Tạp chí Tin học và Điều khiển học, T. 18, S. 4 (2002), 375–383

MỘT SỐ VẤN ĐỀ TRONG MÔ HÌNH HÓA HỆ THỐNG PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

ĐOÀN VĂN BAN, DOÃN NGỌC LIÊN, HOÀNG QUANG

Abstract. Modeling is a central part of all the activities that lead up to the deployment of good software. A model of an object-oriented software system is made in a modeling language, such as UML (Unified Modeling Language). The model has both semantics and notation and can take various forms that include both pictures and text. We build models to better understand the system we are building, often exposing opportunities for simplification of reality and reuse. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system.

Object-oriented software engineering is the process of systematically building software systems using objects and classes and the relationships between them. The process proceeds in stages by systematically building a series of object models, each represented by several diagrams and documents. Each model describes important aspects of the real world or of the software systems that will realize the desired solution.

The purpose of this paper is to present two main issues in software modeling, namely requirement system model- use case diagram and conceptual model-class diagram. The translation of Object Model to Relational Model and the mapping between UML and Entity-Relationship model are also elaborated.

Tóm tắt. Mô hình hóa là nhiệm vụ trọng tâm của mọi hoạt động trong quá trình phát triển một phần mềm tốt, chất lượng cao, đáp ứng các yêu cầu của người sử dụng. Để thực hiện mô hình hoá, tức là để