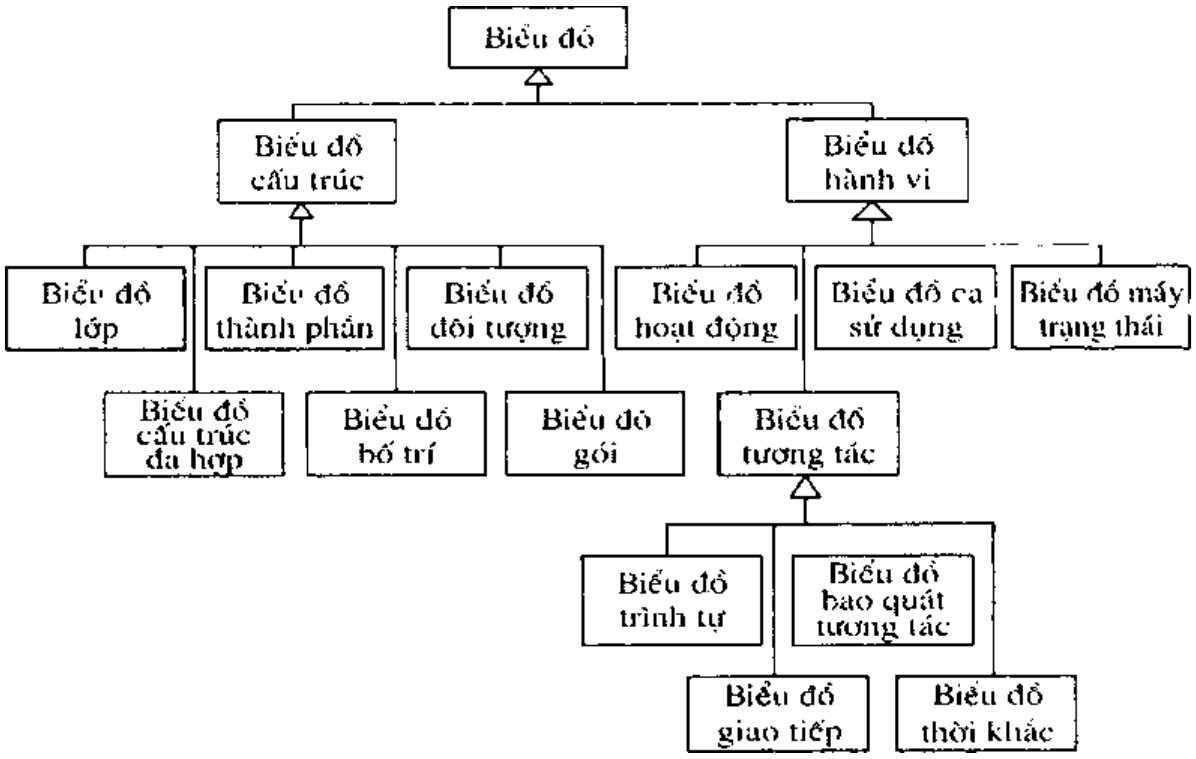
Hình II.2. Một số nút của các biểu đồ

- Các *đường* (paths)[1] là các yếu tố của mô hình có dạng đồ hoạ tuyến tính, như là liên kết, khái quát hoá, phụ thuộc v v..



Hình II.3. Một số đường trong các biểu đồ

UML 1.x có chín loại biểu đồ. UML 2.0 mở rộng thành 13 loại. Chúng được phân thành hai nhóm: các biểu đồ về cấu trúc và các biểu đồ về hành vi. Sự phân loại chi tiết cho trên Hình II.4.

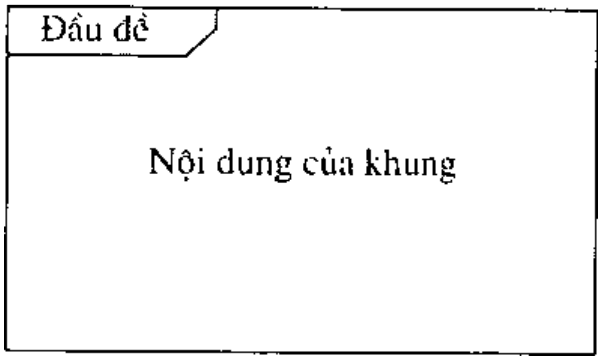


Hình II.4. Phân loại các biểu đồ trong UML 2.0

[1] UML 2.0 dùng thuật ngữ path để trỏ cung hay cạnh vẫn quen dùng trong Lý thuyết đồ thị, có lẽ bởi nó có thể là vô hướng hay có hướng tùy nơi, tùy lúc.

Như vậy so với UML 1.x, thì các biểu đồ cấu trúc đa hợp, bao quát tương tác và thời khắc là hoàn toàn mới. Còn như biểu đồ gói thì trước đây vẫn vẽ song trong các phiên bản UML 1.x lại chưa được kể tới và đặt tên chính thức (trong chín biểu đồ). Ngoài ra thì biểu đồ sơ đồ trạng thái (statechart diagram) nay đổi tên lại là biểu đồ máy trạng thái (state machine diagram) và biểu đồ hợp tác (collaboration diagram) nay đổi tên lại là biểu đồ giao tiếp (communication diagram).

UML 2.0 thường trình bày một biểu đồ bên trong một hình chữ nhật gọi là một *khung* (frame), có gắn ở góc trên trái một *đầu đề* (heading), viết trong một hình chữ nhật nhỏ cắt góc (Hình II.5).



Hình II.5. Khung và đầu đề

Khung là một không gian đặt tên (namespace). Khung là cần thiết khi một biểu đồ được vẽ lồng trong một biểu đồ khác, và nhất là khi mô hình có các phần tử biên, như là các cổng (ports) đối với các lớp và thành phần, các điểm vào/ra (entry/exit points) đối với máy trạng thái. Khi không có các yêu cầu đó thì khung (cùng với đầu đề) có thể bỏ qua.

Đầu đề có cú pháp sau:

[<loại>] <tên> [<tham số>]

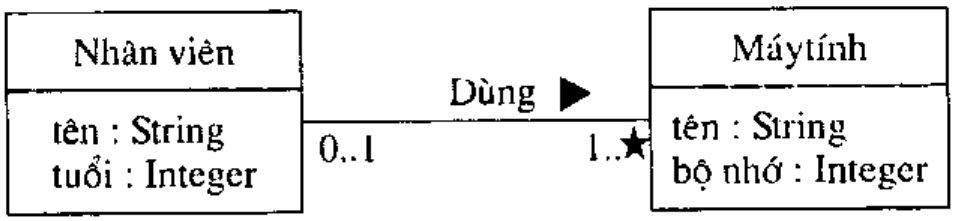
trong đó thì loại trỏ loại của biểu đồ có thể là activity, class, component, interaction, package, statemachine, usecase... (cũng có thể viết tắt, như là pkg thay vì package).

Các chương tiếp sau của cuốn sách sẽ lần lượt trình bày kỹ 13 loại biểu đồ của UML 2.0. Tuy nhiên để bạn đọc có ngay một cái nhìn bao quát, chúng sẽ được giới thiệu một cách tóm tắt tiếp theo đây.

a) Nhóm các biểu đồ về cấu trúc

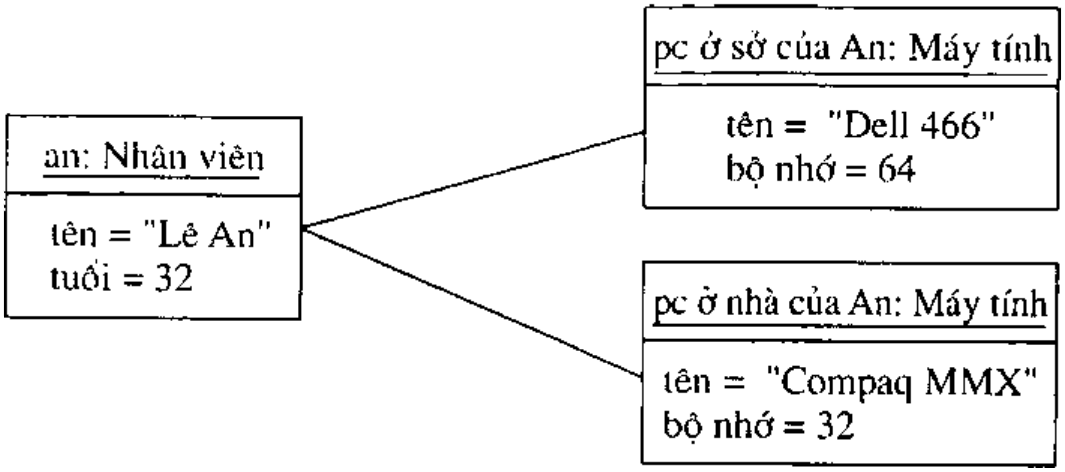
Đây là các biểu đồ dùng để phản ánh sắc thái tĩnh của hệ thống. Có sáu loại biểu đồ thuộc nhóm này.

(1) *Biểu đồ lớp:* Biểu đồ lớp phô bày cấu trúc tĩnh của các lớp trong hệ thống (xem Hình II.6). Các lớp biểu diễn cho các sự vật mà hệ thống quan tâm. Các lớp có thể liên quan với nhau theo nhiều cách: liên kết (kết nối với nhau), phụ thuộc (lớp này phụ thuộc hay dùng một lớp khác), khái quát hoá (một lớp được khái quát hoá thành một lớp khác) và gói (cùng gom vào một gói). Các mối liên quan đó được trình bày trong biểu đồ lớp cùng với cấu trúc bên trong của mỗi lớp, gồm các thuộc tính và các thao tác. Biểu đồ lớp được xem là tĩnh vì cấu trúc mà nó mô tả luôn luôn đúng vào mọi lúc trong chu trình sống của hệ thống. Thông thường thì hệ thống được mô tả bởi một số biểu đồ lớp (chứ không phải là gom tất mọi lớp vào cùng một biểu đồ), và một lớp có thể tham gia vào nhiều biểu đồ.



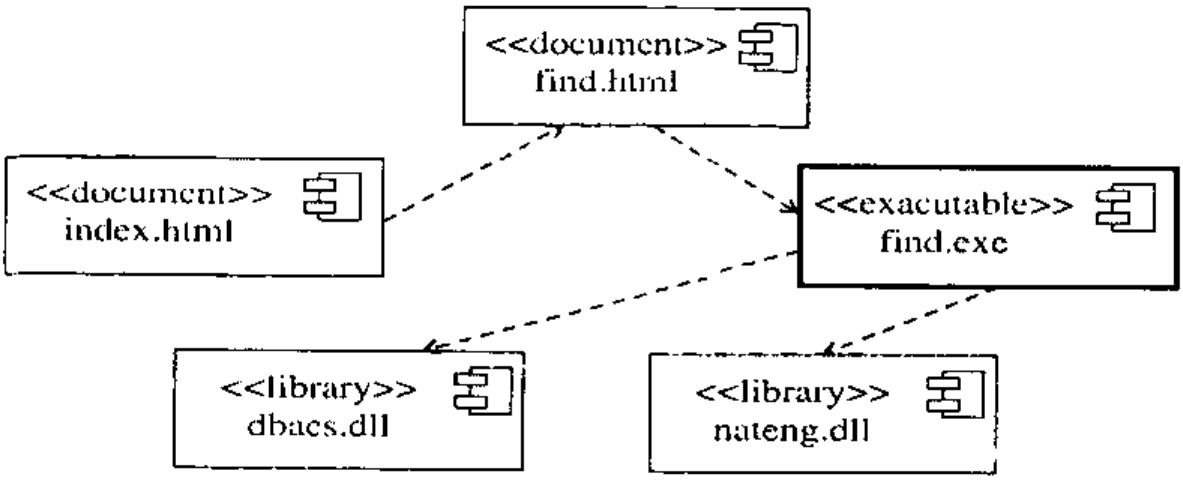
Hình II.6. Một biểu đồ lớp

(2) *Biểu đồ đối tượng:* Biểu đồ đối tượng phô bày các đối tượng thay cho các lớp. Có thể nói đó là một "ảnh chớp" của hệ thống tại một thời điểm, cho thấy có các đối tượng nào đang tồn tại và hoạt động. Nó dùng các ký pháp như biểu đồ lớp (xem Hình II.7), ngoại trừ hai điểm: tên đối tượng được gạch dưới và mọi kết nối cụ thể trong liên kết đều được vẽ ra. Biểu đồ đối tượng ít được dùng, ngoại trừ khi nó được dùng làm nền cho biểu đồ giao tiếp, trên đó được vẽ thêm các thông điệp chuyển giao giữa các đối tượng.



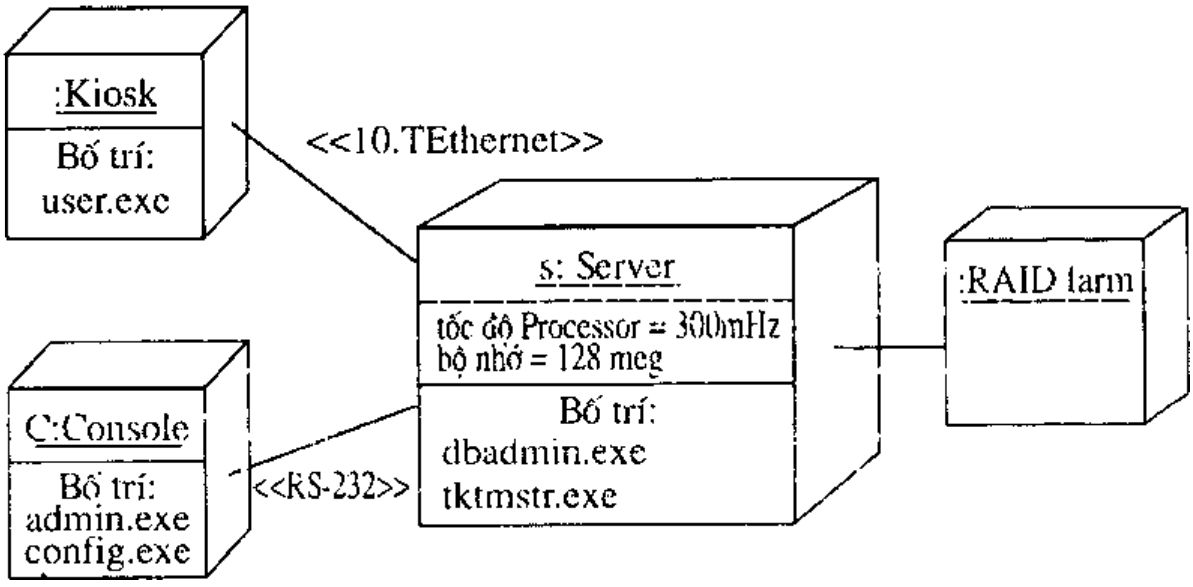
Hình II.7. Biểu đồ đối tượng tương ứng với biểu đồ lớp ở Hình II.6

(3) *Biểu đồ thành phần:* Biểu đồ thành phần trình bày cấu trúc vật lý của chương trình dưới dạng các thành phần, cùng với các mối liên quan phụ thuộc giữa chúng (xem Hình II.8). Một thành phần có thể là thành phần mã nguồn, thành phần mã nhị phân, hoặc thành phần exe. Mỗi thành phần trong biểu đồ thành phần tương ứng với một hay nhiều lớp, giao diện hay hợp tác trong biểu đồ lớp. Biểu đồ thành phần được dùng để trình bày góc nhìn thực thi của hệ thống.



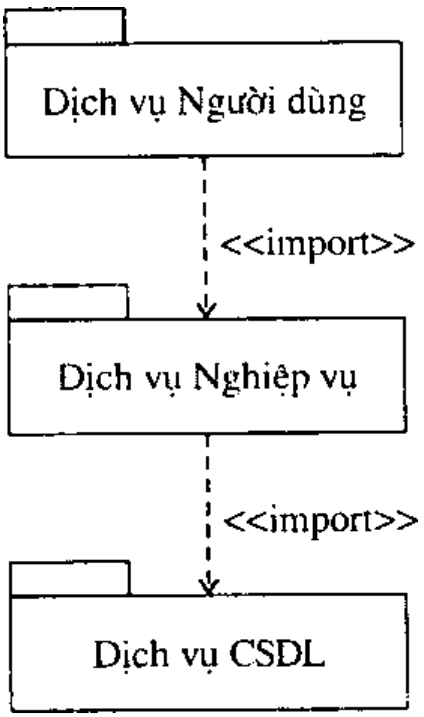
Hình II. 8. Một biểu đồ thành phần

(4) *Biểu đồ bố trí:* Biểu đồ bố trí trình bày kiến trúc vật lý của phần cứng và phần mềm của hệ thống. Nó cho thấy các máy tính và thiết bị (các nút), cùng với các kết nối giữa chúng. Các thành phần thực hiện được (exe) được phân bổ vào các nút, cho thấy các đơn vị phần mềm được thực hiện trên các nút nào (xem Hình II.9).



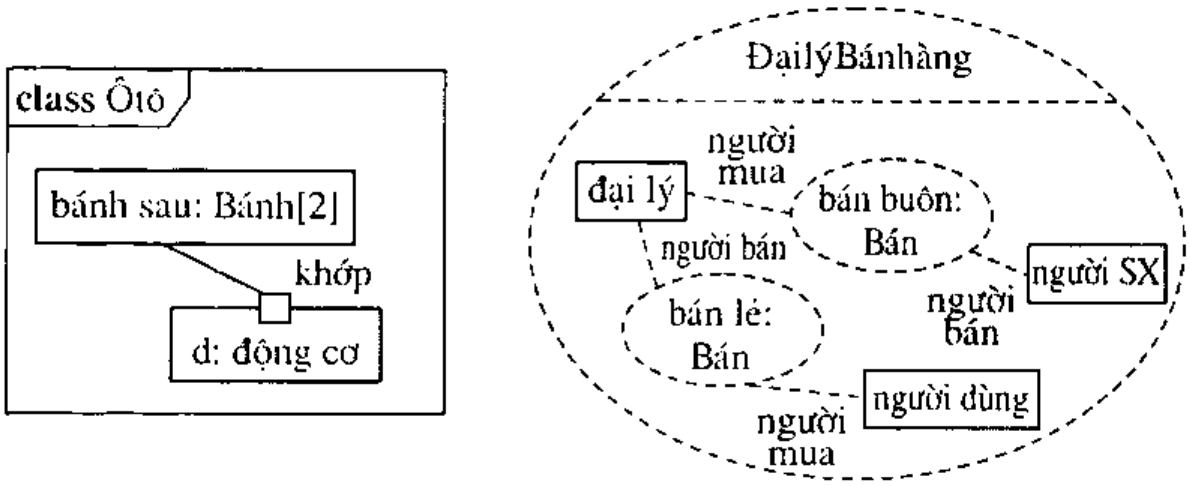
Hình II.9. Một biểu đồ bố trí

(5) *Biểu đồ gói:* Gói là một hình thức gom nhóm các phần tử. Phần tử nói đây có thể là các lớp, các ca sử dụng, các thành phần v.v... Giữa các gói (biểu diễn bằng một hình chữ nhật có quai) có thể có các mối liên quan phụ thuộc, tạo nên một biểu đồ, gọi là biểu đồ gói (Hình II.10).



Hình II.10. Một biểu đồ gói

(6) *Biểu đồ cấu trúc đa hợp:* Là biểu đồ diễn tả cấu trúc bên trong của một loài (classifier), như lớp, gói, giao diện, hợp tác v.v..., chỉ ra các điểm tương tác của loài đó với các thành phần còn lại của hệ thống, cũng như chỉ ra vai trò của các bộ phận tham gia thực hiện hành vi chung của loài đó.

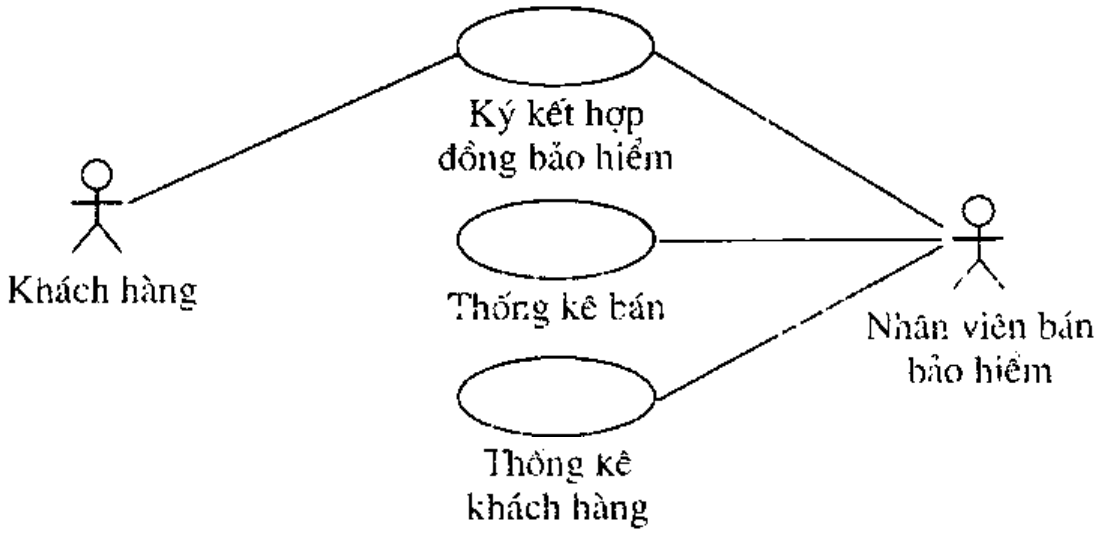


Hình II.11. Hai biểu đồ cấu trúc đa hợp

b) Nhóm các biểu đồ hành vi

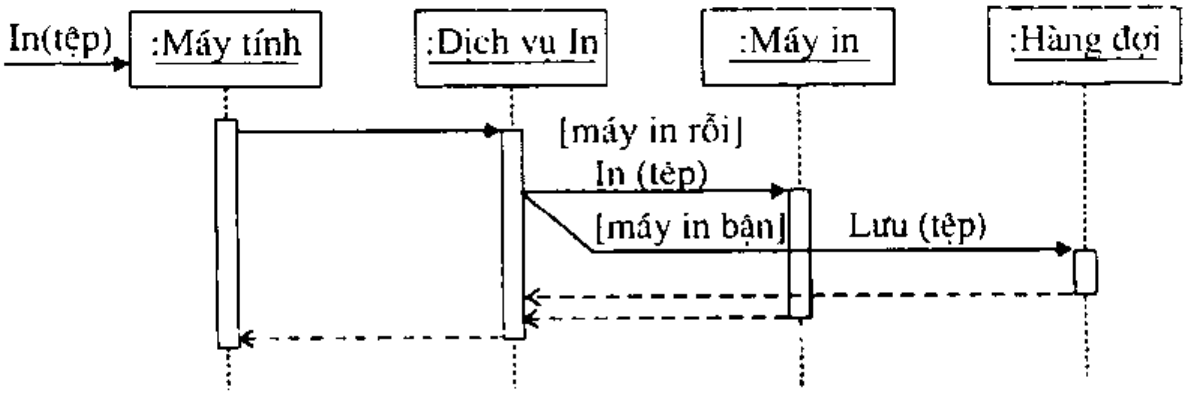
Đây là các biểu đồ dùng để phản ánh sắc thái động của hệ thống. Có bảy loại biểu đồ thuộc nhóm này.

(7) *Biểu đồ ca sử dụng:* Biểu đồ ca sử dụng trình bày một số đối tác (tác nhân ngoài) và sự liên hệ của chúng với các ca sử dụng mà hệ thống cung cấp (xem Hình II.12). Một ca sử dụng là một diễn tả của một chức năng mà hệ thống có khả năng cung cấp. Ca sử dụng có thể được đặc tả (ngoài biểu đồ) bằng một văn bản hay bằng một biểu đồ hoạt động, ở đó chỉ để cập hệ thống làm gì, chứ không nói hệ thống làm như thế nào.



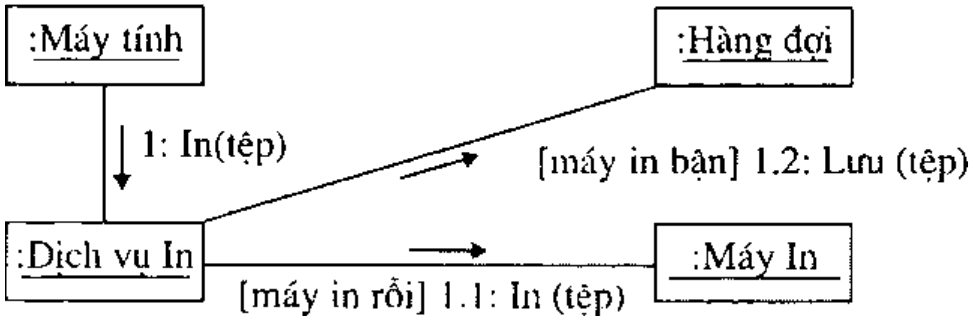
Hình II.12. Một biểu đồ ca sử dụng

(8) *Biểu đồ trình tự:* Biểu đồ trình tự trình bày một số đối tượng với các thông điệp được chuyển giao giữa chúng, đặc biệt làm rõ trình tự các thông điệp chuyển giao này dọc theo trục thời gian (trục thẳng đứng). Biểu đồ trình tự dùng để diễn tả một sự hợp tác (hay tương tác) của một nhóm đối tượng (xem Hình II.13)



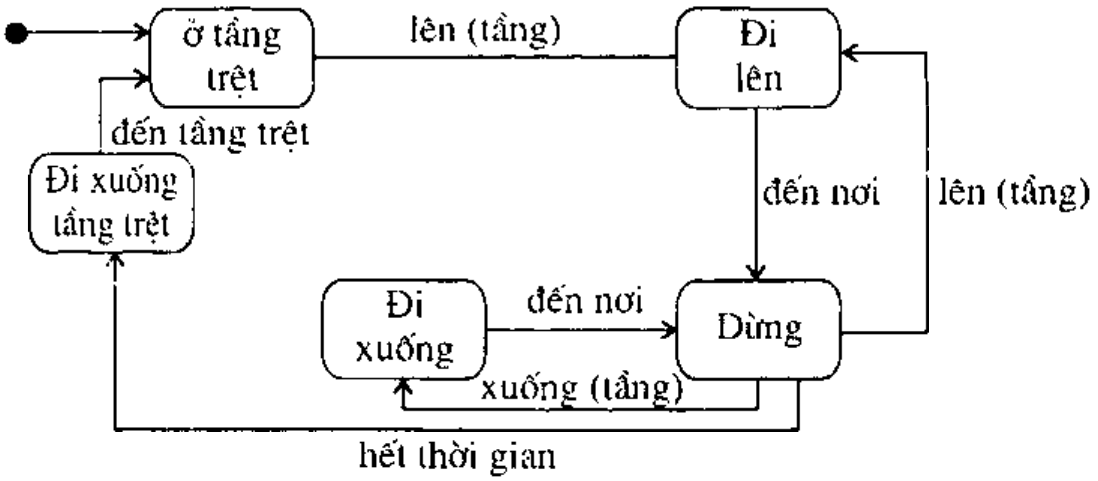
Hình II.13. Một biểu đồ trình tự

(9) *Biểu đồ giao tiếp:* Biểu đồ giao tiếp cũng trình bày một sự hợp tác (tương tác) của một nhóm đối tượng như là biểu đồ trình tự song biểu đồ giao tiếp lại nhấn mạnh tới bối cảnh của sự hợp tác. Nó cũng cho một nhóm các đối tượng cùng với các kết nối giữa chúng, như trong biểu đồ đối tượng, nhưng vẽ thêm các thông điệp (mũi tên nhỏ) dọc theo các kết nối đó. Các thông điệp được đánh số để phân biệt trước sau (xem Hình II.14).



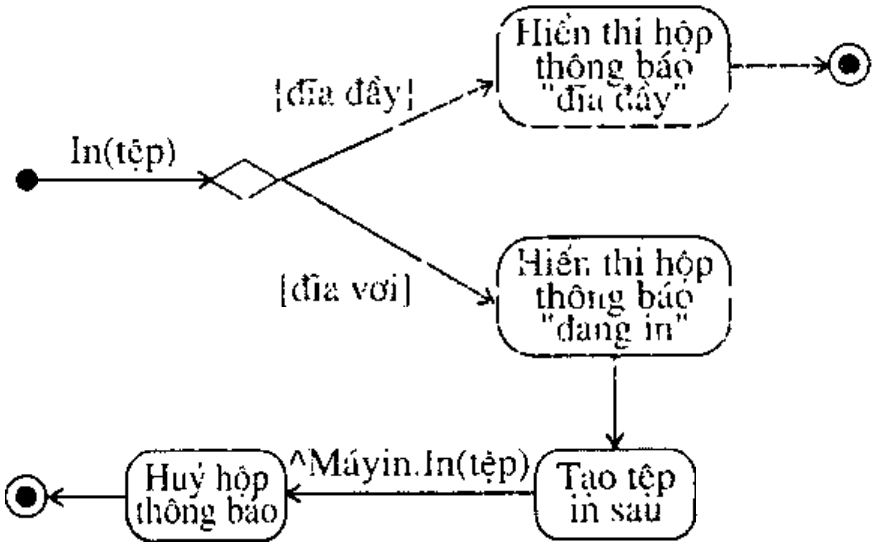
Hình II.14. Một biểu đồ giao tiếp

(10) *Biểu đồ máy trạng thái:* Có những đối tượng có những cách phản ứng linh hoạt trước những sự kiện từ ngoài tới. Hành vi của loại đối tượng này được miêu tả bởi một biểu đồ máy trạng thái. Biểu đồ máy trạng thái trình bày các trạng thái có thể của đối tượng và chỉ rõ các sự kiện nào sẽ làm cho đối tượng thay đổi từ trạng thái này sang trạng thái kia (xem Hình II.15). Một sự kiện có thể là một tín hiệu đặc biệt, một tin báo vừa hết một thời hạn hay một điều kiện nào đó vừa được thoả mãn, mà đối tượng tiếp nhận qua thông điệp gửi tới từ một đối tượng khác. Một sự thay đổi trạng thái gọi là một dịch chuyển. Có thể có các hành động xảy ra gắn với trạng thái hay với bước dịch chuyển. Biểu đồ máy trạng thái không phải được vẽ cho tất cả các lớp, mà chỉ riêng cho các lớp mà đối tượng của nó có khả năng ứng xử trước các sự kiện xảy đến tuỳ thuộc vào trạng thái hiện tại của nó.



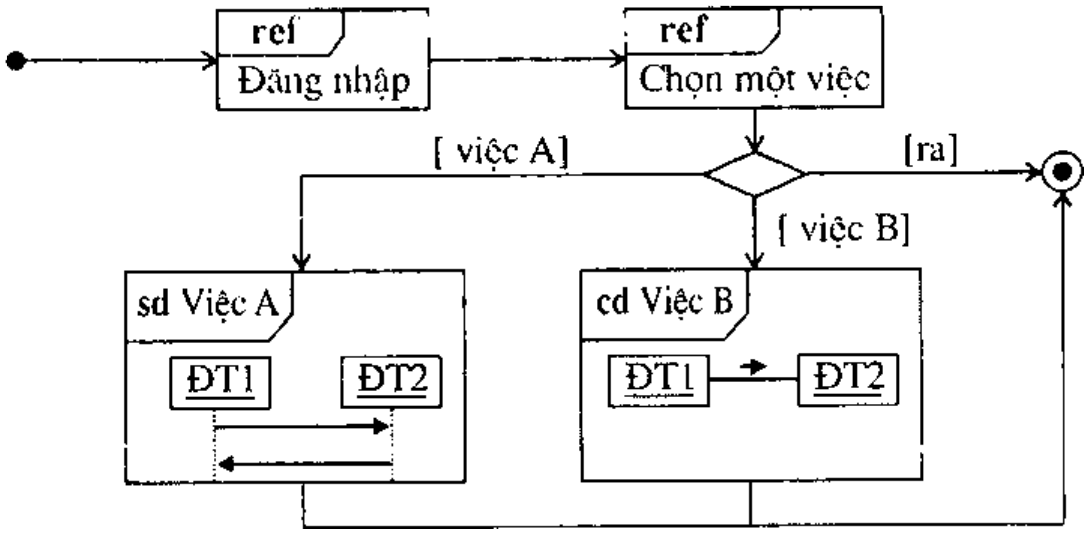
Hình II.15. Một biểu đồ máy trạng thái

(11) *Biểu đồ hoạt động:* Biểu đồ hoạt động trình bày luồng dịch chuyển từ hành động này sang hành động khác, bao gồm sự dịch chuyển tuần tự, rẽ nhánh theo điều kiện hay rẽ nhánh song song (xem Hình II.16). Thông thường thì biểu đồ hoạt động được dùng để đặc tả một thao tác của một lớp, song nó cũng có thể được dùng để mô tả các luồng hoạt động khác, như là ca sử dụng hay tương tác.



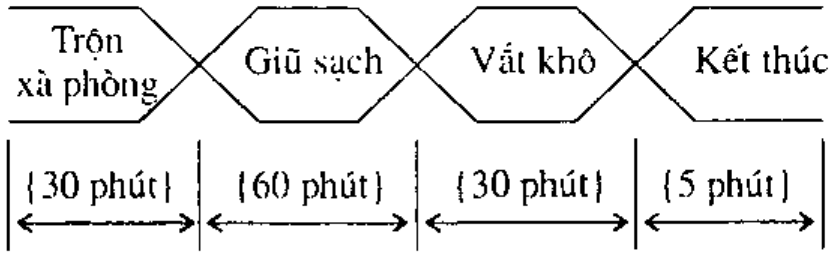
Hình II.16. Một biểu đồ hoạt động

(12) *Biểu đồ bao quát tương tác:* Đây là một biến thể của biểu đồ hoạt động, mà trong đó các nút không những là các hành động, mà còn có thể là các biểu đồ tương tác. Nó cho ta một cái nhìn bao quát đối với một tương tác phức tạp (Hình II.17).



Hình II.17. Một biểu đồ bao quát tương tác

(13) *Biểu đồ thời khắc:* Là biểu đồ diễn tả các giai đoạn trải qua trong thời gian của một (hay nhiều) đối tượng (Hình II.18).



Hình II.18. Một biểu đồ thời khắc

4. MỞ RỘNG Ý NGHĨA CHO CÁC YẾU TỐ MÔ HÌNH

Các yếu tố của mô hình nhiều khi không diễn tả được sát nghĩa mà ta mong muốn, bấy giờ UML cho nhiều cách để gia tăng hay hạn chế ý nghĩa của các yếu tố mô hình cho sát hợp với hoàn cảnh ứng dụng.

a) Đặc tả (specification)

Bên cạnh các ký pháp đồ hoạ, UML cho phép kèm thêm một đặc tả dưới dạng một phát biểu văn tự về cú pháp và ngữ nghĩa của yếu tố mô hình. Chẳng hạn sau biểu tượng khá sơ lược về một lớp ta có thể đưa thêm một đặc tả trong đó cho toàn bộ các thuộc tính, các thao tác, với các chi tiết đầy đủ của lớp đó.

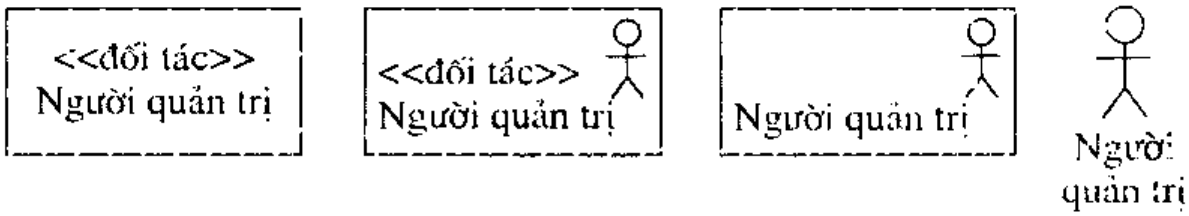
b) Tô điểm (adornment)

Một yếu tố của mô hình có thể được tô điểm thêm để gia tăng một số khía cạnh biểu diễn, ngoài ý nghĩa cơ bản. Chẳng hạn một liên kết vốn được diễn tả bằng một đoạn thẳng nối hai lớp, có thể tô điểm thêm bởi các vai trò, cơ số, hạn định v.v...; một thành phần có thể được vẽ với đường viền đậm nét để chỉ đó là thành phần thi hành được (exe); tên lớp được viết xiên để chỉ đó là một lớp trừu tượng.

c) Khuôn dập (stereotype)

Đó là một xâu (đặt trong các ngoặc kép) hay một biểu tượng gắn thêm cho một yếu tố mô hình đã được định nghĩa sẵn để tạo ra một yếu tố mô hình mới. Khuôn dập làm tăng thêm (thậm chí có thể đè

lấp) ý nghĩa của phần tử mô hình cũ, song không làm thay đổi cấu trúc. Chẳng hạn, đối tác là một khuôn dập đối với lớp. Có thể có bốn cách biểu diễn khác nhau của đối tác: như một lớp có thêm khuôn dập (văn tự hay biểu tượng) hoặc thay bằng biểu tượng mới hoàn toàn (Hình II.19).



Hình II.19. Sử dụng khuôn dập

UML có đưa ra nhiều khuôn dập định nghĩa sẵn, song người dùng cũng có thể đưa thêm khuôn dập của mình.

d) Tính chất (property) và Giá trị gắn nhãn (tagged - value)

Một tính chất, được hiểu theo nghĩa chung, là một giá trị gắn cho một phần tử. Các tính chất được dùng để đưa thêm thông tin cho các phần tử mô hình.

Một giá trị gắn nhãn là một định nghĩa tường minh cho một tính chất, cho dưới dạng một cặp tên - giá trị, trong đó tên còn được gọi là nhãn.

Ký pháp để diễn tả các tính chất là như sau:

{nhãn = giá trị} hoặc {nhãn 1 = giá trị 1, nhãn 2 = giá trị 2} hoặc {nhãn}. Khi nhãn là một nhãn Bun, thì giá trị ngầm định là true cho trường hợp khuyết giá trị. Lấy thí dụ:

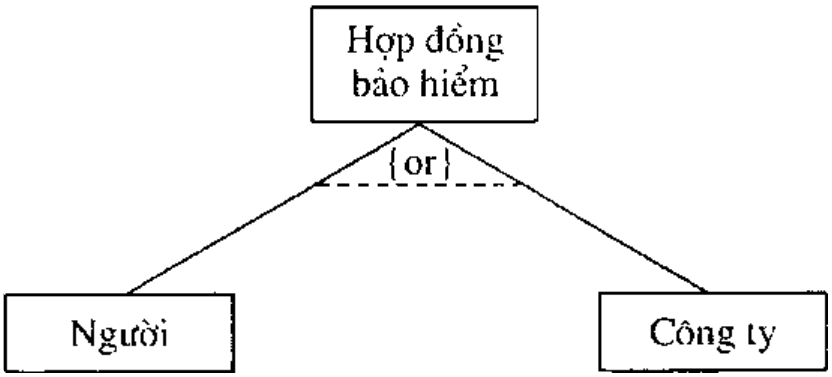
{trừu tượng}

{tình trạng = 'đang xây dựng', người phân tích = 'Liên'}

UML cũng có đưa ra nhiều tính chất và giá trị gắn nhãn định nghĩa sẵn, song người dùng có thể đưa thêm các tính chất và giá trị gắn nhãn của mình.

đ) Ràng buộc (constraint)

Đó là một điều kiện hay một hạn chế phải được tuân thủ đối với một yếu tố mô hình. Ràng buộc được viết giữa một cặp ngoặc ôm, ví dụ {tuổi > 60}, viết bên cạnh yếu tố mô hình. Nếu ràng buộc tác dụng đồng thời lên nhiều yếu tố mô hình thì nó được đặt bên cạnh một đường đứt nét đi ngang qua các yếu tố mô hình đó (xem Hình II.20).



Hình II.20. Ràng buộc {or} tác dụng lên hai yếu tố mô hình (là các liên kết)

UML cũng đưa ra một số các ràng buộc chuẩn, song người dùng cũng có thể đưa ra các ràng buộc của mình.

5. MÔ HÌNH HOÁ VỚI UML

Để mô hình hoá một hệ thống, không phải chỉ dùng một mô hình là đủ mà phải dùng nhiều mô hình (biểu đồ), để diễn tả hệ thống:

- theo nhiều góc nhìn khác nhau, và
- theo các mức độ trừu tượng hoá khác nhau.

a) Mô hình hoá hệ thống theo nhiều góc nhìn

Như trên ta đã thấy, UML đề xuất năm góc nhìn đối với hệ thống và mười ba biểu đồ, trong đó mỗi biểu đồ chỉ dùng để diễn tả hệ thống trong một (hay vài) góc nhìn nào đó mà thôi.

Tuỳ theo đặc điểm của hệ thống mà người phát triển hệ thống sẽ quyết định là cần mô tả nó dưới các góc nhìn nào, và vận dụng các biểu đồ nào. Nếu hệ thống là nhỏ gọn, được cài đặt trên một máy tính duy nhất, thì mô tả nó trên hai góc nhìn là góc nhìn Ca sử dụng (với

biểu đồ ca sử dụng) và góc nhìn Thiết kế (với biểu đồ lớp và biểu đồ tương tác) là đủ. Nếu hệ thống là một hệ phản ứng theo sự kiện thì lại phải chú ý thêm góc nhìn Quá trình, với các biểu đồ máy trạng thái và hoạt động. Nếu hệ thống là một hệ khách hàng/dịch vụ, thì phải đề cập góc nhìn Thực thi (với biểu đồ thành phần) và góc nhìn Bố trí (với biểu đồ bố trí) để mô tả phương diện vật lý của hệ thống. Còn nếu là một hệ phân tán và phức tạp, thì phải dùng toàn bộ các biểu đồ, trên đủ năm góc nhìn.

b) Mô hình hoá hệ thống theo nhiều mức độ trừu tượng hoá

Khi mô hình hoá hệ thống, tuỳ theo giai đoạn và nhu cầu sử dụng mà các biểu đồ được vẽ ở các mức độ trừu tượng hoá cao hay thấp (khái lược hay chi tiết). Vậy cùng một biểu đồ, có thể khi thì được trình bày khái lược (giấu đi nhiều chi tiết), khi thì được vẽ tỉ mỉ (với đầy đủ các chi tiết). Chẳng hạn một biểu đồ lớp dùng cho người phân tích để trao đổi với người dùng, thì chỉ cần vẽ đơn sơ. Nhưng biểu đồ lớp dùng cho người lập trình thì lại phải đầy đủ các chi tiết, với đầy đủ các thuộc tính, các thao tác, các mối liên quan giữa các lớp v.v...

Cần nhớ rằng mỗi mô hình được lập ra phải có một mục đích rõ rệt. Chúng phải là các chặng đường cần thiết để hình thành nên hệ thống, từ những ý tưởng khái lược ban đầu (có khi còn rất mơ hồ), cho tới khi có được một hệ thống chạy thực sự trên các thiết bị phần cứng, đáp ứng được các nhu cầu đặt ra cho nó.

§3. TIẾN TRÌNH RUP

Tiến trình RUP (Rational Unified Process) là một tiến trình mô hình hoá với UML, do "ba người bạn" đưa ra, song nó không phải là chuẩn.

1. CÁC NGUYÊN TẮC CƠ BẢN CỦA RUP

Tiến trình RUP là một tiến trình phát triển phần mềm dựa trên các nguyên tắc: "lặp và tăng trưởng, tập trung vào kiến trúc, dẫn dắt theo các ca sử dụng và khống chế bởi các nguy cơ".

a) Lặp và tăng trưởng

Dự án được cắt thành những vòng lặp hoặc giai đoạn ngắn (khoảng một tháng) cho phép kiểm soát dễ dàng sự tiến triển của dự án. Cuối mỗi vòng lặp thì một phần thi hành được của hệ thống được sản sinh theo cách tăng trưởng (thêm vào) dần dần.

b) Tập trung vào kiến trúc

Toàn bộ hệ thống phức tạp phải được phân chia thành từng phần (các môđun) để có thể dễ dàng triển khai và duy tu, tạo nên một kiến trúc. Kiến trúc này phải được trình bày theo năm góc nhìn khác nhau và bởi các biểu đồ của UML (mà không phải chỉ theo văn bản).

c) Dẫn dắt theo các ca sử dụng

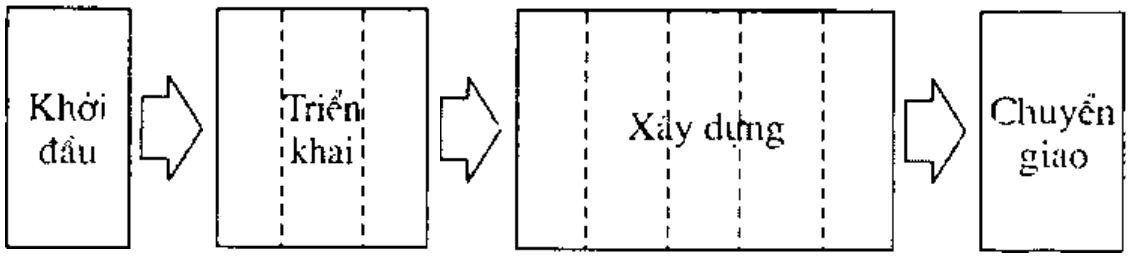
RUP nhấn mạnh sự đáp ứng các nhu cầu của người dùng, thể hiện bởi các ca sử dụng. Do đó các ca sử dụng ảnh hưởng và dẫn đường cho mọi giai đoạn phát triển hệ thống. Nắm bắt nhu cầu là để phát hiện các ca sử dụng. Phân tích là đi sâu vào các ca sử dụng. Thiết kế và cài đặt là để xây dựng hệ thống theo từng ca sử dụng. Kiểm định và nghiệm thu hệ thống thực hiện theo các ca sử dụng. Ca sử dụng là căn cứ để xác định các vòng lặp và tăng trưởng và cũng là căn cứ để phân công công việc trong nhóm phát triển hệ thống.

d) Khống chế bởi các nguy cơ

Các nguy cơ chính đối với dự án phải phát hiện sớm và phải loại bỏ càng sớm càng tốt. Yêu cầu này cũng là căn cứ để xác định thứ tự trước sau của các vòng lặp.

2. CÁC PHA VÀ CÔNG ĐOẠN CỦA TIẾN TRÌNH RUP

Tiến trình RUP được tổ chức thành bốn pha (phase) nối tiếp trong thời gian, là: khởi đầu, triển khai, xây dựng và chuyển giao (Hình II.21).



Hình II.21. Các pha của RUP

a) Pha khởi đầu

Pha khởi đầu (inception) nhằm cho một cái nhìn tổng quát về hệ thống sẽ xây dựng (chức năng, hiệu năng, công nghệ v.v...) và về dự án sẽ triển khai (phạm vi, mục tiêu, tính khả thi v.v...) Từ đó đưa ra kết luận là nên phát triển hay nên loại bỏ dự án.

b) Pha triển khai

Pha triển khai (elaboration) bao gồm một sự phân tích chi tiết hơn về hệ thống, cả về chức năng và cấu trúc tĩnh (dùng các biểu đồ ca sử dụng, biểu đồ lớp, các biểu đồ tương tác). Đồng thời một kiến trúc hệ thống cũng được đề xuất. Kiến trúc này có thể dựng thành nguyên mẫu (prototype), trên đó có thể thử nghiệm nhiều ý đồ đối với hệ thống.

c) Pha xây dựng

Pha xây dựng (construction) tập trung vào việc thiết kế và thực thi (cài đặt) hệ thống. Nếu pha triển khai chỉ mới cho một kiến trúc hệ thống cơ bản, thì ở pha xây dựng kiến trúc đó được tinh chế và chi tiết hoá (thậm chí có thể chỉnh sửa, hay thay đổi). Pha xây dựng kết thúc khi đã phát hành được (ít nhất là trong nội bộ) một hệ thống hoàn chỉnh cùng với các tư liệu kèm theo. Pha xây dựng là pha kéo dài về thời gian và tốn hao sức lực hơn cả.

d) Pha chuyển giao

Pha chuyển giao (transition) nhằm chuyển hệ thống đã xây dựng từ tay những người phát triển hệ thống tới tay các người dùng cuối, bao gồm các công việc như chuyển đổi các dữ liệu, đào tạo người dùng, lắp đặt, kiểm định bêta.

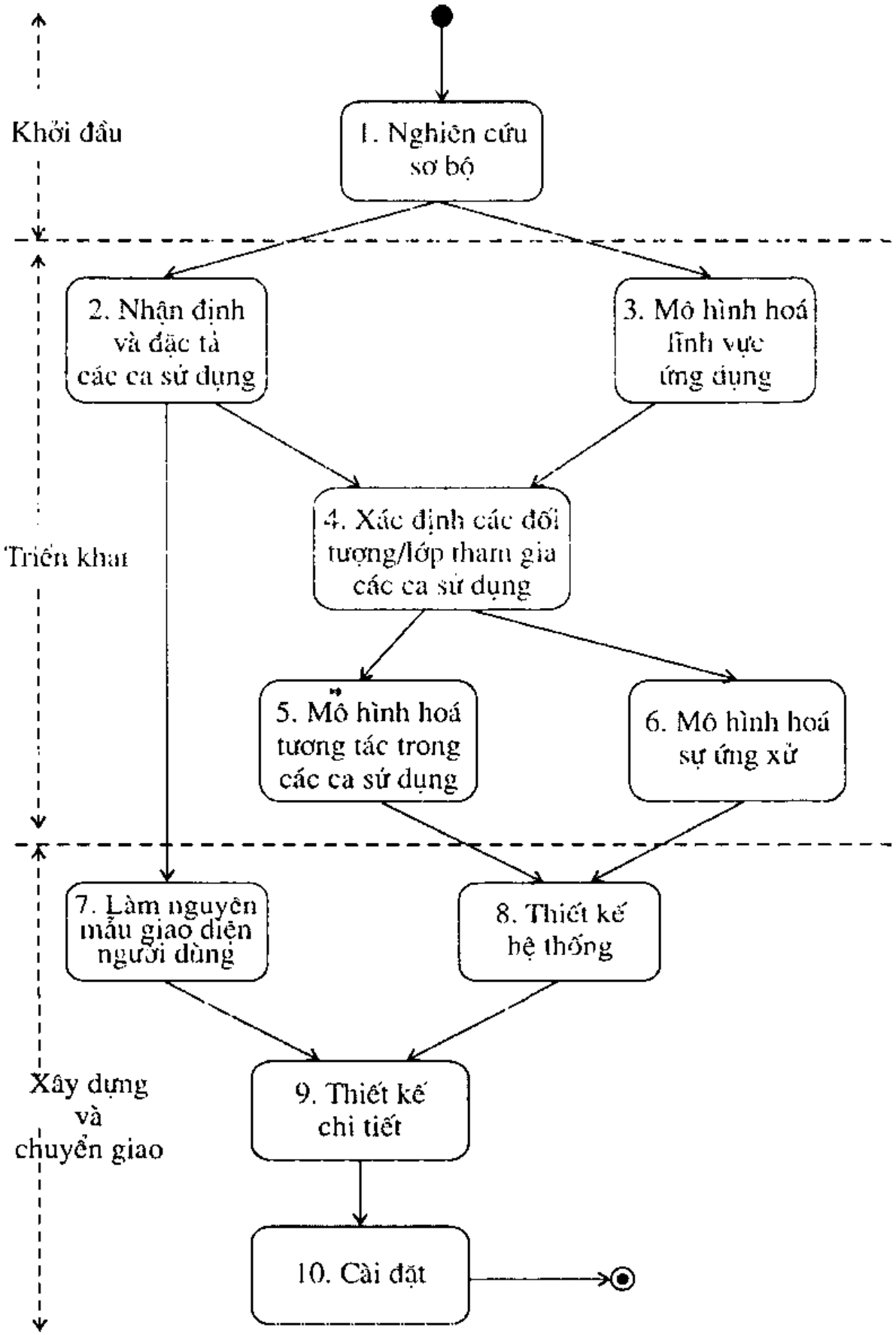
Mỗi pha nói trên, đặc biệt là pha xây dựng và có thể cả pha triển khai, lại được chia thành một số vòng lặp (kéo dài độ 2 đến 4 tuần). Mỗi vòng lặp sẽ hoàn thành một phần của hệ thống và trải qua năm *công đoạn* (workflow) sau: nắm bắt yêu cầu, phân tích và thiết kế, thực thi, kiểm định và bố trí. Đương nhiên liều lượng cho mỗi công đoạn đó trong mỗi vòng lặp là tuỳ thuộc vòng lặp đó ở pha nào: Ở các pha đầu thì các công đoạn đầu (nắm bắt yêu cầu, phân tích thiết kế) được nhấn mạnh, còn ở các pha cuối thì các công đoạn cuối (thực thi, kiểm định, bố trí) lại được nhấn mạnh.

3. MỘT TIẾN TRÌNH ĐƠN GIẢN

Tiến trình RUP như đã giới thiệu ở trên, với các nguyên tắc cơ bản, và với sự phân chia thành các pha, các vòng lặp, các công đoạn, thì vẫn chưa phải là một tiến trình cụ thể, mà chỉ mới là một khuôn khổ hay một họ các tiến trình. Để có một tiến trình, ta còn phải chỉ rõ các bước đi, công việc phải làm trong mỗi bước, sản phẩm phải hoàn thành sau mỗi bước, để có thể dẫn dắt một cách chắc chắn quá trình phát triển một phần mềm từ nhu cầu đặt ra cho tới mã chương trình chạy được và đáp ứng được các nhu cầu một cách đầy đủ và với hiệu năng mong muốn.

Sau đây là một tiến trình đơn giản, tạm gọi là "tiến trình 10 bước" (xem Hình II.22), được vận dụng trong cuốn sách này, với các đặc điểm sau:

- Nó thực hiện mô hình hoá với các biểu đồ UML;
- Nó tuân thủ các nguyên tắc của RUP, song đã cố gắng đơn giản hoá, mặc dù vẫn không bỏ qua những hoạt động cơ bản về phân tích và thiết kế;
- Nó chú trọng nguyên tắc dẫn dắt bởi các ca sử dụng, song cố gắng thể hiện một cách nhẹ nhàng và tự nhiên;
- Nó vẫn có thể được triển khai theo nguyên tắc lặp và tăng trưởng (trong từng bước) và nguyên tắc khống chế bởi các nguy cơ.



Hình II.22. Tiến trình mười bước

Sau đây là nội dung tóm tắt các bước trong tiến trình 10 bước:

(1) *Nghiên cứu sơ bộ:* Nhằm đưa ra một "cái nhìn" khái quát về hệ thống sẽ xây dựng (chức năng, hiệu năng, công nghệ...) và về dự án sẽ triển khai (phạm vi, mục tiêu, tính khả thi, ...). Từ đó đưa ra kết luận nên triển khai tiếp hay nên chấm dứt dự án. Như vậy đây chính là pha khởi đầu của RUP.

(2) *Nhận định và đặc tả các ca sử dụng:* Từ việc nắm bắt các nhu cầu của người dùng mà phát hiện các ca sử dụng. Ca sử dụng là một tập hợp của những dãy hành động mà hệ thống thực hiện để đưa ra một kết quả có ích cho một đối tác của hệ thống. Mỗi ca sử dụng phải được đặc tả dưới dạng văn tự (kịch bản) và/hoặc dưới dạng một biểu đồ trình tự hệ thống.

(3) Mô hình hoá lĩnh vực ứng dụng: Đưa ra một mô hình (dưới dạng một biểu đồ lớp) nhằm phản ánh mọi khái niệm nghiệp vụ (thực thể và liên kết) mà người dùng cũng như người xây dựng hệ thống, khi đề cập tới hệ thống và ứng dụng, đều phải sử dụng đến. Các lớp xuất hiện ở đây đều là các lớp "lĩnh vực" (domain classes), nghĩa là các lớp thuộc lĩnh vực nghiệp vụ của ứng dụng, mà chưa có các lớp phù trợ khác.

(4) Xác định các đối tượng /lớp tham gia các ca sử dụng: Đối với mỗi ca sử dụng, phải phát hiện các lớp lĩnh vực, cùng với các lớp điều khiển và các lớp biên (giao diện) tham gia thực hiện ca sử dụng đó. Như vậy ta lập một biểu đồ lớp (hay biểu đồ đối tượng) làm nền cho mỗi ca sử dụng. Chính trên nền đó mà ta nghiên cứu sự tương tác ở bước sau.

(5) Mô hình hoá tương tác trong các ca sử dụng: Sự tương tác duy nhất có thể có giữa các đối tượng là trao đổi thông điệp. Cần phải nghiên cứu sự tương tác giữa các đối tượng tham gia mỗi ca sử dụng, mà kết quả phải là tạo nên kịch bản của ca sử dụng đó. Sự tương tác được trình bày dưới dạng biểu đồ trình tự hay biểu đồ giao tiếp.

(6) Mô hình hoá ứng xử: Các đối tượng điều khiển khác với các đối tượng thực thể ở chỗ có khả năng ứng xử trước các sự kiện từ ngoài đến để đưa ra các quyết định điều khiển thích hợp. Việc mô tả hành vi ứng xử của các đối tượng điều khiển được thực hiện bởi các biểu đồ máy trạng thái.

(7) Làm nguyên mẫu giao diện người dùng: Với các bộ tạo lập GUI, ta có thể thành lập sớm và nhanh một nguyên mẫu giao diện người dùng, giúp cho việc mô hình hoá và cài đặt hệ thống triển khai dễ dàng hơn.

(8) Thiết kế hệ thống: Đó là sự thiết kế kiến trúc tổng thể của hệ thống, bao gồm việc vỡ hệ thống thành các hệ thống con, chọn lựa loại hình điều khiển thích hợp, miêu tả các thành phần vật lý của hệ thống (dùng biểu đồ thành phần) và bố trí các thành phần khả thi vào các phần cứng (dùng biểu đồ bố trí). Một kiến trúc khách hàng/dịch vụ (client/server) nhiều tầng thường được chọn lựa ở đây.

(9) Thiết kế chi tiết: Đó là bước thiết kế về các lớp, các liên kết, các thuộc tính, các thao tác, thực hiện trên từng tầng của kiến trúc khách hàng/dịch vụ (tầng trình bày, tầng ứng dụng, tầng nghiệp vụ, tầng lưu trữ dữ liệu), và xác định các giải pháp cài đặt trên mạng.

(10) Cài đặt: Đó là bước thực thi hệ thống, bao gồm lập trình và kiểm định. Hệ thống được nghiệm thu dựa trên các ca sử dụng.

Cuốn sách này trình bày các yếu tố của ngôn ngữ mô hình hoá UML, đồng thời trình bày cách triển khai các bước của tiến trình mười bước nói trên:

- Các chương I và II trình bày các khái niệm mở đầu về lập trình hướng đối tượng và mô hình hoá hướng đối tượng.
- Chương III trình bày các bước 1 và 2 cùng với biểu đồ các ca sử dụng.
- Chương IV trình bày biểu đồ lớp, biểu đồ đối tượng, biểu đồ gói, biểu đồ cấu trúc đa hợp cùng với các bước 3 và 4.
- Chương V trình bày các biểu đồ tương tác (biểu đồ trình tự, biểu đồ giao tiếp), cùng với các bước 5 và 6. Mặt khác chương này cũng giới thiệu thêm các biểu đồ hoạt động, biểu đồ bao quát tương tác và biểu đồ thời khắc.
- Chương VI trình bày các bước 7, 8 và 9 cùng với các biểu đồ thành phần và biểu đồ bố trí.
- Các chương VII, VIII, IX trình bày cách cài đặt hệ thống (bước 10) trong ngôn ngữ lập trình C++.

§4. CÁC CÔNG CỤ TRỢ GIÚP

1. TÍNH NĂNG CỦA CÁC CÔNG CỤ TRỢ GIÚP

Các công cụ trợ giúp là các phần mềm hỗ trợ cho quá trình phát triển hệ thống. Không có công cụ hỗ trợ xuyên suốt mọi giai đoạn và mọi công việc phát triển hệ thống, mà thường mỗi công cụ hỗ trợ cho một phần của quá trình đó mà thôi. Đó có thể là:

- Công cụ hỗ trợ cho ngôn ngữ lập trình, như là:
  - Soạn thảo và biên dịch chương trình;
  - Gỡ rối và kiểm định chương trình;
  - Xây dựng và làm nguyên mẫu các giao diện người dùng (GUI), và gắn kết chúng vào hệ thống.
- Công cụ hỗ trợ ngôn ngữ mô hình hoá, như là:
  - Sản sinh, biến đổi và điều chỉnh nhanh các mô hình và biểu đồ;
  - Kiểm tra cú pháp, sự chặt chẽ và sự đầy đủ của các mô hình;
  - Lưu giữ và cho phép xét duyệt các mô hình và các phiên bản theo các góc nhìn khác nhau, tạo các báo cáo và tư liệu;
  - Kiểm thử và đánh giá các mô hình;
  - Mô phỏng và thực hiện mô hình (sản sinh các sườn chương trình từ các mô hình);
  - Thiết lập trở lại các mô hình từ các phần mềm có sẵn (công nghệ ngược).
- Công cụ hỗ trợ tiến trình phát triển hệ thống, như là:
  - Dẫn dắt và hỗ trợ trực tuyến cho việc thực hiện các giai đoạn, chỉ rõ các công việc phải làm, sản phẩm tạo ra;
  - Hỗ trợ tiến trình lặp;
  - Hỗ trợ sự làm việc theo nhóm;
  - Tích hợp được với các công cụ khác;
  - Trợ giúp việc quản lý dự án, giúp người phụ trách dự án lên kế hoạch và theo dõi quá trình phát triển của dự án.

2. CÔNG CỤ TRỢ GIÚP RATIONAL ROSE

Đây là phần mềm trợ giúp được sử dụng khá phổ biến cho quá trình mô hình hoá với UML và RUP, do công ty Rational của "ba người bạn" lập nên (nay đã nhập cùng IBM), các tính năng chính của nó là:

- Tạo một môi trường pnát triển hệ thống thuận tiện từ phân tích, thiết kế, đến sản sinh mã, nhằm đưa ra các giải pháp hợp lý và hiệu quả cho các yêu cầu nghiệp vụ đối với các hệ thống khách/chủ và phân tán, cũng như các hệ thống thời gian thực.
- Hỗ trợ việc tạo lập, đặc tả và kiểm chứng các mô hình UML, trên các góc nhìn ca sử dụng, lôgic, thành phần và bố trí.
- Từ mô hình sản sinh chương trình khung trên Visual Basic, C++ hay Java.
- Dùng công nghệ vòng quanh để làm cho chương trình đồng bộ với thiết kế.
- Dùng công nghệ ngược để lập lại mô hình từ các thành phần và ứng dụng có sẵn.
- Hỗ trợ việc phát triển hệ thống theo nhóm làm việc và trợ giúp việc quản lý dự án, nhờ hệ thống quản lý cấu hình và kiểm soát các phiên bản.

Bạn đọc muốn hiểu sâu hơn và thực hành với Rational Rose xin tham khảo các sách [10] và [23].

Chương III
MÔ HÌNH HOÁ MÔI TRƯỜNG VÀ NHU CẦU

Chương này trình bày các bước 1 và bước 2 trong tiến trình 10 bước, mà mục đích là đưa ra một cái nhìn tổng quát đối với hệ thống và dự án, và tiếp đó là mô hình hoá môi trường và nhu cầu đối với hệ thống, dưới dạng các biểu đồ ca sử dụng.

§1. BƯỚC 1: NGHIÊN CỨU SƠ BỘ

1. MỤC ĐÍCH

Mục đích của bước nghiên cứu sơ bộ (bước 1) là điều tra, tìm hiểu về môi trường, hoàn cảnh nghiệp vụ của hệ thống sắp xây dựng, từ đó nhận định các nhu cầu chức năng và phi chức năng đặt ra đối với hệ thống đó, các nguy cơ và ràng buộc đối với nó, và cuối cùng là xác lập và hoạch định dự án xây dựng hệ thống đó.

Câu hỏi chính phải giải đáp ở đây là: Liệu hệ thống được chọn để xây dựng có thực là đáng chọn, đáng làm và sẽ làm được không?

2. PHƯƠNG PHÁP TIẾN HÀNH

Phương pháp cơ bản là điều tra, khảo sát theo nhiều cách:

a) Nghiên cứu các tài liệu viết

- Tài liệu đã hoàn chỉnh, như:
  + tài liệu giao dịch;
  + tài liệu lưu (sổ sách, tệp);
  + tài liệu tổng hợp (kế hoạch, thống kê).

- Tài liệu để làm tiếp, như:
  + tài liệu để bổ sung (bảng hỏi, phiếu thu thập...);
  + tài liệu chuẩn bị (cho cuộc họp, cho máy tính...).

b) Phỏng vấn (khảo sát bằng lời)

- Phỏng vấn kèm theo tài liệu viết:
  + để giải thích các thông tin viết;
  + để bổ sung các thông tin viết;
  + để kiểm tra các thông tin viết;
  + để cập nhật các thông tin viết
- Phỏng vấn không kèm tài liệu viết:
  + toạ đàm (gợi mở vấn đề);
  + phỏng vấn cá nhân theo chủ đề;
  + phỏng vấn theo nhóm.
- Phiếu điều tra, tờ khai (phỏng vấn gián tiếp):

Là có ích khi diện điều tra là quá rộng. Hai loại câu hỏi trong phiếu điều tra:

  + Câu hỏi đóng (các phương án trả lời đã dự kiến trước);
  + Câu hỏi mở (các phương án trả lời không dự kiến được).

c) Quan sát (khảo sát bằng mắt)

- Quan sát về chất, chẳng hạn quan sát:
  + tiến trình thực hiện một công việc;
  + đường di chuyển của một tài liệu.
- Quan sát về lượng, chẳng hạn:
  + đếm số lần của một loại giao dịch trong ngày;
  + bấm giờ cho việc hoàn thành một công việc nào đó.

3. TỔNG QUAN VỀ HỆ THỐNG

Các thông tin thu thập và điều tra cần phải sắp xếp và biên tập lại, để rồi tổng hợp thành một báo cáo tổng quan về hệ thống.

Hệ thống tin học mà ta sẽ xây dựng luôn luôn phải đặt trong một môi trường nghiệp vụ nào đó, bao gồm các con người, các tổ chức, các thiết bị (kể cả máy tính), cùng nhau hoạt động theo những quy trình chuyên môn nào đó. Chẳng hạn một hệ thống thông tin về quản lý thì đặt trong một môi trường kinh doanh hay dịch vụ (một công ty, một cơ quan), còn một hệ thống điều khiển quá trình thì đặt trong một môi trường kỹ thuật (một dây chuyền sản xuất, một cơ cấu máy). Ở giai đoạn khởi đầu này thì ta chưa thể phân biệt thực rõ ranh giới giữa hệ thống (mà ta sẽ phải xây dựng) và môi trường của nó. Cho nên nói tổng quan về hệ thống là nói sự miêu tả một sự hoạt động chung của con người, thiết bị và máy tính trong một môi trường nghiệp vụ, qua đó đặt ra bài toán và nhiệm vụ phải giải quyết cho hệ thống tin học cần xây dựng.

Nội dung của tổng quan có thể bao gồm các điểm sau:

- Mục đích của hoạt động nghiệp vụ.
- Các nhiệm vụ cơ bản của hoạt động nghiệp vụ. Đây là các chức năng chính phải thực hiện để đạt tới mục đích trên.
- Các quy trình nghiệp vụ. Một quy trình nghiệp vụ (business process) là một tập hợp các hoạt động, thường được phân bổ trên các bộ phận của tổ chức nhưng lại có liên quan lôgic hay liên quan thời gian với nhau, nhằm phối hợp để tiến tới một mục tiêu cụ thể nào đó trong nghiệp vụ.
- Các loại thông tin sử dụng trong hệ thống này bao gồm các thông tin trao đổi cũng như các thông tin lưu giữ.
- Các yêu cầu đặt ra đối với hệ thông tin học tương lai, bao gồm:

+ Các yêu cầu về chức năng;
+ Các yêu cầu phi chức năng, như là chất lượng, hiệu năng v.v...
+ Các ưu tiên, hạn chế, ràng buộc.

4. XÁC LẬP VÀ HOẠCH ĐỊNH DỰ ÁN

Quá trình nghiên cứu sơ bộ dẫn tới việc xác lập và hoạch định một dự án nhằm triển khai hệ thống tin học mới, bao gồm các việc sau:

- Xác định phạm vi và các hạn chế của dự án;
- Xác định các mục tiêu và ưu tiên cho dự án;

- Đề xuất giải pháp thô và chứng tỏ tính khả thi của nó;
- Dự đoán và đánh giá các nguy cơ có thể xảy ra (như nguy cơ về hiểu sai nhu cầu, nguy cơ về công nghệ chọn lựa không thích hợp, nguy cơ về cán bộ không đủ sức v.v...);
- Lập kế hoạch triển khai dự án (nhân sự, tài chính, lịch biểu...).

Thí dụ

Một hệ thống đăng ký môn học (ĐKMH) ở trường đại học

Đây là một thí dụ sử dụng xuyên suốt trong cả cuốn sách, để minh hoạ từng giai đoạn của quá trình phát triển hệ thống. Tuy nhiên, vì khuôn khổ cuốn sách, ta cũng không thể phát triển nó một cách thực sự chi tiết và đầy đủ được. Sau đây là tổng quan tóm tắt về hệ thống này.

Trường ĐHTL áp dụng chế độ học theo tín chỉ và cho phép sinh viên có quyền lựa chọn môn học cho mỗi học kỳ. Trước khi bước vào học kỳ mới, các thầy giáo đăng ký các môn học mà mình có thể dạy trong học kỳ đó. Căn cứ vào đó và vào kế hoạch học tập chung của trường, phòng quản sinh lập và niêm yết một danh sách các môn học có trong học kỳ, kèm với các thông tin cần thiết, như thầy giáo, số giờ các môn phải học trước... để sinh viên có căn cứ lựa chọn. Tiếp đó mỗi sinh viên điền vào một phiếu đăng ký các môn học mà mình chọn, rồi gửi phiếu đó cho phòng quản sinh. Thông thường, thì mỗi sinh viên chọn từ 6 đến 8 môn cho mỗi học kỳ và việc đăng ký được phép thực hiện trong một tuần.

Khi hết hạn đăng ký, cán bộ phòng quản sinh, dựa vào thông tin thu gom được, tổ chức các lớp giảng cho từng môn học. Một lớp giảng không được dưới 10 người và không quá 30 người. Do điều kiện hạn chế này mà có thể xảy ra trường hợp đụng độ, như là lớp giảng quá vắng, không tổ chức được, hoặc lớp giảng quá đông, mà tổ chức thêm một lớp nữa thì lại thiếu người. Trong những trường hợp đó thì phải thông báo cho các sinh viên không được thoả mãn yêu cầu, để họ đăng ký lại.

Khi đã hoàn tất việc xếp lớp, phòng quản sinh thông báo cho từng thầy giáo lịch dạy của mình, và thông báo cho từng sinh viên lịch học của mình. Mặt khác danh sách các môn học cho từng sinh viên cũng được gửi cho phòng tài vụ để tính học phí.

Nhà trường muốn xây dựng một hệ thống thông tin trên máy tính để trợ giúp quá trình đăng ký nói trên, trong đó thầy giáo có thể truy cập trực tuyến để đăng ký môn dạy hay xem danh sách sinh viên lớp mình dạy, còn sinh viên thì được dành một số ngày cho phép truy cập hệ thống để sửa (thêm hay bỏ) các môn học mà mình đã đăng ký.

§2. BƯỚC 2: NHẬN ĐỊNH VÀ ĐẶC TẢ CÁC CA SỬ DỤNG

1. MỤC ĐÍCH

Sau bước 1 nhằm đưa ra một sự diễn tả sơ bộ và phi hình thức về môi trường và nhu cầu đối với hệ thống thì bước 2 này diễn tả lại, chặt chẽ và cô đọng hơn, môi trường và nhu cầu đối với hệ thống thông qua các khái niệm của UML là đối tác và ca sử dụng. Ta sẽ phát hiện các đối tác và ca sử dụng, tìm các mối liên quan giữa chúng và đặc tả chúng, để mở đường cho các bước sau.

2. MÔ HÌNH HOÁ MÔI TRƯỜNG VỚI CÁC ĐỐI TÁC

a) Đối tác

Môi trường của hệ thống được thể hiện bằng một tập hợp các đối tác. Một *đối tác* (actor) hay *tác nhân ngoài* là một vai trò của một hay nhiều người hay vật thể trong sự tương tác với hệ thống.

Nhiều người (hay vật thể) có cùng một vai trò được diễn tả như là một đối tác duy nhất. Nhưng một người (hay vật thể) lại có thể có nhiều vai trò khác nhau (ví như một người có thể khi là người bán hàng, khi lại là thợ sửa chữa hàng bảo hành), và như vậy sẽ được diễn tả bởi nhiều đối tác.

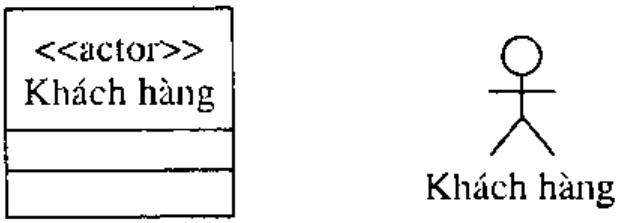
Đối tác phải là người (hay vật thể) có trao đổi thông tin với hệ thống, hay hưởng lợi từ hệ thống và phải có sự tự trị trong quyết định. Cho nên một thiết bị bị động thì không nên xem là đối tác (chẳng hạn màn hình, bàn phím...).

Có bốn loại đối tác:

- Các đối tác chính: đó là những người sử dụng các chức năng chính của hệ thống. Chẳng hạn Khách hàng.
- Các đối tác phụ: Đó là những người làm công việc quản lý, bảo dưỡng hệ thống.
- Các thiết bị ngoài: Đó là những thiết bị được hệ thống điều khiển (nhưng không phải là cái máy tính trên đó cài đặt hệ thống).
- Các hệ thống khác: Đó là các hệ thống không thuộc hệ thống đang xét, nhưng có tương tác với nó.

Để mô tả một đối tác, ngoài tên của nó, ta có thể chỉ rõ:

- Một tập hợp các thuộc tính cho phép phản ánh trạng thái của nó;
- Một tập hợp các tín hiệu mà nó có thể phát hay nhận. Vì những đặc điểm này, mà UML xem đối tác như là một khuôn dập định nghĩa sẵn của lớp, và biểu diễn đối tác dưới dạng một lớp mang khuôn dập <<actor>> hay một biểu tượng có dạng hình nhân như trong Hình III.1.



Hình III.1. Các biểu diễn đồ hoa của đối tác

b) Biểu đồ khung cảnh

Để mô tả môi trường của hệ thống, ta lập một biểu đồ khung cảnh (context diagram), mà thực chất là một biểu đồ giao tiếp đặc biệt, với các yếu tố sau:

- Hệ thống đang xét được biểu diễn bởi một đối tượng trung tâm. Thực chất đó chỉ là một hộp đen (không diễn tả nội dung bên trong). Tuy nhiên khi hệ thống là rất phức tạp, thì trong hộp đó có thể vẽ thêm vài ba hộp con biểu diễn cho các hệ thống con.
- Đối tượng trung tâm được vây quanh bởi các đối tượng biểu diễn cho các đối tác khác nhau của hệ thống.

- Giữa hệ thống và các đối tác có các kết nối (bằng các đoạn thẳng).
- Trên mỗi kết nối có các thông điệp vào và ra hệ thống (không đánh số thứ tự).

Ta gọi *thông điệp* là đặc tả cho một sự trao đổi giữa hai đối tượng, có thể mang theo thông tin và có mục đích khởi phát một hoạt động ở đối tượng nhận.

Sự tiếp nhận một thông điệp thường được xem là một *sự kiện* (event).

Để xác định thông điệp giữa các đối tác và hệ thống:

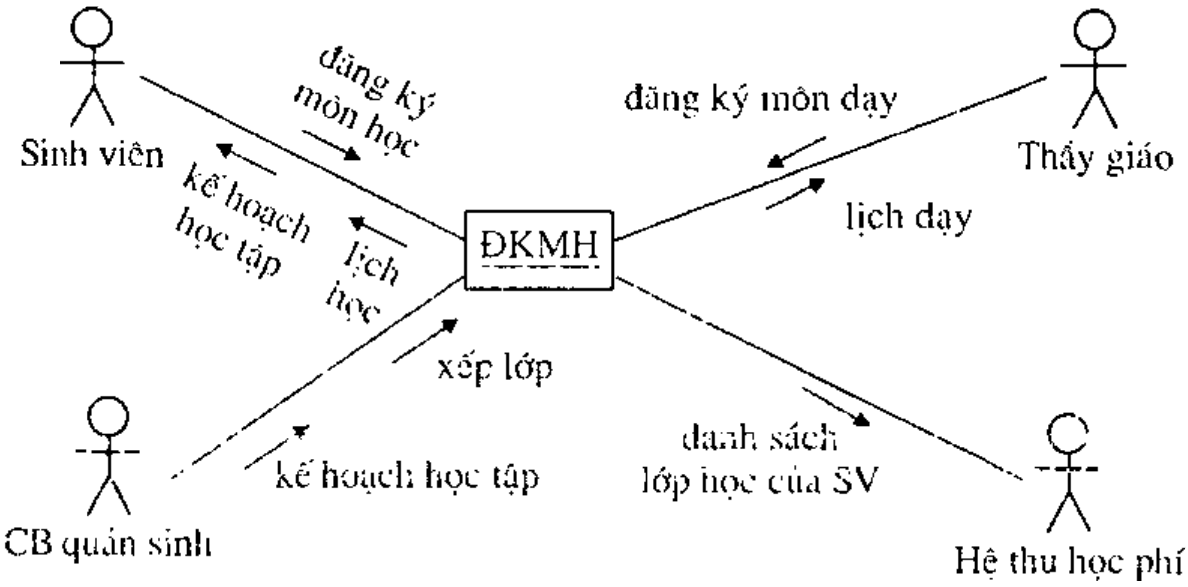
- Với mỗi đối tác, đặt câu hỏi rằng đối tác đó trong hoạt động của mình cần phát những thông điệp nào để kích hoạt một hành vi của hệ thống mà nó mong chờ.
- Với hệ thống, đặt câu hỏi là nó cần phát đi các thông điệp nào tới mỗi đối tác để cung cấp một thông tin cần thiết cho đối tác đó.

Thí dụ

Tiếp tục đi sâu vào hệ ĐKMH và xét các đối tác của nó. Các đối tác được phát hiện ở đây là:

- Sinh viên là người cần được đăng ký vào các lớp học;
- Thầy giáo, là người được xác định giảng dạy cho các lớp học;
- CB quản sinh, là người có trách nhiệm duy trì hoạt động của hệ ĐKMH (lập kế hoạch học tập, đăng ký và xếp lớp cho sinh viên, giữ các thông tin về các môn học, các thầy giáo, các sinh viên);
- Hệ thu học phí, là một hệ thống ngoài có trách nhiệm tính và thu học phí của SV.

Biểu đồ khung cảnh được lập như trên Hình III.2



Hình III.2. Biểu đồ khung cảnh của hệ ĐKMH

3. MÔ HÌNH HOÁ NHU CẦU VỚI CÁC CA SỬ DỤNG

UML diễn tả các nhu cầu đặt ra đối với hệ thống bằng các ca sử dụng, được thu gom lại vào một biểu đồ gọi là biểu đồ ca sử dụng.

a) Nhận định các ca sử dụng

Ca sử dụng (use case) là một biểu diễn của một tập hợp các chuỗi hành động, mà hệ thống thực hiện nhằm cung cấp một kết quả (một giá trị gia tăng) cụ thể cho một đối tác.

Như vậy, thì về thực chất, ca sử dụng là một chức năng của hệ thống. Tuy nhiên không phải chức năng nào của hệ thống cũng có thể được chọn làm ca sử dụng. Bởi vậy khi chọn ca sử dụng, ta cần lưu ý các đặc điểm sau đây:

- Một ca sử dụng phải liên kết với một hay một số đối tác, trong đó có một đối tác chính (đối tác kích hoạt ca sử dụng một cách trực tiếp hay gián tiếp).
- Một ca sử dụng phải dẫn tới một kết quả cụ thể, nghĩa là một kết quả nhận biết được, trọn vẹn và đo đếm được. Như vậy một chức năng quá lớn, với nhiều cái ra khác nhau nên vỡ ra thành nhiều ca sử dụng. Tuy nhiên những chức năng lặt vặt, góp phần vào một phiên làm việc của một đối tác với hệ thống, để dẫn tới một kết quả cụ thể thì lại nên gom lại vào

một ca sử dụng. Chẳng hạn cán bộ quản sinh phải thêm môn học, bỏ môn học, hay điều chỉnh môn học. Đó không phải là ba ca sử dụng mà chỉ là một ca sử dụng là quản lý các môn học, bởi vì đây mới là một công việc trọn vẹn, kích hoạt bởi cùng một đối tác (cán bộ quản sinh), đề cập tới cùng một thực thể trong hệ thống là kế hoạch học tập, để nhằm hoàn tất nó.

- Một ca sử dụng phải là tập hợp của nhiều chuỗi hành động. Gọi mỗi chuỗi hành động đó là một *kịch bản* (scenario). Sở dĩ có nhiều kịch bản, vì để đi tới một kết quả thì có nhiều đường, ứng với mỗi điều kiện đầu vào khác nhau. Trong các kịch bản của một ca sử dụng người ta phân biệt một kịch bản chính, các kịch bản phụ, các kịch bản ngoại lệ và sai hỏng. Như vậy ca sử dụng hiếm khi thu về một kịch bản duy nhất, lại càng không thể là một hành động duy nhất.

Tóm lại, để nhận định các ca sử dụng, ta đặt ra các câu hỏi:

- Mỗi đối tác cần gì ở hệ thống?
- Hệ thống có các cái vào, cái ra nào?

Thí dụ

Đối với hệ ĐKMH, sau khi có các đối tác, ta nhận định các ca sử dụng. Ta thấy các nhu cầu sau được đặt ra với hệ thống ĐKMH:

- Đối tác sinh viên cần dùng hệ thống để đăng ký các môn học.
- Sau khi hoàn tất việc đăng ký môn học, cần cung cấp các thông tin tính học phí cho Hệ Thu học phí.
- Đối tác Thầy giáo cần dùng hệ thống để chọn giáo trình giảng dạy cho một học kỳ và lấy về bản phân công giảng dạy.
- Đối tác CB quản sinh có trách nhiệm sản sinh danh sách các môn học trong một học kỳ, và quản lý mọi thông tin về kế hoạch học tập, các sinh viên và các thầy giáo cần cho hệ thống.

Dựa trên các nhu cầu đó, ta có thể nhận định được các ca sử dụng như sau:

- Đăng ký môn học.
- Chọn môn học để giảng dạy.

- Yêu cầu bản phân công giảng dạy.
- Duy trì thông tin môn học.
- Duy trì thông tin thầy giáo.
- Duy trì thông tin sinh viên.
- Lập bản giới thiệu các môn học.

b) Đặc tả ca sử dụng

Để đặc tả nội dung của một ca sử dụng, ta cần nói rõ ca sử dụng *làm gì*, nhưng lại không được lấn sang câu hỏi *làm như thế nào*. Ngôn ngữ đặc tả nên trước hết là ngôn ngữ tự nhiên (tiếng Việt), bởi vì đặc tả này cần được trao đổi kỹ với người dùng vốn là những người ít hiểu biết về các ký pháp tin học. Nội dung đặc tả chỉ rõ ca sử dụng trao đổi thông tin với các đối tác nào, nó bắt đầu và kết thúc ra sao, nó có thể diễn ra theo các kịch bản khả dĩ nào, trong đó phân biệt rõ trường hợp thông lệ (trường hợp chính), các trường hợp khả dĩ khác, các trường hợp sai hỏng buộc phải ngưng ngắt công việc và không đi đến một kết quả mong muốn.

Thông thường thì một phiếu đặc tả một ca sử dụng bao gồm các khoản mục như sau:

- Mô tả tóm tắt (bắt buộc): gồm tên, mục đích, tóm lược, đối tác, ngày, phiên bản, người lập v.v...
- Mô tả các kịch bản (bắt buộc): chỉ rõ các điều kiện đầu vào và các điều kiện đầu ra, các kịch bản (dãy hành động) thông lệ, khả dĩ, ngoại lệ (sai hỏng).
- Các yêu cầu về giao diện (tuỳ ý): có thể thêm các ràng buộc về giao diện người - máy: hiển thị gì, người dùng có thể khởi phát các thao tác nào.
- Các ràng buộc phi chức năng (tuỳ ý): có thể thêm các thông tin sau: tần suất, khối lượng, khả năng sẵn dùng, mức tin cậy, tính toàn vẹn, tính bảo mật, hiệu năng, sự song hành v.v... Các thông tin này có ích cho việc nắm bắt các nhu cầu về kỹ thuật sau này.

Thí dụ

Với hệ thống ĐKMH, ta cần đặc tả bảy ca sử dụng đã nêu trên. Tuy nhiên ở đây ta chỉ đưa ra một đặc tả cho một ca sử dụng để làm thí dụ, đó là ca sử dụng 'Chọn môn học để giảng dạy'.

(1) Mô tả tóm tắt

- Tên ca sử dụng: Chọn môn học để giảng dạy.
- Mục đích: Giúp người thầy giáo xác định môn mà mình sẽ giảng trong một học kỳ nào đó.
- Tóm lược: Thầy giáo chọn một học kỳ rồi sau đó có thể thêm, bỏ, xem, in các môn và kết thúc.
- Đối tác: Thầy giáo (chính).
- Ngày lập: 02/02/03 Ngày cập nhật: 24/03/03.
- Phiên bản: 1.3 Chịu trách nhiệm: N.V.Ba

(2) Mô tả các kịch bản

- *Điều kiện đầu vào*

Ca sử dụng này, chỉ có thể thực hiện khi kịch bản con 'Lập lớp giảng' (cho các môn học trong một học kỳ) của ca sử dụng 'Duy trì thông tin môn học' đã thực hiện.

- *Kịch bản chính* (thông lệ)

Ca sử dụng này bắt đầu khi người Thầy giáo đăng nhập hệ thống ĐKMH và nhập mật khẩu của mình. Hệ thống kiểm tra thấy mật khẩu đó là đúng đắn (R-1) và nhắc Thầy giáo chọn học kỳ này hay học kỳ sau (R-2). Thầy giáo nhập học kỳ mình muốn. Hệ thống nhắc Thầy giáo chọn việc trong: THÊM, BỎ, XEM, IN, RA.

Nếu THÊM được chọn thì kịch bản con

C-1 : Thêm một lớp giảng được thực hiện.

Nếu BỎ được chọn thì kịch bản con

C-2 : Bỏ một lớp giảng được thực hiện.

Nếu XEM được chọn thì kịch bản con

C-3 : Xem một lịch biểu được thực hiện.

Nếu IN được chọn thì kịch bản con

C-4 : IN một lớp giảng được thực hiện.

Nếu RA được chọn thì ca sử dụng kết thúc.

- *Các kịch bản con* (được dùng trong kịch bản chính)

C-1: Thêm một lớp giảng

Hệ thống hiển thị màn hình môn học bao gồm một trường cho tên môn học và một trường cho mã số môn học. Thầy giáo nhập tên môn học và mã số môn học (R-3). Hệ thống hiển thi các lớp giảng đối với môn học đã chọn (R-4). Thầy giáo chọn một lớp giảng. Hệ thống kết nối Thầy giáo với lớp giảng đã chọn (R-5). Ca sử dụng bắt đầu lại.

C-2: Bỏ một lớp giảng

Hệ thống hiển thị màn hình các lớp giảng trên đó có một trường cho tên của lớp giảng và một trường cho mã số lớp giảng. Thầy giáo nhập tên và mã số lớp giảng (R-6). Hệ thống dỡ bỏ kết nối của lớp giảng với thầy giáo (R-7). Ca sử dụng bắt đầu lại.

C-3: Xem một lịch biểu

Hệ thống tìm kiếm (R-8) và hiển thị các thông tin sau đây đối với các lớp giảng của thầy giáo: tên môn, mã số môn, mã số lớp giảng, các ngày lên lớp trong tuần, thời gian và địa điểm. Khi thầy giáo báo là đã xem xong thì ca sử dụng bắt đầu lại.

C-4: In một lịch biểu

Hệ thống in lịch biểu của thầy giáo (R-9). Ca sử dụng bắt đầu lại.

- *Các kịch bản khả dĩ khác* (rẽ ngang từ kịch bản chính)

R-1: Mật khẩu do thầy giáo đưa vào là không đúng đắn. Người dùng phải đưa lại mật khẩu hoặc kết thúc ca sử dụng.

R-2: Học kỳ đưa vào là không đúng đắn. Người dùng có thể nhập lại học kỳ hoặc kết thúc ca sử dụng.

R-3: Tên/mã số môn học đưa vào là không đúng đắn. Người dùng có thể nhập lại tên/mã số hoặc kết thúc ca sử dụng.

R-4: Các lớp giảng không hiển thị được. Thông báo cho người dùng là chọn lựa đó chưa sẵn sàng ở thời điểm hiện tại. Ca sử dụng bắt đầu lại.

R-5: Kết nối giữa thầy giáo và lớp giảng không thiết lập được. Thông tin được sao lưu và hệ thống sẽ lập kết nối sau. Ca sử dụng tiếp tục.

R-6: Tên/mã số lớp giảng đưa vào là không đúng đắn. Người dùng có thể nhập lại tên/mã số lớp giảng hoặc kết thúc ca sử dụng.

R-7: Kết nối giữa thầy giáo và lớp giảng không dỡ bỏ được. Thông tin được sao lưu và hệ thống sẽ dỡ bỏ kết nối sau. Ca sử dụng tiếp tục.

R-8: Hệ thống không tìm được thông tin về lịch biểu. Ca sử dụng bắt đầu lại.

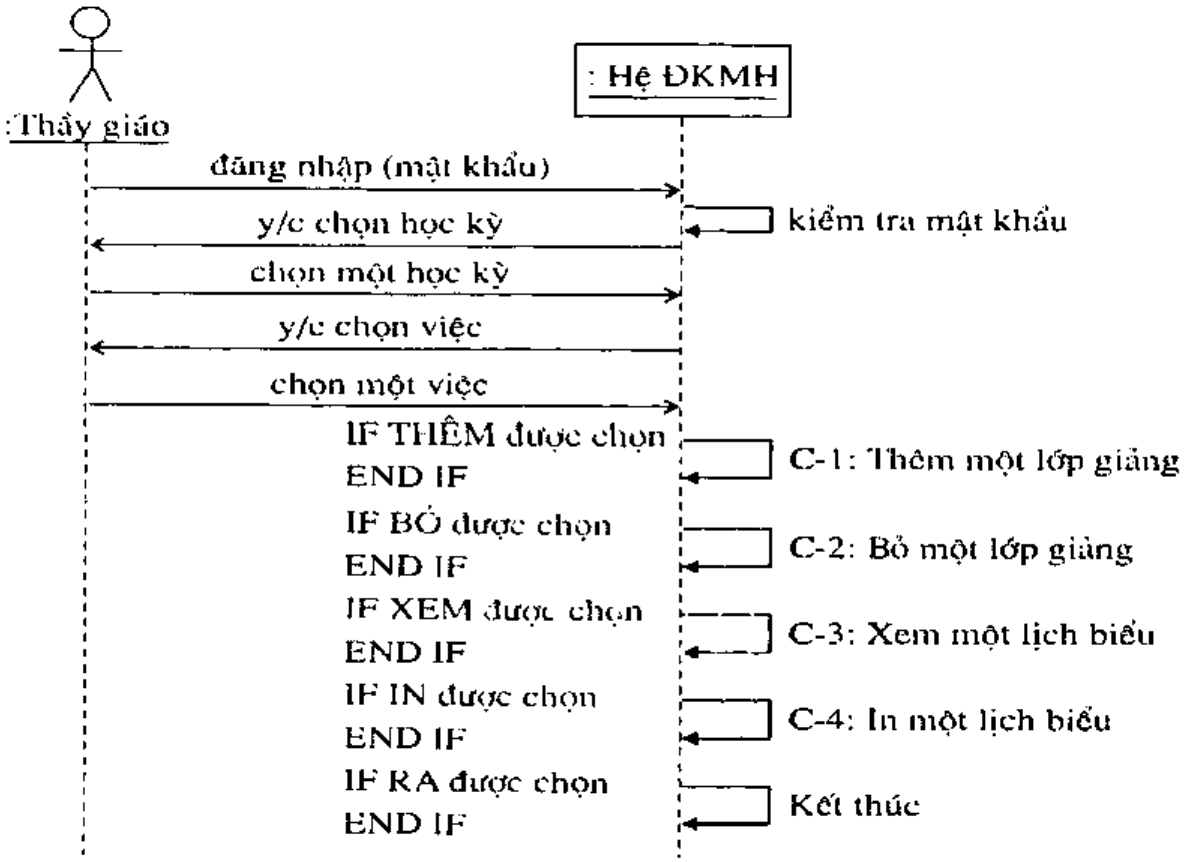
R-9: Lịch biểu không in được. Thông báo cho người dùng là chọn lựa này không sẵn dùng ở thời điểm hiện tại. Ca sử dụng bắt đầu lại.

Chú thích: Ngoài cách mô tả kịch bản bằng ngôn ngữ tự nhiên như trên, đôi khi người ta diễn tả nó dưới dạng một biểu đồ trình tự đặc biệt, gọi là *biểu đồ trình tự hệ thống*. Sự đặc biệt là ở chỗ:

- Hệ thống được xem như là một đối tượng;
- Các đối tác cũng là các đối tượng.

Các thông điệp giữa các đối tác và hệ thống được trình bày dọc theo trục thời gian (trên xuống) theo trình tự trước sau.

Chẳng hạn biểu đồ trình tự hệ thống cho kịch bản chính của ca sử dụng 'Chọn môn học để giảng dạy' là như trên Hình III.3.



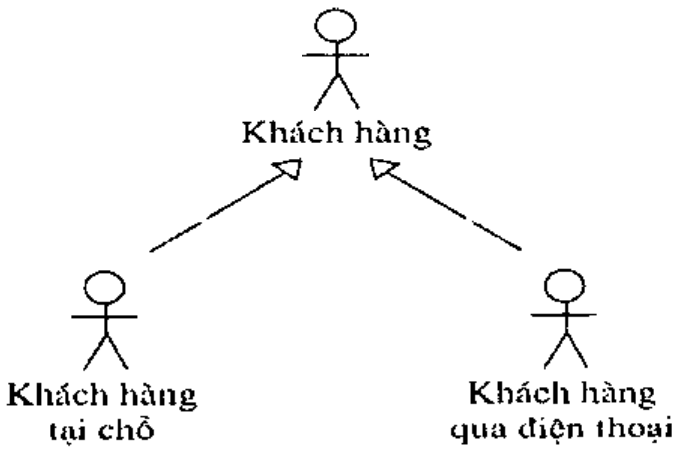
Hình III.3. Biểu đồ trình tự hệ thống diễn tả một kịch bản của ca sử dụng

4. BIỂU ĐỒ CA SỬ DỤNG

Sau khi phát hiện các đối tác và các ca sử dụng ta còn phải tổ chức chúng lại, bằng cách bổ sung các mối liên quan giữa chúng và lập thành biểu đồ ca sử dụng.

a) Liên quan giữa các đối tác

Giữa hai đối tác A, B có thể tồn tại mối liên quan *khái quát hoá*: bấy giờ đối tác B (đối tác chuyên biệt) thừa kế mọi hành vi và ngữ nghĩa của đối tác A (đối tác khái quát), hơn nữa B có thể đè lấp (thay đổi) một phần nào đó các hành vi thừa kế và có thể thêm hành vi mới. UML biểu diễn mối liên quan Khái quát hoá này giữa các đối tác (cũng như sẽ thấy giữa các lớp sau này) bằng một mũi tên có đầu hình tam giác rỗng (xem Hình III.4)



Hình III.4. Liên quan khái quát hoá giữa các đối tác

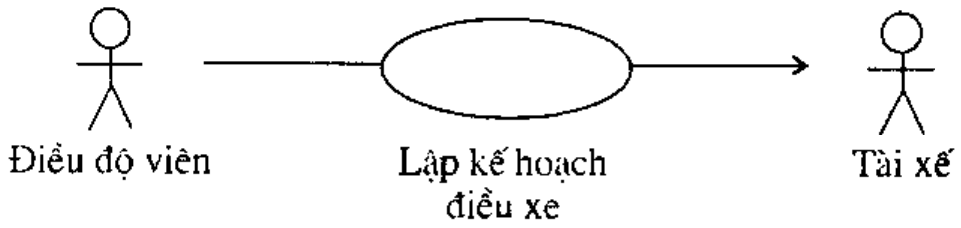
b) Liên quan giữa đối tác và ca sử dụng

UML biểu diễn một ca sử dụng bởi một hình elíp mang theo tên của ca sử dụng đó (viết bên trong hay ở dưới hình elíp).

Giữa một ca sử dụng và một đối tác có thể tồn tại một mối liên quan, thể hiện ở các khía cạnh như:

- Đối tác khởi phát ca sử dụng;
- Đối tác và ca sử dụng trao đổi thông tin cho nhau;
- Đối tác nhận kết quả (giá trị) từ ca sử dụng.

Mối liên quan như trên giữa ca sử dụng và đối tác được gọi là *liên kết giao tiếp* (communication association) và được biểu diễn bởi một nét liền nối chúng, và nét liền đó có thể chuyển thành mũi tên nếu sự trao đổi là một chiều (xem Hình III.5).



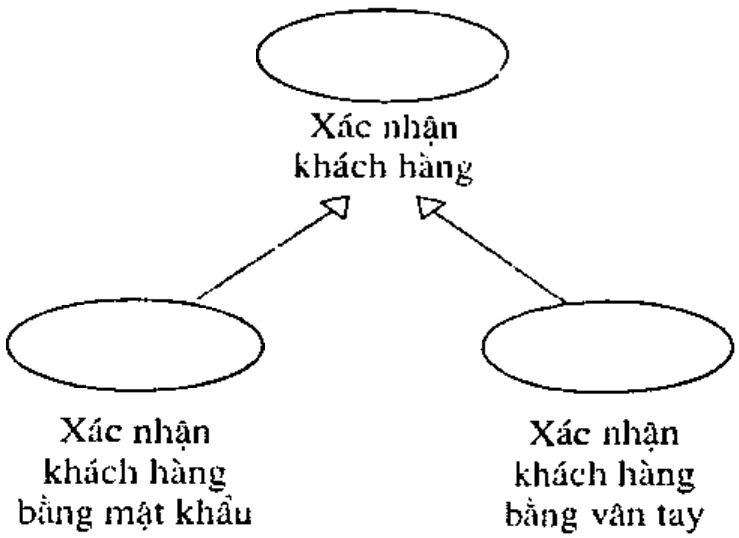
Hình III.5. Liên kết giao tiếp giữa đối tác và ca sử dụng

c) Liên quan giữa các ca sử dụng

UML phân biệt ba mối liên quan giữa các ca sử dụng, đó là liên quan khái quát hoá, liên quan bao hàm và liên quan mở rộng.

- *Liên quan khái quát hoá*

Giữa hai ca sử dụng có thể tồn tại mối liên quan khái quát hoá, biểu diễn bằng một mũi tên đầu tam giác rỗng nối chúng. Ca sử dụng chuyên biệt (ở gốc mũi tên) thừa kế mọi hành vi và ngữ nghĩa của ca sử dụng khái quát (ở đầu mũi tên), hơn nữa nó có thể đè lấp (thay đổi) một phần nào đó các hành vi thừa kế và có thể thêm hành vi mới. Một thí dụ cho trong Hình III.6.



Hình III.6. Liên quan khái quát hoá giữa các ca sử dụng

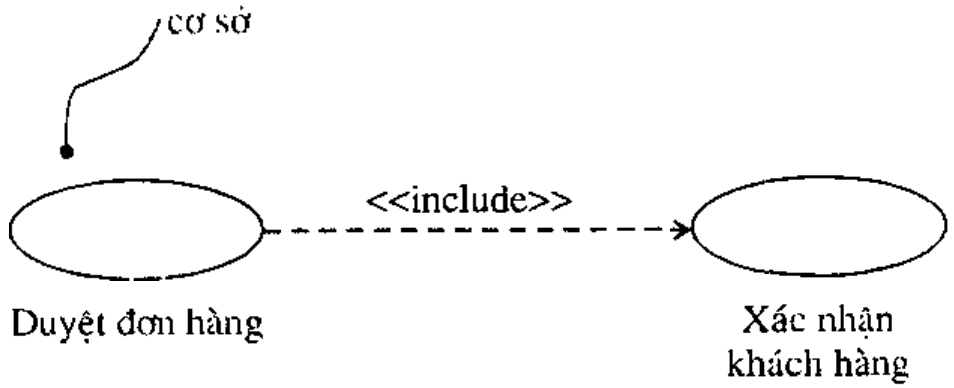
- *Liên quan bao hàm*

Mối liên quan bao hàm giữa hai ca sử dụng được biểu diễn bằng một mũi tên đứt nét nối chúng, có mang theo từ khoá <<include>>. Ca sử dụng ở gốc mũi tên là ca sử dụng cơ sở: nó sáp nhập một cách tường minh (bắt buộc) hành vi của ca sử dụng kia tại vị trí được chỉ rõ trong đặc tả của nó. Người ta thường dùng mối liên quan bao hàm để tách một phần chung của nhiều ca sử dụng. Trong thí dụ cho ở Hình III.7 thì đặc tả của ca sử dụng 'Duyệt đơn hàng' sẽ có kịch bản:

Luồng chính: Nhận và kiểm tra số hiệu đơn hàng.

include (xác nhận khách hàng).

Với mỗi mặt hàng trong đơn hàng, hỏi tình trạng tồn kho, báo lại cho khách hàng.



Hình III.7. Mối liên quan bao hàm

- *Liên quan mở rộng*

Mối liên quan mở rộng giữa hai ca sử dụng được biểu diễn bằng một mũi tên đứt nét nối chúng, có mang theo từ khoá <<extend>>. Ở đây ca sử dụng ở **đầu** mũi tên lại là ca sử dụng cơ sở: nó sáp nhập một cách không tường minh (tuỳ chọn theo những điều kiện nào đó) hành vi của ca sử dụng kia tại một số điểm đặc biệt trong kịch bản của nó (gọi là các điểm mở rộng). Người ta thường dùng mối liên quan mở rộng để chỉ rõ:

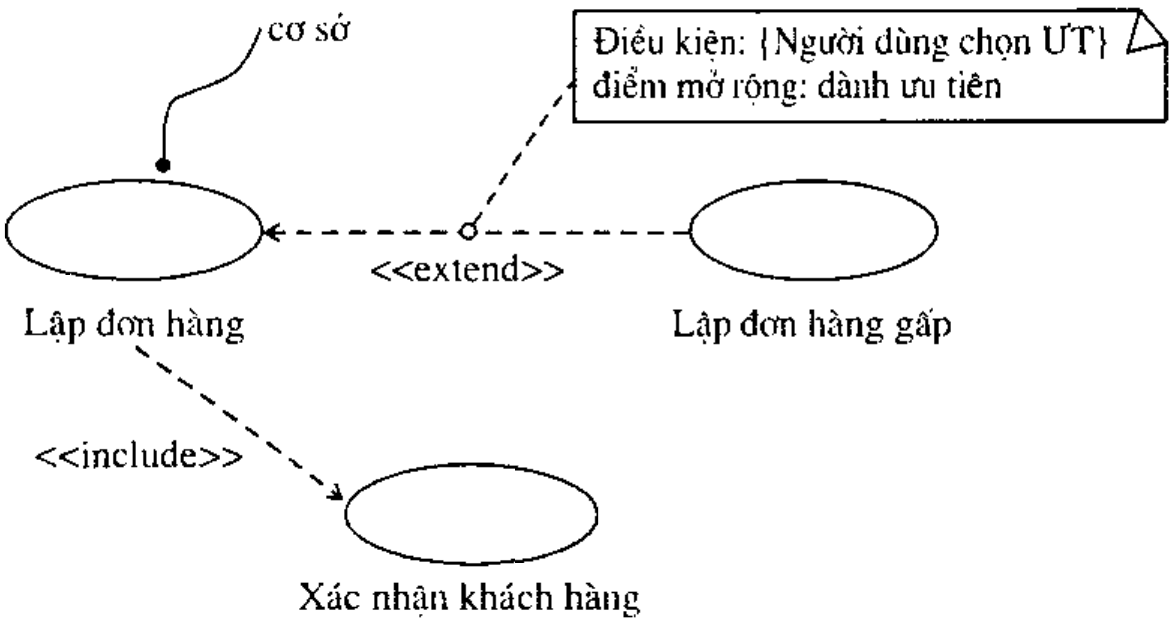
- Hành vi tuỳ chọn;
- Hành vi chỉ xảy ra dưới một số điều kiện nào đó (chẳng hạn kích hoạt một báo động);
- Nhiều luồng sự kiện có thể xảy ra theo sự chọn lựa của đối tác.

Trong thí dụ cho ở Hình III.8 thì đặc tả của ca sử dụng 'Lập đơn hàng' có kịch bản như sau:

Luồng chính: include (Xác nhận khách hàng).

Ghi nhận các mặt hàng được đặt. (dành ưu tiên).

Đưa đơn hàng sang xử lý.



Hình III.8. Mối liên quan mở rộng

Chú thích: Nếu mở rộng là có điều kiện thì điều kiện và điểm mở rộng được viết trong một Chú thích. Nếu mở rộng không có điều kiện thì điểm mở rộng được viết trong hình elíp của ca sử dụng cơ sở. UML diễn tả chú thích trong một mô hình bằng một hình chữ nhật gấp góc.

d) Thành lập biểu đồ ca sử dụng

Các đối tác, các ca sử dụng cùng các mối liên quan giữa chúng được diễn tả chung trong một biểu đồ, gọi là biểu đồ ca sử dụng.

Việc thành lập biểu đồ ca sử dụng có thể thực hiện theo trình tự sau:

- Phát hiện các đối tác theo từng nhóm:
  - + Nhóm đòi hỏi trợ giúp của hệ thống trong hoạt động của mình;
  - + Nhóm cần thiết cho sự thực hiện các chức năng của hệ thống;

+ Nhóm các thiết bị hay hệ thống mềm có tương tác với hệ thống đang xét;
+ Nhóm thực hiện các chức năng phụ trợ như quản trị, duy tu hệ thống.

- Tổ chức các đối tác tương tự với nhau bằng sự phân cấp khái quát hoá.
- Với mỗi đối tác xem xét nó chờ đợi hay yêu cầu những hành vi đáp ứng gì từ hệ thống.
- Đặt tên và diễn tả các hành vi đáp ứng của hệ thống như là các ca sử dụng.
- Rút phần chung của nhiều ca sử dụng lập thành ca sử dụng mới với mối liên quan bao hàm từ các ca sử dụng trên.
- Rút các biến thể hay tuỳ chọn khỏi phần cố định của các ca sử dụng để lập ca sử dụng mới với mối liên quan mở rộng tới ca sử dụng cũ.
- Gom tất cả vào một biểu đồ ca sử dụng, thêm các khuôn dập hay chú thích nếu cần.

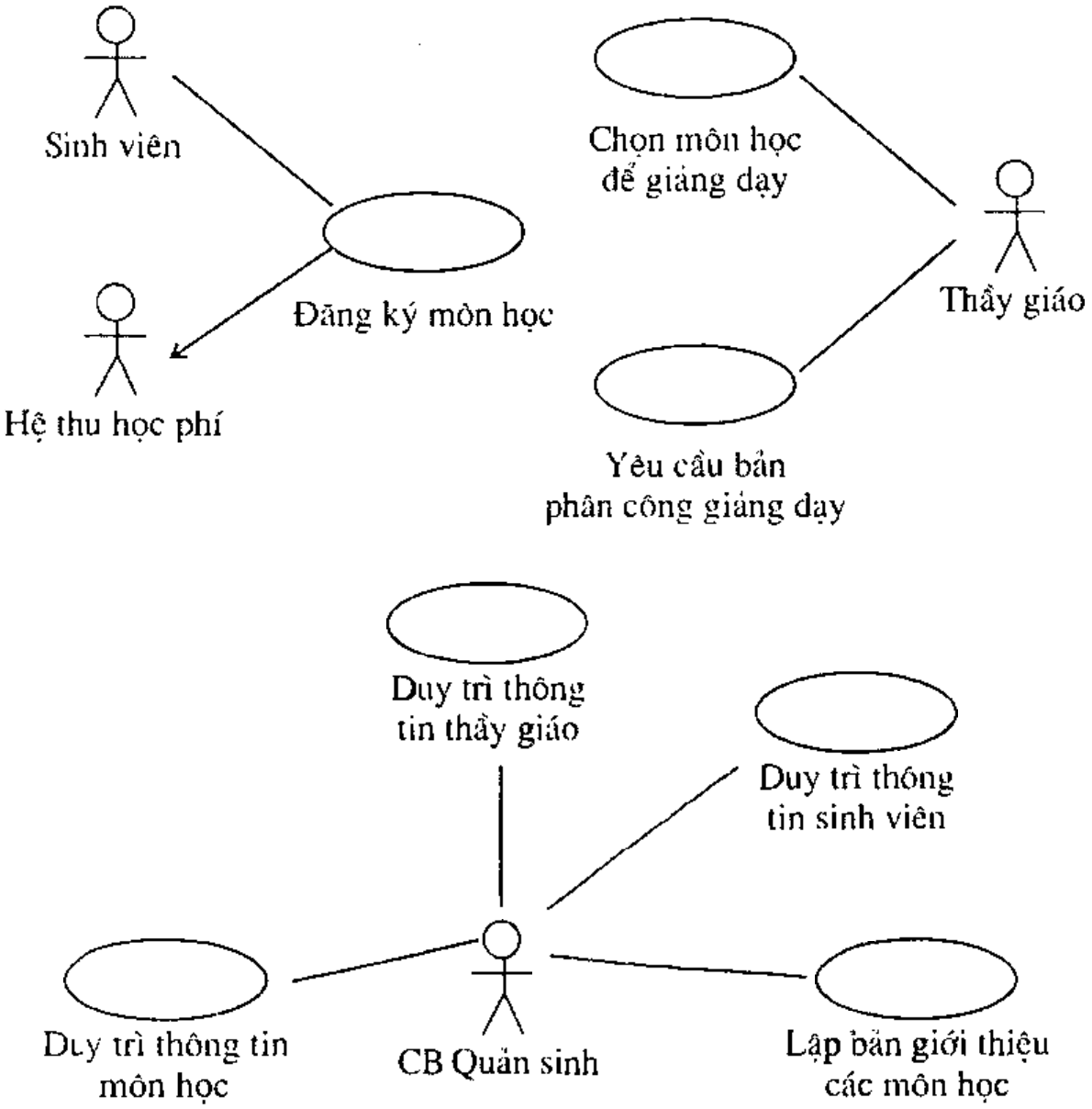
Chú thích: Biểu đồ ca sử dụng gồm tất cả các đối tác và các ca sử dụng của hệ thống thành lập như trên được gọi là biểu đồ ca sử dụng chính. Nếu biểu đồ này là quá phức tạp (gồm hàng trăm ca sử dụng), thì ta nên:

+ Gom nhóm các ca sử dụng gần gũi nhau về chức năng thành gói (gói được biểu diễn dưới dạng một hình chữ nhật có quai);
+ Có thể vẽ tách một số biểu đồ ca sử dụng khác theo những mục đích diễn tả riêng biệt, chẳng hạn:
  - Một biểu đồ trình bày mọi ca sử dụng cho một đối tác nào đó;
  - Một biểu đồ trình bày mọi ca sử dụng được cài đặt cùng nhau trong một bước lặp;
  - Một biểu đồ trình bày một ca sử dụng cùng với các ca sử dụng liên quan với nó.

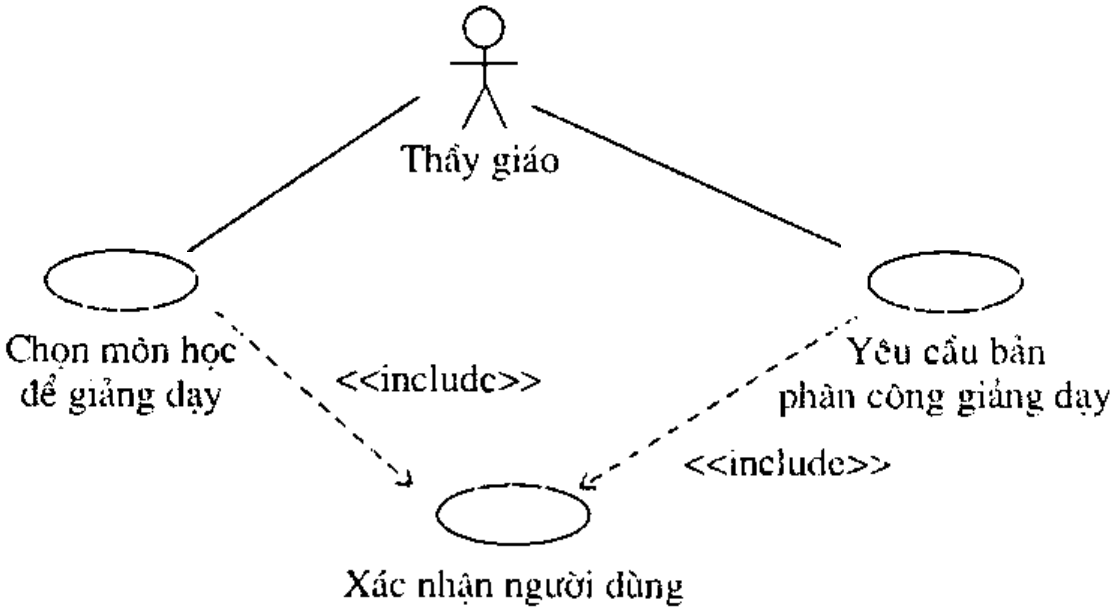
Thí dụ

Trở lại hệ ĐKMH và lập các biểu đồ ca sử dụng cho nó.

Biểu đồ ca sử dụng chính được cho trên Hình III.8 và một biểu đồ ca sử dụng phụ cho trên Hình III.9.



Hình III.8. Biểu đồ ca sử dụng chính



Hình III.9. Một biểu đồ ca sử dụng phụ

Chương IV
MÔ HÌNH HOÁ CẤU TRÚC

Diễn tả hệ thống bằng các ca sử dụng thực chất là một sự diễn tả chức năng (nhìn từ phía người dùng). Một sự phân tích tiếp tục theo hướng chức năng sẽ dẫn ta trở lại con đường của các phương pháp hướng chức năng kinh điển, kéo theo những nhược điểm của nó.

Vì vậy, sau bước diễn tả nhu cầu bằng các ca sử dụng, ta cần chuyển sang cách tiếp cận đối tượng. Việc phát hiện ra các đối tượng và lớp cùng với cấu trúc và hành vi đầy đủ của chúng là cả một chặng đường dài. Ở chương này, ta chỉ xét đến sự phát hiện và khẳng định vai trò của chúng một cách sơ bộ. Có hai nguồn để phát hiện chúng:

- Từ các khái niệm được sử dụng trong lĩnh vực ứng dụng thường là các khái niệm vật thể hoặc sự kiện, ta đề xuất ra các đối tượng và lớp để miêu tả, mô phỏng chúng. Loại đối tượng này thường được gọi là các đối tượng thực thể hay đối tượng lĩnh vực.
- Từ mỗi ca sử dụng, ta nghiên cứu xem cần có sự hợp tác của những đối tượng nào để thực hiện được ca sử dụng này. Qua đó ta có thể gặp lại các đối tượng thực thể ở trên, và như vậy khẳng định được vị thế của chúng trong hệ thống, đồng thời ta lại phát hiện thêm được các loại đối tượng phù trợ, như là các đối tượng biên (giao diện) và các đối tượng điều khiển.

Tuy nhiên trước khi đi sâu vào hai cách tiếp cận này (bước 3 và bước 4), ta hãy xem xét khái niệm chung về đối tượng và lớp cùng với cách diễn tả chúng trong UML đã.

§1. ĐỐI TƯỢNG, LỚP, GÓI VÀ LOÀI

1. ĐỊNH NGHĨA VÀ BIỂU DIỄN CỦA ĐỐI TƯỢNG VÀ LỚP

Đối tượng (tin học) là một biểu diễn trừu tượng của một thực thể (vật lý hay khái niệm) có căn cước và ranh giới rõ ràng trong thế giới thực, cho phép thâu tóm cả trạng thái và hành vi của thực thể đó, nhằm mục đích mô phỏng hay điều khiển thực thể đó.

Trạng thái của đối tượng thể hiện bởi một tập hợp các *thuộc tính*. Ở mỗi thời điểm, mỗi thuộc tính của đối tượng có một giá trị nhất định.

Hành vi của đối tượng thể hiện bằng một tập hợp các *thao tác*, đó là các dịch vụ mà nó có thể thực hiện khi được một đối tượng khác yêu cầu.

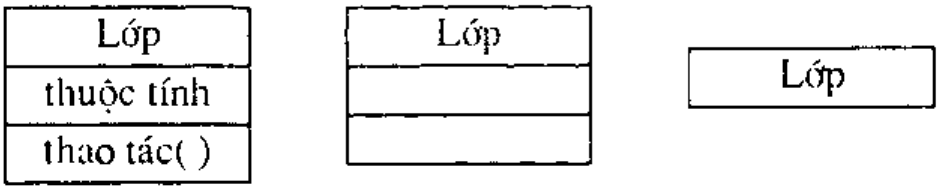
Căn cước của đối tượng là cái để phân biệt nó với đối tượng khác. Căn cước là độc lập với các thuộc tính. Cho nên hai đối tượng có thể có các giá trị thuộc tính trùng nhau, nhưng vẫn được phân biệt nhờ có căn cước riêng của chúng. Trong khi mô hình hóa, thì căn cước luôn luôn được xem là hiện hữu một cách ngầm định. Khi cài đặt, thì căn cước có thể được thực hiện bằng nhiều cách, chẳng hạn bằng định danh hay địa chỉ. Nếu là định danh, thì có thể xem đó là một thuộc tính đặc biệt (có giá trị cố định và duy nhất).

Lớp là một mô tả của một tập hợp các đối tượng cùng có chung các thuộc tính, các thao tác, các mối liên quan, các ràng buộc và ngữ nghĩa. Vậy lớp là một kiểu, và mỗi đối tượng thuộc lớp là một *cá thể* (instance).

UML biểu diễn lớp bằng một hình chữ nhật có ba ngăn:

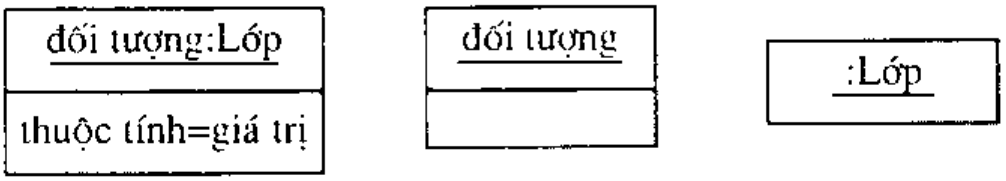
- ngăn thứ nhất dành cho tên lớp (tên lớp phải bắt đầu bằng một chữ cái viết hoa);
- ngăn thứ hai đành cho các thuộc tính (tên thuộc tính phải bắt đầu bằng một chữ cái viết thường);
- ngăn thứ ba dành cho các thao tác (tên thao tác phải bắt đầu bằng một chữ cái viết thường).

Ngoài ra đôi khi có thể vẽ thêm một ngăn thứ tư dành cho các trách nhiệm của lớp. Ngược lại, khi muốn vẽ đơn giản, thì ngoài ngăn tên là bắt buộc, các ngăn khác có thể để trống hay lược bỏ. (Xem Hình IV.1).



Hình IV.1. Các biểu diễn của lớp

Đối tượng được biểu diễn bởi một hình chữ nhật có hai ngăn: ngăn tên và ngăn các giá trị thuộc tính. Tên đối tượng phải được gạch dưới, có thể kèm theo tên lớp (đặt sau dấu ':'). Có khi có thể bỏ hay để trống ngăn giá trị thuộc tính và bỏ cả tên đối tượng nếu đã có lớp (trường hợp khuyết danh) (xem Hình IV.2).



Hình IV.2. Các biểu diễn của đối tượng

Sau đây, ta sẽ trình bày chi tiết hơn về sự mô tả các lớp và đối tượng. Tuy nhiên các chi tiết mô tả do UML đưa ra là rất phong phú, ta không thể kể hết ra đây được. Mặt khác các chi tiết này cũng không phải ngay từ đầu đã có thể phát hiện hết được, mà phải dần dà qua quá trình phân tích, ta mới nhận định và bổ sung chúng cho đầy đủ.

2. CÁC THUỘC TÍNH

Thuộc tính là một tính chất có đặt tên của một lớp và nó nhận một giá trị cho mỗi đối tượng thuộc lớp đó tại mỗi thời điểm. Cú pháp đầy đủ của một thuộc tính là như sau, trong đó trừ tên là bắt buộc phải có, còn các thành tố khác có thể bỏ qua (đặt trong cặp móc vuông []):

[tầm nhìn] [/] tên [: Kiểu] [cơ số] [= giá trị đầu][{xâu tính chất}]

- *Tầm nhìn* (visibility) cho biết thuộc tính đó được thấy và dùng từ các lớp khác như thế nào. Tầm nhìn có thể là:
  - *Công cộng* (public), ký hiệu bởi dấu '+', nếu thuộc tính đó là thấy và dùng được cả bên ngoài lớp;

- *Riêng tư* (private), ký hiệu bởi dấu '-', nếu thuộc tính không thể truy cập từ các lớp khác;
- *Bảo hộ* (protected), ký hiệu bởi dấu '#', nếu thuộc tính có thể truy cập từ các lớp thừa kế (xem mục 5).
- *Gói* (package), ký hiệu bởi dấu '~', nếu thuộc tính có thể truy cập từ các phần tử thuộc cùng một gói (hẹp nhất) với lớp.

- *Cơ số* (multiplicity), trỏ số các giá trị có thể nhận, chẳng hạn [0..1] để trỏ thuộc tính này là tuỳ chọn (Không nhận giá trị nào, hay có nhận một giá trị).
- *Kiểu* (type): đó là kiểu của các giá trị thuộc tính, thông thường thì đó là các kiểu nguyên thuỷ như Integer, Real, Boolean, nhưng cũng có thể là các kiểu khác như Point, Area, Enumeration, kể cả kiểu đó là một lớp khác.
- *Giá trị đầu* (initial value) là giá trị ngầm định gán cho thuộc tính khi một đối tượng được tạo lập từ lớp đó.
- *Xâu tính chất* (property-string) để trỏ các giá trị có thể gán cho thuộc tính, thường dùng đối với một kiểu liệt kê, chẳng hạn:

  tình trạng: Tìnhtrạng = chưatrả {chưatrả, đãtrả}

Ngoài ra, mỗi thuộc tính lại có thể có *phạm vi lớp* (class-scope) nếu nó phản ánh đặc điểm của lớp chứ không phải của riêng đối tượng nào. Chẳng hạn thuộc tính 'số các hoá đơn' trong lớp Hoá đơn. Thuộc tính thuộc phạm vi lớp phải được gạch dưới.

Một thuộc tính là *dẫn xuất*, nếu giá trị của nó được tính từ giá trị của những thuộc tính khác của lớp. Thuộc tính dẫn xuất phải mang thêm dấu gạch chéo '/' ở đầu, chẳng hạn /tuổi (khi đã có ngày sinh).

3. CÁC THAO TÁC

Thao tác là một dịch vụ mà đối tượng có thể đáp ứng được khi được yêu cầu (thông qua một thông điệp). Các thao tác được cài đặt thành các phương thức.

Cú pháp đầy đủ của một thao tác là như sau:

[tầm nhìn] tên[(danh sách tham số)][: Kiểu trả lại][{xâu tính chất}]

- *Tầm nhìn* hoàn toàn giống tầm nhìn của thuộc tính.

- *Danh sách tham số* là một danh sách gồm một số các tham số hình thức, cách nhau bằng dấu phẩy, mỗi tham số có dạng:

  [hướng] tên : Kiểu [= giá trị ngầm định]

  – *Hướng* có thể lấy các giá trị in, out, inout và return tuỳ thuộc tham số là input - không thể điều chỉnh, output - có thể điều chỉnh để đưa thông tin cho bên gọi, hoặc là input có thể điều chỉnh được, hay là để trả lại kết quả cho bên gọi.

  – Giá trị ngầm định là giá trị được sử dụng khi trong lời gọi khuyết tham số thực tương ứng.

- *Xâu tính chất* bao gồm các tiền đề, hậu đề, các tác động lên trạng thái đối tượng...

Thông thường thì các chi tiết mô tả tuỳ chọn của các thuộc tính hay các thao tác không xuất hiện một cách đầy đủ trên mô hình, mà được lưu giữ riêng ra chỗ khác, nhằm làm cho mô hình quang đãng, dễ đọc. Với một công cụ CASE, khi cần ta có thể bấm chuột chẳng hạn để làm lộ diện tất cả các chi tiết đó ra.

4. MỐI LIÊN QUAN PHỤ THUỘC

Giữa các lớp có thể có ba mối liên quan:

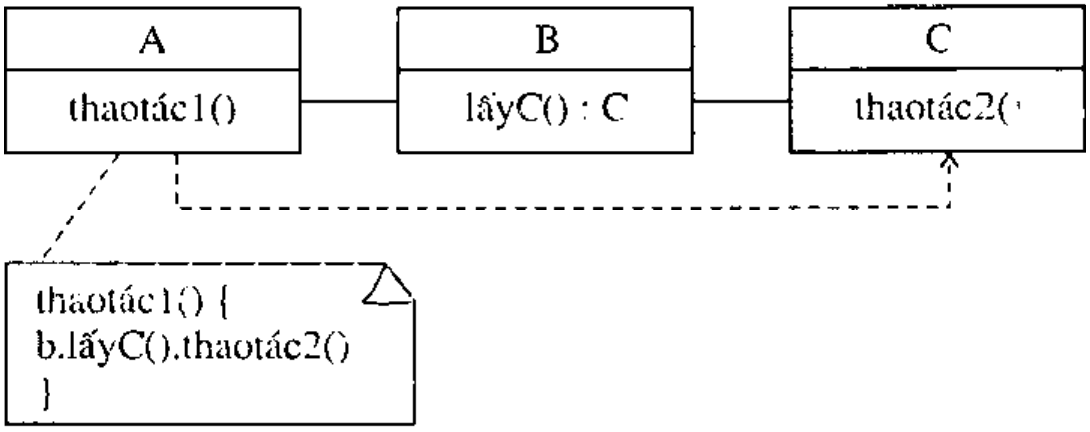
– mối liên quan phụ thuộc,

– mối liên quan khái quát hóa,

– mối liên quan liên kết.

Ở mục này, ta để cập mối liên quan đầu, và các mối liên quan còn lại sẽ được trình bày ở các mục tiếp sau.

Mối liên quan phụ thuộc (dependency relationship) thường dùng để diễn đạt một lớp (bên phụ thuộc) chịu ảnh hưởng của mọi thay đổi trong một lớp khác (bên độc lập), mà ngược lại thì không nhất thiết. Thường thì bên phụ thuộc cần dùng bên độc lập để đặc tả hay cài đặt cho mình. UML biểu diễn mối liên quan phụ thuộc bằng một mũi tên đứt nét (từ bên phụ thuộc sang bên độc lập). Một thí dụ cho trên Hình IV.3.



Hình IV.3. Lớp A phụ thuộc lớp C

Chú thích: Trong Hình IV.3 thì một ghi chú (trình bày trong hình chữ nhật gấp góc) đã cho thấy nội dung cài đặt của thaotác1() của lớp A đã vận dụng thaotác2() của lớp C, và điều này đã làm cho lớp A phụ thuộc vào lớp C.

Có nhiều biểu hiện khác nhau của sự phụ thuộc. Ta thường dùng từ khoá (khuôn dập) để chỉ rõ sự khác biệt đó. Sau đây là vài từ khoá chuẩn:

<<use>>: Chỉ rằng ngữ nghĩa của lớp gốc phụ thuộc vào ngữ nghĩa của lớp ngọn. Đặc biệt là trường hợp lớp gốc dùng lớp ngọn làm tham số trong một thao tác của nó.

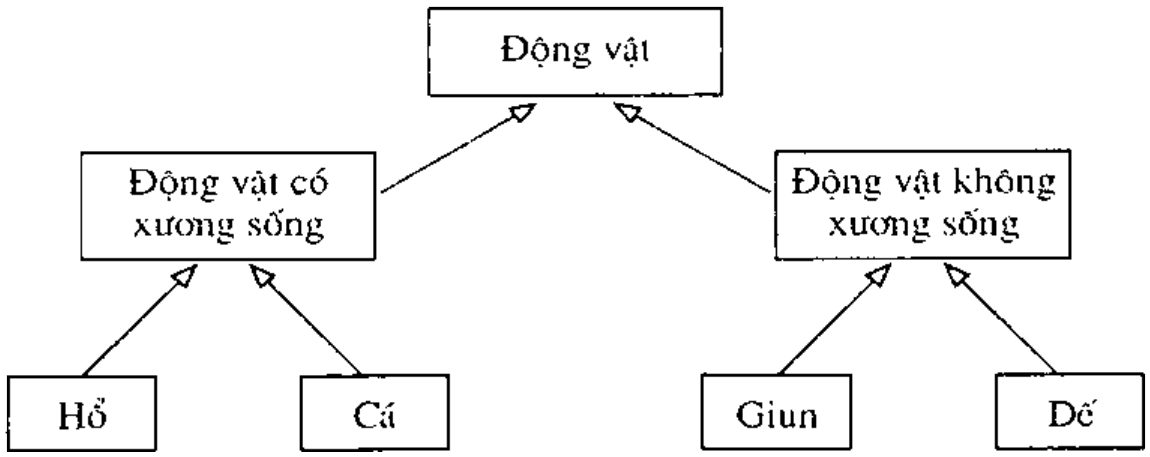
<<permit>>: Chỉ rằng lớp gốc được quyền truy cập một cách đặc biệt vào lớp ngọn (chẳng hạn truy cập cả các thao tác riêng tư). Khái niệm permit tương ứng với khái niệm friend trong C++.

<<refine>>: Chỉ rằng lớp gốc ở một mức độ tinh chế cao hơn từ lớp ngọn. Chẳng hạn một lớp lập ở giai đoạn thiết kế nhằm tinh chế cùng lớp đó lập ở giai đoạn phân tích.

Trong UML mối liên quan phụ thuộc không phải chỉ được thiết lập giữa các lớp, mà còn có thể giữa đối tượng và lớp (với từ khoá <<instanceOf>>), giữa các trường hợp sử dụng (với các từ khoá <<include>> và <<extend>>), giữa các gói (với các từ khoá <<import>>, <<access>> và <<merge>>) v.v...

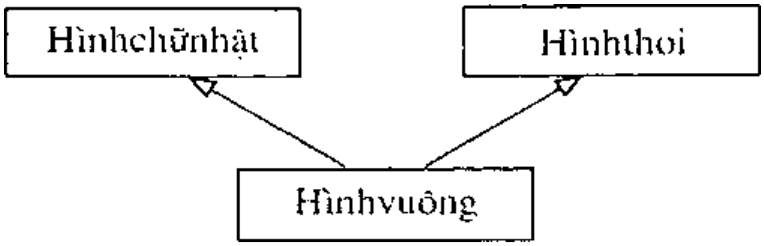
5. MỐI LIÊN QUAN KHÁI QUÁT HOÁ

Khái quát hoá (generalization) là sự rút ra các đặc điểm chung của nhiều lớp để tạo thành một lớp giản lược hơn gọi là *lớp trên* (hay cha). Quá trình ngược lại gọi là *chuyên biệt hoá* (specialization): từ một lớp đã cho ta tăng cường thêm một số đặc điểm mới, tạo thành một lớp chuyên hơn, gọi là *lớp dưới* (hay *con*). Mối liên quan giữa các lớp con và lớp cha gọi là liên quan khái quát hoá, được biểu diễn bằng một mũi tên có đầu tam giác rỗng (thường được đọc là is-a-kind-of). Nếu ta kéo dài sự khái quát hoá hay chuyên biệt hoá, ta lập được một cây phân cấp các lớp, như một thí dụ quen thuộc cho trên Hình IV.4. Cây này thường được gọi là một phả hệ khái quát hoá.



Hình IV.4. Cây phân cấp khái quát hoá

Một lớp có thể không có cha, có một hay nhiều cha. Một lớp không có cha và có một hay nhiều con gọi là *lớp gốc* hay *lớp cơ sở*. Một lớp chỉ có một cha gọi là lớp *thừa kế đơn* (simple inheritance). Một lớp có nhiều cha được gọi là *lớp thừa kế bội* (multiple inheritance). Một thí dụ của thừa kế bội cho ở Hình IV.5.

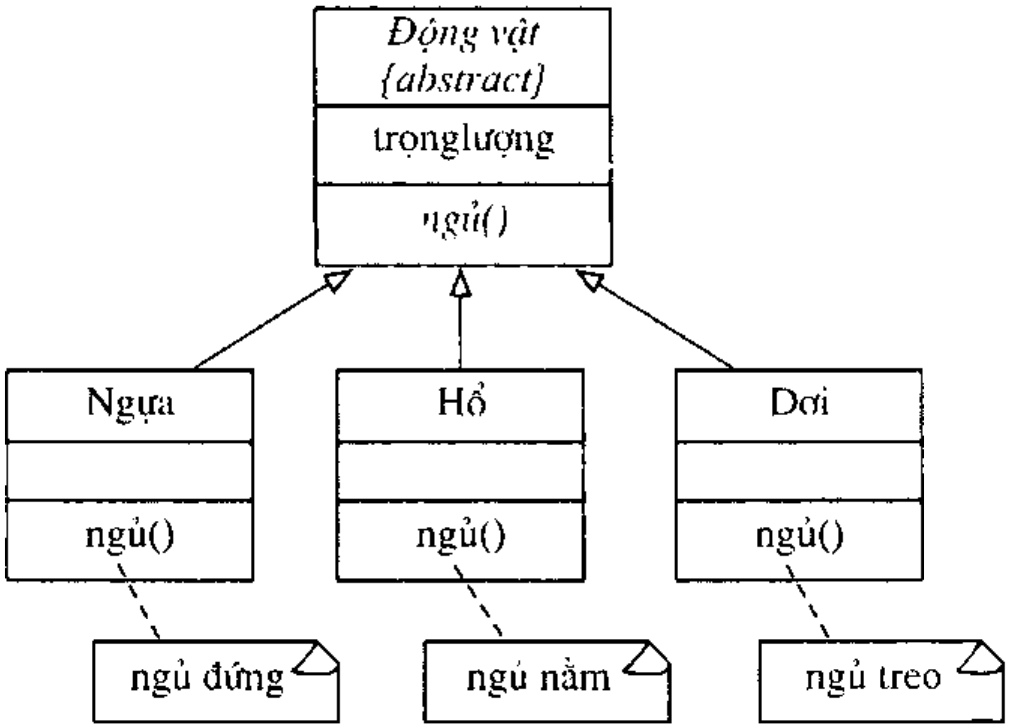


Hình IV.5. Thừa kế bội

Thuật ngữ thừa kế vốn được dùng nhiều trong các ngôn ngữ lập trình, nhằm diễn tả lớp con có mọi thuộc tính, thao tác và liên kết (sẽ nói ở dưới) được mô tả ở lớp cha. Hơn nữa lớp dưới có thể thêm thuộc tính, thao tác và liên kết mới và lại còn có thể định nghĩa lại (đè lấp) một thao tác của lớp trên. Kỹ thuật lập trình mà trong đó một đối tượng của một lớp dưới hoạt động như một đối tượng của lớp trên, nhưng với một vài thao tác đã được định nghĩa lại, được gọi là sự *đa hình* hay *đa xạ* (polymorphism).

Thông qua sự khái quat hoá ta có thể làm xuất hiện các *lớp trừu tượng*, nghĩa là các lớp không có cá thể, mà chỉ dùng cho việc mô tả các đặc điểm chung của những lớp dưới mà thôi. Một lớp trừu tượng thường có chứa các *thao tác trừu tượng*, là các thao tác chỉ có tiêu đề mà không có cài đặt. Thao tác trừu tượng phải được định nghĩa lại và kèm cài đặt ở các lớp dưới. Tên của lớp trừu tượng và tiêu đề của thao tác trừu tượng phải được viết xiên và có thể kèm thêm xâu tính chất {abstract}.

Trong thí dụ cho ở Hình IV.6, thì Động vật là lớp trừu tượng. Nó có một thao tác trừu tượng là ngủ(). Thao tác trừu tượng này sẽ được cụ thể hoá trong các lớp chuyên biệt là Ngựa, Hổ, Dơi theo các cách thức khác nhau (ngủ đứng, ngủ nằm và ngủ treo).



Hình IV.6 Lớp trừu tượng và thao tác trừu tượng

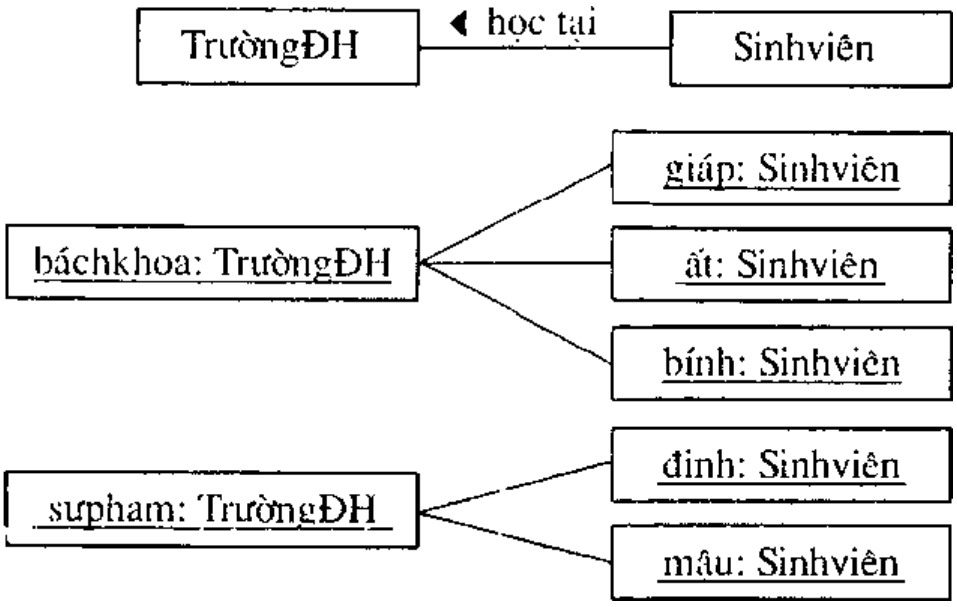
6. LIÊN KẾT

a) Kết nối và liên kết

Giữa các cá thể của hai lớp có thể tồn tại những sự ghép cặp, phản ánh một mối liên hệ nào đó trên thực tế. Gọi đó là một *kết nối* (link). Chẳng hạn kết nối vợ - chồng, kết nối thầy - trò, kết nối xe máy - chủ xe, kết nối khách hàng - hóa đơn v.v... Tập hợp những kết nối cùng loại (cùng ý nghĩa) giữa cá thể của hai lớp tạo thành một mối liên quan giữa hai lớp đó, gọi là một *liên kết* (association). Theo nghĩa đó thì đây chính là một quan hệ (hiểu theo nghĩa toán học) giữa hai tập hợp (là hai lớp).

Liên kết giữa hai lớp được biểu diễn bởi một đường thẳng hay gấp khúc liền nét nối hai lớp, có thể mang theo tên của liên kết. Tên liên kết thường là một động từ với chữ cái đầu viết thường. Nghĩa của động từ thường chỉ đúng về một phía, cho nên ta thường gắn vào tên đó một tam giác đặc ◂ hay ▸ để chỉ hướng áp dụng.

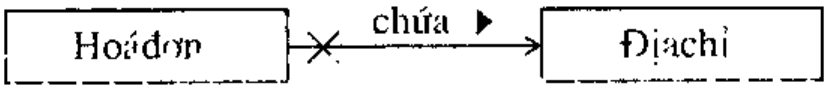
Trong thí dụ ở Hình IV.7 ta có một liên kết giữa lớp TrườngĐH và lớp Sinhviên. Liên kết đó được diễn giải thành các kết nối cụ thể giữa các đối tượng của hai lớp.



Hình IV.7. Một liên kết và các kết nối của nó

b) Sự lưu hành

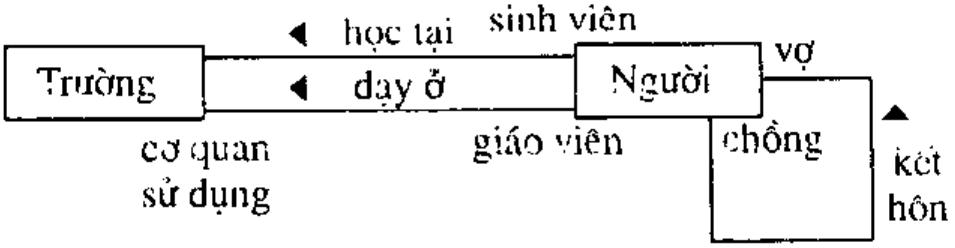
Sự tồn tại một kết nối giữa hai đối tượng (của hai lớp trong một liên kết) có nghĩa là các đối tượng đó "biết nhau". Do đó từ một đối tượng này, nhờ có kết nối mà ta có thể tìm đến được đối tượng kia. Gọi đó là sự *lưu hành* (navigation) trên một liên kết. Nói chung thì sự lưu hành có thể thực hiện theo cả hai chiều (tức là cả ở hai đầu) trên một liên kết. Song cũng có khi sự lưu hành trên liên kết chỉ hạn chế theo một chiều. Để chỉ rõ sự hạn chế này, ta thêm một mũi tên vào đầu được lưu hành và dấu chéo vào đầu không được lưu hành (Hình IV.8). Khi không có mũi tên và dấu chéo thì ta xem là sự lưu hành không được chỉ rõ.



Hình IV.8. Liên kết có hướng

c) Vai trò

Trong một liên kết giữa hai lớp, thì mỗi lớp đóng một *vai trò* (role) khác nhau. Tên vai trò, với chữ cái đầu tiên viết thường, có thể được viết thêm vào mỗi đầu của liên kết, như trong Hình IV.9 (vì vậy mà vai trò cũng được gọi là *tên của một đầu liên kết).* Về ý nghĩa, thì một vai trò biểu diễn cho một tập hợp đối tượng.

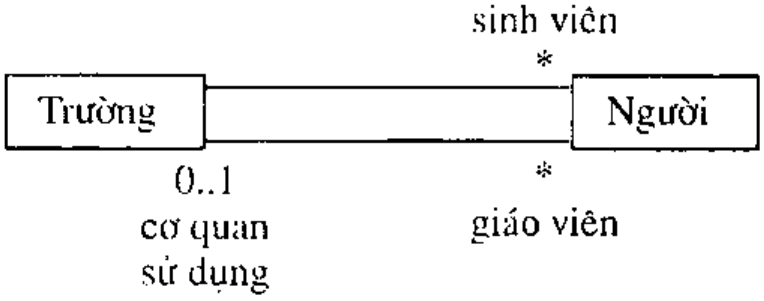


Hình IV.9. Các vai trò trong liên kết

d) Cơ số

Mỗi đầu của liên kết còn có thể chua thêm một *cơ số* (multiplicity), cho biết số cá thể (tối thiểu và tối đa) của đầu đó tham gia liên kết với một cá thể ở đầu kia (xem Hình IV.10). Các giá trị của cơ số thường dùng là:

1	một và chỉ một
0..1	không hay một
m..n	từ m tới n (m và n là các số tự nhiên)
0..* hay *	từ 0 tới nhiều
1..*	một tới nhiều

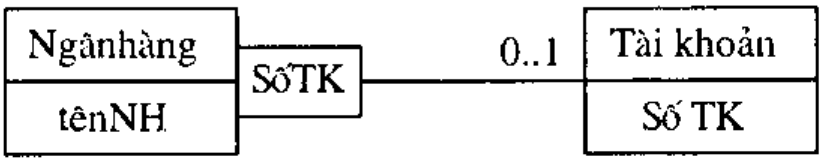


Hình IV.10. Các cơ số trong liên kết

Chú thích: Cơ số với dạng viết như trên còn có thể sử dụng ở nhiều chỗ khác nữa. Chẳng hạn viết trong ngăn tên của một lớp để trỏ số cá thể của lớp đó hoặc trong một thuộc tính để trỏ số các giá trị có thể gán cho thuộc tính đó. Khi cơ số viết cạnh một xâu ký tự, thì nó được đặt giữa hai dấu ngoặc vuông.

đ) Hạn định

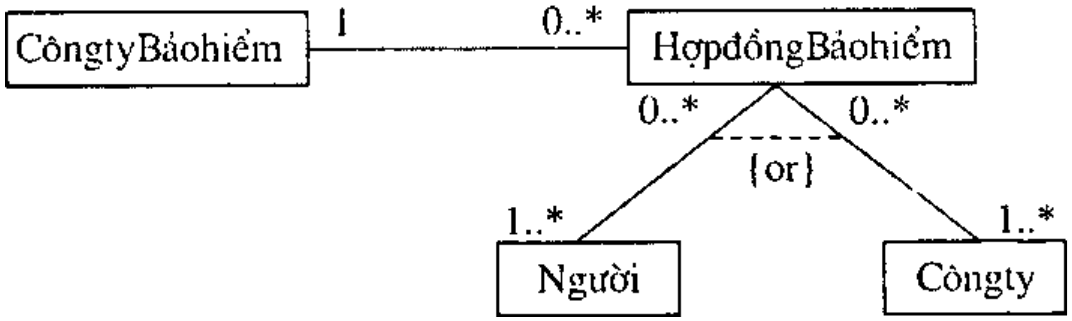
Vấn đề luôn luôn cần giải quyết khi mô hình hóa các liên kết, đó là vấn đề tìm kiếm: cho trước một đối tượng ở một đầu của liên kết, hãy tìm một đối tượng hay tập hợp các đối tượng kết nối với nó ở đầu kia. Để giảm bớt số lượng các đối tượng tìm được, ta có thể hạn chế khu vực tìm kiếm theo (giá trị của) một thuộc tính nào đó. Thuộc tính này (hay cũng có thể là một số thuộc tính) được gọi là một *hạn định* (qualifier) được ghi trong một hộp nhỏ gắn vào đầu mút của liên kết, phía lớp xuất phát của sự lưu hành (xem Hình IV.11). Như vậy hạn định được áp dụng cho các liên kết 1-nhiều hay nhiều-nhiều để giảm từ nhiều xuống 1 hay 0..1, hoặc cũng có thể giảm từ nhiều xuống nhiều, nhưng ít hơn.



Hình IV.11. Liên kết được hạn định

e) Liên kết Hoặc

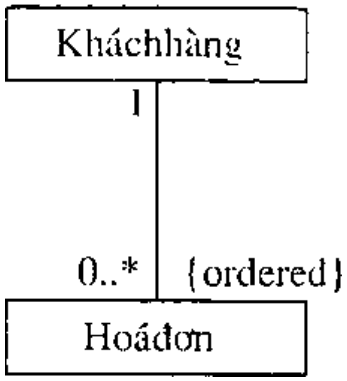
Trong một số mô hình không phải mọi tổ hợp các kết nối của các liên kết đều đúng đắn. Chẳng hạn một cá nhân và một công ty không được cùng kết nối với một hợp đồng bảo hiểm. Bấy giờ ta dùng *liên kết hoặc* (or-association) để diễn tả, thể hiện bằng một đường đứt nét với xâu tính chất {or} vẽ đi ngang qua các liên kết, với nghĩa là: một đối tượng (ở đây là hợp đồng bảo hiểm) chỉ được phép tham gia nhiều nhất vào một trong những liên kết được nối, ở một thời điểm (xem Hình IV.12).



Hình IV.12. Liên kết Hoặc

g) Liên kết có thứ tự

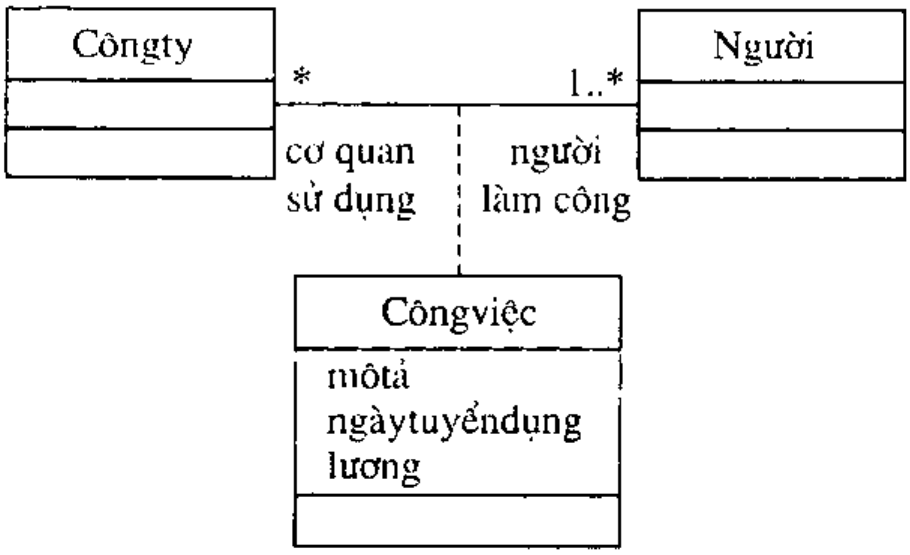
Một cách mặc định, thì các kết nối (với một đối tượng) trong cùng một liên kết là không có thứ tự. Song cũng có khi ta muốn các kết nối được sắp theo một thứ tự nào đó. Gọi đó là *một liên kết có thứ tự* (ordered association). Bấy giờ ta gắn thêm xâu tính chất {ordered} bên cạnh liên kết về phía lớp các đối tượng được sắp. Còn muốn rõ sắp như thế nào, thì có thể thêm một xâu tính chất nữa về cách sắp, chẳng hạn {sắp theo thứ tự thời gian} (xem Hình IV.13).



Hình IV.13. Liên kết có thứ tự

h) Lớp liên kết

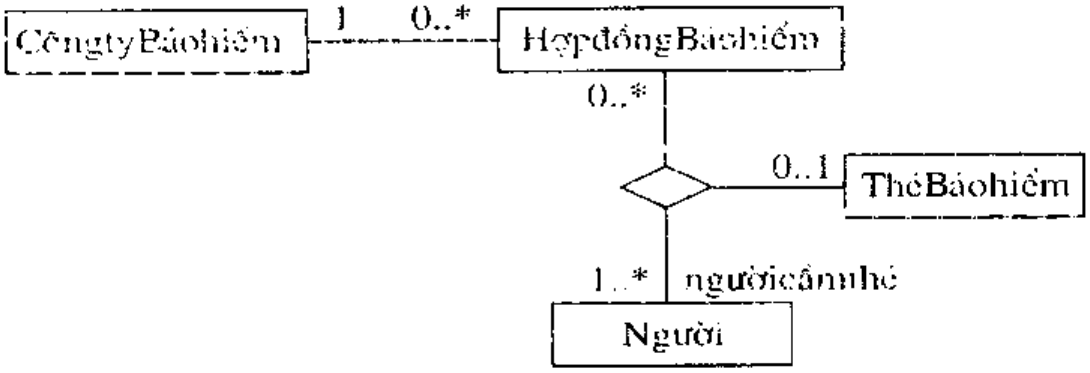
Bản thân các liên kết cũng có thể cần có các tính chất đặc trưng cho nó. Chẳng hạn trong liên kết cơ quan sử dụng và người làm công, thì ta có thể phải mô tả thêm về công việc được giao và đó chính là sự mô tả các đặc điểm của liên kết (chứ không phải của các lớp tham gia liên kết). UML thể hiện điều này bằng các *lớp liên kết* (association class). Lớp liên kết là một lớp như bình thường, nghĩa là có thể có các thuộc tính, các thao tác và tham gia liên kết với những lớp khác, song ngăn tên có thể có tên hay để trống tùy ý, và nó được gắn với liên kết mô tả bởi một đường đứt nét (Hình IV.14).



Hình IV.14. Lớp liên kết

i) Liên kết nhiều bên

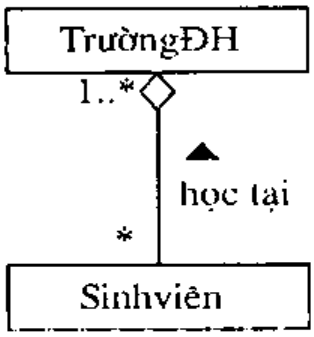
Có thể có liên kết giữa nhiều hơn hai lớp, theo nghĩa của quan hệ nhiều ngôi (một kết nối ở đây là một bộ - n). Lúc đó liên kết được diễn tả bởi một hình thoi nhỏ, nối bằng các đường liền nét với các lớp tham gia (Hình IV.15) cũng có thể có một lớp liên kết cho nó (nối với hình thoi bằng một đường đứt nét).



Hình IV.15. Một liên kết ba bên

k) Kết nhập và hợp thành

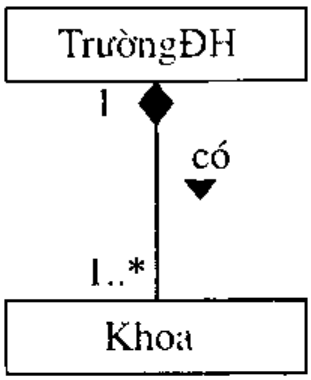
Thông thường thì trong một liên kết, hai bên tham gia được xem là bình đẳng, không bên nào được nhấn mạnh hơn bên nào. Tuy nhiên cũng có lúc ta muốn mô hình hóa mối quan hệ "toàn thể/bộ phận" giữa một lớp các vật thể lớn (cái "toàn thể") với một lớp các vật thể bé (các "bộ phận") bao gồm trong chúng. Đó là một loại liên kết đặc biệt, được gọi là *kết nhập* (aggregation) và được biểu diễn bằng cách gắn thêm một hình thoi nhỏ vào một đầu của liên kết, về phía cái toàn thể (Hình IV.16).



Hình IV.16. Kết nhập

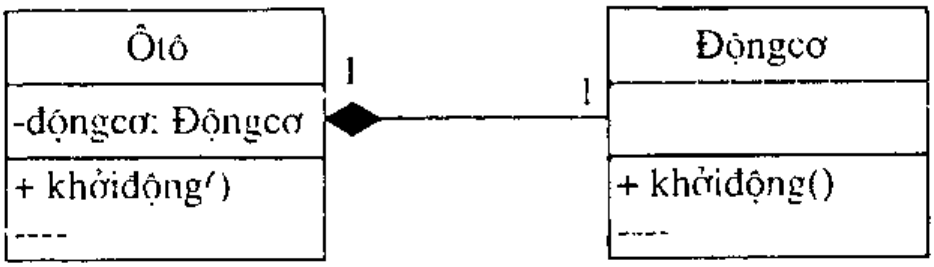
Trong thí dụ Trường ĐH-Sinh viên, thì một sinh viên có thể học tại 1..* trường. Điều đó có nghĩa là trong một kết nhập, thì khi cho một bộ phận, không nhất thiết xác định cái toàn thể duy nhất chứa nó. Hơn nữa không nhất thiết phải có sự gắn kết thời gian sống giữa toàn thể và bộ phận.

Một *hợp thành* (composition) là một loại kết nhập đặc biệt với quan hệ sở hữu mạnh hơn, trong đó một bộ phận chỉ thuộc vào một cái toàn thể duy nhất và cái toàn thể có trách nhiệm tạo lập và hủy bỏ cái bộ phận. Như vậy khi cái toàn thể bị hủy bỏ thì cái bộ phận cũng buộc phải hủy bỏ theo. Hợp thành được biểu diễn bằng cách thay hình thoi rỗng trong kết nhập bởi hình thoi đặc (Hình IV.17).



Hình IV.17. Hợp thành

Một trường hợp đặc biệt của hợp thành là khi thuộc tính của một lớp lại là đối tượng thuộc một lớp khác. Bấy giờ mối liên quan giữa hai lớp đó cũng chính là hợp thành (Hình IV.18).



Hình IV.18. Thuộc tính xem là thành phần trong hợp thành

7. Các khuôn dạng lớp

Khuôn dạng lớp (template class) là một lớp đã tham số hóa. Khái niệm khuôn dạng lớp có trong C++ và Ada, nhưng không có trong

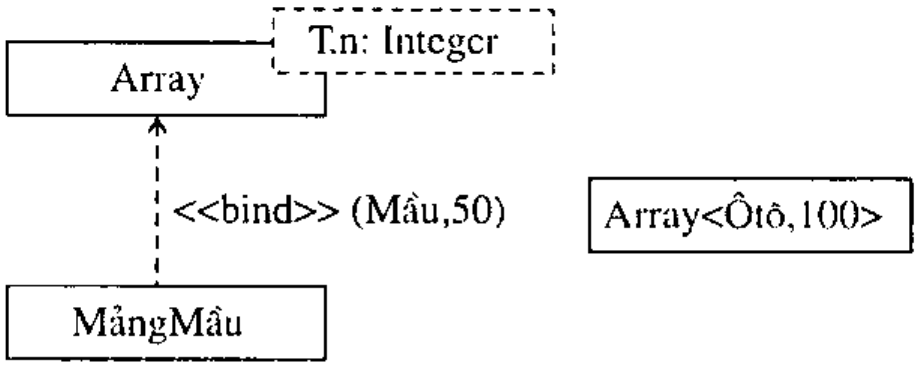
Java. Đây thực ra là một họ các lớp và nó chỉ trở thành một lớp (một cá thể của khuôn dạng) khi các tham số được gán các giá trị.

Các tham số có thể là các lớp, các đối tượng hoặc cũng có thể là các giá trị thuộc các kiểu nguyên thủy (như Integer, Real, Boolean...).

UML biểu diễn khuôn dạng lớp bằng một hình chữ nhật như lớp, song có gắn thêm một hình chữ nhật nhỏ, viền đứt nét, trong đó có chứa danh sách các tham số (các tham số cách nhau bởi dấu phẩy). Kiểu mỗi tham số viết sau dấu ':', song tham số là lớp, thì kiểu của nó (class) có thể chỉ ra hay không tùy ý. Chẳng hạn danh sách [T:class, n: Integer] có thể viết [T,n: Integer], và ở đây T là một lớp, n là một số nguyên.

Có hai cách để biểu diễn một lớp cá biệt của một khuôn dạng lớp:

- Tên lớp là tên khuôn dạng kèm danh sách các giá trị của các tham số (các tham số thực sự);
- Tên lớp là khác tên khuôn dạng, nhưng lớp đó được nối với khuôn dạng bằng một liên quan phụ thuộc kèm với từ khóa <<bind>> theo sau là danh sách các tham số thực sự (Hình IV.19).



Hình IV.19. Khuôn dạng Array có các tham số T (một lớp) và n (một số nguyên). Nó có hai cá thể vẽ theo hai cách khác nhau.

8. CÁC GIAO DIỆN

Một hệ thống lớn nếu muốn có cấu trúc tốt thường phải cắt thành nhiều thành phần độc lập về mặt cài đặt với nhau. Mối liên quan giữa các thành phần lúc đó chỉ còn là mối liên quan cung ứng/sử dụng (có tính chất hợp đồng): bên cung ứng hỗ trợ cho bên sử dụng một số dịch vụ; hợp đồng này là không thay đổi kể cả khi nội bộ mỗi bên có sự thay đổi.

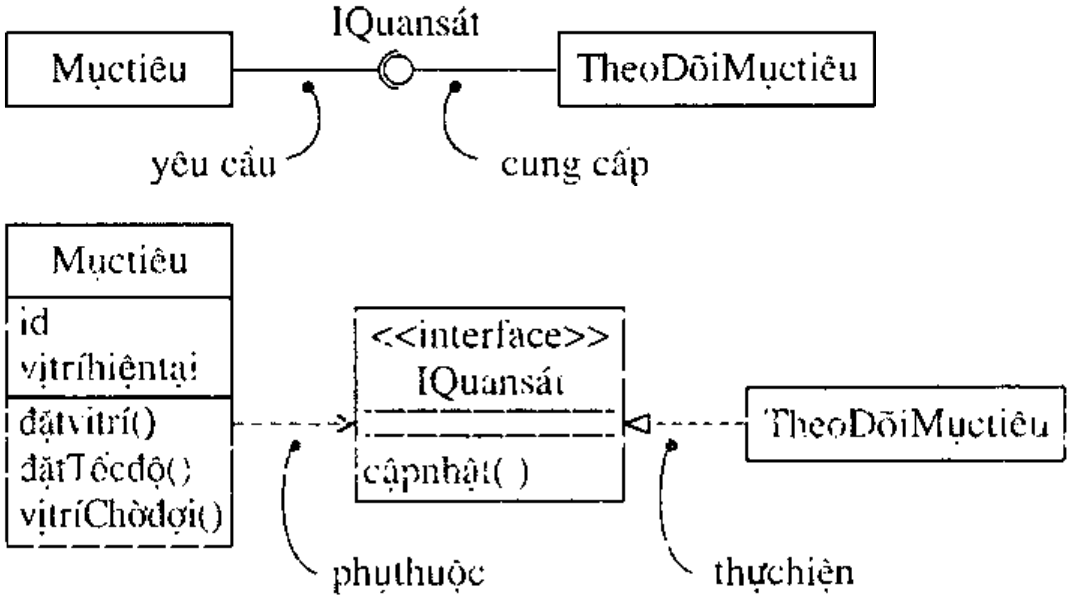
Trong UML để thực hiện một dường phân chia như vậy (còn gọi là một *đường khâu* - seam), ta dùng các giao diện, một khái niệm cũng có trong Java và CORBA IDL.

Một *giao diện* (interface) biểu diễn cho khai báo của một tuyển tập các thao tác, cho thấy khả năng dịch vụ được đưa ra từ một lớp hay một thành phần. Giao diện là không có cấu trúc (có nghĩa là nó không có thuộc tính) và các thao tác của nó là không có cài đặt (chưa có phương thức). Các cài đặt này thuộc trách nhiệm của lớp hay thành phần cung cấp giao diện đó.

Một lớp hay thành phần có thể cung cấp nhiều giao diện. Mỗi giao diện đó thể hiện một vai trò, ví như một Người có thể có nhiều hành vi giao tiếp khác nhau, tuỳ theo vai trò như là mẹ, giám đốc, giáo viên hay hoạ sỹ v.v... Mặt khác thì một giao diện lại có thể được thực hiện bởi nhiều lớp hay thành phần khác nhau.

Có hai cách biểu diễn giao diện:

- Giao diện biểu diễn bởi một hình tròn nhỏ có mang tên (tên giao diện thường được gắn thêm tiền tố I). Bấy giờ giao diện nối với lớp hay thành phần cung cấp nó bằng một nét liền và nối với lớp hay thành phần yêu cầu nó bằng một cái ngàm (Hình IV.20).
- Giao diện cũng có thể biểu diễn bằng một hình chữ nhật chia ngăn, như lớp, để có thể diễn tả nội dung của nó ở bên trong. Bấy giờ giao diện được nối với lớp hay thành phần cung cấp nó bằng một mũi tên đứt nét có đầu tam giác (gọi là mối liên quan thực hiện), và được nối với lớp hay thành phần yêu cầu nó bằng một mũi tên đứt nét đầu mở (mối liên quan phụ thuộc), như trên Hình IV.20.



Hình IV.20. Hai cách biểu diễn giao diện

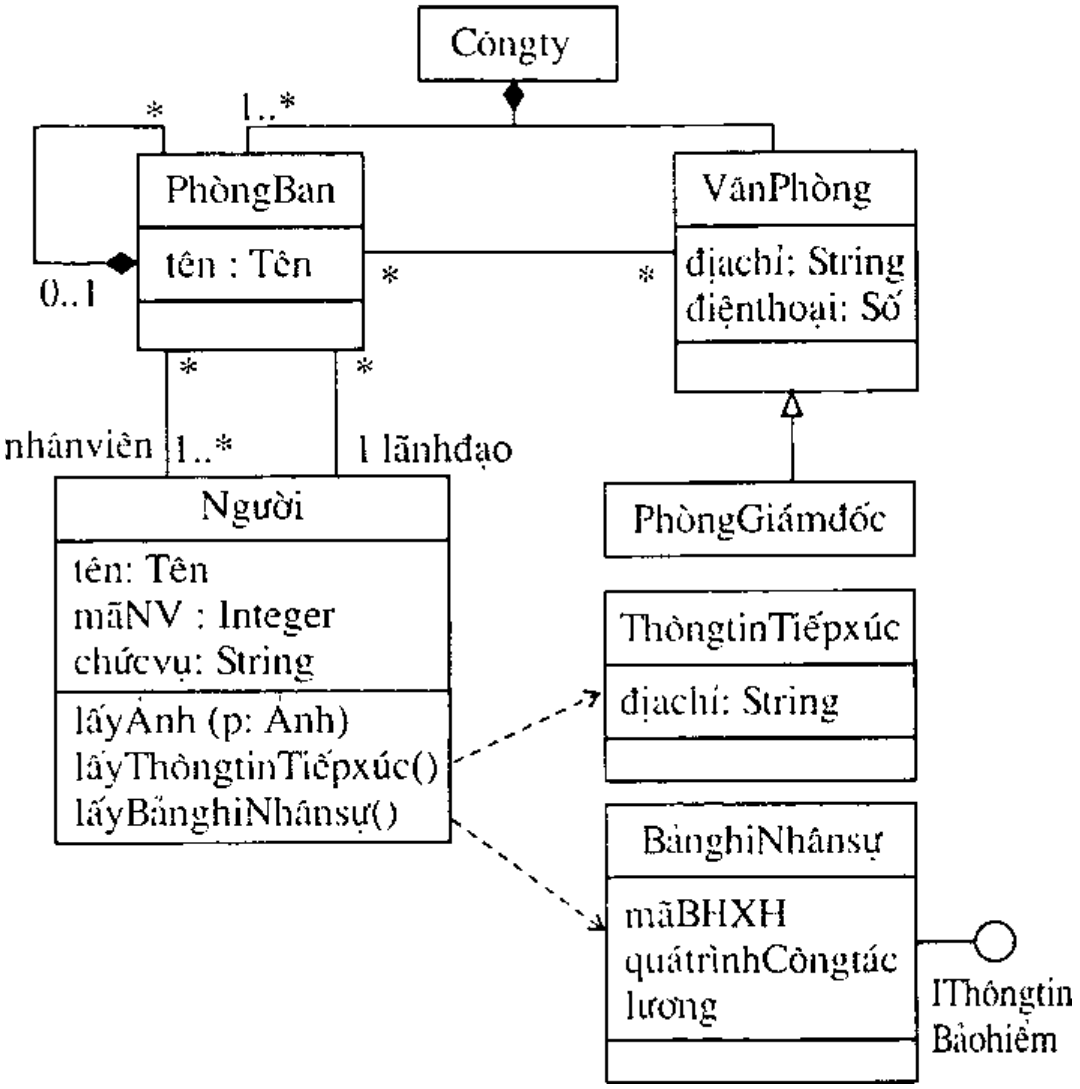
9. BIỂU ĐỒ LỚP VÀ BIỂU ĐỒ ĐỐI TƯỢNG

a) Biểu đồ lớp

Biểu đồ lớp là một biểu đồ phô bày một tập hợp các lớp, các giao diện cùng với các mối liên quan có thể có giữa chúng, như là liên kết, kết nhập, hợp thành, khái quát hoá, phụ thuộc và thực hiện.

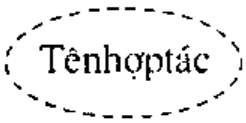
Biểu đồ lớp được dùng để mô hình hoá cấu trúc tĩnh của hệ thống (hay của một phần hệ thống). Nó bao gồm mọi phần tử khai báo, cho nên nó là cái nền (cái nâng đỡ) cho các hoạt động chức năng của hệ thống. Biểu đồ lớp thường được sử dụng theo ba mục đích:

- *Mô hình hoá từ vựng của hệ thống:* Những sự vật mà người dùng và người xây dựng hệ thống quan tâm đến gọi là từ vựng của hệ thống. Chúng sẽ được trừu xuất thành các lớp và được diễn tả trong một biểu đồ lớp (xem Hình IV.21). Vấn đề này sẽ được xem xét ở §2.



Hình IV.21. Biểu đồ lớp diễn tả từ vựng hệ thống

- *Mô hình hoá cấu trúc tĩnh của một hợp tác:* Ta gọi *hợp tác* (collaboration) là một quần thể các lớp, giao diện và các phần tử khác làm việc cùng nhau để thực hiện một hành vi chung. UML diễn tả khái lược một hợp tác bằng một hình elip viền bằng nét đứt, có mang tên hợp tác (Hình IV.22).

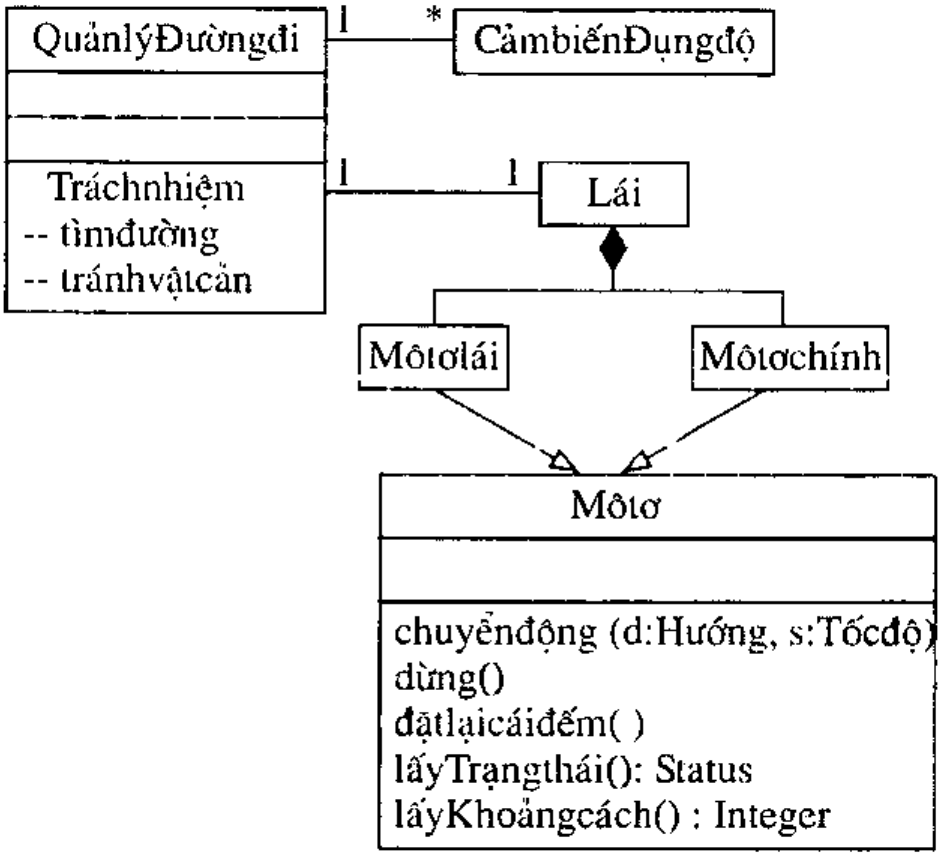


Hình IV.22. Biểu diễn của một hợp tác

Đi sâu vào bên trong thì:

- Người ta biểu diễn *vai trò* của các lớp khi tham gia hợp tác bởi một biểu đồ cấu trúc đa hợp (§1.10 Chương IV);
- Người ta diễn tả *khía cạnh động* của hợp tác bằng một biểu đồ tương tác (§1, Chương V);
- Người ta biểu diễn *khía cạnh tĩnh* của hợp tác bằng một biểu đồ lớp.

Ở §3 ta sẽ đề cập việc mô hình hoá cấu trúc tĩnh của một hợp tác khi nó là một ca sử dụng. Còn ở đây, ta hãy xét một thí dụ về nghiên cứu rô bốt tự động. Rô bốt thể hiện bởi nhiều lớp. Song nếu ta chỉ tập trung vào cơ chế di chuyển rô bốt dọc một đường đi, thì ta có một hợp tác, cấu trúc tĩnh của nó được diễn tả trong biểu đồ lớp cho ở Hình IV.23.



Hình IV.23. Cấu trúc tĩnh của một hợp tác

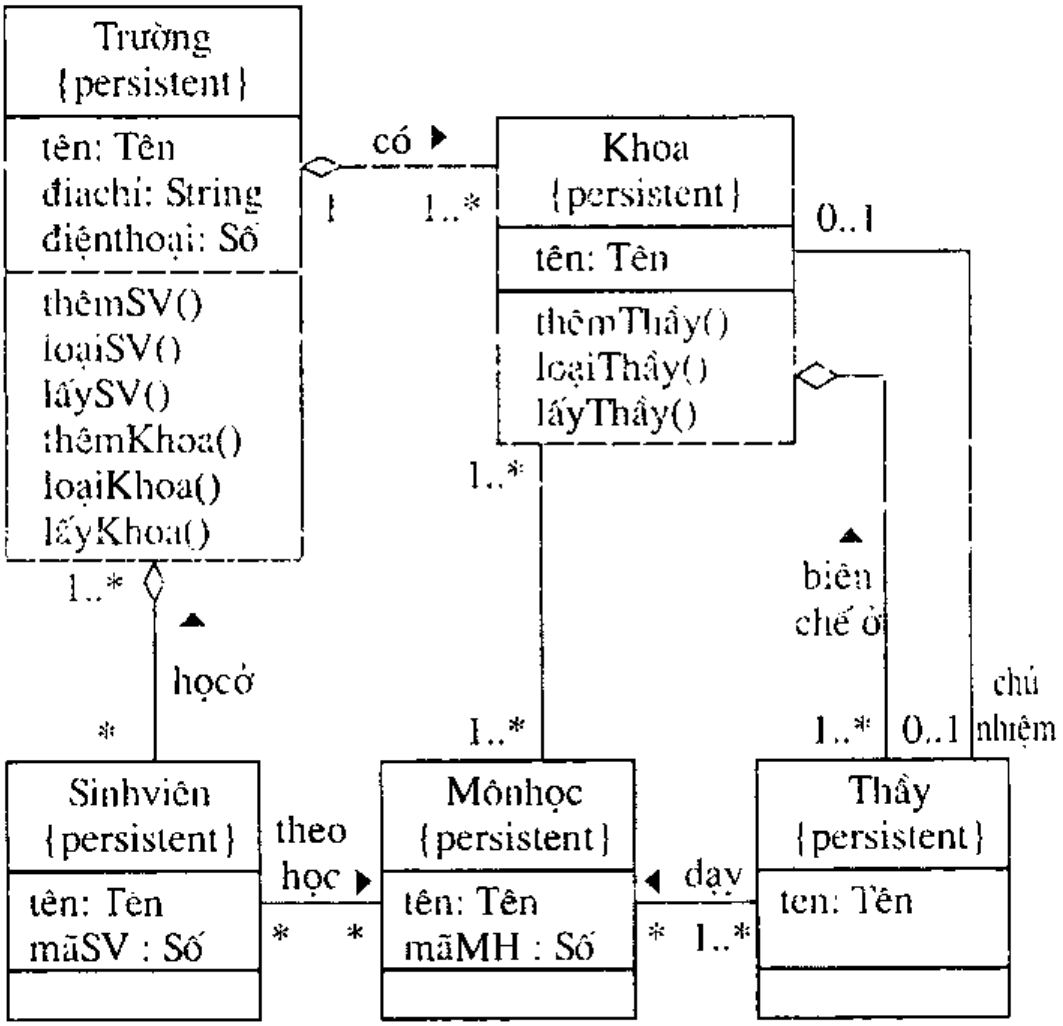
- *Mô hình hoá lược đồ lôgic của cơ sở dữ liệu:*

Có những đối tượng cần phải được lưu vào cơ sở dữ liệu (CSDL) khi chương trình tạo ra chúng đã kết thúc, để rồi sẽ được tìm lại sau này.

CSDL được dùng có thể là CSDL quan hệ, CSDL hướng đối tượng hay CSDL lai đối tượng/quan hệ. Tuy nhiên CSDL quan hệ vẫn là thông dụng, bởi tính ổn định và giá cả của nó. Tiếc thay lại không có ánh xạ 1-1 giữa các đối tượng và các bảng trong CSDL quan hệ. Bởi vậy mà cần có một bước đệm: Lập một biểu đồ lớp phản ánh các lớp mà đối tượng của chúng cần lưu vào CSDL với một số biến đổi nhằm thích ứng với CSDL, như:

- Các lớp đều mang giá trị gắn nhãn {persistent}.
- Các thuộc tính đều có kiểu nguyên thuỷ. Thuộc tính có kiểu lớp phải chuyển thành mối quan hệ kết nhập.
- Các thao tác đều là các thao tác liên quan đến truy cập dữ liệu hay kiểm soát sự toàn vẹn dữ liệu. Các thao tác về nghiệp vụ (gắn liền với ứng dụng) vẫn để ở lớp gốc và vẫn nằm ở tầng ứng dụng (trong kiến trúc phân tầng khách hàng/dịch vụ).
- Các yếu tố gây khó khăn cho việc thiết kế CSDL vật lý, như liên kết đệ quy, liên kết 1-1, liên kết nhiều ngôi phải được diễn tả lại theo những hình thức phù hợp.

Hình IV.24 cho một thí dụ về một lược đồ lôgic của CSDL.

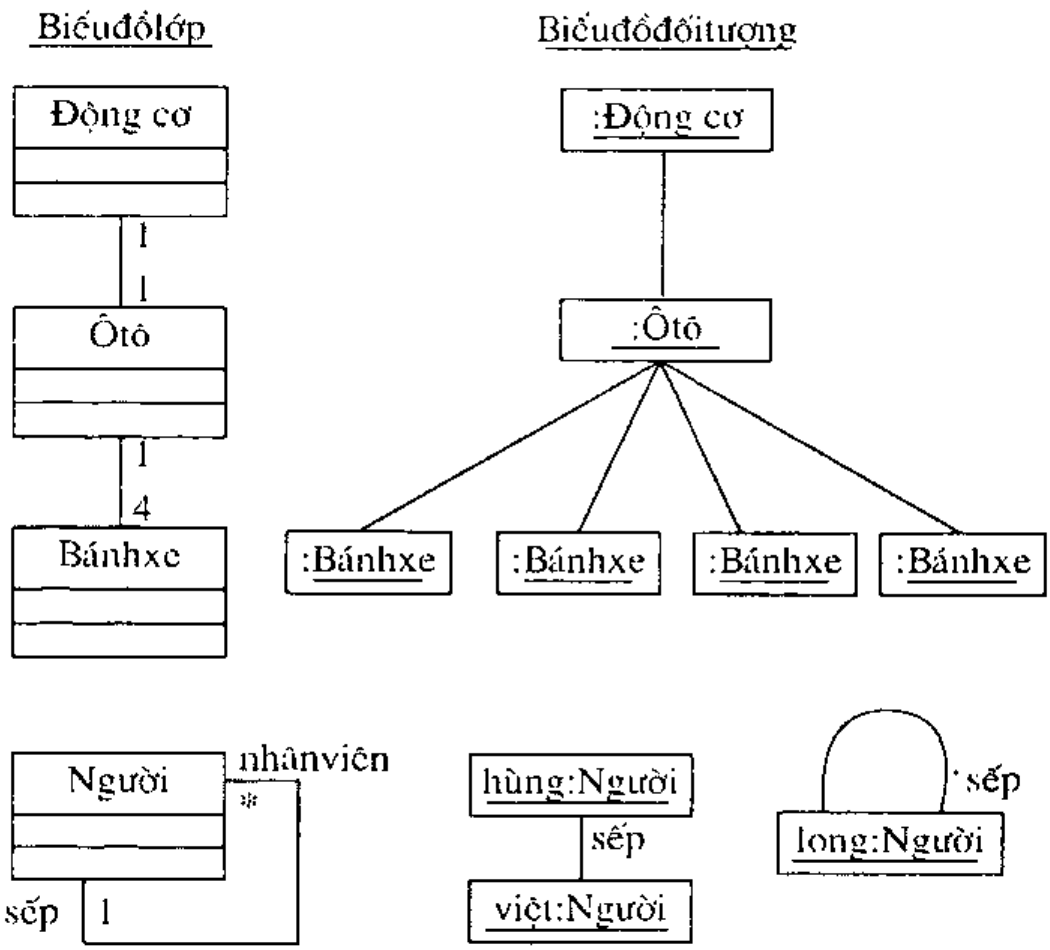


Hình IV.24. Một lược đồ lôgic của CSDL

b) Biểu đồ đối tượng

Biểu đồ đối tượng cũng có mục đích diễn tả cấu trúc tĩnh như biểu đồ lớp, song một cách cụ thể trong một tình huống hay tại một thời điểm nào đó (chẳng hạn trước hay sau một tương tác). Trong biểu đồ đối tượng như vậy chỉ có mặt các đối tượng thay vì các lớp, có mặt các kết nối thay vì các liên kết và phải có sự tương ứng giữa các lớp và đối tượng, liên kết và kết nối.

Hình IV.35 cho hai biểu đồ lớp và hai biểu đồ đối tượng tương ứng với chúng.



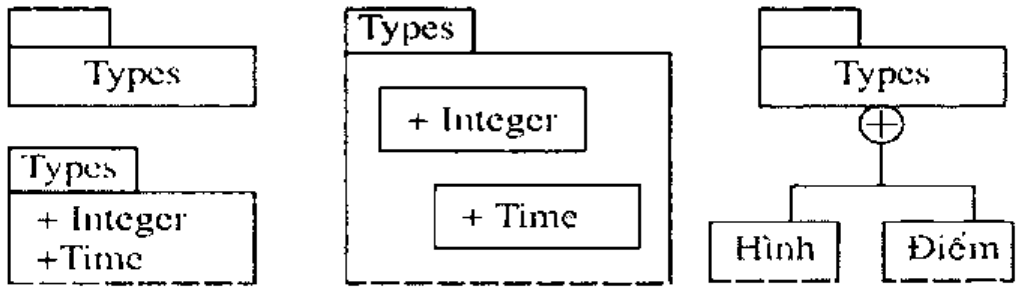
Hình IV.25. Các biểu đồ lớp và các biểu đồ đối tượng tương ứng

10. CÁC GÓI VÀ BIỂU ĐỒ GÓI

a) Gói

Gói (package) là một cơ chế để gom nhiều phần tử vào một nhóm. Các phần tử được đóng gói ở đây có thể là các lớp, các giao diện, các thành phần, các hợp tác, các ca sử dụng, các nút, các biểu đồ và cũng có thể là các gói khác.

Gói được biểu diễn bằng một hình chữ nhật có quai (như một thư mục). Tên của gói được viết bên trong hình chữ nhật nếu ở đó không ghi rõ các phần tử. Còn khi có ghi các phần tử bên trong thì tên của gói ghi trong quai. Các phần tử bên trong gói, không nhất thiết phải liệt kê đầy đủ, có thể cho dưới dạng văn tự hoặc dưới dạng đồ hoạ và có thể kèm theo các ký hiệu tầm nhìn (+, -). Phần tử cũng có thể vẽ bên ngoài gói và kết nối với gói qua ký hiệu + đặt trong một vòng tròn (xem Hình IV.26).



Hình IV.26. Biểu diễn gói cùng với các phần tử của nó

Các phần tử bên trong gói là sở hữu của gói, nghĩa là tồn tại cùng với gói, và mỗi phần tử chỉ được thuộc vào một gói mà thôi. Các phần tử sở hữu của gói được gọi là nội dung của gói.

Gói tạo thành một không gian đặt tên, nghĩa là bên trong một gói thì các phần tử phải có tên phân biệt, nhưng hai phần tử thuộc hai gói khác nhau thì có thể trùng tên. Từ bên ngoài gói, có thể tham chiếu tới một phần tử nhìn được của gói (có tầm nhìn +, hay không có ký hiệu tầm nhìn), bằng cách dùng tên có hạn định (thường gọi là tên có đường dẫn), chẳng hạn Types::Điểm. Quy tắc này có ngoại lệ, đó là khi có mặt mối liên quan nhập khẩu, lúc đó hạn định là không còn cần thiết nữa (xem dưới).

Có hai mối liên quan thường dùng giữa các gói, đó là nhập khẩu và hoà nhập, mà ta sẽ lần lượt đề cập sau đây.

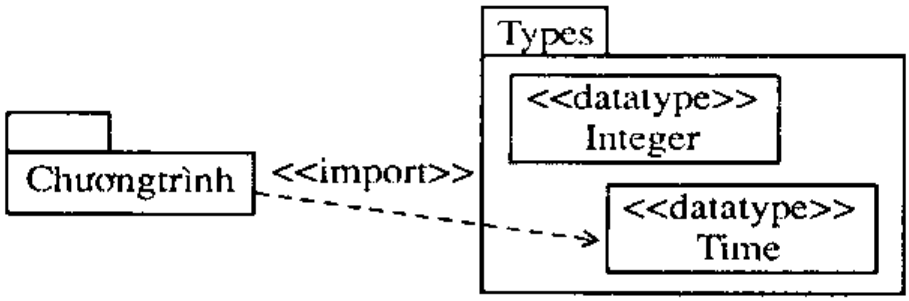
b) Nhập khẩu

Có hai loại nhập khẩu: *Nhập khẩu phần tử* và *nhập khẩu gói*. Nhập khẩu gói chẳng qua là nhập khẩu phần tử đối với tất cả các phần tử thuộc gói. Còn nhập khẩu phần tử là chỉ định một phần tử thuộc gói (phần tử bị nhập khẩu) và cho phép nó được tham chiếu trong một gói khác (gói nhập khẩu) mà không phải dùng hạn định. Nói cách khác đó là sự đưa thêm tên của phần tử (hay các phần tử) bị nhập khẩu vào không gian tên của gói nhập khẩu.

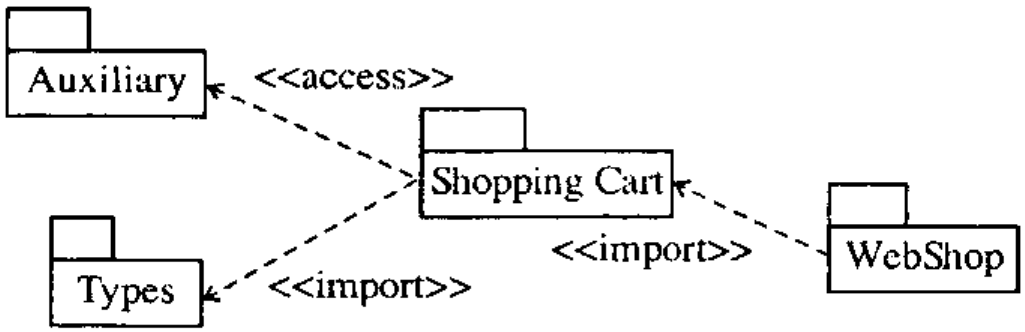
Nếu tên của phần tử bị nhập khẩu trùng với tên của một phần tử thuộc gói nhập khẩu, thì phần tử bị nhập khẩu vẫn phải được tham chiếu với hạn định. Nếu tên của nhiều phần tử bị nhập khẩu từ nhiều nguồn khác nhau mà trùng nhau, thì chúng đều phải được dùng cùng với hạn định. Nếu tên của phần tử bị nhập khẩu trùng với tên của một phần tử thuộc một gói ngoài (bao) của gói nhập khẩu (vốn dĩ vẫn được dùng không có hạn định), thì nay phần tử thuộc gói ngoài này phải

được tham chiếu có hạn định (tức là phần tử bị nhập khẩu che khuất phần tử gói ngoài).

Nhập khẩu (phần tử hay gói) được biểu diễn bằng một mũi tên đứt nét, đầu mở, vẽ từ gói nhập khẩu tới phần tử hay gói bị nhập khẩu (Hình IV.27 và Hình IV.28).



Hình IV.27. Nhập khẩu phần tử



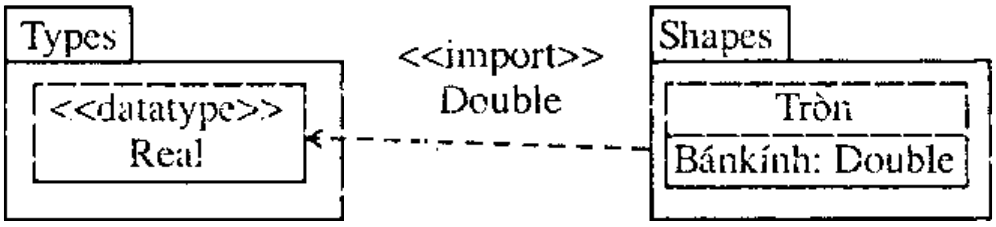
Hình IV.28. Nhập khẩu gói

Khuôn dập cho nhập khẩu có thể là:

<<import>>, nếu phần tử bị nhập khẩu vẫn tiếp tục nhìn thấy được (public) và có thể bị nhập khẩu tiếp. Trên Hình IV.28 thì các phần tử của gói Types được nhập khẩu vào gói Shopping Cart rồi lại được tiếp tục nhập khẩu sang gói WebShop.

<<access>>, nếu phần tử bị nhập khẩu không còn tiếp tục nhìn thấy được nữa (private) và không còn có thể được nhập khẩu tiếp. Trên Hình IV.28 thì các phần tử của gói Auxiliary được nhập khẩu vào gói Shopping Cart, nhưng không được nhập khẩu tiếp sang gói WebShop.

Nếu ta lại muốn dùng một biệt danh (alias) cho phần tử bị nhập khẩu, thì biệt danh đó được viết sau hay dưới từ khoá <<import>> hay <<access>>, như trên Hình IV.29 (biệt danh Double dùng thay Real).



Hình IV.29. Nhập khẩu với biệt danh

c) Hoà nhập

Hoà nhập gói (package) còn gọi *là mở rộng gói* (package extension) là một mối liên quan có hướng giữa hai gói, trong đó nội dung của gói đích (gói bị trỏ tới) được hoà nhập vào nội dung của gói nguồn, thông qua chuyên biệt hoá và định nghĩa lại. Như vậy hoà nhập gói đòi hỏi một tập hợp các biến đổi để chuyển các phần tử của gói đích sang gói nguồn. Mỗi phần tử (lớp hay gói con) có một quy tắc biến đổi riêng.

Nếu phần tử của gói đích là một lớp (hay nói rộng hơn là một loài - xem định nghĩa ở mục tiếp theo), thì nó sẽ được biến đổi thành một lớp (loài) cùng tên trong gói nguồn, ngoại trừ trường hợp ở đó đã có sẵn một lớp (loài) cùng tên. Trong cả hai trường hợp, thì lớp (loài) mới hay cũ đó đều phải thừa kế hay định nghĩa lại các đặc trưng của lớp (loài) cùng tên ở gói đích. Các lớp (loài) cùng tên từ nhiều gói đích khác nhau phải được biến đổi thành một lớp (loài) duy nhất trong gói nguồn với đa thừa kế tới các lớp (loài) nói trên.

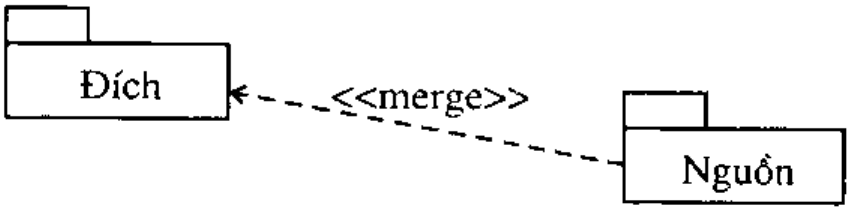
Nếu phần tử của gói đích là một gói con, thì nó sẽ được biến đổi thành một gói con cùng tên trong gói nguồn, ngoại trừ trường hợp ở đó đã có một gói con cùng tên. Trong cả hai trường hợp, thì gói con mới hay cũ này phải chịu mối liên quan hoà nhập tới gói con cùng tên ở gói nguồn. Nhiều gói con cùng tên ở các gói đích khác nhau được biến đổi thành một gói con duy nhất trong gói nguồn, với mối liên quan hoà nhập tới mỗi gói con nói trên.

Một sự nhập khẩu (nhập khẩu phần tử hay nhập khẩu gói) đối với gói đích (trong mối liên quan hoà nhập) sẽ trở thành nhập khẩu đối với gói nguồn. Phần tử nhập khẩu không bị hoà nhập. Tuy nhiên nếu đã có

sẵn một mối liên quan hoà nhập từ gói nguồn tới gói bị nhập khẩu, thì sự nhập khẩu bị sự hoà nhập che khuất, và lúc đó các phần tử nhập khẩu vẫn phải được tham chiếu với tên có hạn định.

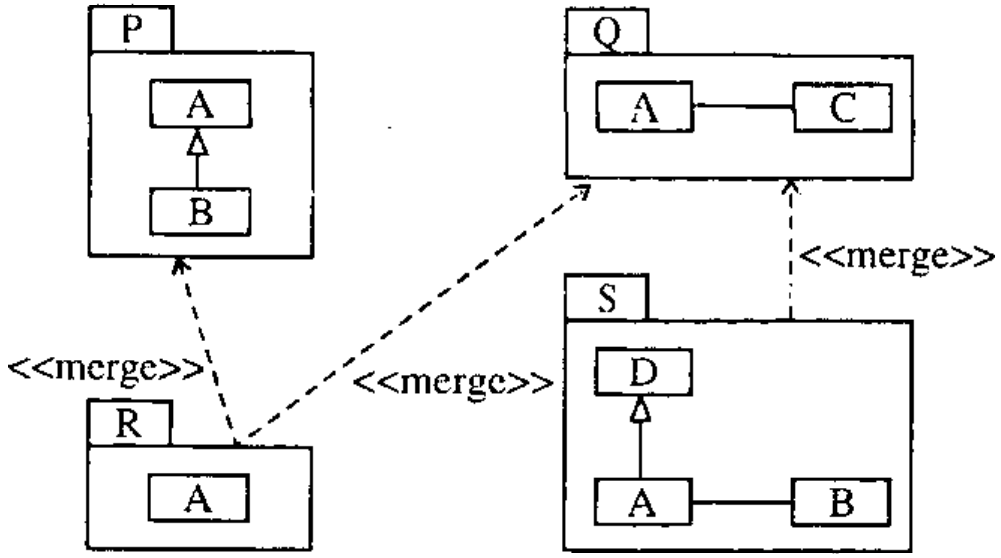
Mối liên quan hoà nhập được ứng dụng mỗi khi ta muốn các phần tử cùng tên và cùng loại trong các gói khác nhau phải được biểu diễn cùng một khái niệm (ở những mức độ khái quát khác nhau). Lưu ý rằng khi một phần tử bị hoà nhập vào một gói khác, thì nó vẫn tồn tại nguyên xi ở gói đích, và vẫn có thể được tham chiếu với tên có hạn định.

Mối liên quan hoà nhập được biểu diễn bằng một mũi tên đứt nét, đầu mở nối từ gói nguồn tới gói đích, có kèm từ khoá <<merge>> (Hình IV.30).



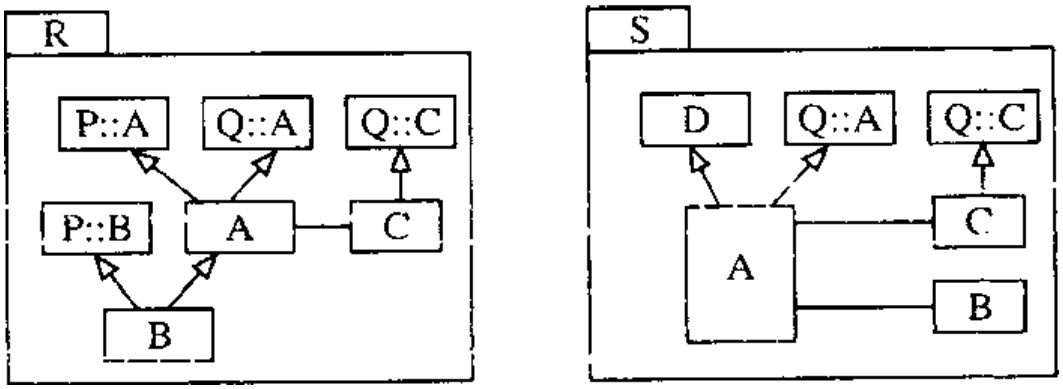
Hình IV.30. Biểu diễn của hoà nhập

Trên Hình IV.31, thì hai gói P và Q bị hoà nhập bởi gói R, còn gói S thì hoà nhập chỉ mỗi một gói Q.



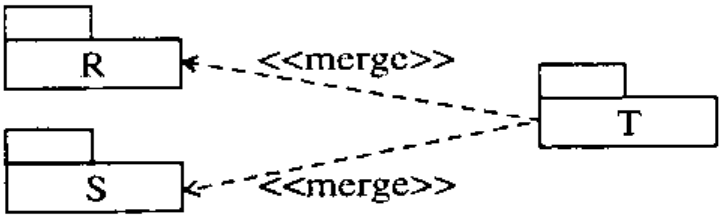
Hình IV.31. Thí dụ về hoà nhập

Các gói R và S được biến đổi (bởi hoà nhập) như Hình IV.32.



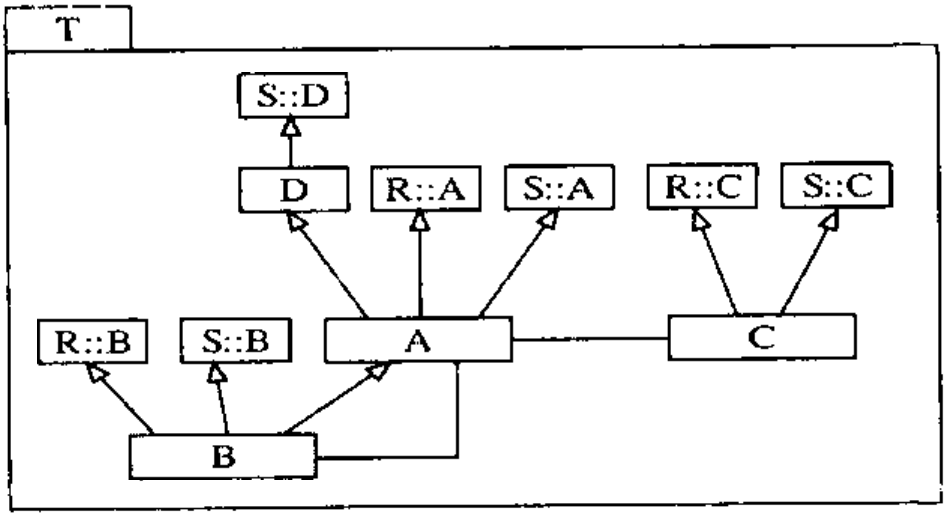
Hình IV.32. Các biến đổi ở gói nguồn

Trên Hình IV.33, ta lại cho thêm một gói T, vốn là một gói rỗng. Nhưng T lại hoà nhập hai gói R và S (đã định nghĩa ở trên).



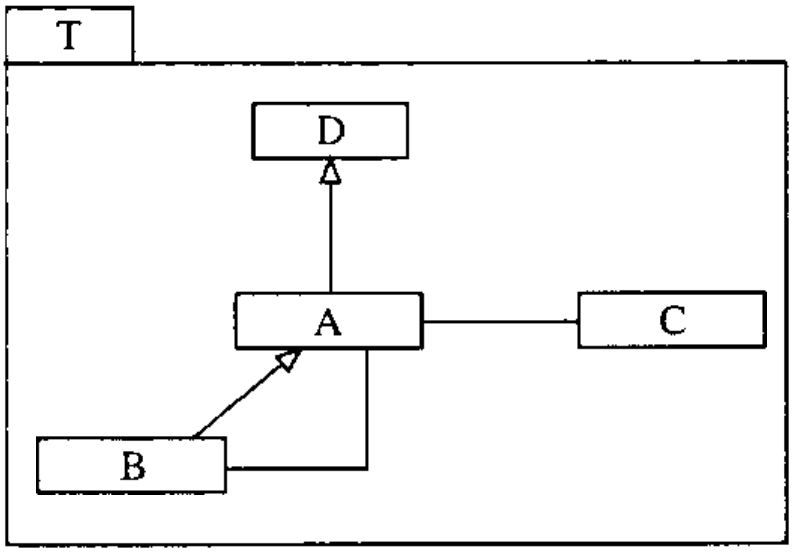
Hình IV.33. Các hoà nhập đưa thêm

Với các hoà nhập đó, thì nội dung của T trở nên như trên Hình IV.34.



Hình IV.34. Kết quả của các hoà nhập đưa thêm

Chú ý rằng việc biểu diễn minh bạch các mối liên quan khái quát hóa giữa các phần tử cùng tên từ gói nguồn đến gói đích là thừa, vì ý nghĩa của chúng đã bao hàm trong sự biến đổi rồi; do đó nội dung của gói T có thể diễn tả lược như trên Hình IV.35 cho quang đãng hơn.



Hình IV.35. Biểu diễn giản lược của gói T

d) Biểu đồ gói

Biểu đồ gói nhằm diễn tả các phần tử của một mô hình được tổ chức thành các gói như thế nào. Các nút của biểu đồ gói là các gói, các đường là các liên quan nhập khẩu hay hoà nhập. Thường dùng gói để gom nhóm các lớp (trong một biểu đồ lớp phức tạp) hoặc gom nhóm các ca sử dụng (trong một biểu đồ ca sử dụng phức tạp).

11. CÁC LOÀI VÀ BIỂU ĐỒ CẤU TRÚC ĐA HỢP

a) Các loài

Trong UML không phải chỉ có lớp mới có cá thể, cấu trúc và hành vi. Còn nhiều phần tử mô hình hoá khác cũng có (toàn bộ hay một phần) các đặc điểm như lớp. Gọi chung các phần tử mô hình đó là các loài.

Loài (classifier) là một sự phân loại các cá thể, nói cách khác đó là một tập hợp các cá thể có đặc điểm chung. Loài là một không gian đặt tên. Loài cũng là một kiểu (type) và có thể chịu sự khái quát hoá (tức là có mối liên quan khái quát hoá tới một loài khác).

Ngoài lớp là một loài thường gặp nhất, ta còn có nhiều loài khác, như là:

- *Giao diện* (interface): là một sưu tập được đặt tên, bao gồm các thao tác diễn tả khả năng dịch vụ của một lớp hay một thành phần.
- *Kiểu dữ liệu* (datatype): là một kiểu mà các giá trị của nó là không có căn cước, bao gồm các kiểu nguyên thuỷ dựng sẵn (như số và xâu) và các kiểu liệt kê (như Boolean).
- *Tín hiệu* (signal): là đặc tả của một kích thích không đồng bộ được truyền đi giữa các cá thể, cho phép kích hoạt một phản ứng ở bên nhận một cách không đồng bộ và không có trả lời. Tín hiệu được định nghĩa một cách độc lập với lớp (loài) đã vận dụng nó.
- *Thành phần* (component): là một bộ phận vật lý và thay thế được của hệ thống, thích ứng và cung cấp sự thực hiện đối với một tập hợp các giao dịch.
- *Nút* (node): là một tài nguyên tính toán lúc chạy, thường có một bộ nhớ và một bộ xử lý. Các đối tượng và các thành phần lúc chạy có thể trú ngụ trên các nút.
- *Ca sử dụng* (use case): là đặc tả của một tập hợp các dãy hành động, bao gồm các biến dị, mà hệ thống thực hiện trong sự tương tác cùng các đối tác của hệ thống.
- *Hệ thống con* (subsystem): là một phần của hệ thống, tạo bởi sự gom nhóm của nhiều phần tử mô hình hoá có liên quan chặt chẽ với nhau (cũng gọi là một gói).

Mặc dù các loài cụ thể nói trên có các biểu diễn khác nhau, song một loài nói chung có thể được biểu diễn bởi một hình chữ nhật nét liền với khuôn dập thích hợp (như <<data tye>>, <<interface>>, ... và riêng với lớp thì không cần khuôn dập). Hình chữ nhật đó cũng có thể chia ngăn nếu loài đó có thuộc tính hay hành vi cần chỉ rõ.

b) Biểu đồ cấu trúc đa hợp

Đây là loại biểu đồ mới được đưa thêm vào từ UML 2.0, nhằm cung cấp một công cụ diễn tả thích hợp với các hệ thống lớn.

Biểu đồ cấu trúc đa hợp (composite structure diagram) là một biểu đồ diễn tả cấu trúc nội tại của một loài (mà chủ yếu là lớp, thành phần, hợp tác). Nó chỉ ra một tập hợp các bộ phận (cũng là những loài) bên

trong của loài đó, kể cả các điểm giao tiếp với phần còn lại của hệ thống, cùng với những cầu nối giữa các bộ phận đó, cho phép chúng trao đổi thông tin với nhau để tạo nên hành vi của loài đó vào một lúc chạy.

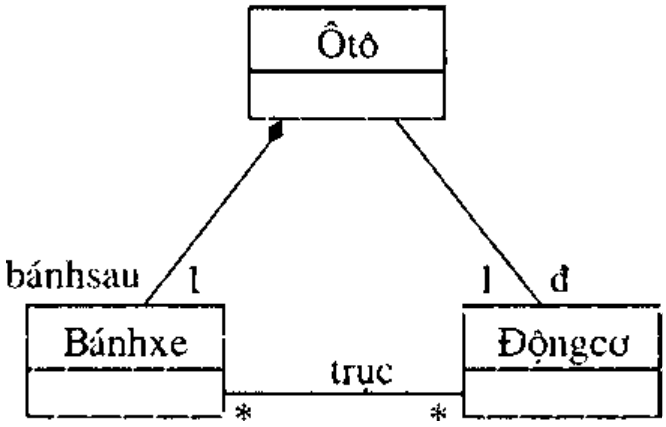
Để hiểu rõ hơn định nghĩa này, ta hãy làm rõ một số khái niệm như: cấu trúc nội tại, bộ phận, cổng (điểm giao tiếp), cầu nối, hợp tác.

- *Cấu trúc nội tại* (internal structure): Đây là nói về cấu trúc của một cá thể của một loài, chỉ ra một số các phần tử của nó được tạo lập vào một thời gian chạy, được kết nối với nhau nhằm hoàn thành một số mục tiêu chung nào đó. Như vậy có nghĩa là sự mô tả chỉ hạn chế trong một hoàn cảnh nhất định (chứ không nhất thiết phải là tổng quát và đầy đủ).
- *Bộ phận* (part): UML, kể cả UML 1.x, có khái niệm *tính chất* (property) với nghĩa là: một giá trị có mang tên trỏ một đặc trưng của một phần tử. Chẳng hạn "tuổi 40" là một tính chất của một người. Nhưng khi ta nói "bánh sau 2 cái" (tính chất của một ô tô), thì cần hiểu ở đây, tên tính chất là một vai trò (bánh sau, nó phân biệt với bánh trước, bánh dự trữ), còn giá trị là 2 đối tượng bánh xe, tức là một tập con của lớp Bánh xe. Lớp Bánh xe gọi là lớp định kiểu cho tính chất nói trên.

UML 2.0 gọi *bộ phận* của một cá thể loài là một tính chất của cá thể đó, tức là một tập con của lớp định kiểu cho tính chất đó, song lớp này phải có mối liên quan hợp thành với loài kể trên, nghĩa là các cá thể của bộ phận phải được tạo lập khi cá thể loài được tạo lập (hoặc sau đó), và bị huỷ bỏ khi cá thể loài bị huỷ bỏ. Một bộ phận được biểu diễn bởi một hình chữ nhật nét liền, trong đó có tênBộphận: TênLớp. Một tính chất không phải là bộ phận (không sở hữu bằng hợp thành) được biểu diễn bởi một hình chữ nhật nét đứt, trong đó có tênTínhchất:TênLớp.

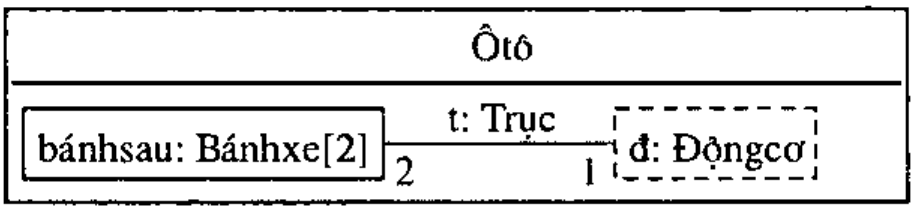
Thí dụ

Giả sử ta có một biểu đồ lớp như trong Hình IV.36



Hình IV.36. Mô tả ô tô

Trong biểu đồ này, ta thấy liên kết giữa Bánh xe và Động cơ là một liên kết nhiều-nhiều, bởi vì đó là một sự diễn tả tổng quát, ở bất cứ nơi đâu, chứ không nhất thiết là ở trong một chiếc ôtô. Tuy nhiên khi mô tả cấu trúc nội tại của một chiếc ôtô, ta sẽ vẽ như trên Hình IV.37. Ta thấy ở đây, bên trong một chiếc ôtô, thì liên kết giữa Bánh xe (với vai trò là bánh sau) với Động cơ lại là liên kết 2-1 (nghĩa là đặt vào hoàn cảnh cụ thể, thì sự ràng buộc sẽ được thắt chặt hơn).

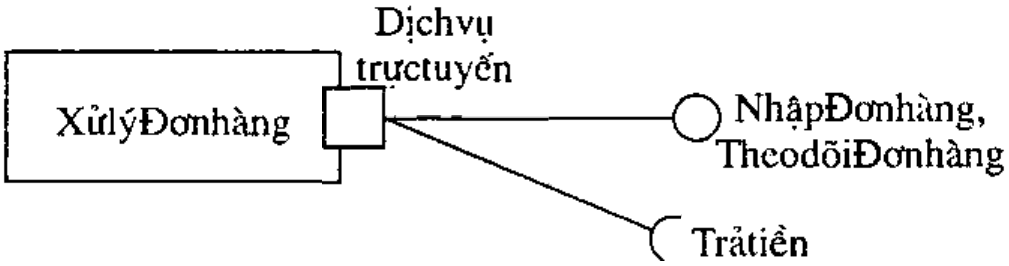


Hình IV.37. Cấu trúc nội tại của ôtô

- *Cổng* (port): Cổng biểu diễn cho một điểm tương tác giữa một loài với môi trường của nó. Một loài có thể có nhiều cổng, cho ta phân biệt các tương tác khác nhau mà loài đó có khả năng trao đổi với môi trường. Nhờ có các cổng mà ta tạo được sự ngăn cách rõ rệt giữa một loài với môi trường, cho phép loài đó có khả năng tái sử dụng ở bất cứ môi trường nào thích ứng với những ràng buộc áp đặt tại các cổng của nó.

Một cổng có thể gắn với một hay nhiều giao diện (giao diện yêu cầu hoặc giao diện cung ứng), thể hiện các yêu cầu hay các dịch vụ của loài đối với môi trường khi tương tác qua cổng đó. Cổng được biểu

diễn bằng một hình vuông nhỏ vẽ trên đường biên của loài, có thể có tên cổng viết bên cạnh như trên Hình IV.38.

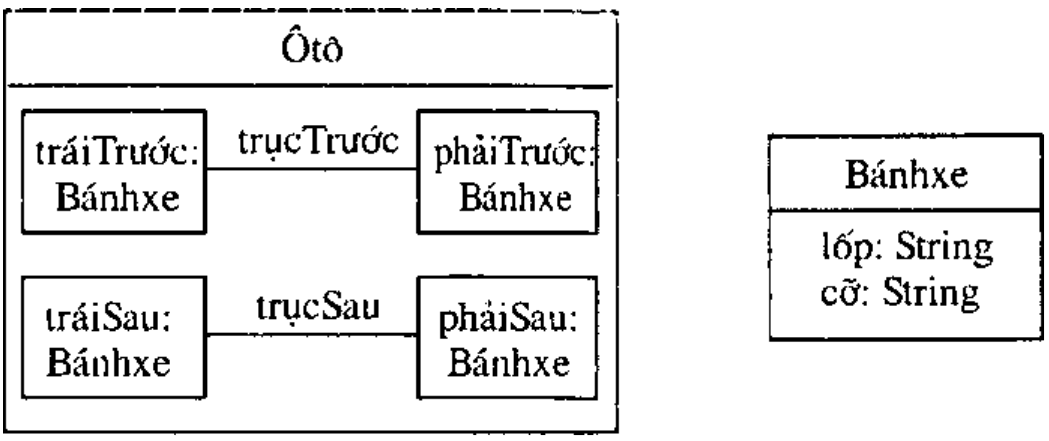


Hình IV.38. Cổng và giao diện

- *Cầu nối* (connector): Là một kết nối cho phép liên lạc giữa hai hay nhiều cá thể. Kết nối đó có thể là một cá thể của một liên kết, hoặc cũng có thể là một khả năng liên lạc do biết được định danh (chẳng hạn qua truyền tham số, hoặc bởi sự tạo lập biến khi chạy,...). Cầu nối có thể thực hiện một cách đơn giản như là bằng một con trỏ, mà cũng có thể thực hiện một cách phức tạp như một đường truyền thông trễn mạng. Cầu nối được biểu diễn theo ký pháp của liên kết, nghĩa là bằng một nét liền với các tô điểm như vai trò, cơ số ở hai đầu mút. Cầu nối có thể mang tên, theo cú pháp:

{{[tên] ':' tênLớp} | tên}

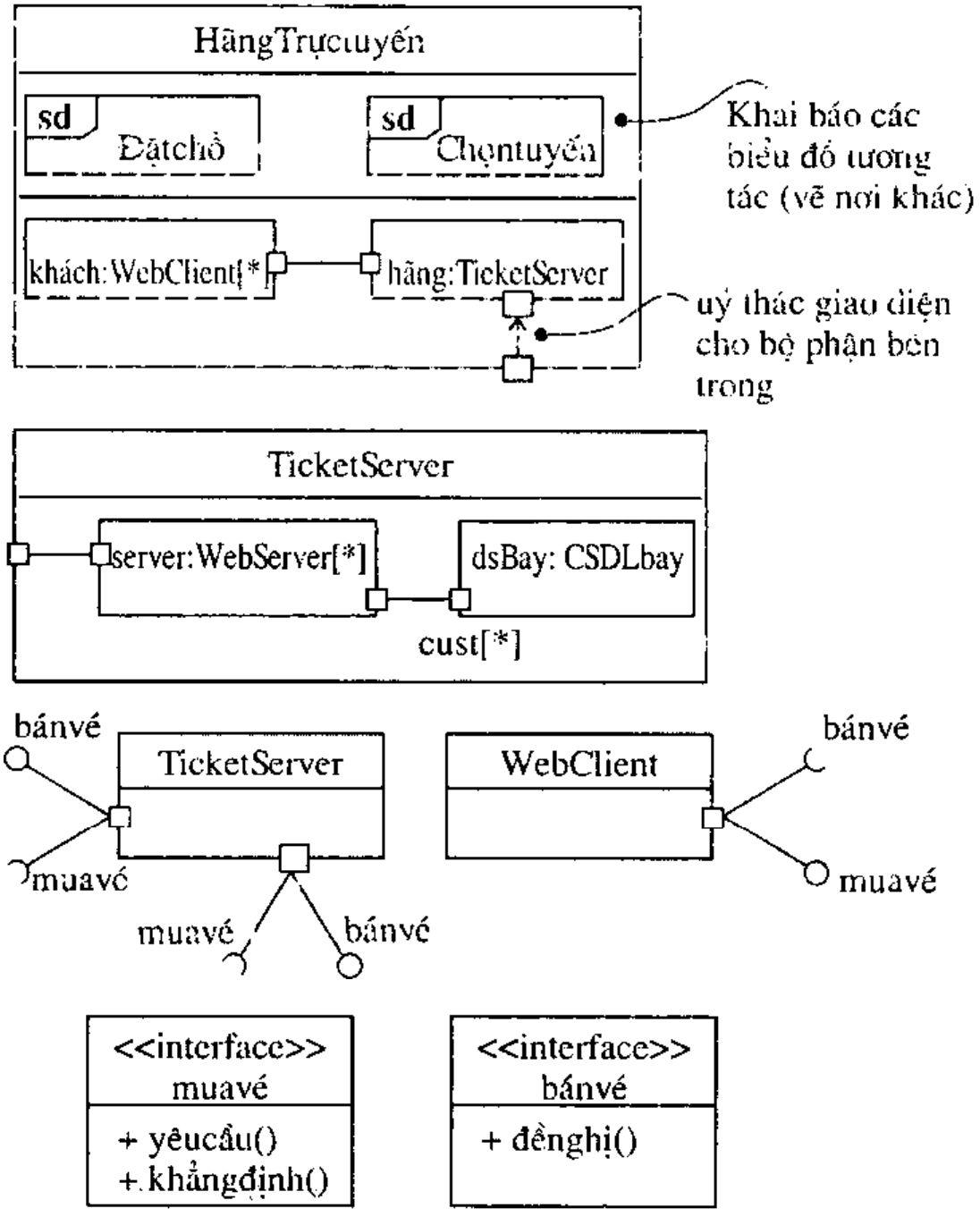
trong đó tênLớp chính là tên của liên kết, như là kiểu của nó. Một khuôn dập có thể đặt trước, hay trên tên cầu nối. Một xâu tính chất có thể đặt sau hay dưới tên cầu nối (Hình IV.39).



Hình IV.39. Bộ phận và cầu nối trong cấu trúc nội tại

Thí dụ

Các biểu đồ cấu trúc đa hợp vẽ trong Hình IV.40, nhằm mô tả một phần hệ thống Bán vé Máy bay, sẽ vận dụng một cách tổng hợp các khái niệm và ký pháp đã được trình bày cho tới đây.



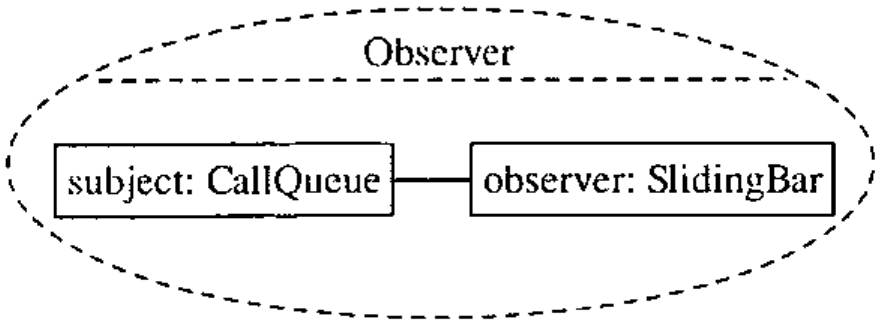
Hình IV.40. Một hệ thống bán vé máy bay trực tuyến

- *Hợp tác* (collaboration): Hợp tác (đã được đề cập sơ qua ở §1.9) cũng được xem là một loài và nó diễn tả một tập hợp phần tử tương

tác cùng nhau (các vai trò), thông qua các đường liên lạc (cầu nối), để hoàn thành một nhiệm vụ hay một tập hợp các nhiệm vụ chung nào đó.

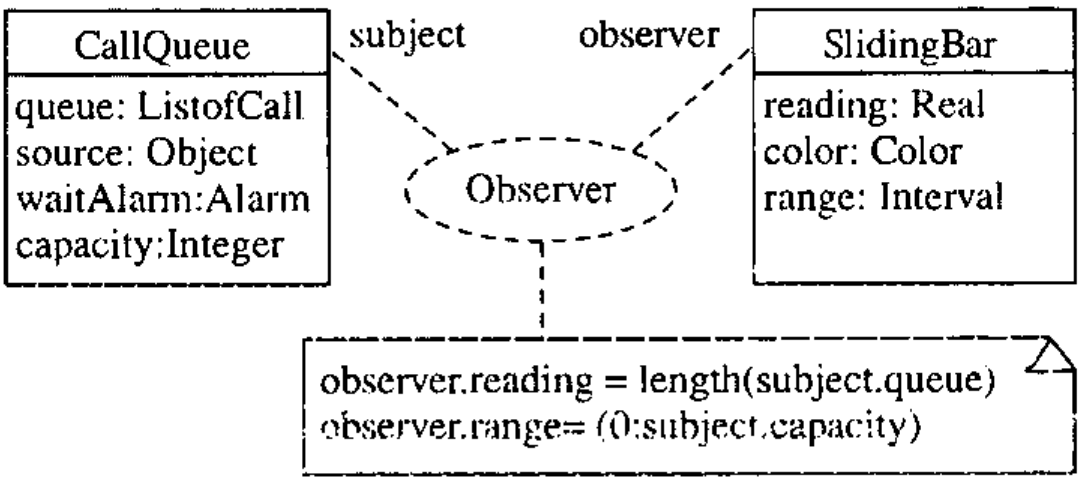
Đây chỉ là một sự diễn tả về cấu trúc, cho nên nó chưa đi sâu vào các chi tiết của sự tương tác (như các biểu đồ tương tác mà ta sẽ đề cập ở Chương V). Mặt khác, các chi tiết về cấu trúc, như là tên và chi tiết đầy đủ của các lớp của các cá thể tham gia hợp tác cũng có thể được bỏ qua, nghĩa là chỉ giữ lại các sắc thái của thực tế được xem là cần thiết cho việc giải thích một sự hợp tác cụ thể mà thôi.

Một hợp tác được biểu diễn bằng một hình elip nét đứt, trong đó có viết tên hợp tác. Có thể thêm một ngăn trong hình elip đó để vẽ cấu trúc nội tại của nó bao gồm các vai trò và các cầu nối giữa chúng (Hình IV.41).



Hình IV.41. Cấu trúc nội tại của một hợp tác

Cũng có thể vẽ một cách khác, có phần chi tiết hơn như trên Hình IV.42, trong đó các tính chất của các lớp CallQueue và SlidingBar, mà mỗi vai trò Subject và Observer cần phải có, đã được nêu rõ. Các đường nối đứt nét giữa hai lớp trên với biểu tượng của hợp tác (trên đó có ghi tên các vai trò) được gọi là các *phân vai* (role binding).



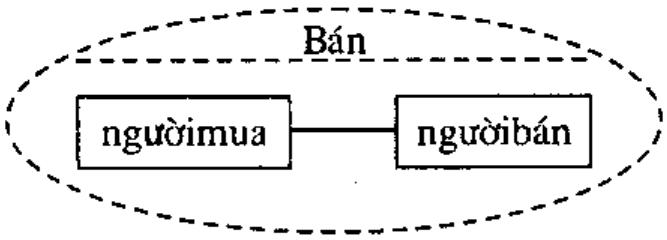
Hình IV.42. Cách biểu diễn khác của hợp tác

- *Biểu hiện Hợp tác* (collaboration occurence): Một biểu hiện hợp tác là một áp dụng của một hợp tác (coi như một hình mẫu) vào một hoàn cảnh cụ thể bao gồm các lớp hay cá thể cụ thể sắm các vai trong hợp tác đó. Có thể ví hợp tác và biểu hiện hợp tác như là vở kịch và suất diễn: ở suất diễn này thì các diễn viên A, B, C đóng các vai trong vở kịch, còn ở suất diễn khác thì lại là các diễn viên I, J, K đóng các vai đó.

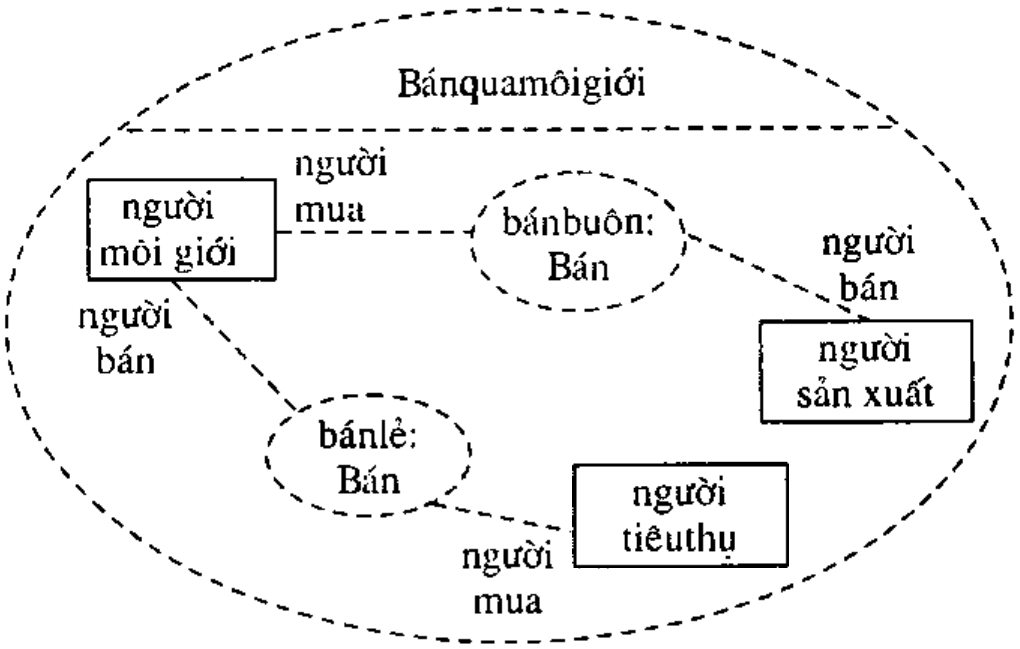
Một biểu hiện hợp tác được biểu diễn bằng một hình elip nét đứt, bên trong có viết tênBiểuhiệnHợptác: TênHợptác.

Thí dụ

Cho hợp tác Bán với hai vai trò là người mua và người bán (Hình IV.43). Lại xét một hợp tác khác là Bán qua môi giới, trong đó có ba vai trò là: người sản xuất, người môi giới và người tiêu thụ. Đặc tả của Bán qua môi giới cho thấy hợp tác này chứa hai biểu hiện của hợp tác Bán (Hình IV.44). Đó là biểu hiện bán buôn giữa người sản xuất và người môi giới, và biểu hiện bán lẻ giữa người môi giới và người tiêu thụ.

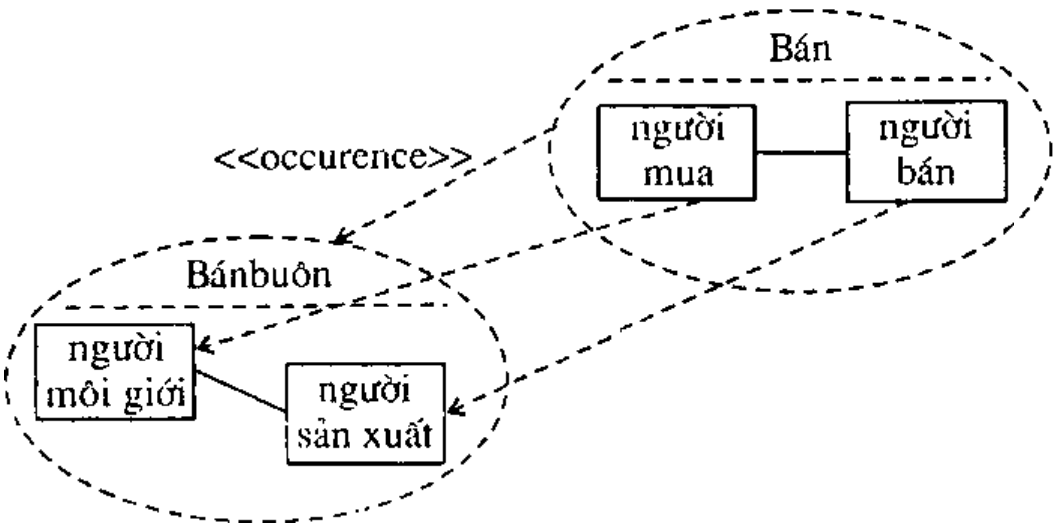


Hình IV.43. Hợp tác bán



Hình IV.44. Hợp tác Bán qua môi giới

Có thể diễn tả mối liên quan giữa một hợp tác với một biểu hiện của nó, cũng như giữa các vai trò và các loài sắm vai trò đó, bằng sự phụ thuộc (mũi tên mở đứt nét) như trên Hình IV.45.



Hình IV.45. Mối liên quan giữa hợp tác và biểu hiện hợp tác

Tóm tắt

Một biểu đồ cấu trúc đa hợp là một đồ thị mà:

- Các nút là: bộ phận, cổng, hợp tác, biểu hiện hợp tác
- Các đường là: cầu nối, phân vai

§2. BƯỚC 3. MÔ HÌNH HOÁ LĨNH VỰC ỨNG DỤNG

1. MỤC ĐÍCH

Mục đích của bước mô hình hoá lĩnh vực ứng dụng (bước 3) là thực hiện việc mô hình hoá từ vựng của hệ thống, đã được giới thiệu qua ở mục 9 trong §1 ở trên. Xuất phát từ các khái niệm về các sự vật trong lĩnh vực ứng dụng, ta trừu tượng hoá chúng thành các lớp gọi là các lớp lĩnh vực, hay lớp nghề nghiệp. Các lớp này thường chỉ dùng để phản ánh và mô phỏng các sự vật trong thế giới thực, cho nên trách nhiệm của chúng cũng thường chỉ là lưu giữ và cung cấp các thông tin về các sự vật đó. Các lớp đó sẽ được biểu diễn trong một (hay một số) biểu đồ lớp, cho thấy rõ các liên kết giữa chúng cũng như thuộc tính của chúng. Tuy nhiên các lớp chưa có các thao tác, vì ở bước này ta mới chỉ tập trung xem xét về cấu trúc mà chưa đả động đến hành vi.

2. TRÌNH TỰ TIẾN HÀNH

Bước 3 được tiến hành qua bốn bước nhỏ sau:

- Nhận định các khái niệm của lĩnh vực.
- Thêm các liên kết và các thuộc tính.
- Khái quát hoá các khái niệm.
- Cấu trúc thành các gói.

Ta sẽ lần lượt xét các bước nhỏ này ở các mục sau.

3. NHẬN ĐỊNH CÁC KHÁI NIỆM CỦA LĨNH VỰC

a) Nguồn tìm kiếm

Các khái niệm của lĩnh vực là những khái niệm về các sự vật (cụ thể hay trừu tượng) mà các người dùng, các chuyên gia nghiệp vụ sử dụng khi nói đến nghề nghiệp và công việc của mình. Bởi vậy để tìm kiếm các khái niệm này, ta dựa vào:

- Các kiến thức về lĩnh vực nghiệp vụ.
- Các cuộc phỏng vấn trao đổi với các người dùng và chuyên gia.
- Bản tổng quan về hệ thống và nhu cầu.

- Các tài liệu miêu tả các ca sử dụng đã lập ở bước trước.

b) Cách nắm bắt các khái niệm

Có thể tham khảo sự hướng dẫn của một số tác giả như sau:

- Abbott (1983) đề nghị: cứ đọc văn bản miêu tả hệ thống (tức bản phát biểu nhu cầu),
  - các danh từ sẽ là đối tượng hay thuộc tính,
  - các động từ có thể là các thao tác.

Cách làm này là triệt để, khó bỏ sót khái niệm, song cũng dễ hồ đồ, vì:

- Sẽ có nhiều khái niệm (danh từ) không là khái niệm nghề nghiệp.
- Không phải lúc nào cũng dễ phân biệt giữa đối tượng và thuộc tính. Nói chung thì đối tượng là một khái niệm có mang tính chất, có sự hoạt động chức năng trong hệ thống, và đôi khi còn có đời sống, có trạng thái. Còn thuộc tính chỉ là một tính chất lượng hoá được hay gán giá trị được.

- Shlaer & Mellor, Ross, Coad & Yourdon lại đề nghị các khái niệm được chuyển thành đối tượng có thể là:
  - các *thực thể* vật chất, như xe đạp, máy bay, cảm biến...
  - các *vai trò* như mẹ, giáo viên, cảnh sát...
  - các *sự kiện* như hạ cánh, ngắt, đăng ký xe máy...
  - các *tương tác* như cho vay, hội thảo...
  - các *tổ chức* như công ty, khoa, lớp,...

  v.v...

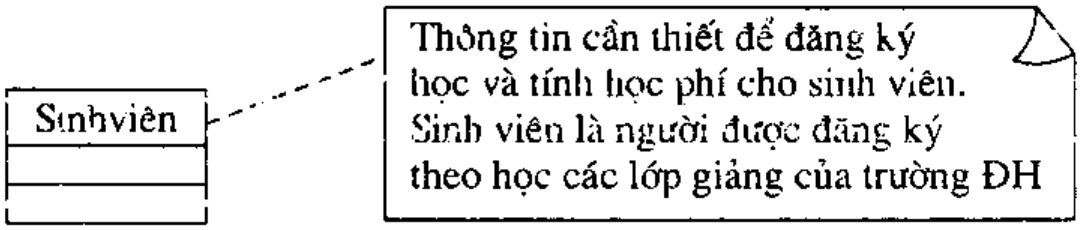
c) Đặt tên và gán trách nhiệm

Từ mỗi khái niệm nghiệp vụ được phát hiện như trên, ta lập một lớp và việc đầu tiên là cho nó một cái tên. Nên giữ nguyên tên các khái niệm như trong thực tế và không được dùng nhiều tên cho một khái niệm, dù rằng trong thực tế có vậy.

Tiếp đến là gán trách nhiệm cho lớp vừa mới thành lập. Trách nhiệm mô tả nghĩa vụ và mục đích của lớp, chứ không phải là cấu trúc của nó, dù rằng sau này trách nhiệm sẽ cho phép ta định ra cấu trúc

(thuộc tính và liên kết) cùng với hành vi (các thao tác) của lớp. Chỉ nên diễn tả trách nhiệm với dăm ba dòng; nếu ta có dùng công cụ CASE thì công cụ sẽ cất giữ nó, còn nếu không ta có thể viết nó trong một ghi chú cho lớp, hoặc trong ngăn thứ tư của lớp.

Một thí dụ là lớp Sinh viên có thể gán trách nhiệm như trên Hình IV.46.



Hình IV.46. Gán trách nhiệm cho một lớp

Trái lại một mô tả trách nhiệm như sau là không tốt:

"Tên, địa chỉ, số điện thoại của một sinh viên"

Mô tả này chỉ cho ta các thuộc tính cần thiết song lại không nói được vì sao ta cần có lớp này.

Việc gán tên và trách nhiệm cho một lớp đề cử cũng đã có thể cho ta hay là việc chọn lựa là có hợp lý không:

- Nếu chọn được tên và gán được trách nhiệm rõ ràng, chặt chẽ thì lớp đề cử là tốt.
- Nếu chọn được tên, song trách nhiêm lại giống trách nhiệm của một lớp khác, thì nên gộp hai lớp đó làm một.
- Nếu chọn được tên, song trách nhiệm lại quá dông dài, thì nên tách nó ra nhiều lớp.
- Khó chọn được tên hợp lý hay khó mô tả trách nhiệm, thì nên phân tích sâu thêm vào nó để chọn những biểu diễn thích hợp.

4. THÊM CÁC LIÊN KẾT VÀ CÁC THUỘC TÍNH

Tiếp đến là phát hiện các liên kết và các thuộc tính của các lớp. Nhiều thuộc tính và liên kết của các lớp lĩnh vực đã có thể phát hiện trực tiếp từ bản miêu tả hệ thống và nhu cầu, từ ý kiến của các chuyên gia lĩnh vực và người dùng và từ các trách nhiệm của các lớp mà ta vừa gán ở trên. Đương nhiên như thế là chưa đủ, sau này sẽ bổ sung dần

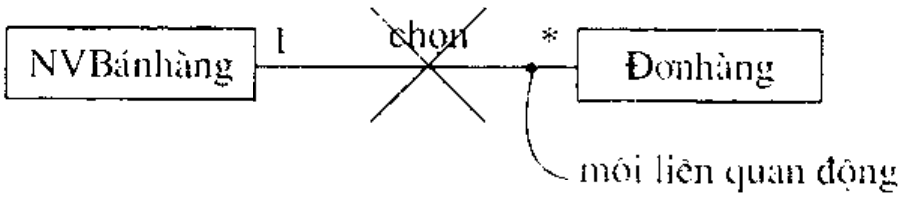
các liên kết và các thuộc tính, cũng như bổ sung các thao tác cho các lớp (lớp lĩnh vực và cả các loại lớp khác đưa thêm sau này), khi ta nghiên cứu sâu vào hành vi (tương tác và ứng xử) của hệ thống. Vậy đây chỉ là sự phát hiện khởi đầu các liên kết và thuộc tính.

Chẳng hạn, từ văn bản trách nhiệm của lớp Sinh viên, ta có:

- Câu "Thông tin cần thiết để đăng ký học và tính học phí" cho ta suy ra một số thuộc tính như tên, mã SV, địachỉ ... cho lớp Sinhviên.
- Câu "Sinh viên là người được đăng ký theo học các lớp giảng của trường ĐH" cho ta suy ra là có liên kết giữa lớp Sinhviên và Lớpgiảng, với tên liên kết có thể là "đăng ký".

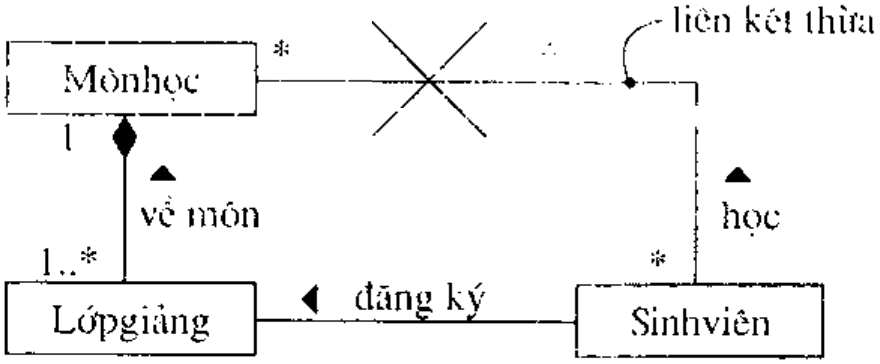
Việc chọn lựa liên kết và thuộc tính này cũng sẽ gặp phải nhiều tình huống mà ta cần cân nhắc cẩn thận, như sẽ trình bày tiếp đây:

- *Liên kết không có cấu trúc:* Đó chỉ là các mối liên quan động, xảy ra nhất thời (Hình IV.47). Các liên kết (có cấu trúc) phải ổn định và tồn tại trong một thời gian nào đó.



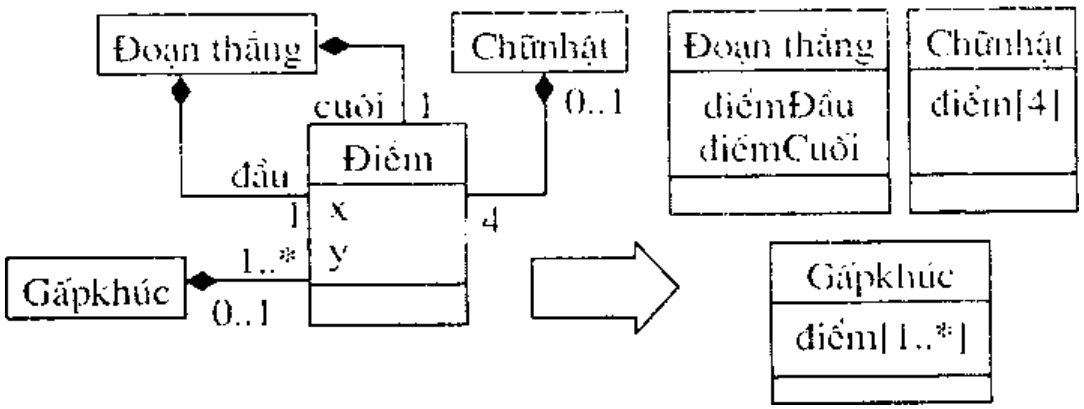
Hình IV.47. Liên kết không đúng đắn

- *Liên kết thừa:* Đó là những liên kết tìm lại được nhờ lưu hành theo những liên kết khác đã có (Hình IV.48).



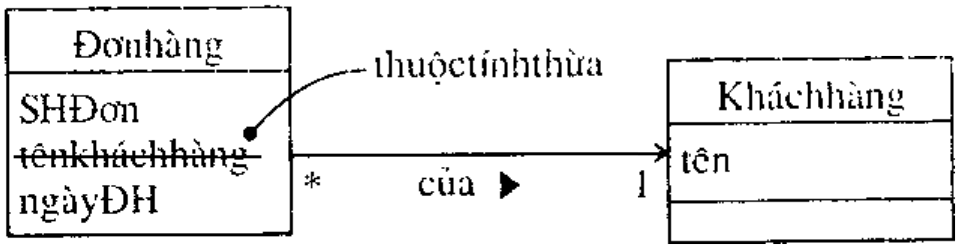
Hình IV.48. Liên kết thừa

- *Phân biệt thuộc tính và lớp:* Như trên đã nói, nhiều khi khó phân biệt thuộc tính với lớp. Nói chung thì lớp phải là một khái niệm phức tạp, ta có thể đặt cho nó nhiều câu hỏi. Chẳng hạn với lớp Sinh viên ta có thể hỏi: tên sinh viên là gì, địa chỉ ở đâu, đăng ký học lớp giảng nào, thậm chí là đã học xong các môn học nào, với các thầy nào. Trái lại với thuộc tính thì chỉ có thể hỏi một câu là nó có giá trị bao nhiêu. Tuy nhiên, cũng cần lưu ý là thuộc tính có thể có giá trị cấu trúc (nhiều thành phần) hoặc có khi là đa trị. Điều này làm cho chúng ta bối rối. Thí dụ ở Hình IV.19 là một thí dụ về xử lý các hình như Đoạn thẳng, Hình chữ nhật, Đường gấp khúc. Đương nhiên đó là các lớp. Còn Điểm cũng có thể xem là lớp, với các thuộc tính là các toạ độ x, y của nó. Song nếu ta biết người dùng không có xử lý nào về điểm, thì lại nên chuyển nó thành thuộc tính đa trị trong các lớp hình học kể trên (mỗi trị của nó là một cặp toạ độ). Biểu đồ sẽ gọn hơn.



Hình IV.49. Thuộc tính hay lớp

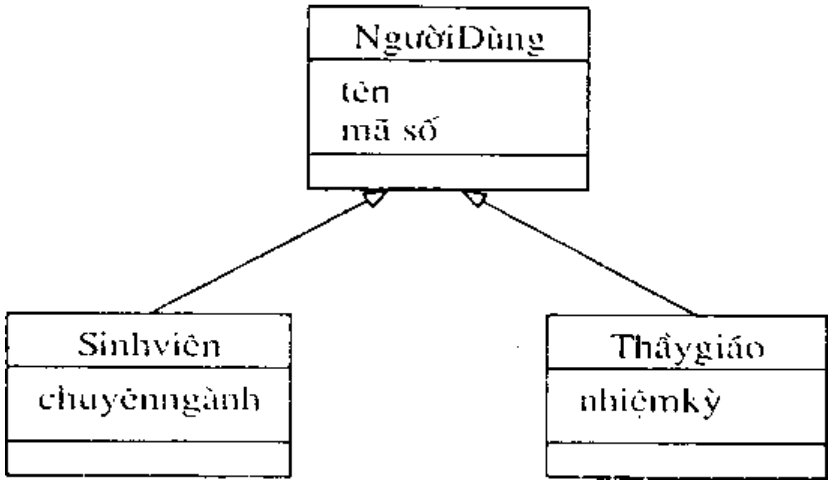
- *Thuộc tính thừa vì đã có liên kết:* Đó là khi ta đưa vào một thuộc tính mà thực ra nghĩa của nó đã biểu hiện trong một liên kết (Hình IV.50).



Hình IV.50. Thuộc tính thừa vì có liên kết

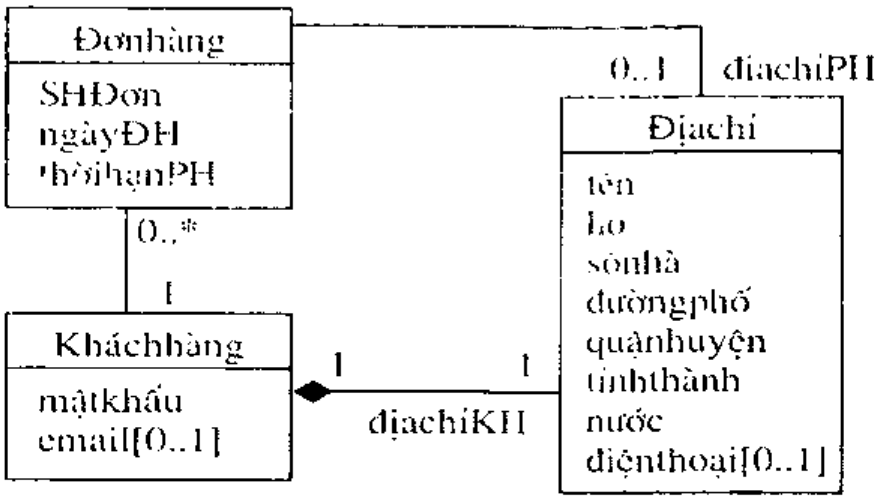
5. KHÁI QUÁT HOÁ CÁC LỚP

Để cho mô hình thêm tốt, ta tìm cách rút ra các phần chung giữa các lớp để lập thành lớp khái quát. Chẳng hạn trong hệ ĐKMH (thí dụ xuyên suốt) thì Sinh viên và Thầy giáo có những thuộc tính chung, như tên, mã số, có thể rút ra để lập một lớp khái quát là NgườiDùng (Hình IV.51).



Hình IV.51. Thực hiện khái quát hoá

Lưu ý rằng việc rút ra phần chung không nhất thiết là phải dẫn tới khái quát hoá, chẳng hạn khi phần chung đó chỉ là yếu tố phụ, và bấy giờ ta thay khái quát hoá bằng liên kết hay kết nhập. Trong Hình IV.52, thì hai lớp Đơnhàng và Kháchhàng đều có thuộc tính là địa chỉ (địa chỉ giao hàng và địa chỉ khách hàng), lập một lớp là Vật có địa chỉ (?!) và xem là khái quát hoá của Đơn hàng và Khách hàng là không thuận, cho nên ta đã chuyển thành liên kết.



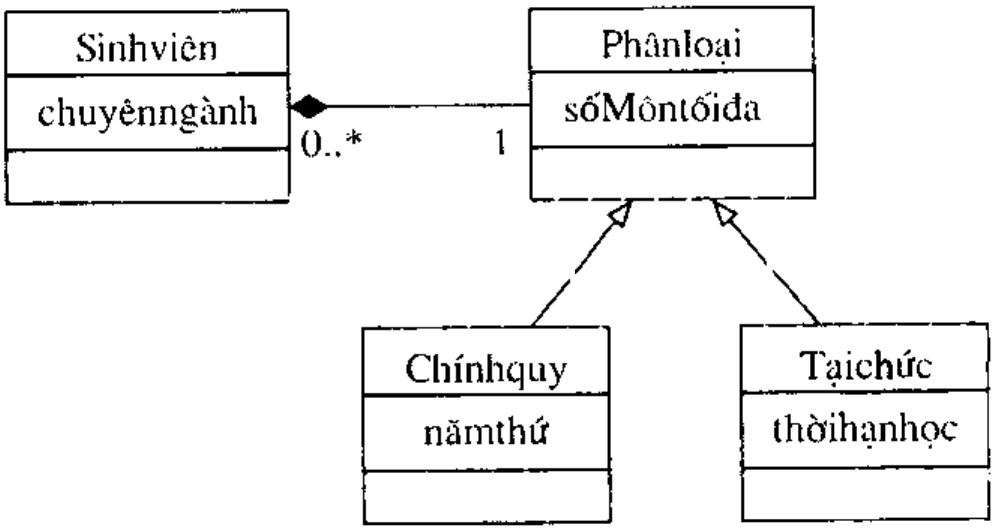
Hình IV.52. Tách phần chung thành liên kết

Một khả năng nữa là khi tách phần chung có thể dẫn tới nhiều lớp khái quát (thừa kế bội). Có nhiều rắc rối trong cài đặt xảy ra với thừa kế bội, như đụng độ về tên gọi, tìm kiếm ngược theo khái quát hóa khó khăn..., cho nên thừa kế bội không phải là phương án ưu tiên mà chỉ là phương án bất đắc dĩ.

Nhiều khi để có một mô hình tốt, ta phải phối hợp giữa kết nhập và thừa kế khi tách phần chung của các lớp. Chẳng hạn trong hệ ĐKMH, nếu có sự phân biệt giữa Sinh viên chính quy và tại chức, thì thông thường ta lập hai lớp chuyên biệt cho lớp sinh viên. Tuy nhiên phương án này có những bất hợp lý như:

- Một sinh viên chính quy muốn chuyển sang hệ tại chức, sẽ dẫn tới việc một đối tượng phải thay lớp của mình (dẫn theo nhiều rắc rối trong lập trình).
- Nếu lại có thêm một sự phân loại nữa như là Học tại trường và Học từ xa, thì sẽ dẫn tới thừa kế bội và phải quản lý các trường hợp tổ hợp khác nhau.

Để tránh các điều rắc rối này, ta cho lớp Sinh viên một phân loại (kết nhập) và phân loại này có thể hoặc là chính quy, hoặc là tại chức (thừa kế), như trên Hình IV.53.



Hình IV.53. Phối hợp kết nhập và thừa kế

6. ĐÓNG GÓI CÁC LỚP

a) Mục đích đóng gói

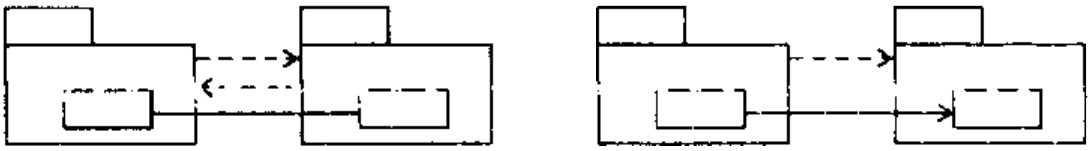
Các lớp cùng với các liên kết, các thuộc tính được biểu diễn chung vào một biểu đồ lớp. Tuy nhiên khi biểu đồ là lớn, ta nên gom các lớp trên biểu đồ vào các gói. Mục đích cấu trúc biểu đồ thành gói là:

- Khắc phục sự phức tạp: Nếu hệ thống là quá phức tạp, thì ta thường nhận định các gói trước, rồi phát triển mỗi gói thành biểu đồ lớp sau. Còn nếu hệ thống không phức tạp lắm, thì có thể lập biểu đồ lớp trước rồi chia cắt thành gói sau, để cho biểu đồ dễ đọc, dễ xử lý. Cũng cần chú ý là đây chỉ mới là giai đoạn mô hình hoá lĩnh vực, các lớp cho vào các gói chỉ mới là các lớp lĩnh vực. Sau này hệ thống phát triển thêm, sẽ có nhiều lớp khác mà ta sẽ bổ sung thêm vào các gói cũ hay lập thành các gói mới.
- Tạo thuận lợi cho việc sử dụng lại: Việc gom các gói theo hai tiêu chí cố kết mạnh và tương liên yếu (sẽ nói ở dưới) sẽ cho phép ta tách các lớp mà cá thể của chúng có đời sống ngắn ngủi với các lớp mà cá thể của chúng có đời sống lâu dài, thậm chí là dùng lại được cho nhiều bài toán, nhiều ứng dụng. Trong lập trình hướng đối tượng thì khả năng sử dụng lại luôn luôn là một mục tiêu cần quan tâm.
- Tạo thuận lợi cho lập trình hướng thành phần: Ngày nay lập trình luôn luôn hướng tới sử dụng các thành phần phần mềm đã lập sẵn. Việc tổ chức biểu đồ thành các gói cũng nhằm đưa dần hệ thống đang lập vào quỹ đạo của lập trình hướng thành phần.
- Làm căn cứ để tổ chức các nhóm làm việc: Việc tổ chức thành gói cũng giúp cho người điều hành dự án dễ dàng phân công công việc cho các nhóm làm việc trong dự án. Thông thường thì mỗi gói nên bao gồm trung bình khoảng 10 lớp. Gói bé quá sẽ có trách nhiệm nhỏ, có khi là tủn mủn, trong khi tương liên giữa các gói lại phức tạp. Gói lớn quá thì trách nhiệm lớn, thường đó là sự gom nhóm của nhiều trách nhiệm khác nhau, khó có thể quán xuyến bởi một nhóm làm việc.

b) Tiêu chí đóng gói

Để gom các lớp vào thành các gói, ta căn cứ vào hai tiêu chí sau:

- *Cố kết cao*: Tính cố kết (cohesion) biểu hiện ở sự gắn gũi về ngữ nghĩa, gắn kết về hành vi. Tính cố kết trong một gói phải càng cao càng tốt. Muốn thế thì:
  - Các lớp trong cùng một gói phải hoạt động hợp tác với nhau để hướng tới cùng một mục đích.
  - Các gói tách biệt các lớp mà đối tượng của chúng có thời gian sống rất khác nhau, đặc biệt là tách biệt các lớp lĩnh vực với các lớp ứng dụng, các lớp phù trợ.
- *Tương liên yếu*: Tương liên (coupling) là sự ràng buộc ảnh hưởng lẫn nhau giữa các gói. Tương liên càng yếu, càng đơn giản càng tốt. Mối liên quan chính giữa các gói là sự phụ thuộc (nhập khẩu). Sự phụ thuộc đó lại bắt nguồn từ các liên kết giữa các lớp: Nếu giữa hai lớp thuộc hai gói có một liên kết hai chiều (lưu hành theo cả hai phía), thì sẽ có hai phụ thuộc ngược chiều nhau giữa hai gói. Nếu liên kết đó lại chỉ là một chiều (lưu hành chỉ theo một phía), thì chỉ có một phụ thuộc giữa hai gói (xem Hình IV.54). Kiến trúc đóng gói ưu tiên là chỉ có sự phụ thuộc một chiều giữa từng cặp gói (sự phụ thuộc Khách hàng/Cung ứng). Vì vậy khi đóng gói ta cần chú ý:
  - các liên kết vắt qua giữa hai gói là không có nhiều.
  - các liên kết đó có khả năng chuyển thành một chiều (nghĩa là không dùng chiều ngược lại).



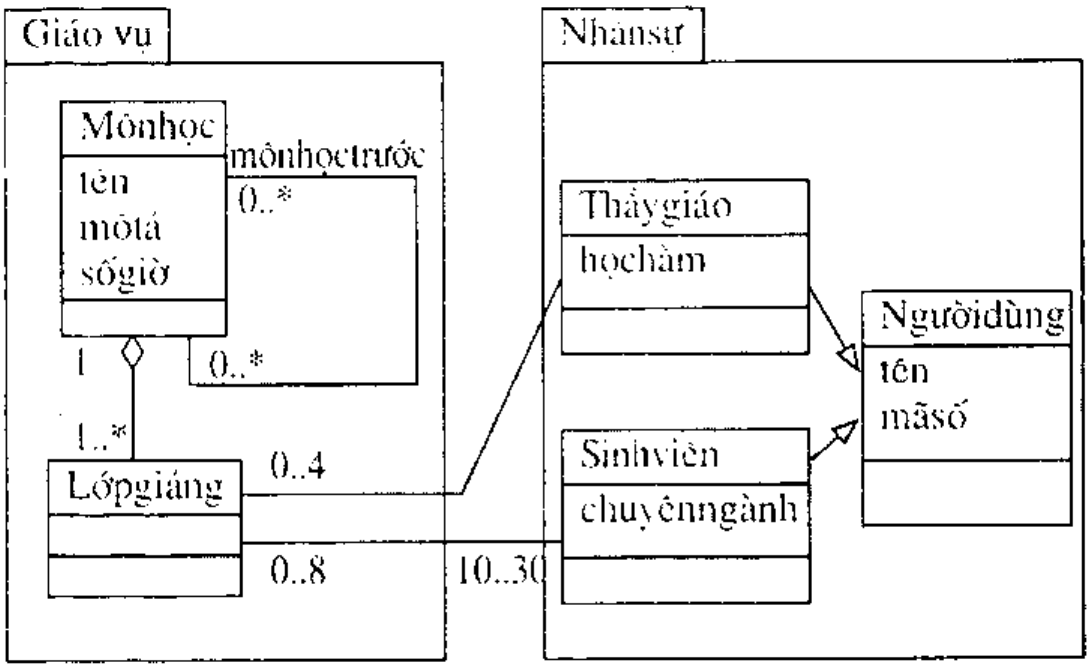
Hình IV.54. Phụ thuộc hai chiều và phụ thuộc một chiều

Thí dụ

Trở lại hệ ĐKMH và tiến hành mô hình hoá lĩnh vực cho nó.

Hệ ĐKMH là bé. Chỉ có bốn lớp lĩnh vực là: Thầygiáo, Sinhviên, Mônhọc và Lớpgiảng. Chú ý rằng ta có bốn đối tác là: Thầygiáo, Sinhviên, CBQuảnsinh và Hệthuhọcphí, song chỉ có Thầygiáo và Sinhviên được xem là lớp thôi, vì chỉ hai đối tác đó là cần được lưu giữ và quản lý thông tin về chúng, còn hai đối tác kia thì không.

Biểu đồ lớp cho như trên Hình IV.55, trong đó việc tách thành hai gói chỉ là thí dụ. Việc tách đó là có cần thiết và có hợp lý không, ta sẽ tiếp tục cân nhắc thêm.



Hình IV.55. Biểu đồ lớp lĩnh vực của hệ ĐKMH

§3. BƯỚC 4: XÁC ĐỊNH CÁC ĐỐI TƯỢNG VÀ LỚP THAM GIA CÁC CA SỬ DỤNG

1. MỤC ĐÍCH

Trong bước 3 (bước mô hình hoá lĩnh vực) ta đã tiến hành nghiên cứu lĩnh vực, mà không xem xét tới ứng dụng. Trên một lĩnh vực (chẳng hạn một hãng hàng không hay một ngân hàng...), có thể có nhiều ứng dụng (chẳng hạn quản lý bay, quản lý vật tư, quản lý ngoại hối v.v...). Cho nên các lớp lĩnh vực (lớp thực thể) mà ta đã phát hiện được ở bước 3, dù là sẽ được dùng phổ biến trong nhiều ứng dụng trên cùng một lĩnh vực, song lại chưa phản ánh thực sự đặc thù của mỗi ứng dụng cụ thể.

Đặc thù của mỗi ứng dụng cụ thể nằm ở các ca sử dụng (được phát hiện ở bước 2). Bởi vậy, để đi sâu nghiên cứu một ứng dụng, ta phải bám sát các ca sử dụng.

Như đã nói ngay đầu chương này, một ca sử dụng thực chất là một chức năng (lớn) của hệ thống cần xây dựng. Để đi sâu phân tích một ca sử dụng, ta có hai con đường khác biệt:

- *Phân tích hướng chức năng:* Phân rã ca sử dụng thành nhiều chức năng con, và làm mịn dần qua nhiều bước. Đó là con đường phân tích trên xuống truyền thống, mà ta đã chủ ý rời bỏ.
- *Phân tích hướng đối tượng:* Đây chính là con đường mà ta chọn. Ca sử dụng được hình dung như là một hợp tác của một số đối tượng. Với cách tiếp cận này thì ta sẽ phải lần lượt thực hiện các việc sau:

  (1) Phát hiện các đối tượng tham gia ca sử dụng.

  (2) Xác định vai trò của mỗi đối tượng trong ca sử dụng.

  (3) Nghiên cứu các liên kết giữa các lớp tham gia ca sử dụng.

  (4) Nghiên cứu sự tương tác giữa các đối tượng tham gia ca sử dụng.

Các điểm (1), (2), (3) đề cập các đặc điểm tĩnh (cấu trúc) của sự hợp tác, và đó chính là mục đích của bước 4 này. Còn điểm (4) đề cập đặc điểm động (hành vi) của sự hợp tác, và đó sẽ là công việc thực hiện trong bước 5 (Chương V).

Dưới đây ta sẽ lần lượt trình bày các việc (1), (2), (3), mà các kết quả thu được sẽ được diễn tả ban đầu bởi một biểu đồ cấu trúc đa hợp, và sau đó là một biểu đồ lớp.

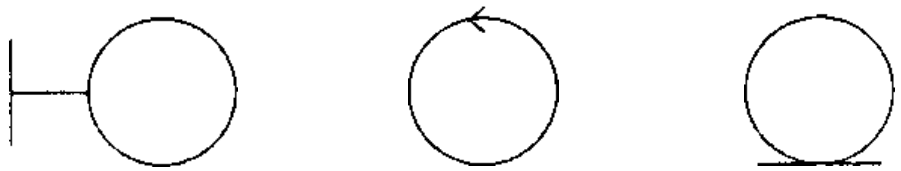
2. PHÁT HIỆN CÁC ĐỐI TƯỢNG/LỚP THAM GIA CA SỬ DỤNG

Đầu vào của bước 4 này là các kết quả từ bước 2 (các ca sử dụng cùng với các kịch bản của chúng) và các kết quả từ bước 3 (các lớp lĩnh vực với các trách nhiệm của chúng). Các ca sử dụng được đem ra nghiên cứu, hoặc là lần lượt từng cái một (quá trình lặp), hoặc là song song (phân công cho nhiều nhóm làm việc), để phát hiện các đối tượng/lớp tham gia từng ca sử dụng đó. Các lớp tham gia ca sử dụng được gọi chung là các lớp phân tích.

Để thực hiện được các kịch bản của một ca sử dụng, thì các lớp phân tích phải gồm đủ ba loại:

- *Các lớp biên* (boundary) hay lớp *đối thoại* (dialog): Đó là các lớp nhằm chuyển đổi thông tin giao tiếp giữa các đối tác và hệ thống. Điển hình là các màn hình giao lưu với các người dùng, cho phép thu thập thông tin hay xuất các kết quả. Nhưng đó cũng có thể là các giao diện (cứng và mềm) chuyển đổi tương tự/số giữa hệ thống và các thiết bị mà nó điều khiển hay thu thập thông tin. Cứ mỗi cặp đối tác - ca sử dụng thì phải có ít nhất một lớp biên. Lớp biên chính này lại có thể cần đến các lớp biên phù trợ để nó uỷ thác một phần nào đó trong các trách nhiệm quá lớn của nó. Nói chung thì các đối tượng biên có đời sống kéo dài cùng với ca sử dụng liên quan.
- Các lớp *điều khiển* (control): Đó là các lớp điều hành sự diễn biến trong một ca sử dụng; có thể nói đó là cái "động cơ" làm cho ca sử dụng chuyển vận được. Các lớp này chứa các quy tắc nghiệp vụ và đứng trung gian giữa các lớp biên với các lớp thực thể, cho phép từ màn hình có thể thao tác được các thông tin chứa đựng trong các thực thể. Cứ mỗi ca sử dụng ta lập ít nhất một lớp điều khiển. Các lớp điều khiển khi bước qua giai đoạn thiết kế không nhất thiết là sẽ còn tồn tại như một lớp thực sự, vì nhiệm vụ của nó có thể bị phân tán hay gửi gắm vào các lớp khác, song trong giai đoạn phân tích, nhất thiết phải có chúng để bảo đảm không bỏ sót các chức năng hay hành vi các ca sử dụng.
- Các lớp *lĩnh vực* (domain) hay *thực thể* (entity): Đây là các lớp nghiệp vụ, đến trực tiếp từ biểu đồ lĩnh vực, song chúng sẽ được khẳng định vị thế khi được xuất hiện trong các ca sử dụng. Nói chung thì đấy là các lớp trường cửu, nghĩa là các lớp mà các dữ liệu và các mối liên quan của chúng còn được lưu lại (trong cơ sở dữ liệu hay trên các tệp) sau khi ca sử dụng của chúng đã kết thúc. Lớp thực thể được chọn tham gia ca sử dụng khi thông tin chứa đựng trong nó là được đề cập đến trong ca sử dụng.

Các lớp biên, điều khiển và thực thể được biểu diễn một cách phân biệt nhờ các biểu tượng như trên Hình IV.56 hoặc nhờ các từ khoá <<boumdary>>, <<control>>, <<entity>> khi biểu diễn dưới dạng lớp bình thường. Tên của các lớp biên thường được gắn thêm tiền tố W_ (Window) hay I_ (Interface).



Hình IV.56. Các biểu tượng cho các lớp biên, điều khiển và thực thể

Thí dụ

Trở lại hệ ĐKMH để nghiên cứu sự hợp tác trong các ca sử dụng. Hãy xem lại biểu đồ các ca sử dụng của hệ thống này đã lập ở §2, Chương III. Có nhiều ca sử dụng trên biểu đồ, mà ta đều phải lần lượt nghiên cứu

Để làm thí dụ, ở đây ta chỉ xem xét một ca sử dụng. Đó là ca sử dụng 'Chọn môn học để giảng dạy', mà kịch bản của nó cho bằng văn tự ở mục 3b, §2, Chương III, hoặc cho bằng biểu đồ trình tự hệ thống ở Hình III.3.

Từ kịch bản đó, ta phát hiện ra các lớp tham gia như sau:

- Các lớp thực thể, lấy từ biểu đồ lĩnh vực sang và có mặt ở đây bởi thông tin về chúng được đề cập trong kịch bản, là: Lớp giảng, Môn học, Thầy giáo.
- Các lớp biên gồm:
  - Lớp W_Thầy là màn hình chính giao tiếp với đối tác Thầy giáo.
  - Lớp W_Lớpgiảng và lớp W_Lịchgiảng là các màn hình phụ, dùng tương ứng cho các trường hợp Thêm/Bớt lớp giảng và Xem/In lịch giảng.
- Lớp điều khiển chỉ cần có một, lấy tên là QLLớpThầy.

3. DIỄN TẢ CÁC VAI TRÒ TRONG HỢP TÁC BẰNG MỘT BIỂU ĐỒ CẤU TRÚC ĐA HỢP

Đương nhiên khi ta phân loại các lớp tham gia thành các lớp biên, lớp điều khiển, lớp thực thể thì có thể coi là đã gán các vai trò cho các lớp tham gia hợp tác rồi. Tuy nhiên đó chỉ là vai trò chung, ta nên chỉ ra vai trò cụ thể hơn của mỗi lớp, với từ vựng của ứng dụng.

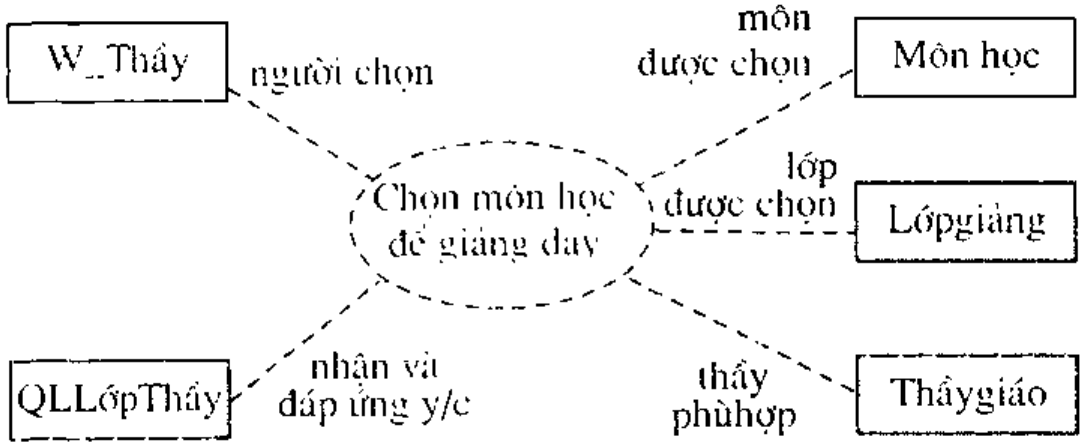
Chẳng hạn, với ca sử dụng 'Chọn môn học để giảng dạy' ở trên, thì vai trò các lớp tham gia có thể là:

- Lớp W_Thầy, đại diện cho đối tác là Thầy giáo, là Người chọn;
- Lớp Mônhọc là Môn được chọn;
- Lớp Lớpgiảng là Lớp giảng được chọn
- Lớp Thầygiáo là Thầy giáo phù hợp;
- Lớp QLThầy là nơi nhận và đáp ứng yêu cầu.

Các lớp W_Lịch giảng và W_Lớp giảng chỉ là các lớp biên phù trợ, san sẻ trách nhiệm với lớp W_Thầy, ta có thể tạm gác sang bên.

UML 2.0, như ta đã biết, cung cấp một loại biểu đồ để biểu diễn vai trò của các lớp tham gia một hợp tác, ấy là biểu đồ cấu trúc đa hợp (xem lại §1.11).

Trong biểu đồ cấu trúc đa hợp thì hợp tác được diễn tả bởi một hình elip viền nét đứt, còn các lớp tham gia (biểu diễn bằng các hình chữ nhật như thường lệ) được kết nối với hợp tác, trên đó có ghi vai trò của mỗi lớp trong hợp tác. Với hợp tác 'Chọn môn học để giảng dạy' thì biểu đồ cấu trúc đa hợp được vẽ như trên Hình IV.57.

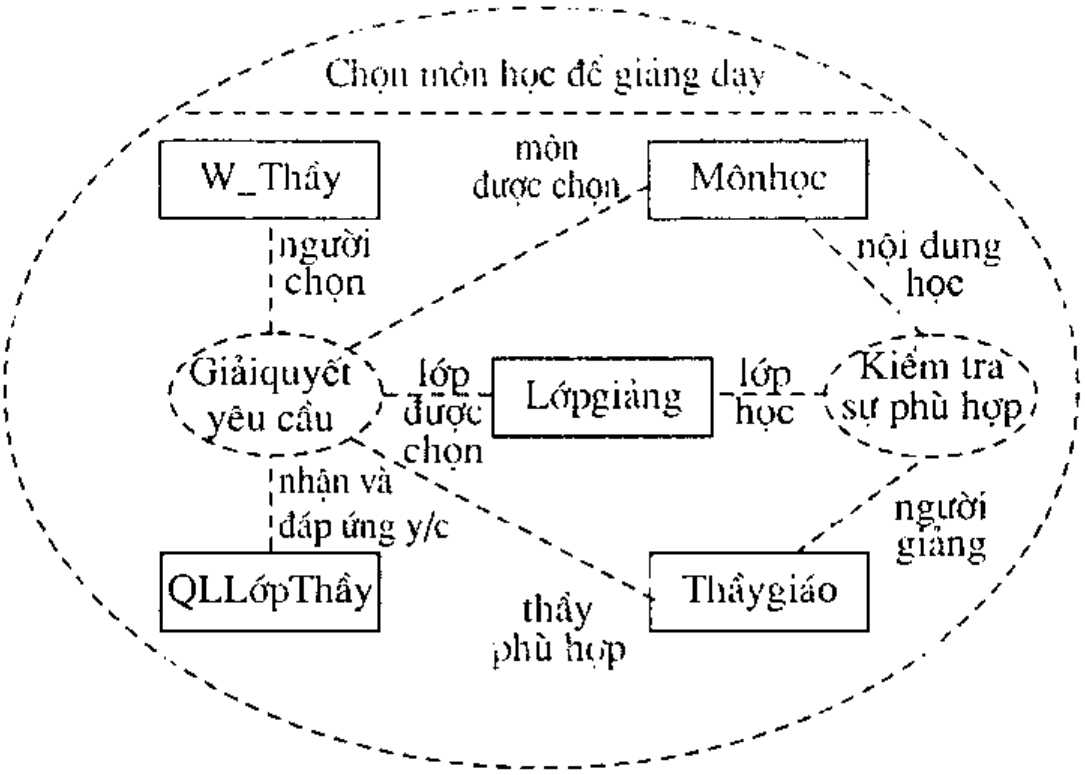


Hình IV.57. Một biểu đồ cấu trúc đa hợp cho ca sử dụng Chọn môn học để giảng dạy

Rõ ràng là cách diễn tả của biểu đồ cấu trúc đa hợp như trên là còn rất sơ lược. Mặc dù biểu đồ đã chỉ ra các lớp tham gia hợp tác cùng với vai trò của mỗi lớp, song cơ chế bên trong của hợp tác vẫn chưa được phô bày. Để phô bày thêm chút ít cơ chế bên trong của hợp tác, UML

2.0 còn cung cấp một hình thức nữa của biểu đồ cấu trúc đa hợp, trong đó có sự lồng nhau của các hợp tác. Nhờ có sự lồng nhau của hợp tác, mà ta có thể diễn tả bên trong một hợp tác lớn giữa nhiều lớp, có những hợp tác nhỏ, cục bộ trong từng nhóm lớp, để thực hiện từng phần việc nhỏ của hợp tác lớn. Có thể xem đó là một cách tiếp cận sự phân tích ca sử dụng. Nó gợi nhớ phần nào tới cách phân rã chức năng truyền thống. Tuy nhiên ta không nên kỳ vọng nhiều ở cách làm này, vì thực chất của sự hợp tác giữa các đối tượng là sự trao đổi thông điệp, thì ta vẫn chưa có cách để diễn tả ở đây.

Với ca sử dụng 'Chọn môn học để giảng dạy', nếu muốn đi sâu vào bên trong, làm lộ diện các hợp tác cục bộ, thì ta có thể diễn tả với biểu đồ cấu trúc đa hợp như trên Hình IV.58.



Hình IV.58. Biểu đồ cấu trúc đa hợp có sự lồng ghép các hợp tác

4. DIỄN TẢ CẤU TRÚC TĨNH CỦA HỢP TÁC BẰNG MỘT BIỂU ĐỒ LỚP

Mục đích cuối cùng của bước 4 là lập một biểu đồ lớp cho mỗi ca sử dụng, phản ánh cấu trúc tĩnh của sự hợp tác. Chính biểu đồ lớp này, mà ta thường gọi là biểu đồ các lớp tham gia ca sử dụng, sẽ là cái nền

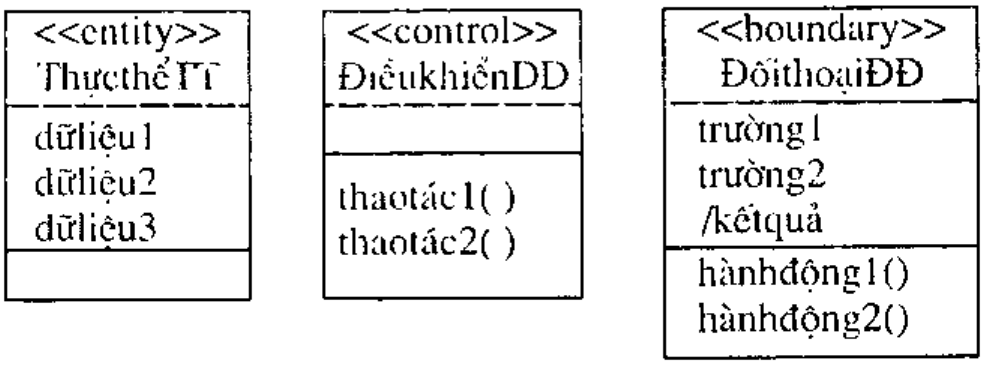
trên đó diễn ra các hoạt động tương tác giữa các lớp, mà ta đi sâu tìm hiểu trong bước 5.

Biểu đồ các lớp tham gia một ca sử dụng phải bao gồm đủ các yếu tố cấu trúc (thuộc tính, thao tác, liên kết) cần thiết cho sự hợp tác giữa các lớp. Các thuộc tính phải lưu giữ đủ những thông tin về các đối tượng, cần dùng trong hợp tác. Các thao tác cung cấp các khả năng dịch vụ cần thiết của mỗi lớp khi tham gia vào hợp tác. Các liên kết tạo ra các cầu nối giữa các lớp, cho phép chúng biết nhau và trao đổi với nhau khi hợp tác.

Các thuộc tính và thao tác được phát hiện dựa trên vai trò của mỗi lớp trong hợp tác, mà ta đã chỉ ra ở phần trên. Các liên kết được phát hiện dựa trên ngữ nghĩa của mỗi lớp, cũng như sự xuất hiện từng cụm trong hợp tác cục bộ, được chỉ ra trong biểu đồ cấu trúc đa hợp ở trên.

Nói cụ thể hơn, thì đầu tiên ta phát hiện và bổ sung các thuộc tính và thao tác cho từng lớp tham gia ca sử dụng (đã được tuyển chọn ở trên) theo các hướng dẫn sau (xem Hình IV.59):

- Các lớp thực thể tạm thời chỉ có các thuộc tính. Các thuộc tính này diễn tả các thông tin trường cữu của hệ thống.
- Các lớp điều khiển lại chỉ có các thao tác. Các thao tác này diễn tả lôgic của ứng dụng, các quy tắc nghiệp vụ, cấc hành vi của hệ thống tin học. Mỗi ca sử dụng có một hay nhiều lớp điều khiển tuỳ thuộc vào số các hành vi và trách nhiệm cần diễn tả.
- Các lớp biên có cả thuộc tính và thao tác. Các thuộc tính diễn tả các trường thu thập thông tin hay xuất kết quả. Các kết quả được phân biệt bằng ký hiệu thuộc tính dẫn xuất. Các thao tác biểu diễn những hành động mà người dùng thực hiện trên màn hình giao diện.

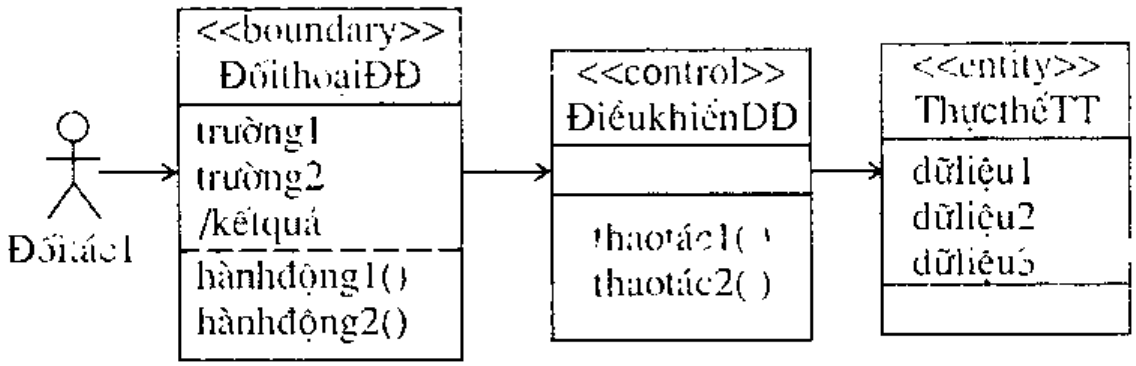


Hình IV.59. Thuộc tính và thao tác của ba loại lớp

Tiếp theo thì các liên kết sẽ được thêm vào giữa các lớp phân tích theo các hướng dẫn sau đây:

- Các lớp biên chỉ được nối với các lớp điều khiển hay với các lớp biên khác. Nói chung thì các liên kết là một chiều từ lớp biên đến lớp điều khiển, trừ khi lớp điều khiển lại tạo lập một đối thoại mới (chẳng hạn một trang các kết quả).
- Các lớp thực thể chỉ được nối với các lớp điều khiển hay lớp thực thể khác. Liên kết với các lớp điều khiển luôn luôn là một chiều (từ lớp điều khiển tới lớp thực thể).
- Các lớp điều khiển được phép truy cập tới mọi loại lớp, bao gồm các lớp điều khiển khác.

Cuối cùng thì ta thêm các đối tác vào biểu đồ, theo nguyên tắc sau: một đối tác chỉ được nối với một (hay một số) lớp biên (Hình IV.60).

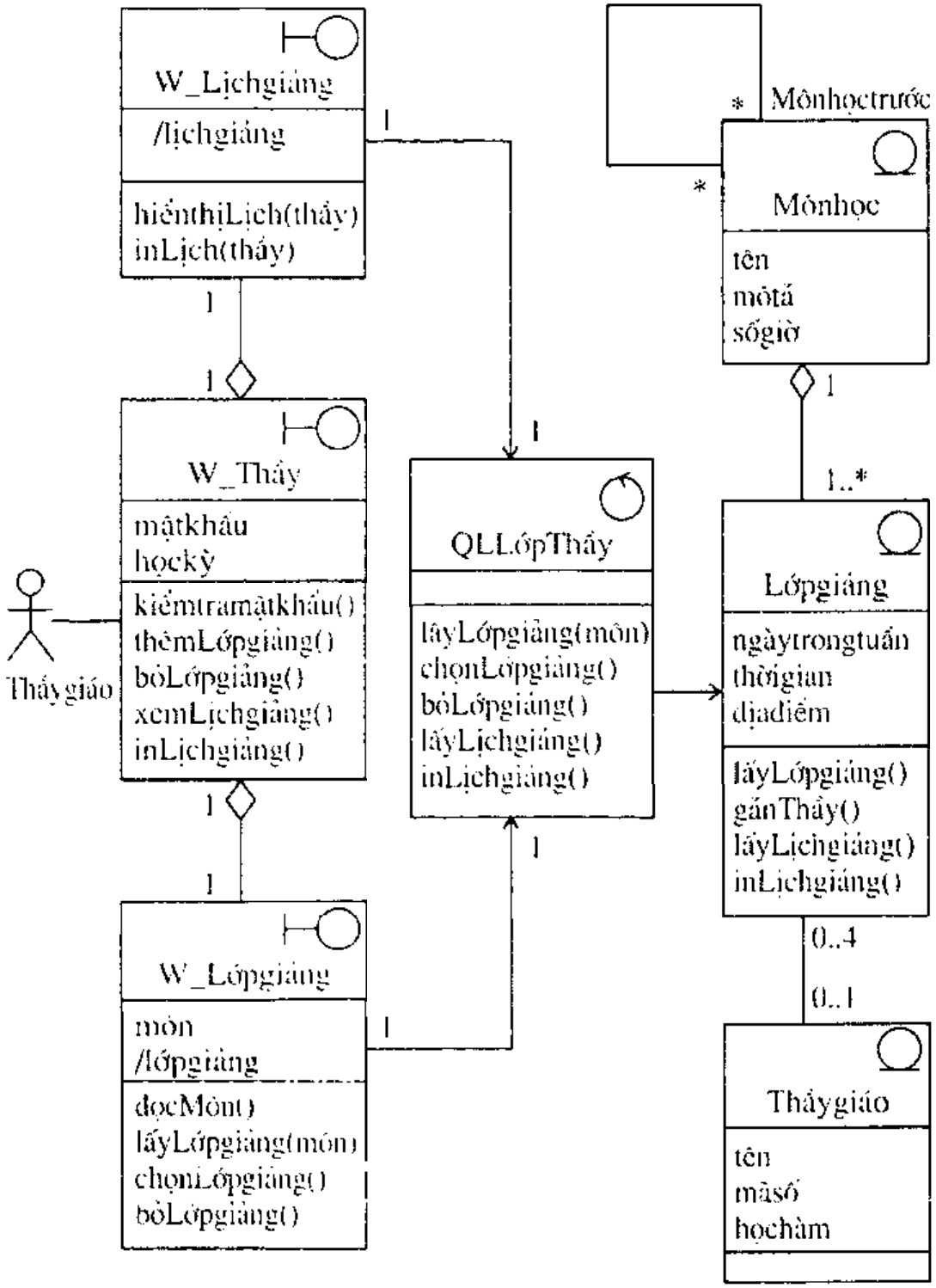


Hình IV.60. Một biểu đồ các lớp tham gia

Thí dụ

Với ca sử dụng 'Chọn môn học để giảng dạy' trong hệ ĐKMH, ta lập được biểu đồ các lớp tham gia như trên Hình IV.61. Trên hình này, thì có lẽ các thuộc tính và các liên kết là dễ nắm bắt theo các hướng dẫn đã nêu trên. Song với các thao tác, thì cũng cần giải thích thêm một chút. Đầu tiên, trực tiếp từ kịch bản của ca sử dụng, ta suy ra các thao tác trên các lớp biên: đó chính là các hành động mà người dùng (thầy giáo) muốn kích hoạt khi vào ca sử dụng. Nhưng nhớ rằng các lớp biên thì chẳng có khả năng xử lý, cho nên chúng lại phải gửi các yêu cầu đó về lớp điều khiển, rồi lớp điều khiển lại uỷ thác cho các lớp thực thể thực hiện (vì các thông tin gốc nằm ở các lớp thực thể). Rốt

cục thì ta thấy tên các thao tác cứ được lặp lại trên các lớp, dù rằng nội dung xử lý của chúng là khác nhau.



Hình IV.61. Biểu đồ các lớp tham gia của ca sử dụng 'Chọn môn học để giảng dạy'

Chương V
MÔ HÌNH HOÁ HÀNH VI

Hành vi (behavior) là cách hành động, cách cư xử của một hệ thống. Mô hình hoá hành vi đối với hệ thống là sự diễn tả khía cạnh động của hệ thống, đối lập với mô hình hoá cấu trúc, là sự diễn tả khía cạnh tĩnh của hệ thống.

Có ba cách tiếp cận hành vi của hệ thống:

- Hành vi thể hiện trong sự tương tác giữa các đối tượng, tạo nên những kịch bản.
- Hành vi biểu lộ cách ứng xử của mỗi đối tượng, trước những sự kiện xảy đến với nó.
- Hành vi bộc lộ ở công việc và các luồng công việc.

UML có những biểu đồ khác nhau để đáp ứng ba cách tiếp cận nói trên đối với hành vi hệ thống. Chương này sẽ lần lượt trình bày cả ba cách tiếp cận đó: mô hình hoá tương tác, mô hình hoá ứng xử và mô hình hoá luồng công việc; trong đó hai cách tiếp cận đầu lại chính là các bước 5 và 6 trong tiến trình 10 bước. Còn cách tiếp cận thứ ba không phải là một bước trọn vẹn trong tiến trình 10 bước, song đâu đó trong quá trình mô hình hoá cũng có lắm lúc ta cần vận dụng cách tiếp cận này.

§1. BƯỚC 5: MÔ HÌNH HOÁ SỰ TƯƠNG TÁC

1. MỤC ĐÍCH

Mục đích của bước mô hình hoá tương tác (bước 5) là dùng các biểu đồ tương tác (cơ bản là các biểu đồ trình tự và biểu đồ giao tiếp) để diễn tả sự tương tác giữa các đối tượng nhằm tạo ra các kịch bản của mỗi ca sử dụng của hệ thống.

Ở bước 2, ta đã chỉ ra các ca sử dụng và đặc tả các ca sử dụng đó bằng văn tự, hoặc bằng biểu đồ trình tự hệ thống. Đó chỉ mới là sự đặc tả ca sử dụng nhìn từ phía ngoài hệ thống (phía người dùng). Qua bước 4, ta đã thâm nhập vào bên trong hệ thống và chỉ ra các đối tượng/lớp tham gia vào các ca sử dụng. Nhưng đó lại chỉ mới là sự đề cập cấu trúc tĩnh của ca sử dụng. Giờ đây, ở bước 5 này, ta nghiên cứu sự tương tác giữa các lớp tham gia nói trên nhằm thực hiện thực sự các kịch bản của mỗi ca sử dụng, tức là để cập tới khía cạnh động của ca sử dụng.

Hình thức tương tác duy nhất có thể có giữa các đối tượng là chuyển giao thông điệp và có hai biểu đồ chính được sử dụng để diễn tả sự tương tác (một cách tương đương với nhau) là biểu đồ trình tự và biểu đồ giao tiếp.

Dù hai biểu đồ này là khác nhau về hình thức, song khi thành lập chúng, ta có thể áp dụng chung các nguyên tắc sau đây:

- Các đối tác chỉ có thể tương tác (gửi thông điệp) tới các đối tượng biên.
- Các đối tượng biên chỉ có thể tương tác tới các đối tượng điều khiển hay đối tượng biên khác.
- Các đối tượng điều khiển có thể tương tác tới các đối tượng biên, các đối tượng thực thể hay các đối tượng điều khiển khác.
- Các đối tượng thực thể chỉ có thể tương tác với các đối tượng thực thể mà thôi.

2. CÁC THÔNG ĐIỆP

Thông điệp (message) là một đặc tả sự giao lưu giữa hai đối tượng, bao gồm sự truyền đạt một số thông tin và/hoặc sự yêu cầu thực hiện một hoạt động nào đó thuộc khả năng của bên nhận.

Hành động tạo nên bởi một thông điệp có thể là các hành động sau:

- *Gọi* (call): Yêu cầu thực hiện một thao tác của đối tượng nhận. Một đối tượng có thể gửi một thông điệp cho chính nó mà kết quả là huy động một thao tác riêng tư của nó (gọi cục bộ).
- *Trả lại* (return): Trả lại một giá trị cho bên gọi.
- *Gửi* (send): Gửi một tín hiệu tới một đối tượng.
- *Tạo lập* (create): Tạo lập một đối tượng mới.

- *Huỷ bỏ* (destror): Huỷ một đối tượng. Một đối tượng cũng có thể huỷ bỏ chính nó (terminate).

Khi một đối tượng gửi một thông điệp cho một đối tượng khác, thì đối tượng này trong hoạt động đáp ứng thông điệp trên lại có thể gửi thông điệp cho đối tượng khác, cứ thế tạo thành một luồng kích hoạt lan dần. Gọi đó là một *lộ trình điều khiển* (thread of control). Lộ trình điều khiển có thể *phẳng* (tuyến tính) hay *lồng* (do có sự trả lại). Bởi thế về hình thức tiếp nối thông điệp, ta phân biệt:

- *Thông điệp đồng bộ*, biểu diễn bằng mũi tên đầu tam giác đặc: ——►
  - Đó là một chuyển giao điều khiển lồng, tức là một lời gọi thao tác: bên gọi chuyển điều khiển cho bên bị gọi, rồi tạm ngưng để chờ bên bị gọi trả lại điều khiển.
  - Bên bị gọi thực hiện thao tác được yêu cầu, nếu cần có thể chuyển điều khiển cho một đối tượng khác và khi thao tác hoàn thành trả điều khiển về bên gọi, có thể kèm theo kết quả trả lời.
  - Thông điệp trả về có thể biểu diễn tường minh bởi mũi tên đứt nét <---- hoặc có thể bỏ qua, vì nó là mặc định ở thời điểm kết thúc thao tác.
- *Thông điệp không đồng bộ*, biểu diễn bằng mũi tên thường: ——→ (UML 1.3 trở về trước dùng mũi tên nửa ——\)
  - Đó là một chuyển giao điều khiển phẳng, thông qua sự gửi đi một tín hiệu. Thông diệp đi vào hàng đợi của bên nhận.
  - Bên gửi không cần biết thông điệp đã nhận chưa, mà tiếp tục làm việc ngay (tức là làm việc đồng thời).
  - Bên nhận thực hiện một thao tác và cũng có thể trả về một thông tin cho bên gửi. Nhưng nếu có sự trả lại, thì phải biểu diễn tường minh.

Ngoài ra, còn tùy thuộc vào nơi phát và nơi nhận thông điệp là có được biết rõ không mà UML 2.0 còn đưa thêm hai loại thông điệp nữa là:

- *Thông điệp mất hút* (lost message): là thông điệp mà nơi phát thì biết rõ, song nơi nhận thì không được biết (vì ở ngoài phạm vi mô tả, hoặc vì đó là một sự phát tán). Thông điệp mất hút được biểu diễn bằng một mũi tên có hình tròn đen ở cuối: ——►●

- *Thông điệp kiếm được* (fount message): là thông điệp mà nơi nhận thì biết rõ, song nơi phát thì không biết là đâu (vì ở ngoài phạm vi mô tả). Thông điệp kiếm được được biểu diễn bằng một mũi tên có hình tròn đen ở gốc: ●⟶

Chú thích: Các tác giả của UML khuyên rằng: Nên ít dùng thông điệp không đồng bộ. Đặc biệt là trong biểu đồ trình tự hệ thống, ở đó hệ thống là một khối nguyên và các tương tác giữa các đối tác và hệ thống là những điều khiển theo bước, chưa đi vào bên trong hệ thống để có các lời gọi theo kiểu thủ tục, vậy ta dùng thông điệp không đồng bộ. Còn lại nói chung thì ta nên dùng thông điệp đồng bộ, vì nó diễn tả các lời gọi thao tác lồng thông thường, được hỗ trợ trong phần lớn các ngôn ngữ lập trình. Còn các thông điệp mất hút và thông điệp kiếm được thì lại còn ít dùng hơn, bởi vì đó là các diễn tả cho các trường hợp hy hữu mà thôi.

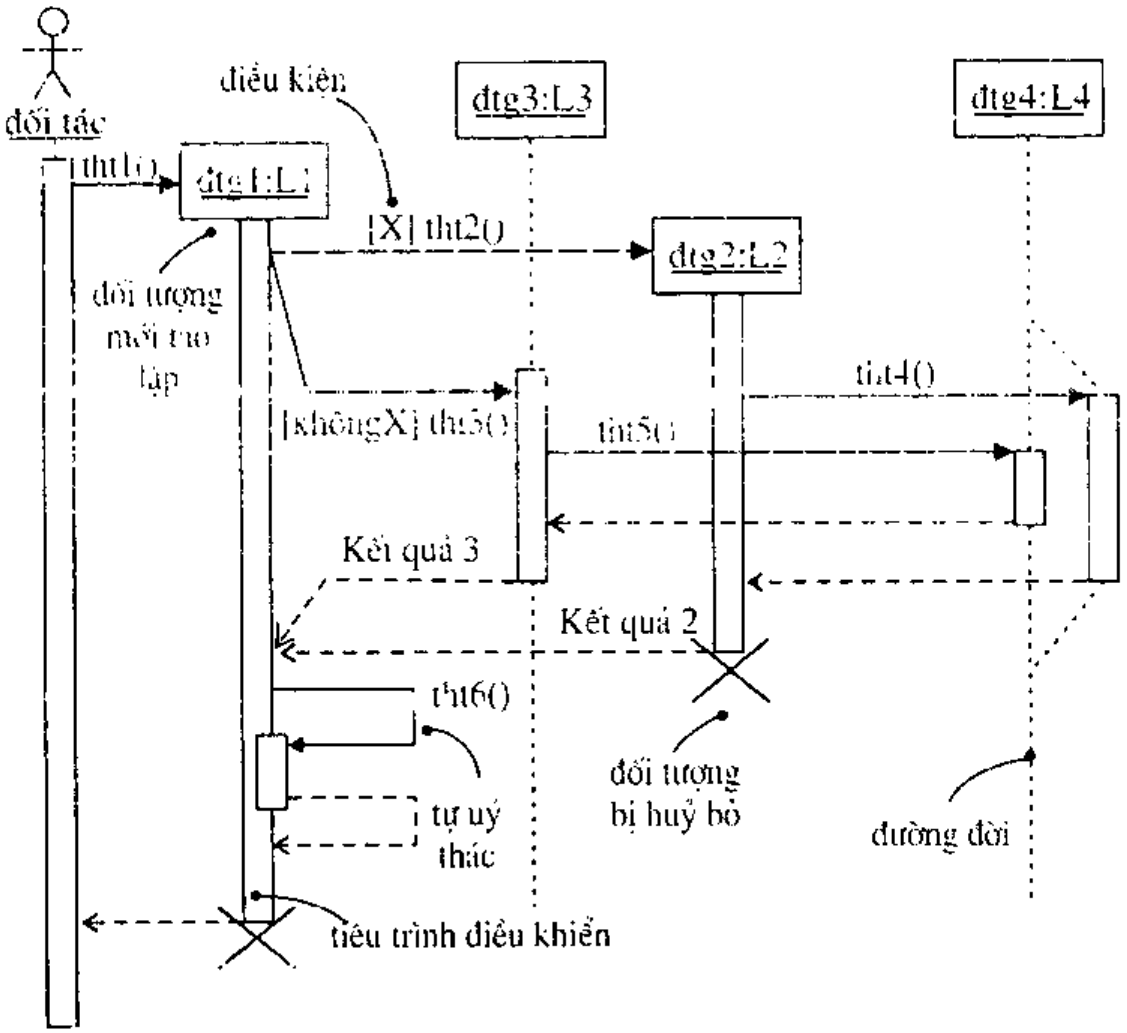
3. BIỂU ĐỒ TRÌNH TỰ

Biểu đồ trình tự (Sequence Diagram) là một trong hai biểu đồ tương tác chính, với chủ ý làm nổi bật trình tự theo thời gian của các thông điệp. Nó trình bày một tập hợp các đối tượng cùng với những thông điệp chuyển giao giữa chúng với nhau. Các đối tượng nói đây thường là các cá thể có tên hay khuyết danh của các lớp, song thế vào chỗ các đối tượng, cũng còn có thể là các đối tác, các hợp tác, các thành phần, các nút.

Biểu đồ trình tự được trình bày theo hai chiều:

- Chiều ngang bố trí các đối tượng. Các đối tượng được vẽ theo dạng hình chữ nhật hoặc bằng biểu tượng, dàn thành một hàng ngang trên đỉnh biểu đồ. Trật tự các đối tượng là không quan trọng, song các đối tượng khởi phát thông điệp nên vẽ ở phía trái. Các đối tượng mới được tạo lập thì vẽ thấp xuống, ngang với thông điệp tạo lập chúng.
- Chiều dọc là trục thời gian (hướng xuống dưới). Mỗi đối tượng có mang một trục đứng (vẽ đứt nét), gọi là *đường đời*. Đường đời của đối tượng sẽ kết thúc bằng một dấu gạch chéo, khi đối tượng bị huỷ bỏ. Các thông điệp (đồng bộ, không đồng bộ hay trả lời) là những mũi tên nằm ngang nối đường đời của hai đối tượng và được vẽ lần lượt từ trên xuống theo thứ tự thời gian. Nếu muốn làm rõ thời kỳ

hoạt động (tức là lúc đối tượng nắm giữ điều khiển) và làm rõ sự lồng nhau của các thông điệp, ta vẽ thêm trên đường đời một hay một số dải hẹp hình chữ nhật, gọi là *tiêu trình điều khiển* (focus of control). Lề phải và lề trái của biểu đồ có thể dùng để ghi các giải thích, các ràng buộc (Hình V.1).



Hình V.1. Biểu đồ trình tự với tiêu trình điều khiển, điều kiện chọn, đệ quy, tạo lập và huỷ bỏ.

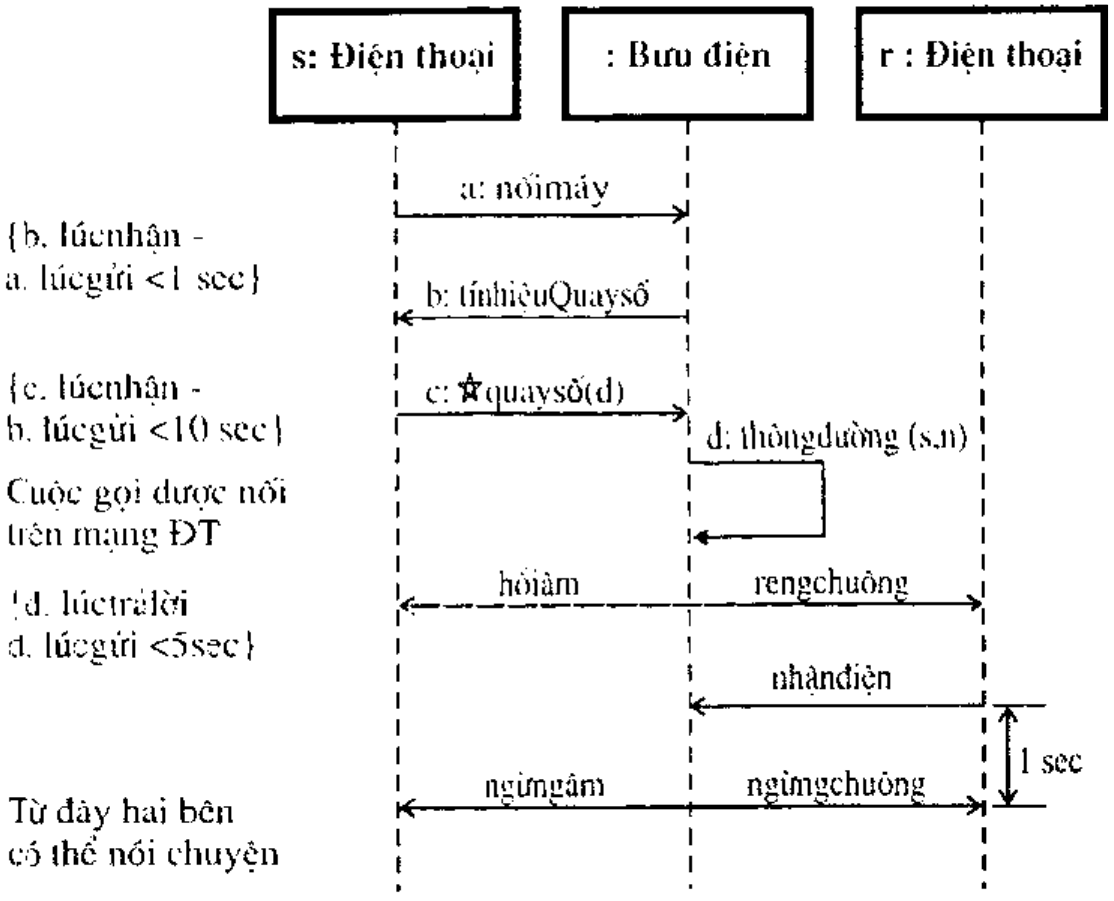
Mỗi thông điệp mang theo tên thông điệp dưới dạng thôngđiệp(ds tham số). Tên thông điệp lại có thể gắn thêm các tiền tố với các ý nghĩa như sau:

- *Một biểu thức trình tự* có dạng a: thường thì a là số thứ tự của thông điệp, nhưng cũng có thể là một nhãn (ký tự). Vì trật tự thông điệp đã biểu hiện rõ, nên ở biểu đồ trình tự các biểu thức trình tự thường ít dùng.

- *Một điều kiện chọn*, ở dạng [điều kiện], với nghĩa là thông điệp chỉ được gửi đi khi điều kiện này thoả mãn. Nếu vẽ nhiều thông điệp cùng xuất phát ở một điểm, mỗi thông điệp mang một điều kiện riêng thì ta có thể diễn tả:
  - Một rẽ nhánh chọn, nếu các điều kiện loại trừ lẫn nhau.
  - Một rẽ nhánh song song, nếu các điều kiện đó không loại trừ lẫn nhau.
- Một ký hiệu lặp ở dạng *, với nghĩa là thông điệp được lặp lại nhiều lần (thường thì sự lặp thực hiện trên nhiều đối tượng, do đối tượng nhận là một đối tượng bội).

Thí dụ:

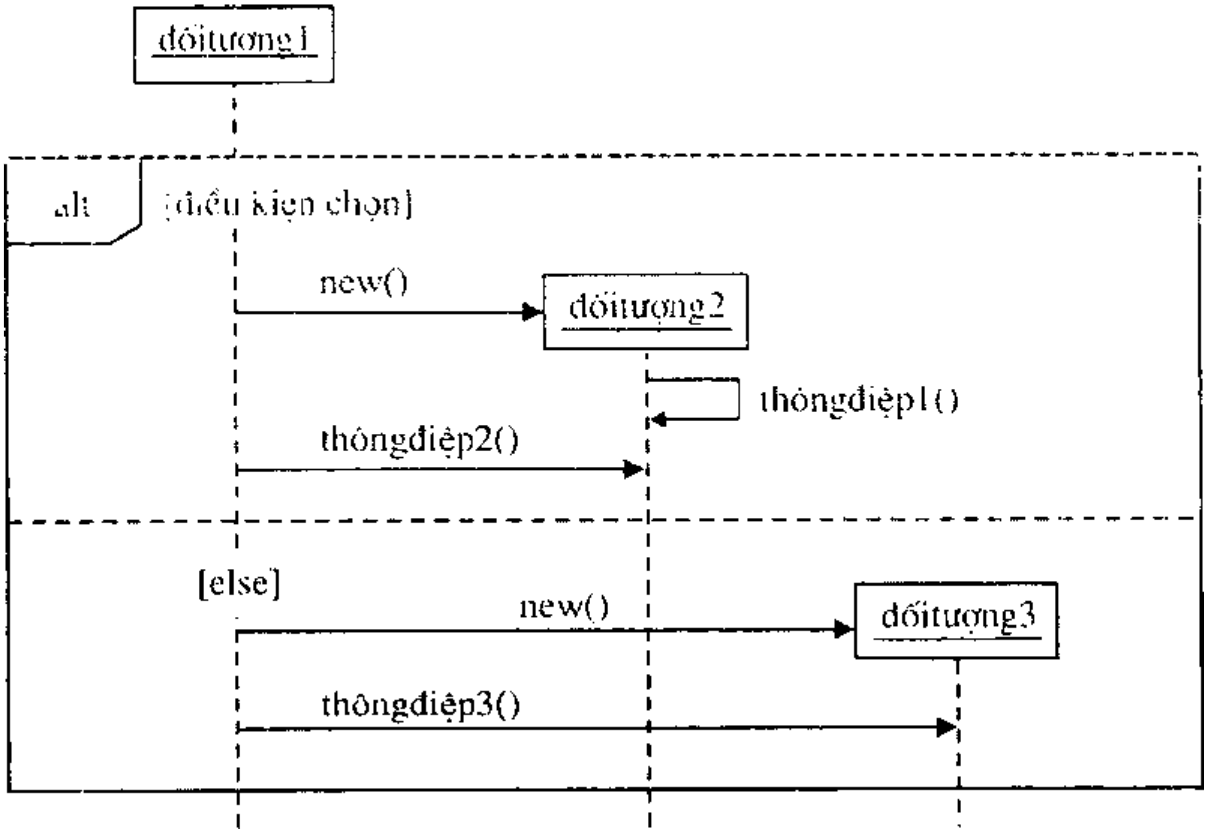
Hình V.2. thể hiện các quy ước trên trong một biểu đồ trình tự diễn tả một cuộc liên lạc bằng điện thoại. Các đối tượng đều là các đối tượng tương tranh (làm việc song song), do đó đều được vẽ với viền đậm. Các thông điệp đều là không đồng bộ (trừ d).



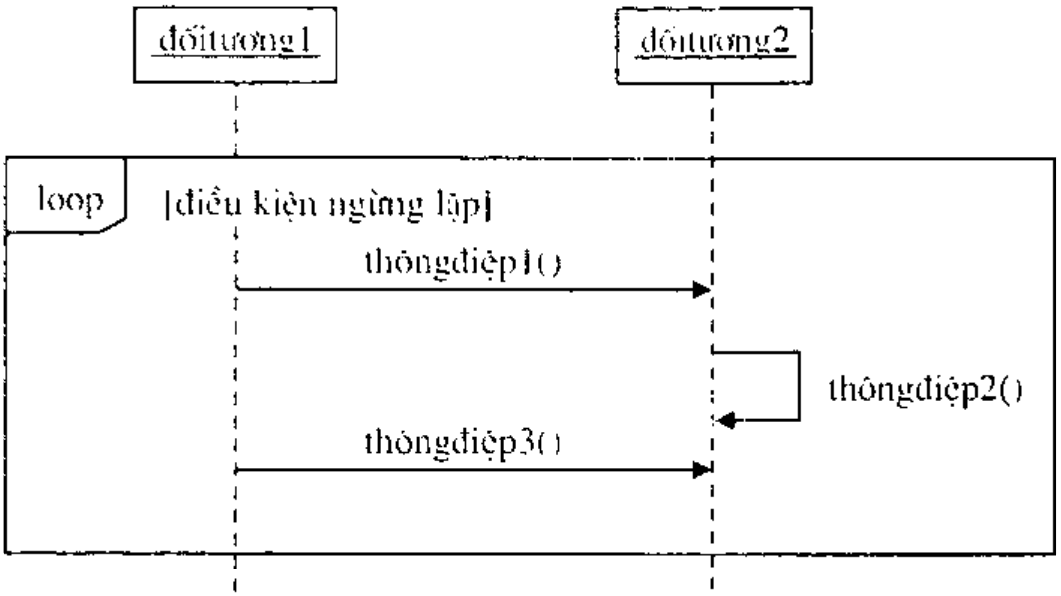
Hình V.2. Biểu đồ trình tự của một cuộc liên lạc bằng điện thoại

Chú thích

Cách thể hiện phép chọn và phép lặp nói ở trên chỉ áp dụng cho những trường hợp đơn giản, trong đó phần được chọn hay được lặp chỉ là một thông điệp. Còn trong trường hợp mà phần được chọn hay được lặp gồm nhiều thông điệp (nghĩa là chiếm cả một đoạn của biểu đồ) thì UML 2.0 đặt chúng vào trong một khung, với nhãn là alt (chọn) hay loop (lặp) với điều kiện chọn hay điều kiện lặp viết kèm, như ở các hình V.3 và V.4 dưới đây



Hình V.3. Chọn trong biểu đồ trình tự



Hình V.4. Lặp trong biểu đồ trình tự

5. CÁCH MÔ HÌNH HÓA TƯƠNG TÁC TRONG CA SỬ DỤNG VỚI BIỂU ĐỒ TRÌNH TỰ

Một ca sử dụng có thể gồm nhiều kịch bản tương ứng với nhiều luồng điều khiển khác nhau. Một biểu đồ trình tự chỉ có thể diễn tả một luồng điều khiển, mặc dầu trong biểu đồ cũng có thể diễn tả sự rẽ nhánh hay lặp đơn giản. Vậy thông thường ta lập một số biểu đồ trình tự cho một ca sử dụng. Một vài biểu đồ trong đó là chính, số biểu đồ còn lại diễn tả các lối rẽ khả dĩ hay các trường hợp mắc lỗi.

Để lập một biểu đồ trình tự cho một kịch bản của ca sử dụng ta tiến hành các bước sau:

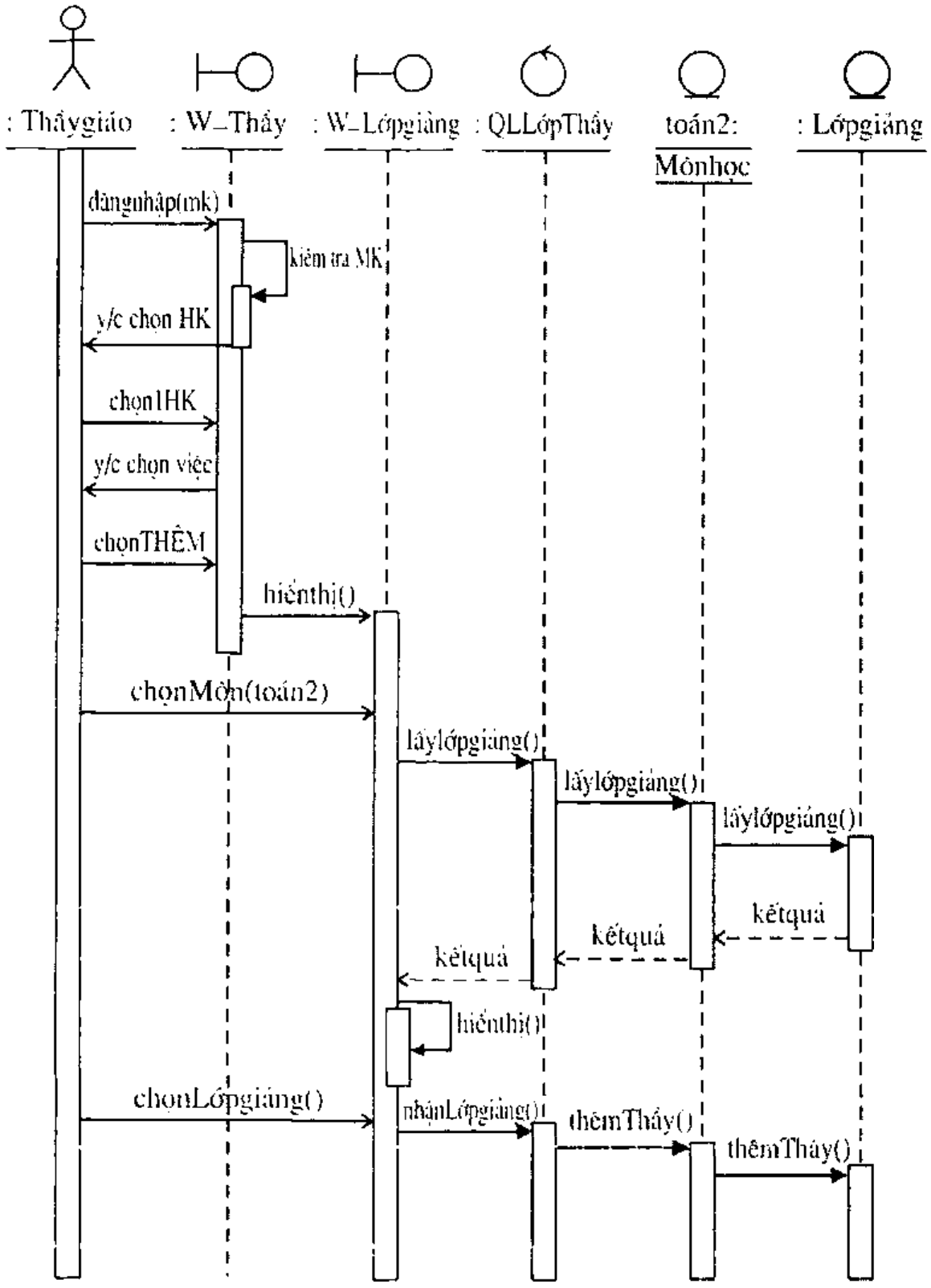
- Xem lại biểu đồ các lớp tham gia của ca sử dụng (đã lập ở bước 4) để xác định các cá thể nào của những lớp trong biểu đồ đó tham gia thực sự vào kịch bản đang xét. Muốn thế, dõi theo từng bước trong kịch bản để xem các đối tượng đóng vai trò gì trong bước đó.
- Dàn các đối tượng thành hàng ngang trên đỉnh biểu đồ trình tự. Bố trí các đối tượng quan trọng ở bên trái, các đối tượng phù trợ đặt ở bên phải. Đối tác, nếu có, thì đặt ở lề trái.
- Vẽ đường đời cho mỗi đối tượng (và đối tác). Trong đa số các trường hợp thì đối tượng tồn tại suốt thời gian tương tác. Tuy nhiên cũng có đối tượng được tạo lập và bị huỷ bỏ trong thời gian tương tác, cần được vẽ đường đời ngắn, từ lúc sinh tới lúc chết; với các

khuôn dập thích hợp cho các thông điệp đã tạo lập hay huỷ bỏ chúng.

- Xuất phát với thông điệp đã khởi đầu tương tác. Bố trí các thông điệp tiếp theo lần lượt từ trên xuống dưới, giữa các đường đời. Chỉ rõ đặc điểm của mỗi thông điệp (như là các tham số của nó). Nếu cần thì cho thêm giải thích về ngữ nghĩa của tương tác.
- Nếu thấy cần làm rõ thời kỳ hoạt động của các đối tượng và làm rõ sự lồng nhau của các thông điệp, thì vẽ thêm các tiêu trình điều khiển trên mỗi đường đời của đối tượng.
- Nếu cần làm rõ các ràng buộc phải có về thời gian và không gian, thì cho thêm các ký hiệu thời gian (như biểu thức trình tự, ký hiệu lặp *) và các ràng buộc không thời gian thích hợp.
- Nếu thấy cần diễn tả luồng điều khiển này một cách hình thức hơn, hãy đưa thêm các tiền đề và hậu đề cho mỗi thông điệp.

Thí dụ

Trở lại với hệ ĐKMH. Với ca sử dụng *Chọn môn học để giảng dạy*, thì ta đã chỉ ra trong bước 4 các lớp tham gia ca sử dụng đó (Hình IV.42). Giờ đây, với biểu đồ trình tự, ta có thể diễn tả sự tương tác giữa các đối tượng của các lớp đó để tạo nên kịch bản *Chọn môn học để giảng dạy/Thêm một lớp giảng*, như trên Hình V.5.



Hình V.5. Biểu đồ trình tự cho
"Chọn môn học để giảng dạy/Thêm một lớp giảng"

6. BIỂU ĐỒ GIAO TIẾP

Biểu đồ giao tiếp (Communication Diagram)[1] là một trong hai biểu đồ tương tác chính, với chủ ý làm nổi bật khung cảnh tổ chức của sự tương tác. Nó trình bày một tập hợp các đối tượng, các kết nối giữa các đối tượng đó cùng với những thông điệp chuyển giao giữa chúng với nhau. Các đối tượng nói đây thường là các cá thể có tên hay khuyết danh của các lớp, song thế vào chỗ các đối tượng, cũng còn có thể là các đối tác, các hợp tác, các thành phần, các nút.

Thông điệp được biểu diễn bằng một mũi tên nhỏ, vẽ dọc theo một kết nối giữa hai đối tượng, với hàm ý rằng nhờ có kết nối đó, mà bên gửi biết bên nhận để có thể gửi thông điệp. Nói cách khác, thông điệp không thể gửi đi nếu bên nhận không ở trong "tầm nhìn" của bên gửi. Có năm trường hợp của kết nối thể hiện tầm nhìn.

- Tồn tại liên kết giữa hai lớp của hai đối tượng. Bấy giờ kết nối được chua thêm từ khoá <<association>>.
- Đối tượng nhận là toàn cục đối với đối tượng gửi. Bấy giờ kết nối được chua thêm từ khoá <<global>>
- Đối tượng nhận là cục bộ trong thao tác (của bên gửi) đã gửi thông điệp đi. Bấy giờ kết nối được chua thêm từ khoá <<local>>.
- Đối tượng nhận là tham số của thao tác (của bên gửi) đã gửi thông điệp đi. Bấy giờ kết nối được chua thêm từ khoá <<parameter>>.
- Đối tượng nhận cũng chính là đối tượng gửi. Bấy giờ kết nối được chua thêm từ khoá <<self>>.

Các đối tượng cũng như các kết nối có thể được tạo lập hay bị huỷ bỏ trong thời gian thực hiện. Để phân biệt, ta chua thêm các xâu tính chất:

- {new} cho đối tượng hay kết nối mới được tạo lập;
- {destroyed} cho đối tượng hay kết nối bị huỷ bỏ;
- {transcient} cho đối tượng hay kết nối được thành lập trong thời gian thực hiện, rồi bị huỷ bỏ.

Mũi tên thông điệp (ở dạng ⟶ hay ⟶) phải kèm theo tiêu đề của thông điệp mà cú pháp đầy đủ là như sau:

[1] Tên cũ trong UML 1.x là biểu đồ hợp tác (Collaboration diagram).

Bướctrước Biểuthứctrìnhtự Trảlời := TênThôngđiệp(DsThamsố) ong đó trừ TênThôngđiệp là bắt buộc, còn các phần khác đều là ỳ ý.

Bướctrước: Một danh sách các số thứ tự, cách nhau bằng dấu phẩy và kết thúc bằng gạch chéo ('/'). Thông điệp chỉ được chuyển giao khi các thông điệp tương ứng với các số thứ tự cho trong danh sách này đã chuyển giao rồi. Vậy đây là một yêu cầu đồng bộ hoá của lộ trình điều khiển.

Thí dụ: 1.1, 2.3/

BiểuthứcTrìnhtự: là dãy số và ký tự kết thúc bởi dấu hai chấm (':') điển tả trình tự tiếp nối các thông điệp.

- Trường hợp đơn giản, thì đó chỉ là một số thứ tự, viết theo dạng ký pháp chấm: khi một thông điệp kích hoạt một thao tác mà bên trong thao tác này có một loạt thông điệp được gửi đi, thì những thông điệp này mang số thứ tự là số thứ tự của thông điệp trên kèm một dấu chấm rồi các số 1, 2, 3... lần lượt. Cách đánh số này cho thấy sự lồng nhau của các thông điệp.

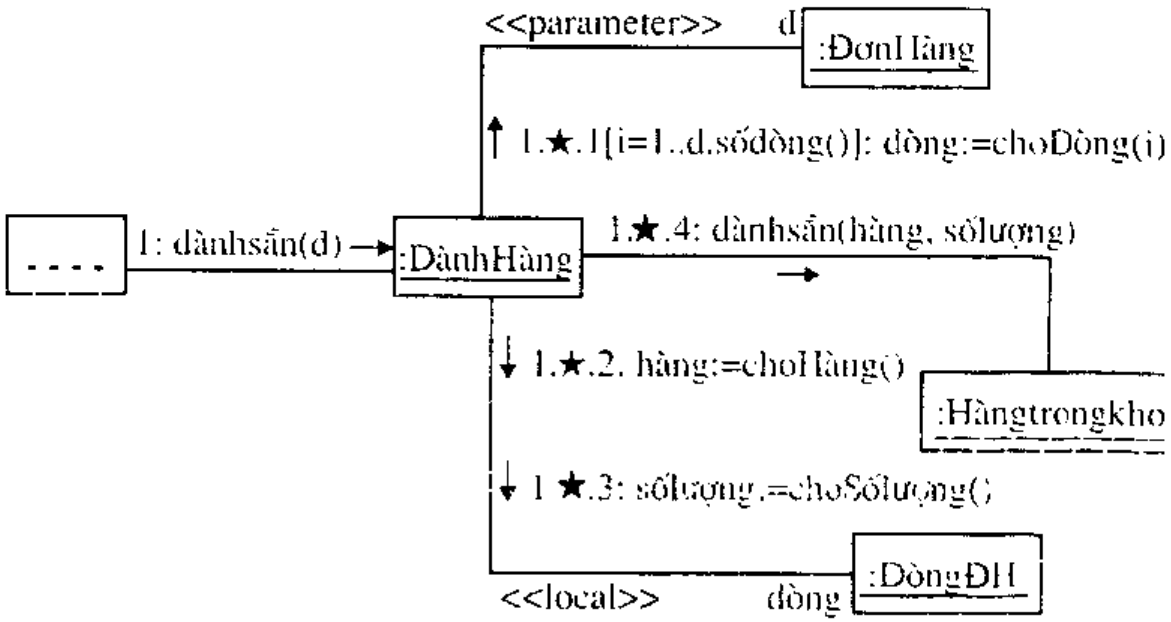
Chẳng hạn thông điệp 2.1.3 tiếp sau thông điệp 2.1.2 và cả hai đều ược gửi đi bởi cùng một thao tác thực hiện thông điệp 2.1.

- Các số lại có thể thay bởi các dãy ký tự, nhằm mục đích:
  - Diễn tả lộ trình điều khiển, như là A3, B4/ C3.1.2:
  - Diễn tả điều kiện chọn, như là 4.2. [x<0]:
  - Diễn tả thông điệp lặp, như là 1.1.*: hay 2.1.* [i:=1..n]:

Trảlời: là tên của kết quả trả về bên gửi. Tên này sau đó có thể dùng làm tham số cho thông điệp khác. Tên này có thể lấy là một biến cục bộ trong thao tác gửi thông điệp, hoặc là một thuộc tính của đối tượng gửi.

TênThôngđiệp(DsThamsố): là tên của thông điệp, kèm theo danh sách tham số (có thể rỗng). Tên này thường lấy trùng với thao tác cần được huy động ở bên nhận.

Một thí dụ về biểu đồ giao tiếp, vận dụng các ký pháp kể trên được cho trên Hình V.6.



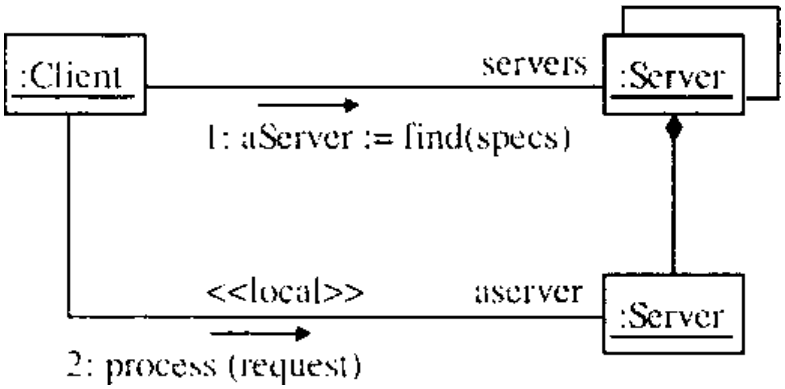
Hình V.6. Biểu đồ giao tiếp cho Đơn hàng đặt trước

Chương trình sau đây thực hiện biểu đồ giao tiếp trên, cho phép t hiểu rõ hơn về các ký pháp vận dụng trong đó:

```
dànhsẵn(d : Đơnhàng)
{
    DòngĐH dòng; Hàng hàng; int sốlượng;
    for (int i=1; i < d.sốdòng(); i++) {
        dòng = d.choDòng (i);
        hàng = dòng.choHàng();
        sốlượng = dòng.choSốlượng();
        hàngtrongKho.dànhsẵn(hàng, sốlượng);
    }
}
```

Nhiều khi một thông điệp gửi đi không hẳn là đến một đối tượng mà đến nhiều đối tượng một lúc. Chẳng hạn thông điệp tìm một đố tượng thích hợp tại đầu "nhiều" của một liên kết, hoặc tìm một cuố sách trong catalô. Bấy giờ ta nên diễn tả bằng đối tượng bội. *Đối tượn bội* (multiobject) là một ký pháp UML dùng để biểu diễn trong cl một ký hiệu nhiều đối tượng của cùng một lớp. Nó được vẽ thành hà đối tượng chồng lên nhau. Dùng đối tượng bội, ta không phải đề cậ quá sớm vào cách thức cài đặt (chẳng hạn không cần biết đấy sẽ là lớ

Vector hay lớp ArrayList, hay là gì). Hình V.7 cho một thí dụ về sử dụng đối tượng bội.



Hình V.7. Sử dụng đối tượng bội

6. CÁCH MÔ HÌNH HOÁ TƯƠNG TÁC TRONG CA SỬ DỤNG VỚI BIỂU ĐỒ GIAO TIẾP

Biểu đồ trình tự và biểu đồ giao tiếp là tương đương về ngữ nghĩa. Ta có thể chuyển đổi từ biểu đồ này sang biểu đồ kia mà không làm mất mát thông tin cơ bản. Tuy nhiên chúng có những thế mạnh riêng, và có những sắc thái biểu diễn mà bên này có, còn bên kia không có. Chẳng hạn biểu đồ trình tự có các tiêu trình điều khiển, cho thấy rõ đường đời và thời kỳ đang hoạt động hay tạm ngưng, lúc thành lập, lúc huỷ bỏ của đối tượng. Biểu đồ giao tiếp không làm rõ được các điều này. Ngược lại thì biểu đồ giao tiếp cho thấy các đối tượng kết nối với nhau như thế nào (<<global>>, <<local>>,...) hoặc diễn tả trường hợp đối tượng bội. Biểu đồ trình tự lại không thể hiện được các yếu tố này.

Vậy dùng biểu đồ nào để mô hình hoá sự tương tác? Câu trả lời là tuỳ ý, nghĩa là tuỳ lúc, tùy nơi và cả ... tuỳ thích. Đương nhiên là không nên dùng cả hai biểu đồ đồng thời để diễn tả cùng một tương tác, mà chỉ dùng một biểu đồ là đủ.

Để lập biểu đồ giao tiếp cho một kịch bản của ca sử dụng ta tiến hành các bước sau:

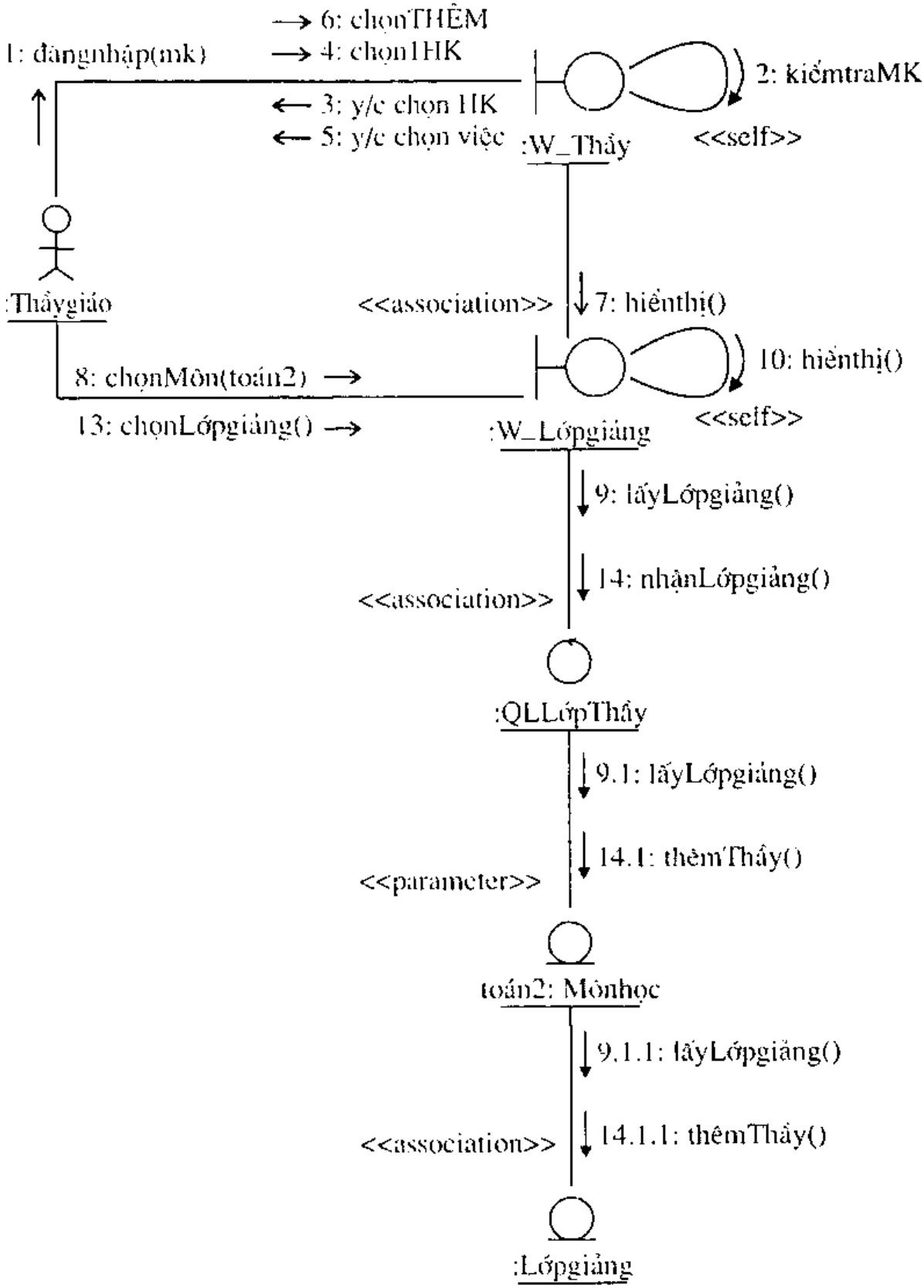
- Xem lại biểu đồ các lớp tham gia của ca sử dụng (đã lập ở bước 4) để xác định các cá thể nào của những lớp trong biểu đồ đó tham gia thực sự vào kịch bản đang xét. Muốn thế, dõi theo từng bước trong kịch bản để xem đối tượng đóng vai trò gì trong bước đó.

- Vẽ các đối tượng như là các đỉnh của đồ thị. Bố trí các đối tượng quan trọng vào giữa, các đối tượng khác ở xung quanh.
- Gán những tính chất đầu tiên cho mỗi đối tượng. Nếu có đối tượng nào đó mà tính chất của nó (giá trị thuộc tính, giá trị gắn nhãn, trạng thái, vai trò) thay đổi một cách đáng kể trong thời gian tương tác, hãy vẽ thêm bản sao cho các đối tượng đó, cập nhật chúng với các giá trị mới và nối với đối tượng gốc bằng một thông điệp với khuôn dập như là <<become>> hay <<copy>>.
- Xác định các kết nối giữa các đối tượng, cùng với các thông điệp có thể có trên đó.
  - Vẽ các kết nối liên kết trước; đo là những kết nối quan trọng nhất vì chúng diễn tả cấu trúc tĩnh.
  - Tiếp theo vẽ các kết nối khác và ghi chú với các khuôn dập thích hợp (như là <<global>>, <<local>>) để làm rõ các đối tượng liên hệ với nhau theo kiểu gì.
- Xuất phát với thông điệp đã khởi đầu tương tác. Bố trí lần lượt các thông điệp tiếp theo trên các kết nối thích hợp, thêm các số thứ tự thích hợp. Dùng ký pháp chấm để làm rõ sự long nhau.
- Nếu thấy cần các rang buộc về không gian và thời gian, hãy thêm vào thông điệp các đấu hiệu thời gian và đính kèm các ràng buộc về không gian hay thời gian.
- Nếu thấy cần diễn tả luồng điều khiển này một cách hình thức hơn, hãy đưa thêm tiền đề và hậu đề cho mỗi thông điệp.

Cũng như với biểu đồ trình tự, một biểu đồ hợp tác chỉ có thể diễn tả một luồng điều khiển, mặc dù trong biểu đồ cũng có thể diễn tả sự rẽ nhánh hay lặp đơn giản. Vậy thông thường ta lập một số biểu đồ hợp tác cho một ca sử dụng. Một vài biểu đồ trong đó là chính, số biểu đồ còn lại diễn tả các lối rẽ khả dĩ hay các trường hợp mắc lỗi.

Thí dụ:

Trở lại hệ ĐKMH, với biểu đồ giao tiếp ta có thể diễn tả kịch bản *Chọn môn học để giảng dạy/Thêm một lớp giảng* như trên Hình V.8.



Hình V.8. Biểu đồ giao tiếp cho
'Chọn môn học để giảng dạy/Thêm một lớp giảng'

7. ĐỐI CHIẾU VÀ CHỈNH SỬA CÁC MÔ HÌNH CẤU TRÚC VÀ TƯƠNG TÁC

Tới đây ta đã lập các biểu đồ cấu trúc (biểu đồ các lớp lĩnh vực, biểu đồ các lớp tham gia) và đã lập các biểu đồ tương tác (biểu đồ trình tự, biểu đồ giao tiếp). Ta cần phải đối chiếu các biểu đồ đó với nhau để chỉnh sửa lại các chỗ không phù hợp giữa chúng.

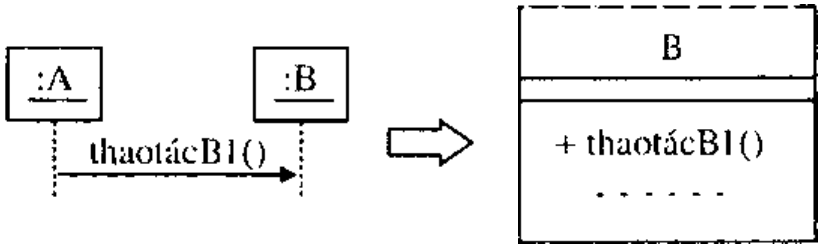
a) Thêm bớt các lớp

Một số lớp đã được phát hiện thêm khi lập các biểu đồ tương tác, cần bổ sung chúng vào các biểu đồ cấu trúc. Ngược lại có những lớp trong biểu đồ cấu trúc không hề có đối tượng tham gia vào một biểu đồ tương tác nào, cần phải loại bỏ chúng đi.

b) Thêm và chỉnh lý các thao tác trong các lớp

Khi có một thông điệp được gửi từ một đối tượng của lớp A sang một đối tượng của lớp B (Hình V.9), thì:

- Trong lớp B (bên nhận) phải có một thao tác công cộng tương ứng với thông điệp đó. Tên thông điệp và thao tác phải phu nợp ngữ nghĩa với nhau, thông thường thì thông điệp nên lấy trùng tên với thao tác. Danh sách các tham số của thao tác phải phù hợp các yêu cầu của thông điệp.
- Trong lớp A (bên gửi) phải có một thao tác có trách nhiệm gửi đi thông điệp trên.



Hình V.9. Tương ứng giữa thông điệp và thao tác

Tuy nhiên quy luật trên cũng có những ngoại lệ:

- *Thông điệp đến một đối tác:*

- Nếu đối tác là người, thì thông điệp chỉ yêu cầu người làm một việc gì đó ghi trong bản hướng dẫn sử dụng, chứ không phải là một thao tác (tức là một chương trình máy tính).
- Nếu đối tác là một thiết bị, thì thông điệp chỉ kích hoạt một cảm biến hay một động cơ ở thiết bị đó.
- Nếu đối tác là một hệ thống ngoài, thì nó phải được đại diện bởi một lớp chứa các giao thức thích hợp. Bấy giờ thông điệp sẽ kích hoạt một thao tác trong lớp này.

- *Thông điệp đến một lớp giao diện GUI:*

Yêu cầu của thông điệp được cài đặt thành các yếu tố đồ hoạ của GUI, như là trường hay nút.... chứ không là một thao tác.

- *Các thao tác riêng tư:*

Đó là những thao tác dùng trong nội bộ, chỉ tương ứng với các thông điệp đệ quy, chứ không tương ứng với thông điệp từ đối tượng khác gửi đến.

c) Thêm các kiểu cho các thuộc tính, các tham số và các trả lời

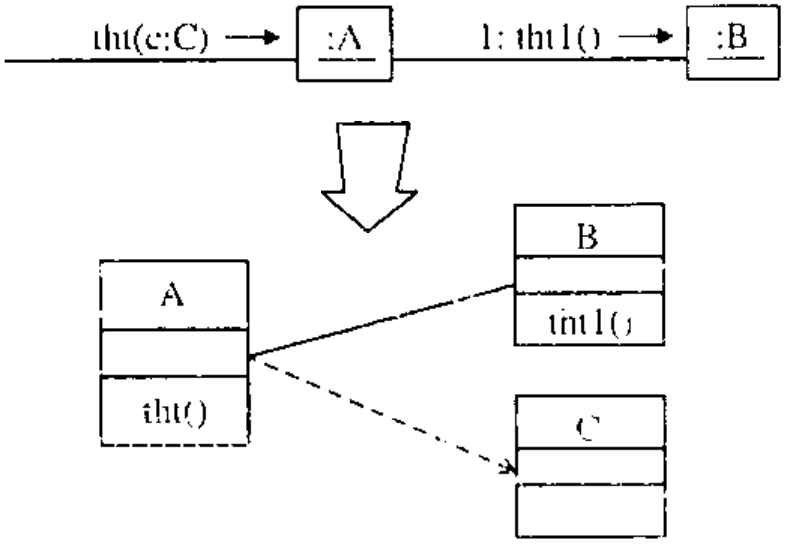
Các kiểu có thể là các kiểu nguyên thuỷ hay các kiểu do người dùng đưa vào, hoặc các lớp. Tuy nhiên chưa nên vội dùng tên các kiểu của một ngôn ngữ lập trình cụ thể nào, mà vẫn dùng các tên chung, như vậy linh hoạt hơn.

d) Thêm và chỉnh lý các mối liên quan giữa các lớp

Nghiên cứu các biểu đồ tương tác, ta có thể xác lập và chỉnh sửa các mối liên quan (liên kết và phụ thuộc) giữa các lớp trong mô hình tĩnh.

Khi đối tượng của một lớp A gửi một thông điệp cho một đối tượng của một lớp B, thì giữa hai đối tượng đó có một kết nối với ý nghĩa là bên nhận thuộc phạm vi hiểu biết (hay ở trong tầm nhìn) của bên gửi:

- Nếu sự hiểu biết đó là vốn có từ lâu, thì đây là một biểu hiện của sự liên kết (association) giữa A và B.
- Nếu sự hiểu biết đó là mới được cung cấp (chẳng hạn thông qua việc truyền tham số hay việc nhận kết quả trả lời), thì đấy là một biểu hiện của sự phụ thuộc (dependency) của A vào B (Xem hình V.10).



Hình V.10. Liên kết hay phụ thuộc

Đối với các liên kết, nhờ biểu đồ tương tác, ta nên xác định chiều lưu hành trên đó, nếu quả thật các thông điệp gửi đi trên đó chỉ đi theo một chiều.

§ 2. BƯỚC 6: MÔ HÌNH HOÁ SỰ ỨNG XỬ

1. MỤC ĐÍCH

Về mặt hành vi ứng xử, ta có thể phân biệt hai loại đối tượng:

- *Các đối tượng bị động:* Đó là các đối tượng mà cách ứng xử của chúng không hề bị thay đổi theo thời gian. Cùng một thông điệp, thì dù được gửi đến bất cứ lúc nào, cũng luôn được đáp ứng theo một cách nhất định (kết quả đáp ứng có thể là một thông tin hay một dịch vụ).
- *Các đối tượng chủ động:* Đó là các đối tượng mà cách ứng xử của chúng thay đổi theo thời gian. Khi một thông điệp đến, thì còn phải tuỳ thuộc vào trạng thái bên trong của đối tượng vào lúc đó, mà đối tượng có những cách đáp ứng khác nhau. Loại đối tượng này có đời sống thực sự: sinh ra, trải qua một số trạng thái trong đời và chết. Chính nhờ có trạng thái mà đối tượng chủ động có khả năng điều khiển.

Mục đích của việc mô hình hoá sự ứng xử (bước 6 trong tiến trình 10 bước) chính là mô tả cách phản ứng của các đối tượng chủ động trước các sự kiện (thông điệp) đến với chúng. Công cụ mô tả được dùng ở đây là các biểu đồ máy trạng thái.

Biểu đồ máy trạng thái (State Machine Diagram[1]) là một đồ thị hữu hạn có hướng, trong đó mỗi nút là một trạng thái, mỗi cung là một dịch chuyển trạng thái. Nó diễn tả quy luật thay đổi trạng thái và hành vi của một đối tượng (chủ động) tuỳ thuộc vào các sự kiện xảy đến với nó. Vậy đây chính là một ô tô mát hữu hạn có cái vào và cái ra. Cái vào là các sự kiện, cái ra là các hành động, các hoạt động và cả các sự kiện.

Dưới đây ta sẽ lần lượt trình bày rõ hơn về các yếu tố tạo nên một biểu đồ máy trạng thái, và sự vận dụng chúng để miêu tả sự ứng xử của đối tượng.

2. CÁC SỰ KIỆN

Sự kiện (event) là một điều gì đó xảy ra bất chợt đối với hệ thống, và có ảnh hưởng tới hành vi của hệ thống. Phân biệt *sự kiện trong*, bắt nguồn từ bên trong hệ thống (chẳng hạn một lỗi tràn ô), với *sự kiện ngoài*, bắt nguồn từ bên ngoài hệ thống (chẳng hạn một cái nhấp chuột).

Đối với một đối tượng, thì một sự kiện có thể đến với nó thông qua sự tiếp nhận một thông điệp từ một đối tượng khác gửi tới. Trong cái vỏ thông điệp này, thì ta có hai loại sự kiện:

- *Sự kiện gọi* (call event): Đó là sự tiếp nhận một lời gọi tới một thao tác và đây là một sự kiện đồng bộ.
- *Sự kiện tín hiệu* (signal event): Đó là sự tiếp nhận một tín hiệu và đây là một sự kiện không đồng bộ.

  Tín hiệu là một đối tượng có tên, được gửi đi một cách không đồng bộ bởi một đối tượng và được tiếp nhận bởi một đối tượng khác. Về ngữ nghĩa, thì đó là một đặc tả cho một kích thích được truyền đi giữa các đối tượng.

Ngoài ra, một sự kiện có thể xảy tới cho một đối tượng mà không phải là thông qua sự tiếp nhận thông điệp. Đó là các sự kiện diễn tả cho một sự thay đổi khách quan nào đó. Có hai loại sự kiện này:

---

[1] Trong UML 1.x gọi là State Chart Diagram

- *Sự kiện thời gian:* Đó là sự kiện biểu diễn cho sự đi qua một mốc thời gian nào đó. Thường diễn tả với từ khoá after, chẳng hạn *after 2 seconds*.
- *Sự kiện thay đổi:* Đó là sự kiện biểu diễn cho một sự thay đổi trạng thái, hoặc sự thoả mãn một điều kiện nào đó. Thường diễn tả với khoá when, chẳng hạn *when độcao < 1000* Khi diễn tả như vậy, thì ta xem như là điều kiện nói trên đã được kiểm tra một cách liên tục, tuy nhiên khi cài đặt thì phải thực hiện sự kiểm tra đó theo thời gian rời rạc.

Đối với một đối tượng chủ động, thì các sự kiện chính là các kích thích đã gây nên sự thay đổi trạng thái (và hành vi) của nó. Vì vậy các sự kiện là những cái vào đối với biểu đồ máy trạng thái của đối tượng.

3. CÁC TRẠNG THÁI

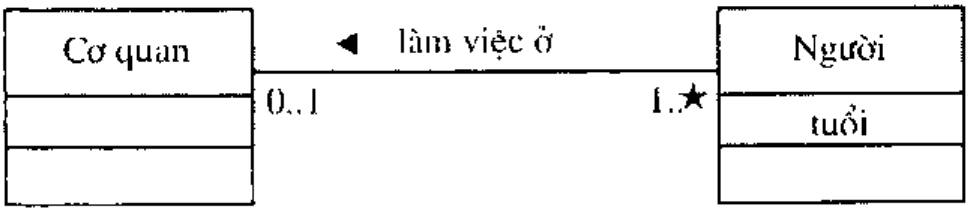
Trạng thái (state) của một đối tượng thuộc một lớp là một sự trừu tượng hoá hay sự tổ hợp của một tập hợp các giá trị thuộc tính và kết nối có thể, mà đối tượng của lớp đó có thể nhận. Nói một cách chính xác, với ngôn từ của toán học, thì trạng thái là một lớp tương đương trong tập hợp các giá trị thuộc tính và kết nối của đối tượng.

Đặc điểm của trạng thái là có tính hữu hạn, tính kéo dài trong thời gian và tính ổn định.

- *Tính hữu hạn:* Số trạng thái có thể có của một đối tượng phải là hữu hạn. Nói cách khác là cho dù tập hợp các giá trị thuộc tính có thể là vô hạn, thì ta cũng không chấp nhận một sự phân hoạch tập hợp đó (theo một quan hệ tương đương nào đó) thành vô hạn trạng thái được.
- *Tính kéo dài theo thời gian:* Ở mỗi thời điểm, một đối tượng (chủ động) đang tồn tại phải luôn luôn có một trạng thái nhất định, nghĩa là không thể có trạng thái không biết hay không xác định. Trạng thái đó không phải là nhất thời, mà phải kéo dài trong một khoảng thời gian hữu hạn. Chính vì vậy mà có thể quan niệm một trạng thái như là một giai đoạn (có thể lặp lại) trong đời của một đối tượng.
- *Tính ổn định:* Trong thời gian kéo dài của trạng thái nói trên, thì hành vi của đối tượng (thực hiện một hoạt động nào đó, hay chờ một sự kiện nào đó) là không thay đổi.

Thí dụ:

Cho một biểu đồ lớp như trên Hình V.11.



Hình V.11. Một biểu đồ lớp

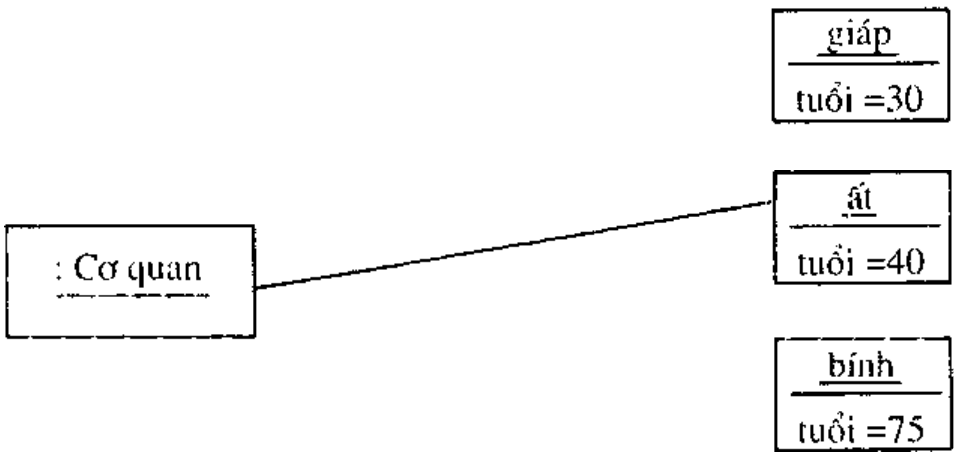
Với mỗi đối tượng trong lớp Người, thì tuỳ thuộc vào:

- tuổi của người đó, và
- có kết nối với một cơ quan nào không.

mà đối tượng đó là ở một trong ba trạng thái sau đây:

+ Có việc làm (có kết nối với một cơ quan)
+ Thất nghiệp (tuổi < 60 và không kết nối với cơ quan nào)
+ Nghỉ hưu (tuổi > 60)

Chẳng hạn, biểu đồ đối tượng trên Hình V.12 thể hiện một trường hợp cụ thể của biểu đồ lớp cho ở trên, trong đó Giáp là thất nghiệp, Ất có việc làm và Bính nghỉ hưu.



Hình V.12. Một biểu đồ đối tượng

Như trên đã nói, trong biểu đồ máy trạng thái, thì mỗi nút là một trạng thái, và nó được biểu diễn theo ba dạng như trên Hình V.13. Mỗi biểu đồ máy trạng thái có một và chỉ một trạng thái đầu, không hoặc nhiều trạng thái kết thúc.

Trạng thái đầu Trạng thái trung gian Trạng thái kết thúc

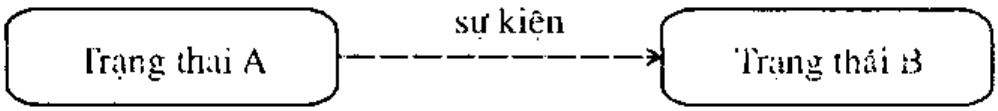
Hình V.13. Biểu diễn của trạng thái

4. CÁC DỊCH CHUYỂN

Một *dịch chuyển*, nói rõ hơn là dịch chuyển trạng thái (state transition), là một sự thay đổi từ một trạng thái này sang một trạng thái khác.

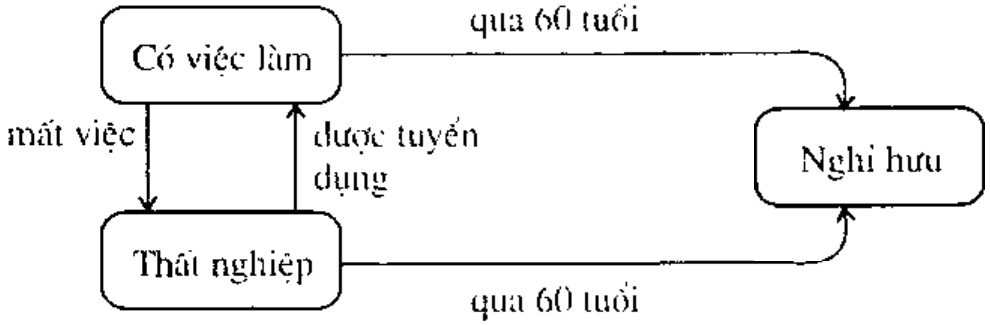
Bước dịch chuyển là tức thời (không mất thời gian), vì đối tượng luôn luôn phải ở trong một trạng thái nhất định. Mặt khác thì một bước dịch chuyển chỉ xảy ra khi có một sự kiện nhất định xuất hiện, gọi là sự kiện kích thích bước dịch chuyển.

Trong biểu đồ máy trạng thái thì một dịch chuyển được biểu diễn bằng một cung nối liền hai trạng thái, trên đó có ghi sự kiện kích thích nó (Hình V.14).



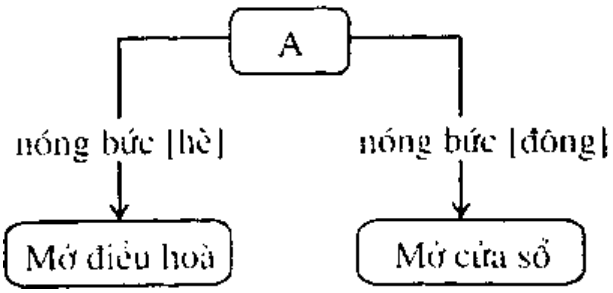
Hình V.14. Biểu diễn dịch chuyển

Trở lại thí dụ Người làm công ở trên, thì giữa ba trạng thái của Người, ta có các dịch chuyển như trên Hình V.15.



Hình V.15. Dịch chuyển trạng thái với một Người

Đôi khi bước dịch chuyển trạng thái xảy ra không những chỉ vì sự xuất hiện của sự kiện kích thích nó, mà còn đòi hỏi thêm một điều kiện Bun kèm theo phải được thoả mãn. Điều kiện kèm theo đó được gọi là một *cảnh giới* (guard), và được viết tiếp theo với sự kiện, trong một cặp ngoặc vuông. Cảnh giới có thể được viết theo văn tự tự do (như trên Hình V.16), nhưng tốt hơn là viết theo một cú pháp hình thức (chẳng hạn Java hay OCL), vì như thế sẽ thuận lợi cho việc sản sinh mã của các công cụ CASE. Người ta thường dùng cảnh giới để làm cho ô tô mát trở nên đơn định.



Hình V.16. Sự kiện có cảnh giới

5. CÁC CÁI RA

Như trên đã nói, máy trạng thái là một ô tô mát hữu hạn có vào/ra. Các cái vào là những sự kiện xảy đến đối với đối tượng. Còn các cái ra thì có ba loại: hành động, hoạt động và sự kiện.

Hành động (action) là một việc làm mà thời gian thực hiện không đáng kể, có thể xem là tức thời và trọn gói (không bao giờ thực hiện dở dang). Một hành động có thể là một lời gọi đến một thao tác, một sự

gửi tín hiệu đến một đối tượng khác, một sự tạo lập hay huỷ bỏ đối tượng.

Hoạt động (activity) là một thao tác mà thời gian thực hiện là đáng kể và có thể bị ngưng ngắt nửa chừng.

Các cái ra của máy trạng thái (hành động, hoạt động và sự kiện) luôn luôn chỉ xuất hiện khi có một sự kiện nhất định xảy tới. Ta ghi nhận điều đó (trên biểu đồ) bằng cách viết:

sự kiện tới/ hành động ra

sự kiện tới/hoạt động ra

sự kiện tới ^ sự kiện ra

Tuy nhiên sự kiện tới ở đây có thể là một sự kiện kích thích một bước dịch chuyển trạng thái, mà cũng có thể là một sự kiện không kích thích dịch chuyển trạng thái (xảy ra bên trong trạng thái). Như vậy cái ra của máy trạng thái gắn liền với cả dịch chuyển và trạng thái.

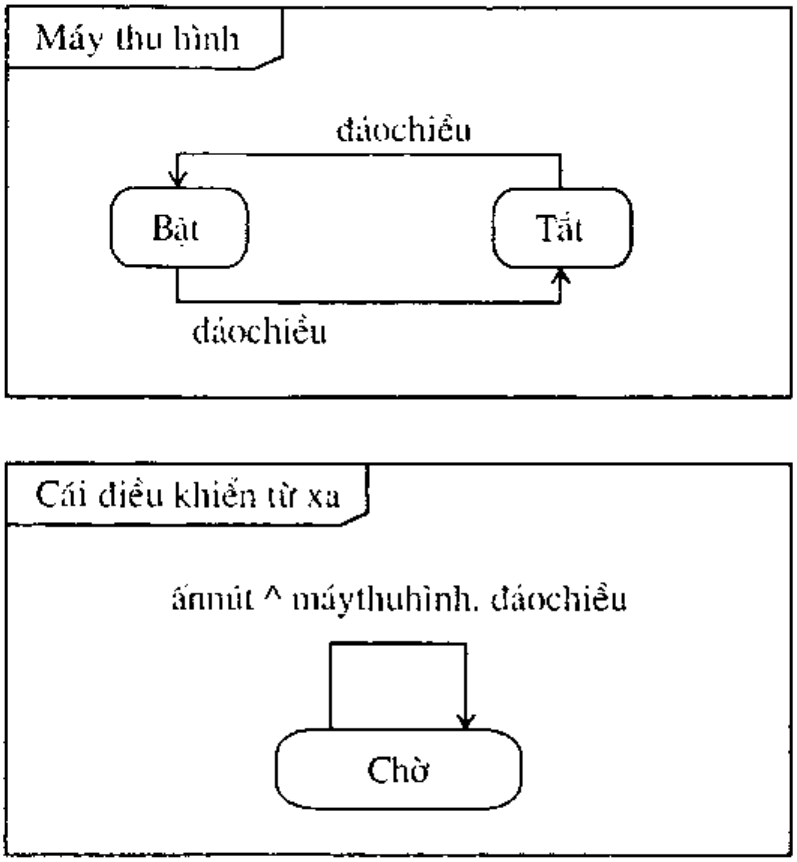
a) Ra theo bước dịch chuyển

Vì bước dịch chuyển là tức thời, vậy cái ra theo bước dịch chuyển chỉ có thể là hành động và sự kiện, mà không thể là hoạt động. Hành động ra theo bước dịch chuyển có thể sử dụng các tham số của sự kiện vào tương ứng và các thuộc tính của đối tượng. Sự kiện ra là một sự kiện được gửi tới một đối tượng đích nào đó. Như vậy cú pháp hoàn chỉnh cho một dịch chuyển sẽ là:

sựkiệntới(cácthamsố)[điềukiện]/thaotác(cácthamsố)^
đốitượngđích. sựkiệngửiđi (cácthamsố)

Thí dụ

Một máy thu hình có thể bật hay tắt nhờ một cái ngắt đảo chiều. Cái điều khiển từ xa có một nút ấn on/off, cứ mỗi lần ấn nút thì bật hay tắt máy thu hình. Các máy trạng thái diễn tả hành vi của máy thu hình và của cái điều khiển từ xa sẽ bao gồm các yếu tố (trong đó có sự kiện xuất), như trên Hình V.17.



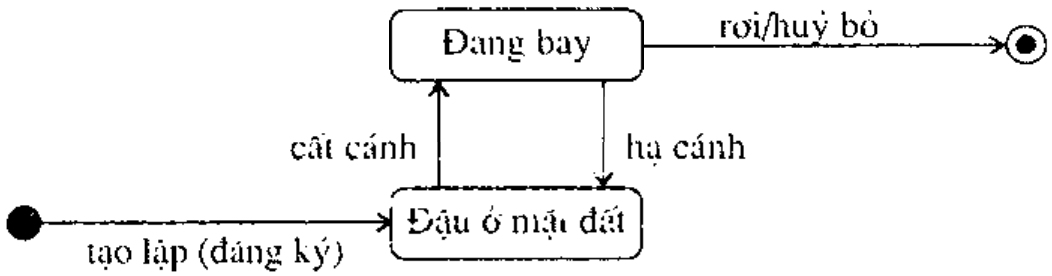
Hình V.17. Gửi sự kiện đi từ một dịch chuyển

Trường hợp đặc biệt là các dịch chuyển liên quan tới trạng thái vào hay trạng thái kết thúc:

- Dịch chuyển từ trạng thái vào phải được kích thích bởi sự kiện tạo lập đối tượng;
- Dịch chuyển đến một trạng thái kết thúc phải đồng thời huỷ bỏ đối tượng.

Thí dụ:

Hình V.18. diễn tả đời sống của một đối tượng máy bay. Sự kiện tạo lập cho phép đăng ký một máy bay. Khi bị rơi, đối tượng máy bay đó phải bị huỷ bỏ.



Hình V.18. Tạo lập và huỷ bỏ đối tượng

b) Ra ở một trạng thái

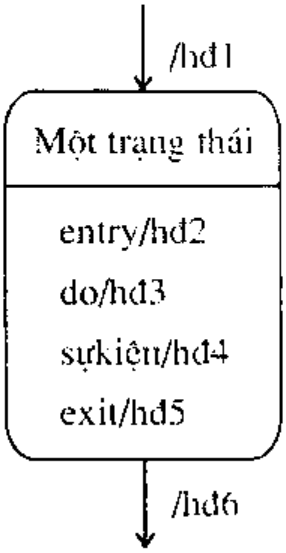
Có những sự kiện xảy ra khi đối tượng đang ở một trạng thái, nhưng không gây ra một bước dịch chuyển trạng thái, mà chỉ kích hoạt một hành động hay hoạt động nào đó. Các cặp sự kiện/cái ra này sẽ được liệt kê trong lòng (tức là trong hình chữ nhật tròn góc) của trạng thái.

Trong số các sự kiện loại này, có ba sự kiện mặc định và đó là:

- entry: sự kiện vào trạng thái, chỉ có thể kích hoạt một hành động;
- exit: sự kiện ra khỏi trạng thái, chỉ có thể kích hoạt một hành động;
- do: sự kiện kích hoạt một hoạt động nhất thiết phải diễn ra ở trạng thái tương ứng.

Tóm lại, thì có sáu thời điểm xung quanh một trạng thái ở đó có thể xuất hành động hay hoạt động, được chỉ ra trên Hình V.18 và kể lần lượt như sau:

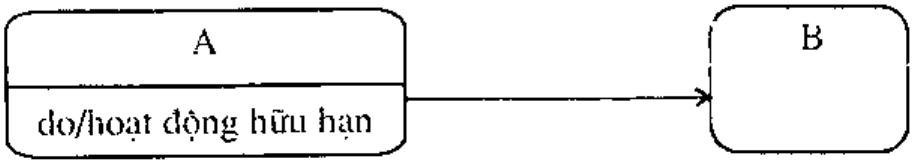
- hành động gắn với bước dịch chuyển tới trạng thái (hđ1);
- hành động khi vào trạng thái (hđ2);
- hoạt động định sẵn trong trạng thái (hđ3);
- hành động gắn với các sự kiện trong (hđ4);
- hành động khi ra khỏi trạng thái (hđ5);
- hành động gắn với bước dịch chuyển tới một trạng thái khác (hđ6).



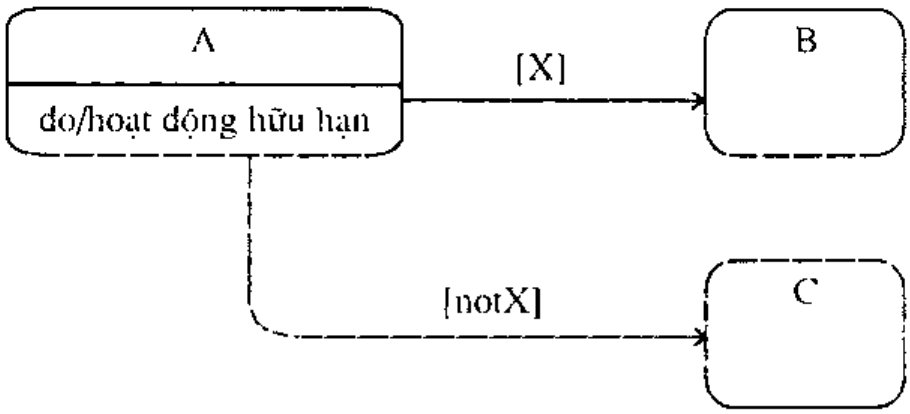
Hình V.19. Sáu cơ hội xuất hành động và hoạt động

Nhớ rằng hoạt động khác hành động ở chỗ cần có thời gian, và như thế có thể bị ngưng ngắt bất cứ lúc nào, khi xảy ra bước dịch chuyển rời khỏi trạng thái. Về mặt này, ta phân biệt hai hoạt động:

- *Hoạt động liên tục*, là hoạt động lặp và kéo dài, chỉ dừng khi có bước dịch chuyển rời khỏi trạng thái;
- *Hoạt động hữu hạn*, là hoạt động tuần tự và nhất thiết sẽ dừng sau một khoảng thời gian nhất định hoặc khi một điều kiện nào đó được thoả mãn. Khi một hoạt động loại này đi đến kết thúc, thì phải ra khỏi trạng thái. Bước dịch chuyển đó không do một sự kiện từ bên ngoài kích thích và được gọi là bước *dịch chuyển tự động* (Hình V.120). Tuy không có sự kiện, song trong bước dịch chuyển tự động, ta có thể dùng các cảnh giới để thực hiện một sự lựa chon (Hình V.21).



Hình V.20. Dịch chuyển tự động

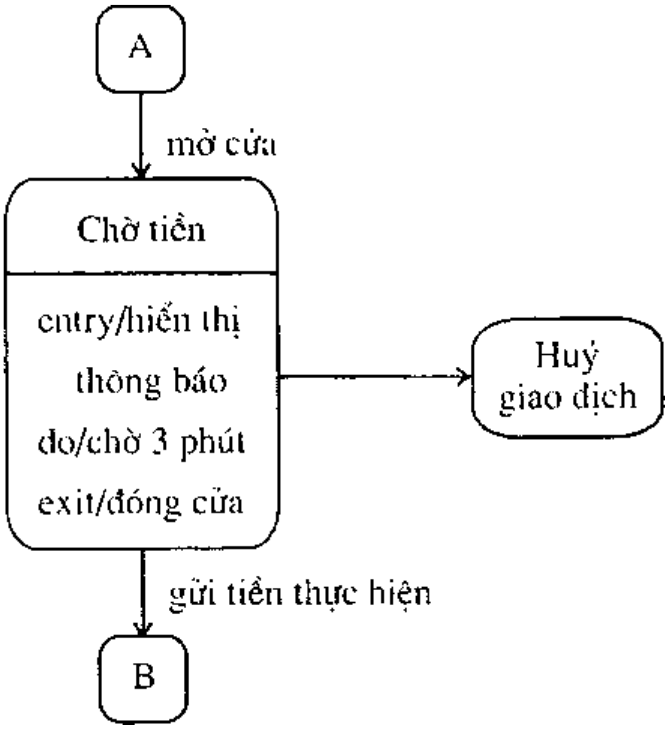


Hình V.21. Dịch chuyển tự động có cảnh giới

Một loại hoạt động đặc biệt là hoạt động chờ. Đó là một hoạt động, mà thực chất là không làm gì, kéo dài cho đến khi một sự kiện chờ đợi xuất hiện. Sự kiện này gây nên sự dịch chuyển ra khỏi trạng thái.

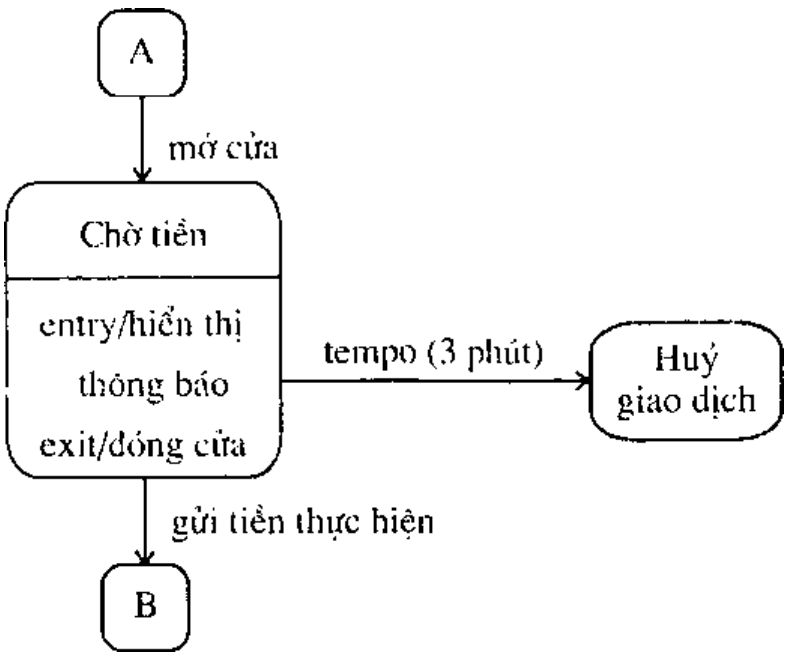
Thí dụ

Một ghi sê tự động của ngân hàng, khi khe cửa bị mơ ra sẽ đưa ra một thông báo là việc gửi tiền sẽ được chờ không quá 3 phút. Nếu khách hàng thực hiện việc gửi tiền trong vòng 3 phút, thì hoạt động chờ bị ngắt bởi dịch chuyển sang trạng thái B. Ngược lại hết 3 phút mà tiền vẫn chưa được khách cho vào, thì sẽ có dịch chuyển tự động sang trạng thái Huỷ giao dịch. Trong mọi trường hợp, khi ra khỏi trạng thái chờ khe cửa đều phải được sập lại. Các yêu cầu như trên sẽ được thực hiện bởi một trạng thái chờ tiền như trên Hình V.22.



Hình V.22. Hoạt động của một ghi sê

Việc chờ đợi (temporisation) có thể diễn tả bằng một cách khác, gọn hơn, nhờ sự kiện định nghĩa sẵn tempo(thời hạn chờ) gắn với bước dịch chuyển hết thời hạn chờ đợi, như trên hình V.23.

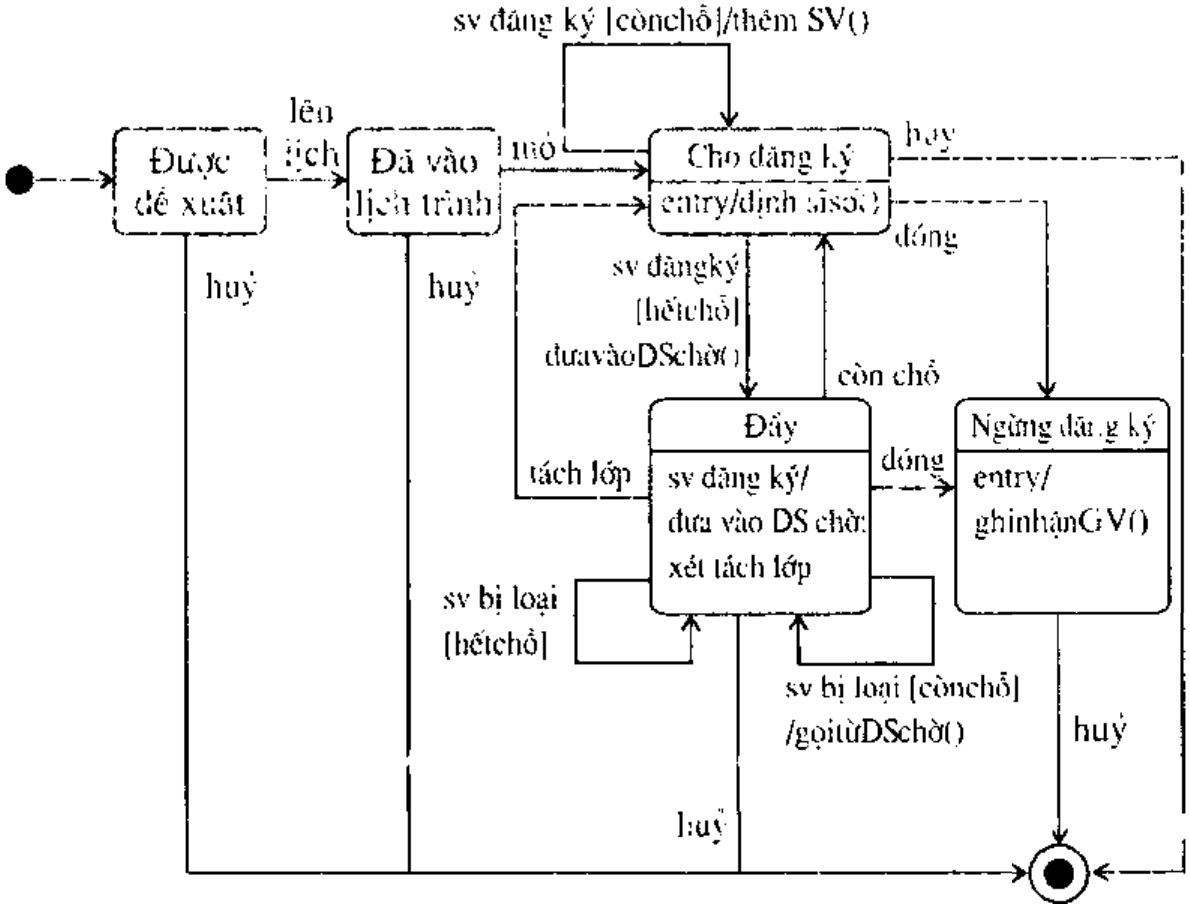


Hình V.23. Chờ đợi nhờ sự kiện tempo

Dưới đây là một thí dụ tổng hợp các yếu tố của một biểu đồ máy trạng thái, đã được trình bày cho tới đây.

Thí dụ:

Trở lại hệ ĐKMH ở trường đại học. Hình V.24. trình bày một biểu đồ máy trạng thái cho một Lớp giảng trong thời gian đăng ký. Có 5 trạng thái: Được đề xuất, Đã vào lịch trình, Cho đăng ký, Đầy, Ngừng đăng ký (không kể hai trạng thái đầu và cuối).



Hình V.24. Một Lớp giảng trong thời gian đăng ký

6. GOM NHÓM VÀ PHÂN RÃ TRẠNG THÁI

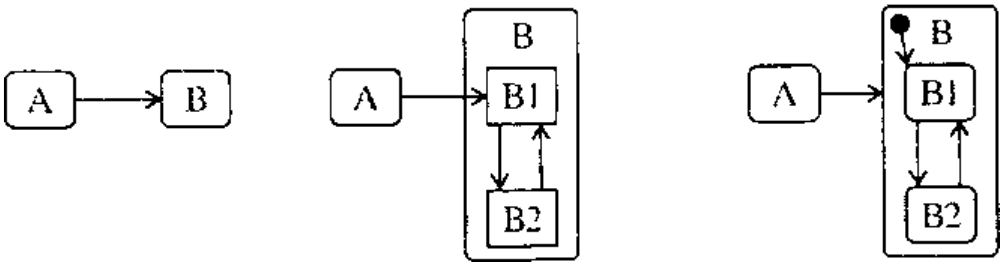
Nhiều khi ta muốn gom nhóm nhiều trạng thái vào một trạng thái để làm cho biểu đồ quang đãng, dễ đọc hơn. Ngược lại cũng có khi ta lại muốn phân rã một trạng thái thành một biểu đồ con gồm nhiều trạng thái để thấy được các diễn biến bên trong trạng thái lớn ban đầu.

Việc gom nhóm hay phân rã trạng thái được thực hiện theo hai hình thức khác nhau, với ngữ nghĩa khác nhau. Đó là sự khái quát và sự kết nhập.

a) Khái quát hoá các trạng thái

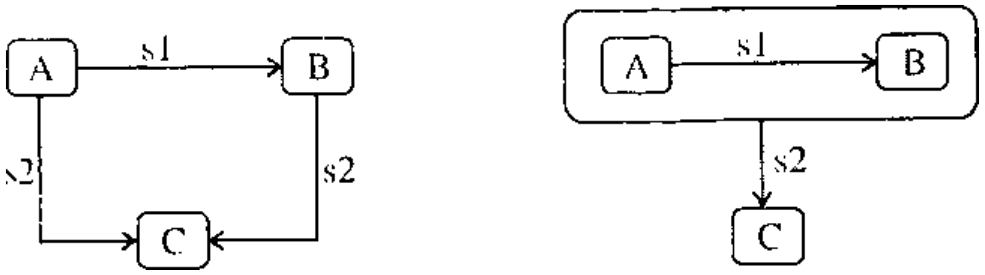
Khái quát hoá (generalization) là sự gom cụm một số trạng thái với các bước dịch chuyển giữa các trạng thái đó vào thành một trạng thái. Quá trình ngược lại là sự chuyên biệt hoá (specialization). Sự khái quát hoá/chuyên biệt hoá phải tuân thủ các quy tắc sau:

- Khi đối tượng đang ở trạng thái cha, thì nó cũng phải ở một và chỉ một trạng thái con. Chính vì vậy mà người ta thường gọi đây là sự phân rã tuyển (hay phân rã theo phép hoặc có loại trừ).
- Một dịch chuyển vào trạng thái cha không được thừa kế bởi mọi trạng thái con, mà chỉ có một trạng thái con tiếp nhận bước dịch chuyển đó. Có hai cách diễn tả quy tắc này, như ta thấy trên Hình V.25.



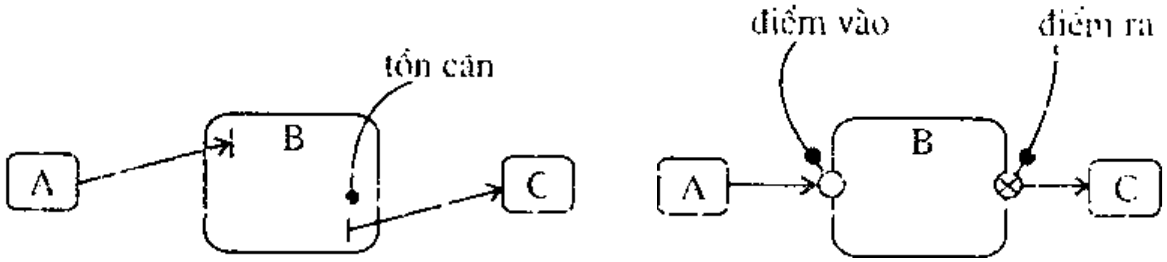
Hình V.25. Hai cách vẽ bước dịch chuyển vào trạng thái cha

- Một dịch chuyển ra khỏi trạng thái cha có thể được thừa kế bởi mọi trạng thái con (và bấy giờ nó được vẽ từ biên của trạng thái cha, như trên hình V.26), mà cũng có thể chỉ một trạng thái con được thiết kế để thực hiện bước dịch chuyển đó.



Hình V.26. Sự thừa kế dịch chuyển ra từ cha sang các con

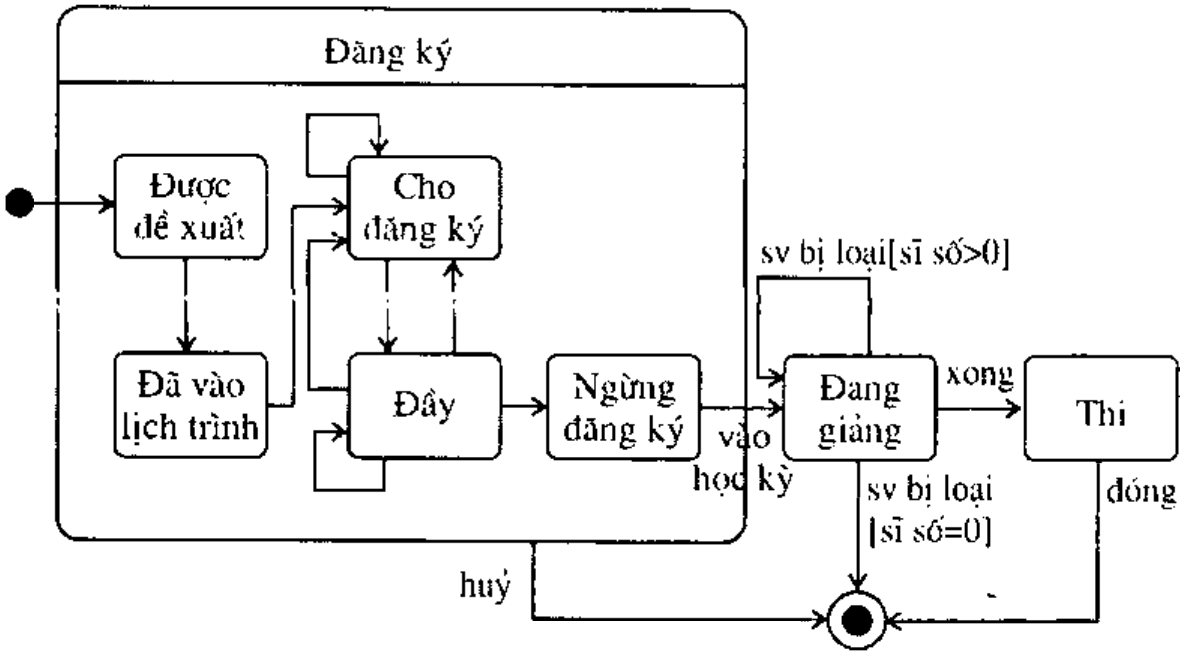
Khi ta muốn tạm giấu các trạng thái con trong một trạng thái cha, nhưng muốn tránh sự hiểu lầm về tính thừa kế, ta dùng các "tồn căn" (theo UML 1.x) hoặc các điểm vào, điểm ra (theo UML 2.0) như trên Hình V.27.



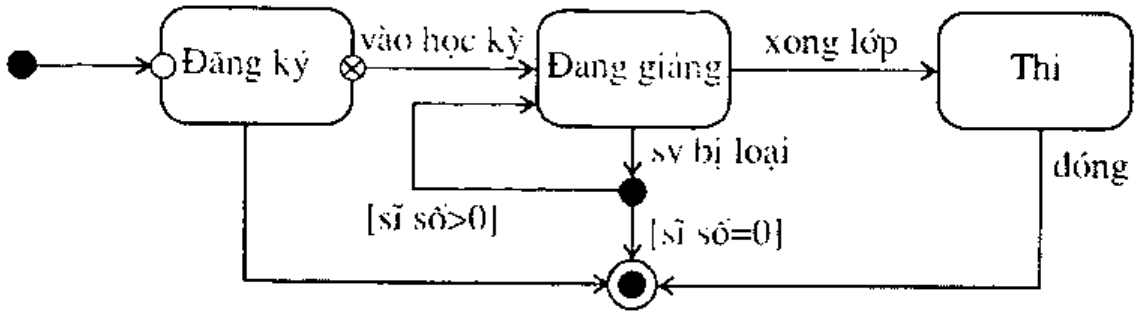
Hình V.27. Giấu các trạng thái con

Thí dụ

Trở lại thí dụ về hệ ĐKMH. Toàn bộ biểu đồ cho ở Hình V.22 có thể khái quát hoá thành một trạng thái trong vòng đời đầy đủ của một lớp giảng, như trên Hình V.28. Biểu đồ máy trạng thái ở mức đỉnh của lớp giảng cho trên Hình V.29.

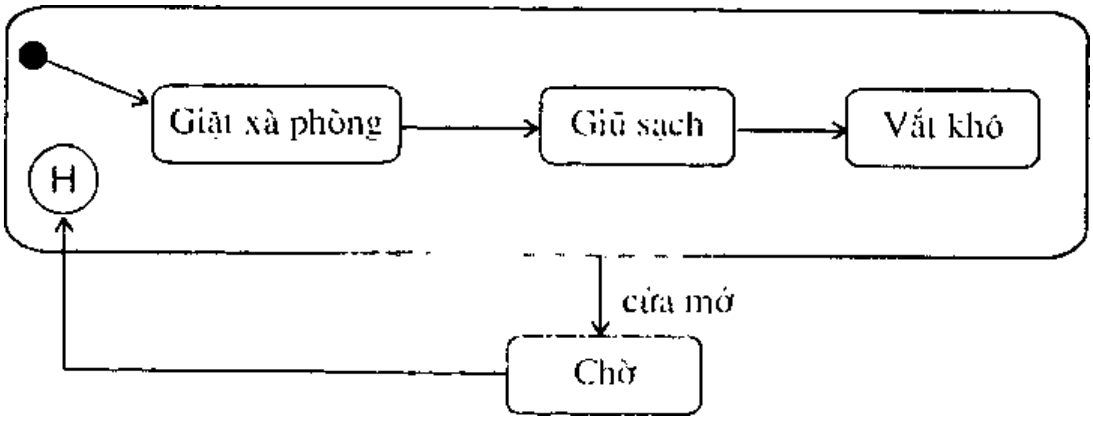


Hình V.28. Vòng đời đầy đủ của lớp giảng



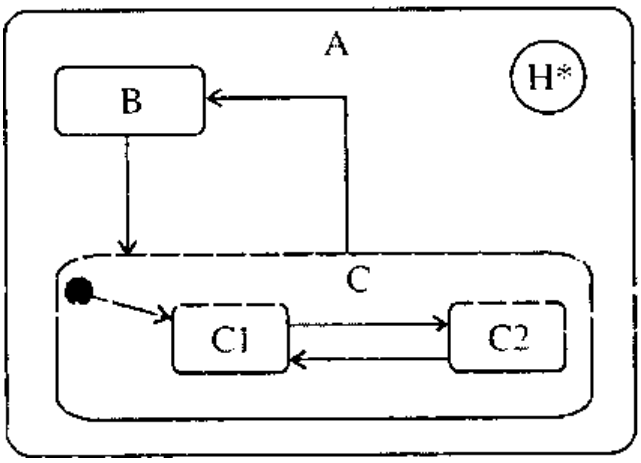
Hình V.29. Mức đỉnh của biểu đồ máy trạng thái

Nhiều khi ta cần phải nhớ trạng thái con cuối cùng trước khi rời khỏi trạng thái cha, để khi trở lại trạng thái cha, thì quá trình sẽ tiếp tục từ trạng thái con này, mà không làm lại từ đầu. Muốn vậy, ta dùng một ký hiệu H (history) đặt đâu đó trong trạng thái cha (thường là ở góc dưới, bên trái) và bước dịch chuyển trở lại trạng thái cha chạm đến H. Chẳng hạn, một máy giặt có chu trình làm việc gồm ba giai đoạn: giặt xà phòng, giũ sạch và vắt khô. Bất cứ lúc nào, ta có thể mở cửa và quy trình tạm ngưng. Khi ta đóng cửa trở lại, thì quy trình lại tiếp tục từ trạng thái vừa bị ngắt. Hành vi của máy giặt đó được mô tả như trên Hình V.30.



Hình V.30. Máy giặt có khả năng ghi nhớ lịch sử

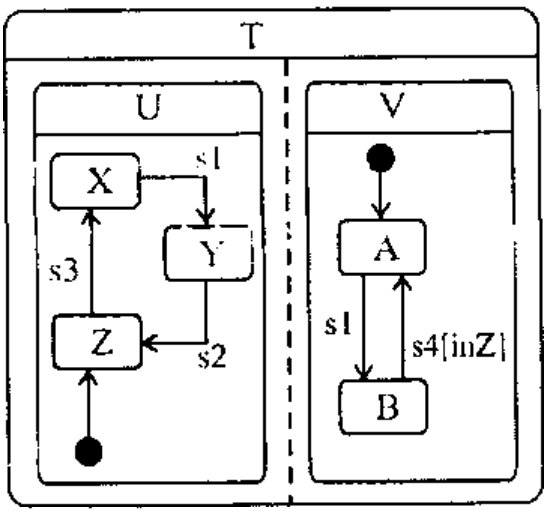
Nếu sự gom bọc các trạng thái được thực hiện theo nhiều tầng bao nhau, thì thay vì phải viết ký hiệu H cho mỗi tầng, ta chỉ viết một ký hiệu H* ở tầng ngoài cùng, như trên Hình V.31.



Hình V.31. Lịch sử trên nhiều tầng

b) Kết nhập các trạng thái

Kết nhập (aggregation) là sự gom nhiều trạng thái vào một trạng thái cha theo kiểu hội (theo phép và): Bấy giờ đối tượng khi vào trạng thái cha thì buộc phải ở đồng thời trong mọi trạng thái con. Như vậy, kết nhập diễn tả một dạng song hành trong một ô tô mát. Người ta dùng các đường đứt nét để ngăn cách các trạng thái con, tạo nên những tuyến bơi (swimlane) cho sự song hành.

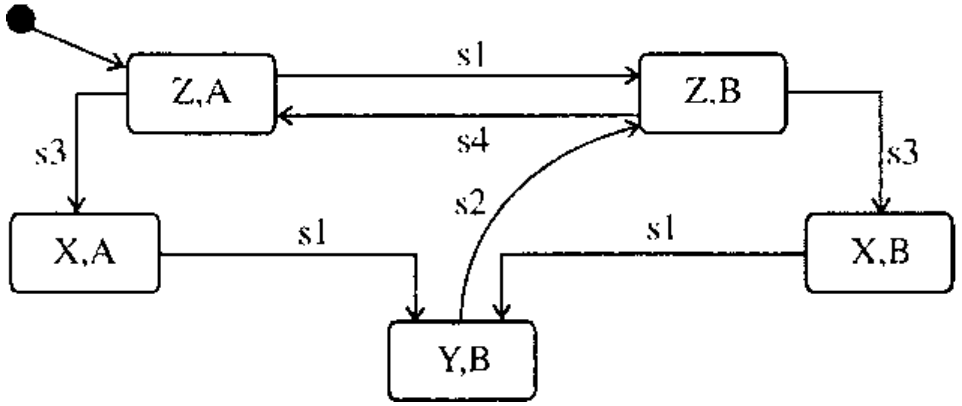


Hình V.32. Kết nhập các trạng thái

Trong Hình V.32 thì T là kết nhập của hai trạng thái U và V. U thì phân rã thành X, Y, Z và V thành A, B. Vậy miền của T là tích Đề các U ×V. Một địch chuyển đi vào T sẽ khởi động đồng thời cả hai ô tô

mát U và V, và như vậy đối tượng sẽ bắt đầu ở trạng thái (Z, A). Khi sự kiện s3 xảy ra, thì do hai ô tô mát U và V phát triển độc lập, đối tượng sẽ chuyển từ trạng thái (Z, A) sang trạng thái (X,A). Tiếp đó nếu sự kiện s1 xảy ra, thì đối tượng chuyển đến trạng thái (Y,B). Cảnh giới [inZ] có nghĩa là sự kiện s4 sẽ có tác dụng dịch chuyển trạng thái B sang trạng thái A chỉ khi đối tượng đang đồng thời ở trạng thái Z.

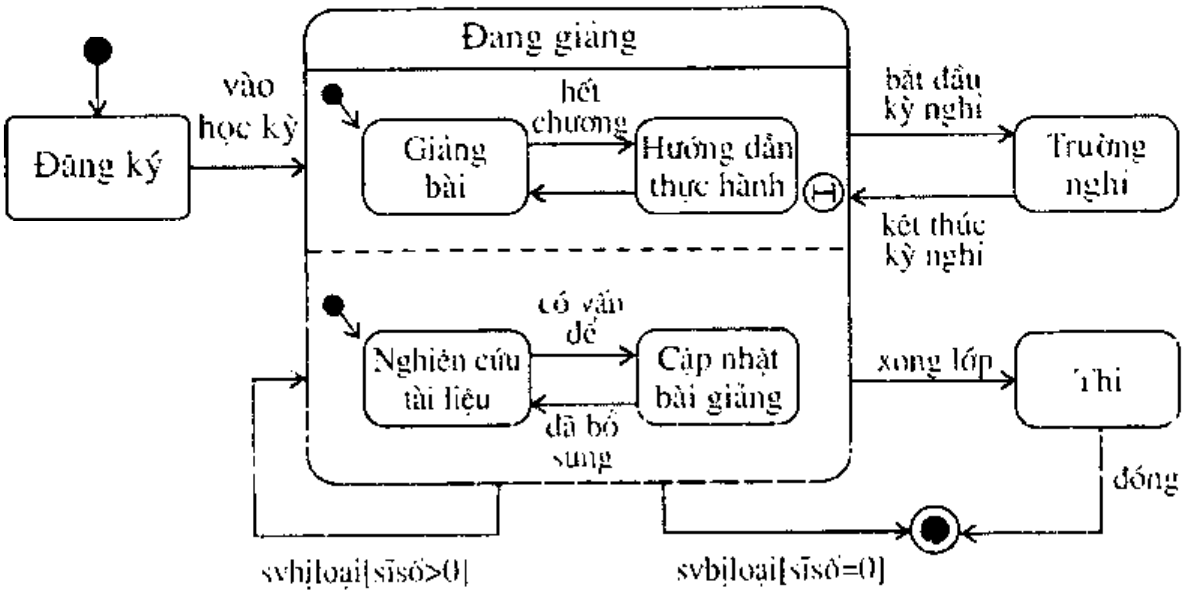
Kết nhập cũng là một cách để đơn giản hoá ô tô mát. Nếu không dùng kết nhập, thì ô tô mát tương đương với ô tô mát trên Hình V.32 sẽ là ô tô mát như trên Hình V.33.



Hình V.33. Ô tô mát tương đương

Thí dụ

Trở lại hệ ĐKMH. Nghiên cứu vào bên trong trạng thái Đang giảng bài của một lớp giảng (Hình V.29), ta thấy có hai quá trình song song: trên lớp (giảng bài và hướng dẫn thực hành) và ngoài lớp (nghiên cứu tài liệu và cập nhập bài giảng). Vậy ta phân rã trạng thái này theo kiểu kết nhập như trên Hình V.34.



Hình V.34. Phân rã trạng thái Đang giảng

7. CÁCH MÔ HÌNH HOÁ ỨNG XỬ VỚI BIỂU ĐỒ MÁY TRẠNG THÁI

Thay đổi hành vi theo trạng thái là đặc điểm của các đối tượng chủ động, chứ không phải của mọi đối tượng. Vậy trước hết cần chọn lớp để mô hình hoá sự ứng xử. Đó là các lớp có ít nhất một trong hai đặc điểm sau:

- Đối tượng của lớp đó có thể có sự phản ứng khác nhau trước cùng một sự kiện xảy đến, mỗi cách ứng xử tương ứng một trạng thái.
- Đối tượng của lớp đó có yêu cầu tổ chức một số thao tác vào một trật tự nhất định. Lúc đó các trạng thái tiếp nối nhau sẽ cho phép mô tả lịch trình đưa ra các kích hoạt cho các thao tác cần thiết.

Thông thường thì chỉ có độ 10% đến 20% số các lớp là có yêu cầu phải mô tả sự ứng xử với biểu đồ máy trạng thái. Ngoài ra cũng có khi ta dùng biểu đồ máy trạng thái để mô tả hành vi không phải là của một đối tượng, mà là của một nhóm đối tượng hợp tác cùng nhau, của một hệ thống con, thậm chí là của cả một hệ thống, hoặc của một đối tác, một thiết bị...

Để thành lập một biểu đồ máy trạng thái, ta tiến hành theo các bước như sau:

- Tìm các trạng thái: Không có một chỉ dẫn chính xác cho việc này. Vì vậy xác định các trạng thái của một đối tượng phức tạp cũng khó như xác định các lớp tham gia một ca sử dụng vậy. Tuy nhiên cũng có ba cách tiếp cận có thể giúp ta tìm kiếm các trạng thái;
  - Với quan niệm trạng thái là một giai đoạn ổn định trong đời sống nhiều biến động về hành vi của đối tượng, thì các giai đoạn này thường đã được nhắc đến trong ngôn từ của các chuyên gia lĩnh vực; ta có thể phát hiện chúng từ các tài liệu, các báo cáo và các chuyên gia này.
  - Với quan niệm trạng thái là một sự phản ánh "thô" của các giá trị thuộc tính và kết nối, thì ta có thể dõi theo sự biến thiên của các thuộc tính và kết nối, qua đó phát hiện ra các giá trị ngưỡng, là các giá trị mà vượt qua đó thì đối tượng có một sự thay đổi trong hành vi. Căn cứ vào các giá trị ngưỡng đó, ta có một sự phân chia ra các giai đoạn (tức là các trạng thái).
  - Cuối cùng thì có thể nghiên cứu các biểu đồ trình tự có chứa đối tượng thuộc lớp mà ta đang xét. Trên đường đời của đối tượng này, thì mỗi khoảng cách giữa hai lần nhận thông điệp liên tiếp tương ứng với một trạng thái của đối tượng. Cách làm này dường như chỉ phát hiện ra các trạng thái nối tiếp nhau một cách tuyến tính (theo thời gian). Thực ra, thường có sự vòng lại: một trạng thái phát hiện sau có thể trùng với một trạng thái trước đó. Vậy cần đi sâu vào ngữ nghĩa của từng trạng thái (điều kiện phải được thoả mãn, hoạt động cần thực hiện, sự kiện được chờ đợi ở trang thái đó) để phát hiện sự trùng lắp và giảm bớt số các trạng thái xuống.

Lưu ý rằng số các trạng thái phải là hữu hạn. Không cần vẽ các biểu đồ máy trạng thái có dưới ba trạng thái. Bởi vì đối tượng một trạng thái không phải là đối tượng chủ động, còn đối tượng hai trạng thái thì có hành vi đơn giản (kiểu "on/off"), có thể mô tả trực tiếp. Mặt khác thì, ít nhất là trong bước đầu, không nên đưa ra nhiều trạng thái quá, vì biểu đồ với quá nhiều trạng thái sẽ rối rắm, khó đọc. Thà rằng sau đó, nếu đối tượng là rất phức tạp, ta lại phải phân rã các trạng thái lớn thành các trạng thái con (theo kiểu khái quát hoá hay kết nhập).

- Tổ chức các trạng thái thành biểu đồ: Đó là bổ sung các dịch chuyển giữa các trạng thái, cùng với các sự kiện tương ứng. Việc

này có thể thực hiện theo kiểu tăng trưởng dần dần dựa trên ba giai đoạn sau:

- Trước tiên biểu diễn dãy các trạng thái mô tả cho hành vi chính danh của đối tượng, từ khi sinh ra cho đến khi chết, cùng với các dịch chuyển liên quan;
- Bổ sung dần dần các dịch chuyển tương ứng với các lối rẽ, diễn tả cho các hành vi khả dĩ;
- Bổ sung dần dần các dịch chuyển tương ứng với các hành vi sai lỗi;
- Thêm các hành động và các hoạt động;
- Gom nhóm hay phân rã các trạng thái nếu biểu đồ là quá phức tạp.

Mỗi biểu đồ máy trạng thái phải có một trạng thái đầu. Dùng nhiều trạng thái cuối với tên gọi khác nhau để làm cho biểu đồ dễ đọc, hoặc để phân biệt rõ các hoàn cảnh kết thúc khác nhau. Cần chú ý khi dùng dịch chuyển đệ quy (dịch chuyển từ một trạng thái trở lại chính trạng thái đó): Nó buộc phải thực hiện các hành động exit và entry, mỗi lần ra và vào trạng thái. Nếu không muốn điều đó, thì thay dịch chuyển đệ quy bằng một dịch chuyển trong, tức là dịch chuyển giữa hai trang thái con của trạng thái đang xét.

8. ĐỐI CHIẾU BIỂU ĐỒ MÁY TRẠNG THÁI VỚI CÁC BIỂU ĐỒ TƯƠNG TÁC

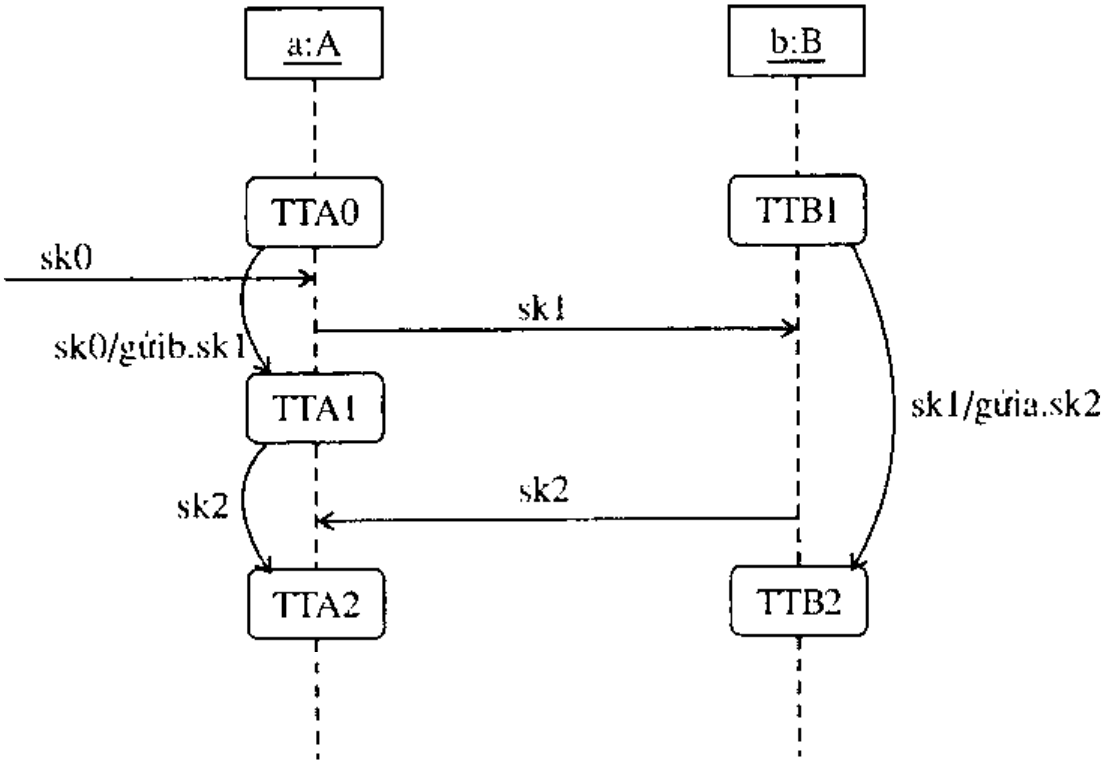
Các biểu đồ tương tác (biểu đồ trình tự và biểu đồ giao tiếp) diễn tả sự chuyển giao thông điệp giữa các đối tượng. Còn biểu đồ máy trạng thái lại diễn tả cách phản ứng của mỗi đối tượng khi nhận thông điệp (sự kiện). Có thể nói đó chỉ là hai cách nhìn đối với một vấn đề (nhìn rộng và nhìn sâu), và chúng phải có mối liên hệ và bổ sung thông tin cho nhau. Bởi vậy khi lập xong biểu đồ máy trạng thái của một đối tượng, ta phải đối chiếu biểu đồ đó với các biểu đồ tương tác đề cập đến đối tượng đó để chỉnh sửa cho phù hợp.

Có thể nói là các biểu đồ máy trạng thái (cách nhìn sâu) đã mang lại sự chính xác và đầy đủ, cho nên ta có thể lấy đó làm căn cứ để nghiệm thu và bổ sung các biểu đồ tương tác liên quan. Thậm chí biểu đồ máy trạng thái còn có thể dẫn ta tới chỗ phải lập thêm các biểu đồ tương tác mới cho đầy đủ.

Hình V.35 cho ta thấy mối liên quan giữa hai loại biểu đồ này. Đây là một biểu đồ trình tự đơn giản, gồm hai đối tượng: đối tượng a của lớp A và đối tượng b của Lớp B. Chúng tương tác với nhau bởi hai sự kiện tiếp nối.

- Sau khi nhận sự kiện sk0, thì a gửi sự kiện sk1 tới b;
- Đáp lại, thì b gửi sự kiện sk2 tới a.

Các biểu đồ máy trạng thái của các lớp A và B nhất thiết phải tương thích với sự tương tác này, cho dù chúng còn bao gồm thêm nhiều hành vi khác.



Hình V.35. Liên quan giữa tương tác và dịch chuyển trạng thái

Ta đưa thêm ở hình này các trạng thái của đối tượng dọc theo đường đời của nó. Như vậy, đối tượng a ở trạng thái TTA0 trước khi nhận sự kiện sk0, và nó sẽ chuyển sang trạng thái mới TTA1 sau khi đã gửi đi sự kiện sk1 cho b. Sự kiện sk1 được ô tô mát của lớp B tiếp nhận, sẽ làm dịch chuyển trạng thái TTB1 sang trạng thái TTB2 trong b, sau khi đáp lại sk2 cho a. Điều này lại chứng tỏ rằng ở trạng thái TTA1, thì đối tượng a phải có khả năng xử lý sk2 (bằng một hoạt động nào đó tại trạng thái này).

Cứ tiếp tục như vậy dọc theo đường đời của mỗi đối tượng, ta có thể kiểm tra sự tương thích giữa hai loại biểu đồ.

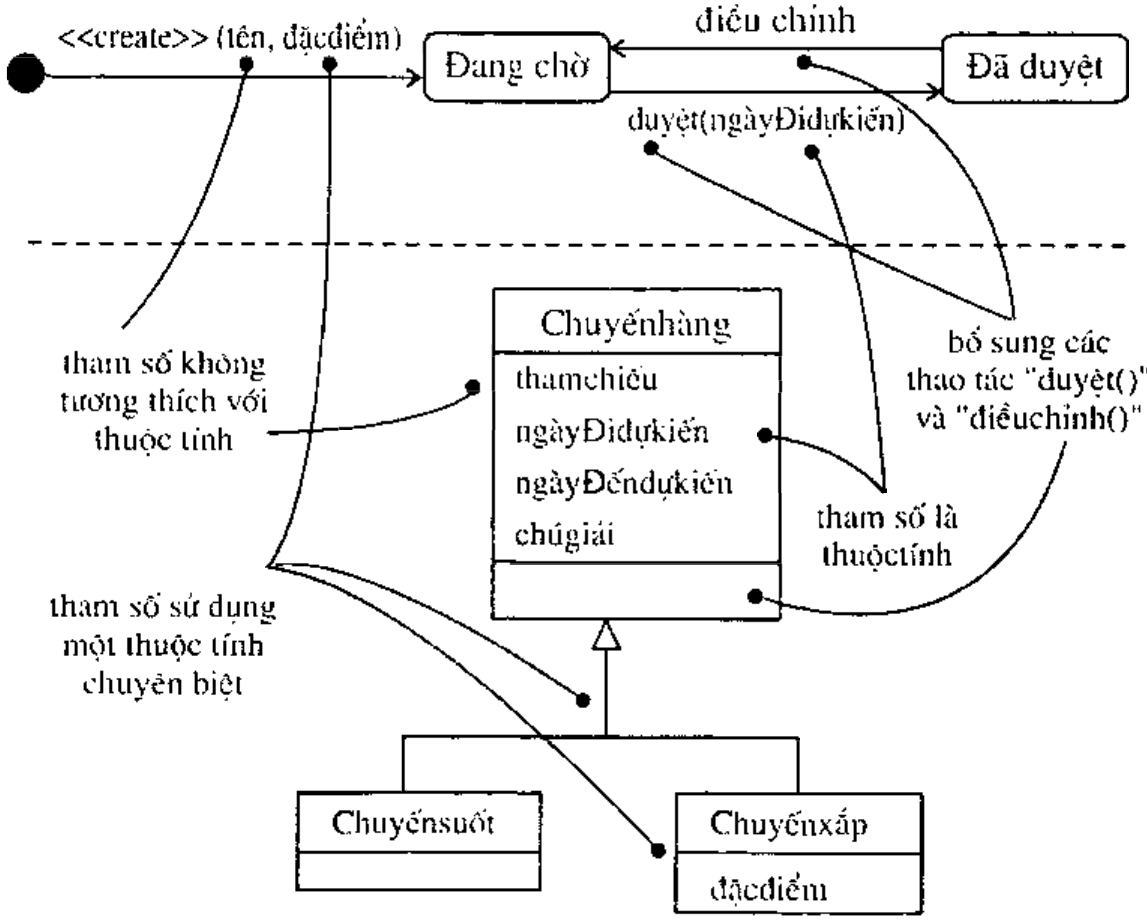
9. ĐỐI CHIẾU CÁC MÔ HÌNH ĐỘNG VỚI MÔ HÌNH TĨNH

Sau khi lập các mô hình động (biểu đồ tương tác, biểu đồ máy trạng thái), ta lại phải đối chiếu chúng với mô hình tĩnh (biểu đồ lớp) để chỉnh sửa lại cho ăn khớp. Nhớ rằng khi lập biểu đồ lớp, thì ta chưa có đủ các căn cứ để chỉ ra đầy đủ các thuộc tính, liên kết và thao tác, và nay chính là lúc phải bổ sung chúng cho đầy đủ. Mặt khác, khi lập các biểu đồ tương tác và máy trạng thái, ta đặt tên cho các thông điệp, các sự kiện, các tham số một cách tuỳ tiện theo bản chất của chúng, mà chưa nhìn lại nguồn gốc của chúng trong biểu đồ lớp để gọi tên một cách chính xác. Chính vì vậy mà giờ đây phải có sự chỉnh sửa cả đôi bên cho thật ăn khớp.

Vậy các yếu tố trong biểu đồ lớp như là *thao tác, thuộc tính và liên kết* có mối liên hệ gì với các yếu tố trong biểu đồ tương tác và biểu đồ máy trạng thái như là *thông điệp, sự kiện, hành động, hoạt động, cảnh giới, tham số.v.v...*? Có thể kể ra các mối liên hệ chính, để làm căn cứ cho việc kiểm tra tính tương thích, như sau đây:

- Đối với các thao tác:
  - một thông điệp có thể là một lời gọi tới một thao tác có trong một đối tượng (bên nhận), phát ra từ một thao tác của một đối tượng khác (bên gửi);
  - một sự kiện hoặc một hành động trên một dịch chuyển có thể là một lời gọi tới một thao tác,
  - một hoạt động trong một trạng thái có thể xem là sự thực hiện của một thao tác phức tạp hoặc một sự tiếp nối của nhiều thao tác.
- Đối với các thuộc tính và các liên kết:
  - một điều kiện cảnh giới hoặc một sự kiện thay đổi có thể đề cập các thuộc tính hoặc các kết nối tĩnh;
  - một hành động trên một dịch chuyển có thể đề cập các thuộc tính hoặc các kết nối tĩnh;
  - một tham số của một thông điệp có thể là một thuộc tính hoặc một đối tượng trọn vẹn.

Hình V.36 minh hoạ sự đối chiếu giữa biểu đồ máy trạng thái và biểu đồ lớp đối với lớp Chuyến hàng (trích trong một hệ thống vận chuyển hàng hoá).



Hình V.36. Đối chiếu biểu đồ máy trạng thái và biểu đồ lớp

§3. MÔ HÌNH HOÁ HÀNH VI TRÊN CÁC GÓC ĐỘ KHÁC

1. MỤC ĐÍCH

Cùng với các bước 5 và 6 ta đã mô hình hoá hành vi trên hai góc độ: tương tác và ứng xử. Tuy nhiên, do những đặc điểm khác biệt của nhiều hệ thống, có những khi ta còn cần mô hình hoá hành vi trên các góc độ khác nữa. Vì vậy UML còn cung cấp thêm cho chúng ta một số mô hình động thái, như là:

- biểu đồ hoạt động, mô tả hành vi theo luồng công việc;
- biểu đồ bao quát tương tác, mô tả sự tiếp nối của nhiều tương tác phức tạp;
- biểu đồ thời khắc, mô tả hành vi theo sự tiếp nối của các giai đoạn.

Các biểu đồ này - mà ta sẽ lần lượt tìm hiểu ở dưới đây - đều không gắn kết nhất định vào một bước nào trong tiến trình 10 bước mà ta đang theo. Song trong khi mô hình hoá hành vi, nếu thấy thích hợp, ta đều có thể vận dụng chúng nơi này hay nơi khác một cách tuỳ ý, để bổ sung các khía cạnh của hành vi chưa được diễn tả trong các biểu đồ tương tác và máy trạng thái.

2. BIỂU ĐỒ HOẠT ĐỘNG

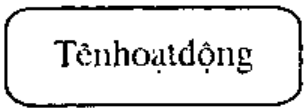
Biểu đồ hoạt động (Activity Diagram) là biểu đồ mô tả một nội dung hoạt động, theo các luồng đi từ việc này sang việc khác. Nó thường được dùng để diễn tả lôgic của một ca sử dụng, một kịch bản, một nhóm ca sử dụng, một quy tắc hay một thao tác phức tạp.

Có thể nói biểu đồ hoạt động là mô hình UML tương đương với sơ đồ khối hoặc với biểu đồ luồng dữ liệu trong các phương pháp phân tích và thiết kế cũ.

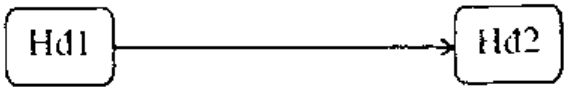
a) Các hoạt động và dịch chuyển

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* (Activity) là một công việc, có thể là được xử lý bằng tay, như là *Điền mẫu*, hoặc xử lý bằng máy tính, như là *Hiển thị Màn hình Đăng ký*. Trong biểu đồ thì một hoạt động được biểu diễn bằng một hình chữ nhật tròn góc, có mang tên của hoạt động:



- *Dịch chuyển* (Transition) là sự chuyển tiếp từ hoạt động này sang hoạt động khác. Trong biểu đồ thì dịch chuyển được biểu diễn bằng một mũi tên nối từ một hoạt động này sang một hoạt động khác:



Như vậy có thể nói biểu đồ hoạt động là một biểu đồ máy trạng thái giản lược, với hai đặc điểm:

- Các trạng thái không cần mang tên, và đồng nhất với hoạt động của nó;
- Các dịch chuyển đều là dịch chuyển tự động (khởi phát bởi sự hoàn tất một hoạt động, mà không phải là do một sự kiện từ ngoài đến).

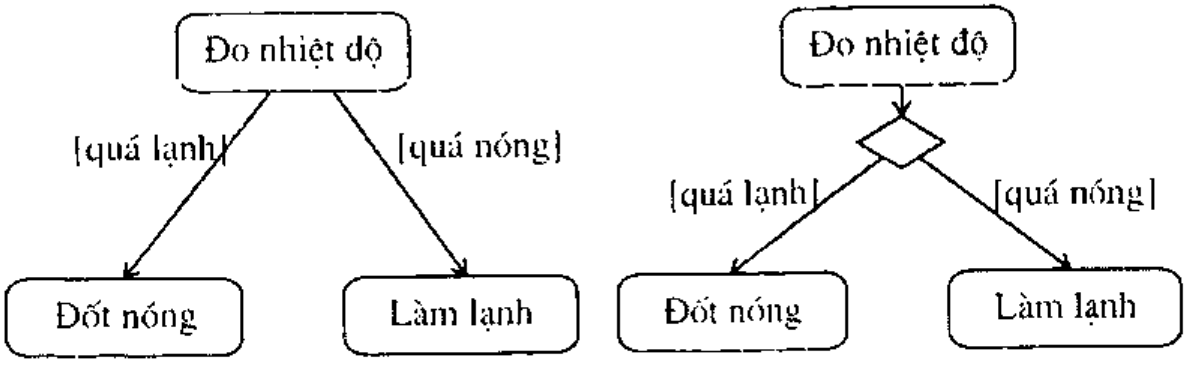
Cũng như trong biểu đồ máy trạng thái, biểu đồ hoạt động cũng có nút khởi đầu, dưới dạng hình tròn đặc (không nhất thiết phải có, song có nó thì biểu đồ dễ đọc hơn), và cũng có nút kết thúc, dưới dạng hình tròn đặc có viền (có thể có không hoặc nhiều nút kết thúc).

b) Các cảnh giới

Cũng như trong biểu đồ máy trạng thái, ta có thể dùng các cảnh giới (các điều kiện Bun loại trừ nhau) để thực hiện sự rẽ nhánh. UML dùng một hình thoi nhỏ để diễn tả:

- hoặc là một *quyết định* (decision), khi nó có một luồng vào và nhiều luồng ra (các luồng ra đều phải có cảnh giới);
- hoặc là một *hoà nhập* (merge) khi nó có nhiều luồng vào và một luồng ra (điểm hoà nhập sẽ được vượt qua khi có một luồng vào xuất hiện).

Như vậy hai biểu đồ trên Hình V.37 là tương đương.

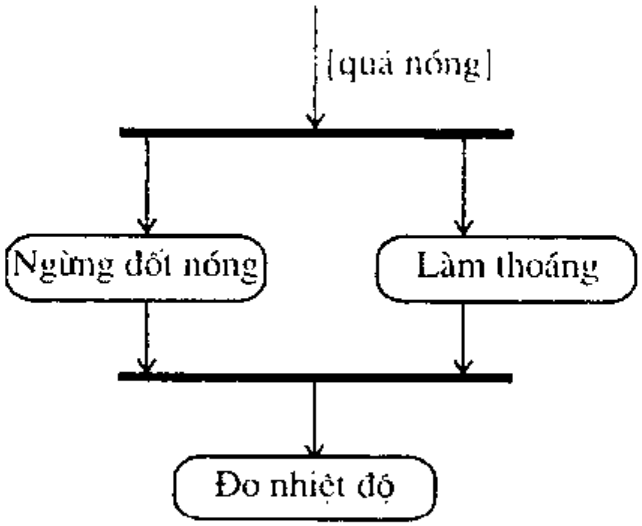


Hình V.37. Hai cách diễn tả sự rẽ nhánh

c) Đồng bộ hoá

Trong biểu đồ hoạt động, ta có thể dùng các *thanh đồng bộ hoá* (synchronization bars) để mở hay đóng các nhánh thực hiện song song (Hình V.38):

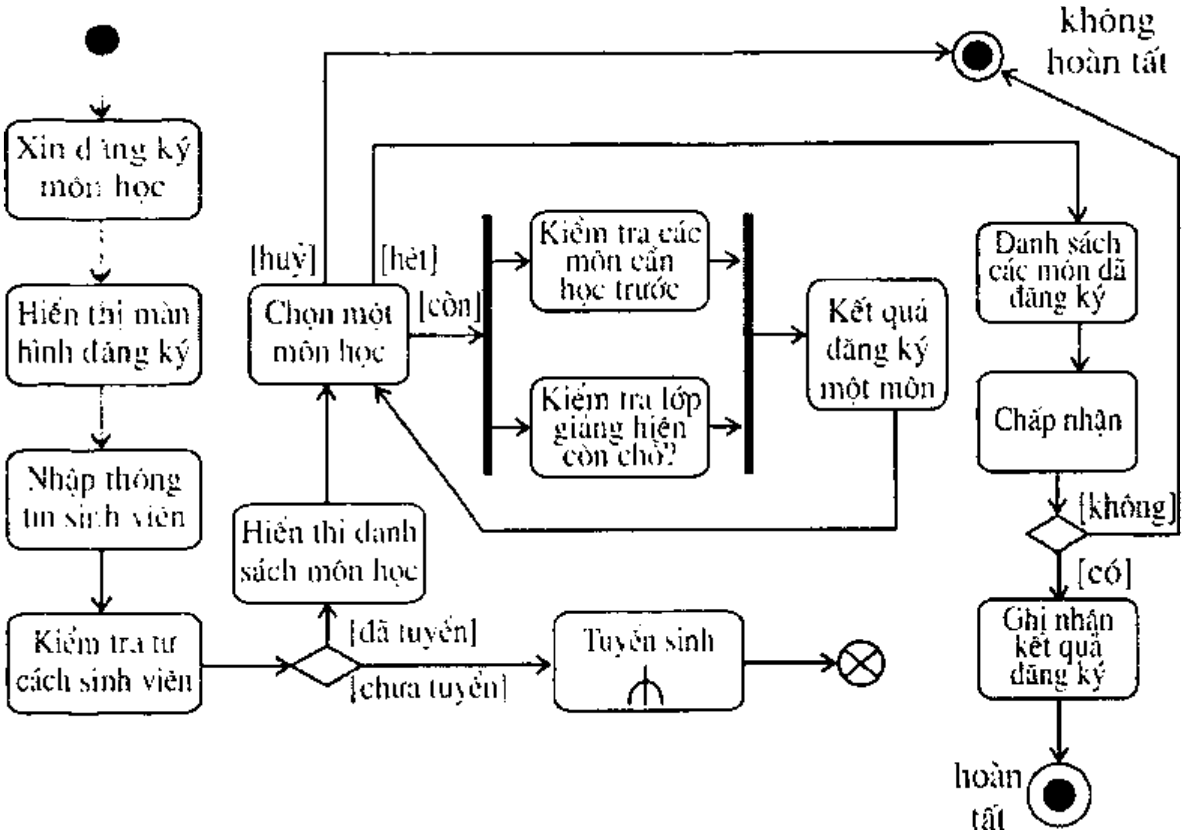
- mở các nhánh song song bằng một thanh đồng bộ hoá khi nó có một dịch chuyển vào và nhiều dịch chuyển ra - ta gọi đó là một *chạc* (fork);
- đóng các nhánh song song bằng một thanh đồng bộ hoá khi nó có nhiều dịch chuyển vào và một dịch chuyển ra - ta gọi đó là một *chụm* (join). Chụm chỉ có thể vượt qua khi mọi nhánh vào nó đều đã hoàn tất.



Hình V.38. Mở và đóng các nhánh song song

Thí dụ:

Trở lại hệ ĐKMH. Hình V.39 cho một biểu đồ hoạt động (bản phác thảo) dùng để diễn tả ca sử dụng *Đăng ký môn học*.



Hình V.39. Biểu đồ hoạt động cho ca sử dụng Đăng ký môn học

Trong Hình V.39 có hai ký hiệu mới:

- Ký hiệu ⑃ ở trong nút Tuyển sinh, để nói rằng đây là một hoạt động phức tạp, cần được diễn tả bằng một biểu đồ hoạt động khác;
- Ký hiệu ⊗ để chỉ rằng một nhánh được ngắt ở đây (có thể là do ta không muốn triển khai xa hơn). Nhiều khi ký hiệu này lại được thế bằng một hình tròn nhỏ mang những chữ cái A, B, ... để tạo ra một *mối nối* tới một biểu đồ khác, mà lối vào cũng có cùng ký hiệu đó.

d) Phân tuyến và phân vùng

Biểu đồ hoạt động có thể được phân tuyến, như trong một bể bơi. Mỗi hoạt động phải đặt gọn trong một tuyến, và mỗi tuyến giành cho một hay một số đối tượng thực hiện. Các dịch chuyển có thể đổi tuyến tự do.

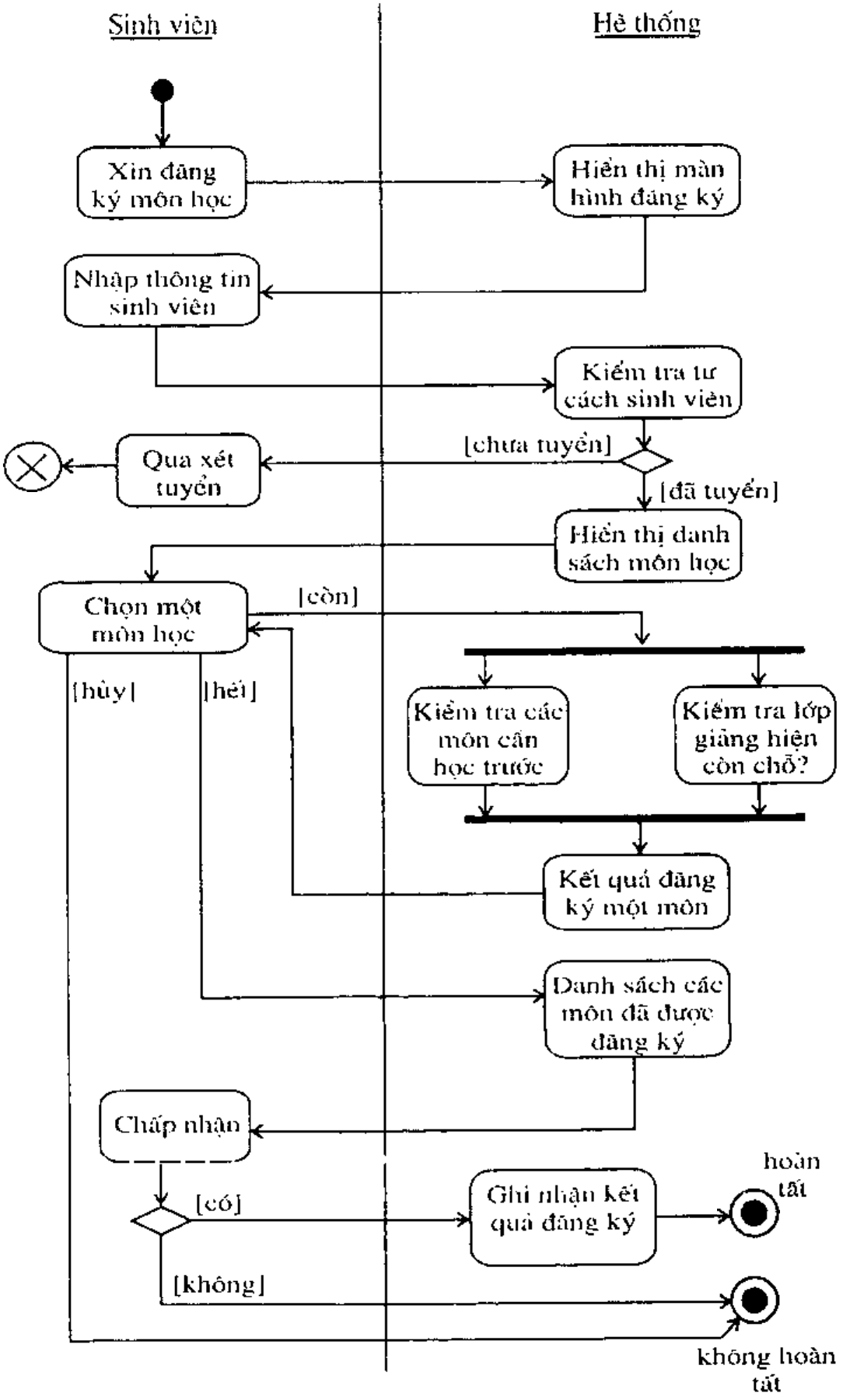
Biểu đồ hoạt động cũng có thể được phân vùng, mỗi vùng gom các hoạt động cùng hướng vào một mục đích chung nào đó. Có thể có một

vùng thực hiện kịch bản chính (lộ trình chính), còn các vùng khác thực hiện các kịch bản phụ (chẳng hạn các trường hợp biến dị, các trường hợp đặc biệt, các trường hợp sai lỗi...).

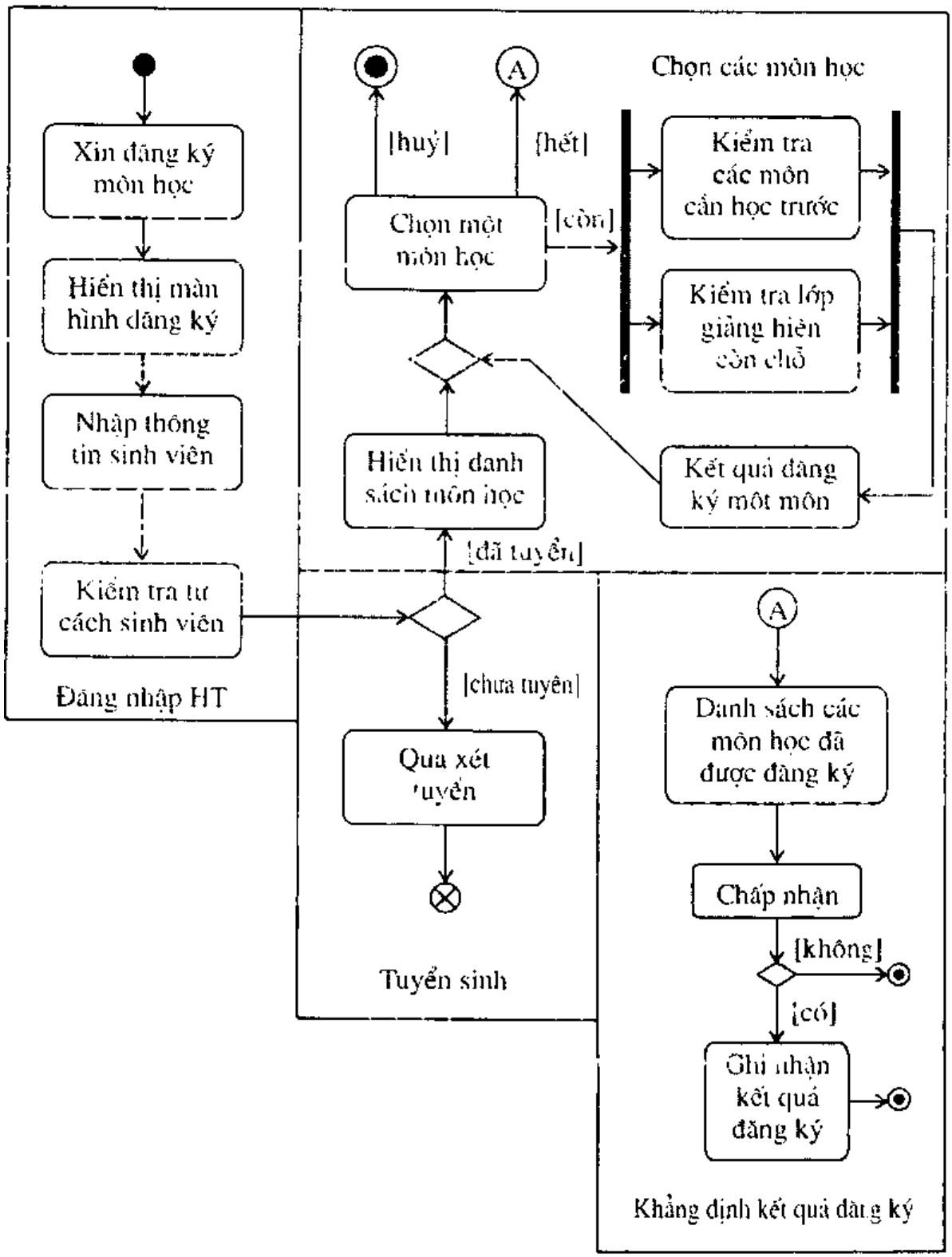
Thí dụ

Hình V.40 trình bày lại biểu đồ hoạt động cho ca sử dụng *Đăngký môn học* ở trên, song đã được phân tuyến theo hai tác nhân là Sinh viên và Hệ thống.

Còn Hình C.41 lại trình bày biểu đồ hoạt động đó với sự phân vùng, theo các mục đích con như là Đăng nhập, Chọn các môn học, Khẳng định kết quả đăng ký, Tuyển sinh.



Hình V. 40. Biểu đồ hoạt động có phân tuyến



Hình V.41. Biểu đồ hoạt động có phân vùng

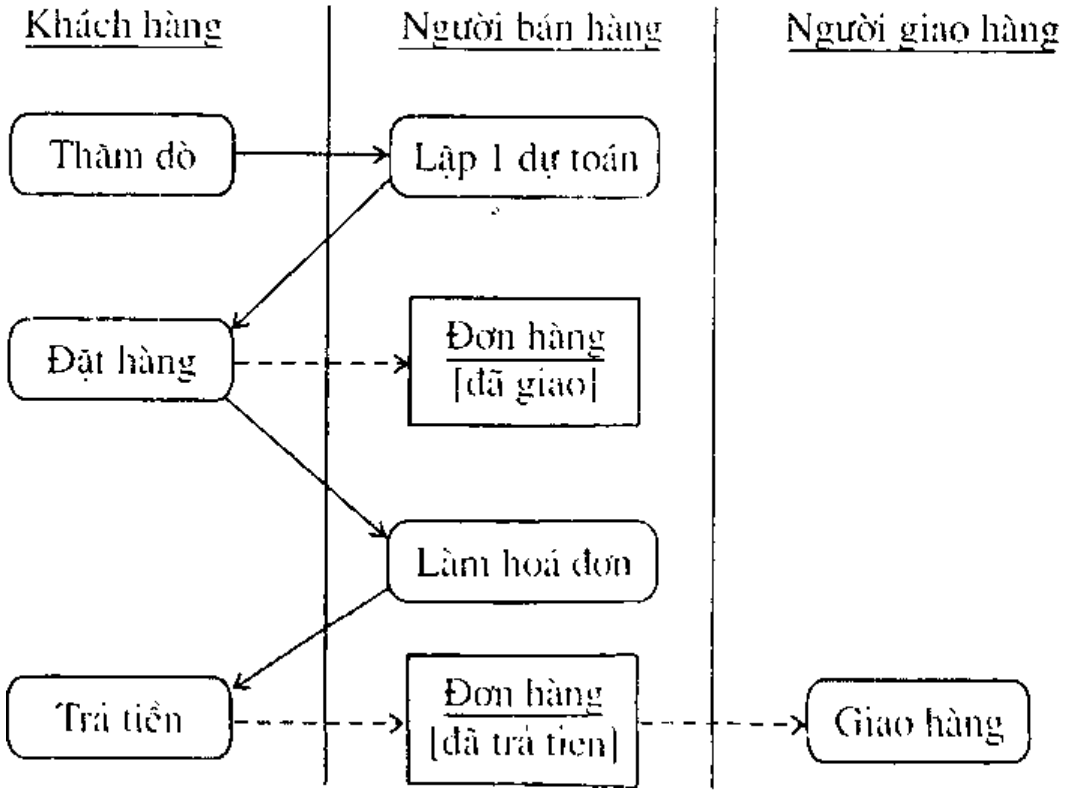
đ) Tạo lập đối tượng và xuất nhập sự kiện

Nhiều khi trong một biểu đồ hoạt động, ta lại muốn chỉ rõ rằng có một hoạt động nào đó tạo lập (hay làm thay đổi trạng thái) một đối tượng, hoặc nhận một đối tượng để xử lý. Bấy giờ ta vẽ thêm đối tượng

(với trạng thái nếu cần) vào trong biểu đồ hoạt động và nối nó với hoạt động liên quan bằng một mũi tên đứt nét (gọi là một luồng đối tượng). Một luồng đối tượng từ một hoạt động đến một đối tượng, rồi lại tiếp tục đi vào một hoạt động khác có thể xem là một luồng điều khiển (một dịch chuyển) giữa hai hoạt động đó.

Thí dụ

Hình V.42 diễn tả một quy trình bán hàng, trong đó có chỉ rõ một đối tượng Đơn hàng đã được tạo lập và thay đổi trạng thái qua các bước hoạt động.



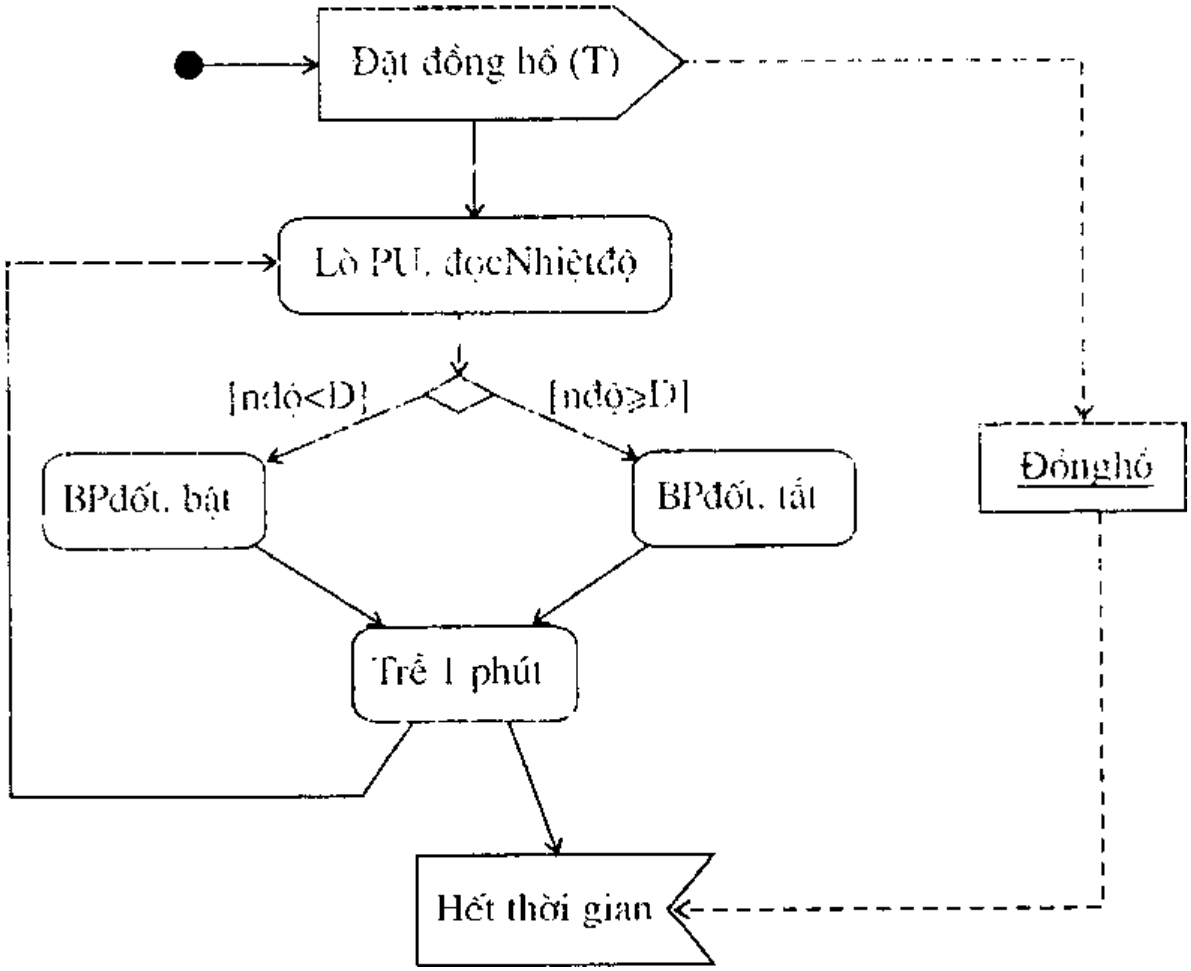
Hình V.42. Biểu đồ hoạt động có thêm luồng đối tượng

Lại có khi ta muốn chỉ rõ một hoạt động xuất một sự kiện tới một đối tượng, hoặc nhập một sự kiện từ một đối tượng. Lúc đó ta vẽ thêm luồng đối tượng như trên, và ngoài ra:

- hoạt động gửi sự kiện (hay tín hiệu) được vẽ dưới dạng một ngũ giác lồi;
- hoạt động nhận sự kiện (hay tín hiệu) được vẽ dưới dạng một ngũ giác lõm.

Thí dụ

Hình V.43 diễn tả một quy trình ủ nhiệt (trong một hệ Điều khiển phản ứng nhiệt): Giữ nhiệt độ trong lò phản ứng luôn ở nhiệt độ D trong một khoảng thời gian T.



Hình V.43. Biểu đồ hoạt động có xuất, nhập tín hiệu

3. BIỂU ĐỒ BAO QUÁT TƯƠNG TÁC

Biểu đồ bao quát tương tác (Interation Overview Diagram) là một trong bốn biểu đồ mới của UML 2.0, và không có trong các UML 1.x. Nó là một biến thể của biểu đồ hoạt động với sự khác biệt là: mỗi nút của biểu đồ không phải là một hoạt động thông thường mà là một *khung* (frame). Có hai loại khung:

- *khung tương tác*, trong đó vẽ một biểu đồ tương tác UML nào đó (như là biểu đồ trình tự, biểu đồ giao tiếp, biểu đồ thời khắc và biểu đồ bao quát tương tác);

– *khung tương tác cụ thể*, trong đó chỉ ra một hoạt động hoặc một thao tác được gọi.

Ngoài ra, thì các cung trong biểu đồ vẫn giữ nguyên mọi quy ước như trong biểu đồ hoạt động, và nay diễn tả cho các luồng điều khiển đại thể.

Thí dụ

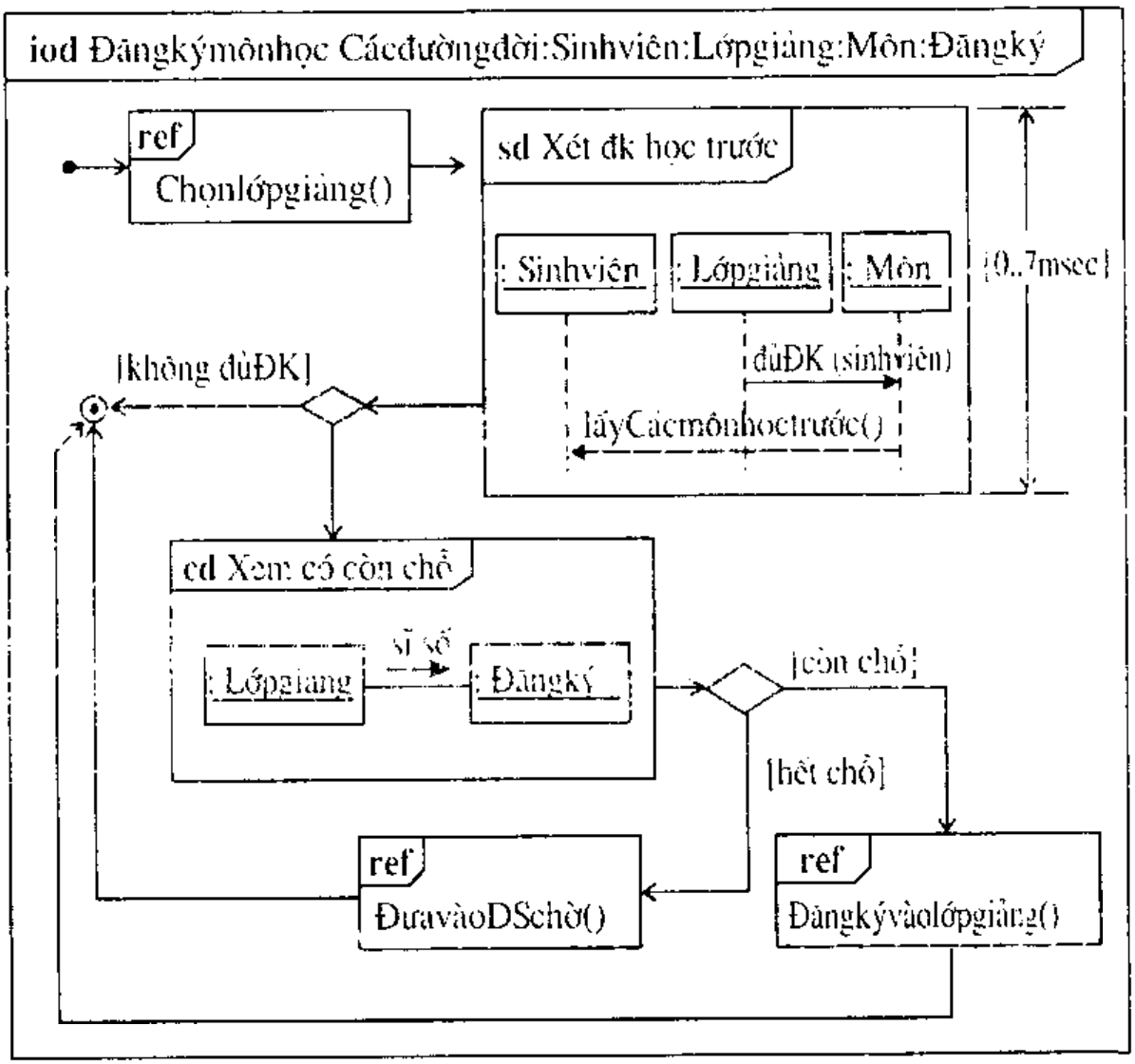
Trở lại với hệ ĐKMH. Hình V.44 cho một biểu đồ bao quát tương tác đối với ca sử dụng Đăngký môn học. Lưu ý rằng ở đây mọi khung đều được chỉ rõ loại biểu đồ vẽ bên trong, theo quy ước:

– sd cho biểu đồ trình tự;
– cd cho biểu đồ giao tiếp;
– td cho biểu đồ thời khắc;
– iod cho biểu đồ bao quát tương tác.

Các biểu đồ này có thể được cho thêm tên và một số thông tin liên quan khác (như là các đường đời của các đối tượng tham gia). Các khung tương tác cụ thể, được chỉ định bởi ký hiệu ref, lại không cần phải có tên, vì tên của hoạt động hay thao tác bên trong đủ nói lên điều gì xẩy ra rồi.

Ta vẫn dùng các cách diễn tả về các trạng thái đầu, trạng thái cuối, các cảnh giới, các quyết định... như trong biểu đồ hoạt động. Các ràng buộc về thời gian, như là {0..7sec}, cũng được điễn tả theo quy ước dùng trong các loại biểu đồ tương tác khác.

Nhận xét: UML 2.0 đưa thêm loại biểu đồ bao quát tương tác này vì nó cho ta một cái nhìn bao quát về sự tương tác, khá thú vị. Song có lẽ loại biểu đồ mới này sẽ khó vận dụng vào thực tế, vì các khung của nó khó có đủ không gian để vẽ được các biểu đồ tương tác phức tạp bến trong. Nhận xét này có thể là không đúng, và ta vẫn còn phải chờ sự thử thách của thời gian đối với nó.



Hình V.44. Biểu đồ bao quát tương tác

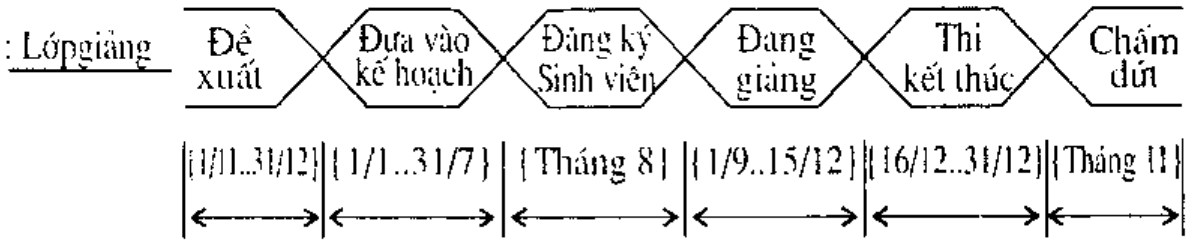
4. BIỂU ĐỒ THỜI KHẮC

Biểu đồ thời khắc (Timing Diagram) cũng là một biểu đồ tương tác mới có trong UML 2.0. Nó được dùng để diễn tả hành vi của một hay nhiều đối tượng bằng cách chỉ ra các giai đoạn trải qua trong thời gian. Nó hay được dùng trong thiết kế các phần mềm nhúng (chẳng hạn phần mềm điều khiển trong một hệ cung cấp chất đốt), tuy nhiên nó có thể dùng trong các phần mềm quản lý.

Có hai dạng cơ bản của biểu đồ thời khắc: dạng súc tích và dạng tăng cường.

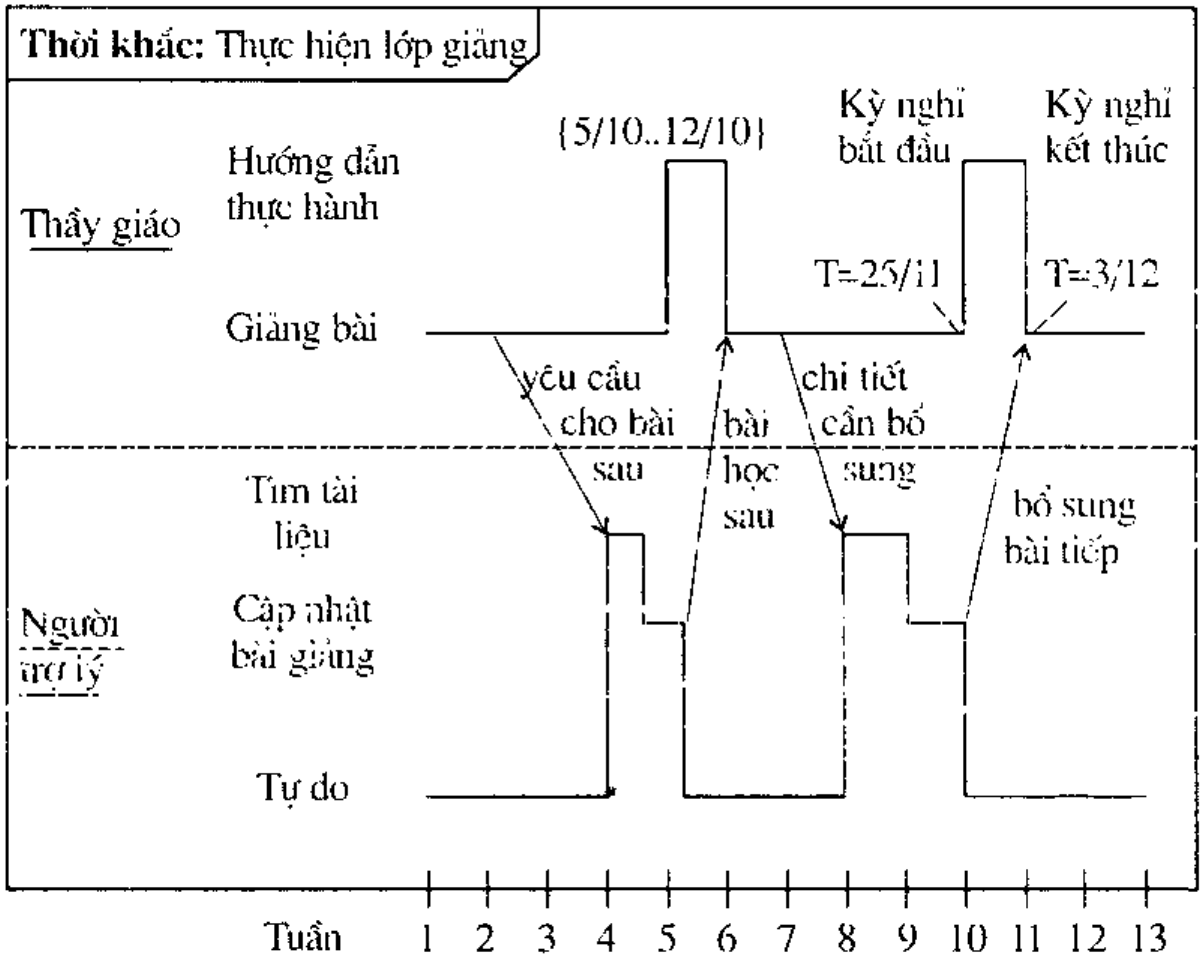
Hình V.45 cho biểu đồ thời khắc, ở dạng súc tích, của một đối tượng Lớp giảng. Nó cho ta thấy rõ các giai đoạn (trạng thái) mà Lớp giảng đó trải qua trong thời gian: Đề xuất, đưa vào kế hoạch, Đăng ký

sinh viên, Đang giảng, Thi kết thúc và Chấm dứt. Bên dưới các giai đoạn, ta chỉ ra các thời gian tương ứng.



Hình V.45. Biểu đồ thời khắc dạng súc tích

Hình V.46 cho một biểu đồ thời khắc ở dạng tăng cường diễn tả những gì xảy ra khi một lớp giảng đang thực hiện (đang giảng). Có hai đối tượng tham gia việc thực hiện lớp giảng, đó là Thầy giáo và Người trợ lý. Thầy giáo thực hiện công việc trên lớp, bao gồm giảng bài, hướng dẫn thực hành và cho điểm; còn người trợ lý làm các công việc ngoài lớp, như nghiên cứu tài liệu mới để bổ sung kịp thời vào bài giảng. Đối với mỗi đối tượng đó, ta vẽ một đường thời gian, và đường thời gian này chuyển đổi giữa các công việc theo từng thời kỳ. Các mũi tên nối giữa hai đường thời gian là các thông điệp giữa các đối tượng. Các ràng buộc về thời gian, như là {5/10... 10/10}, hoặc các kích thích như là Kỳ nghỉ bắt đầu, Kỳ nghỉ kết thúc, T = 25/11... có thể thêm vào ở những chỗ cần thiết. Mép dưới của biểu đồ là một thước thời gian, được khắc theo tuần.



Hình V.46. Biểu đồ thời khắc dạng tăng cường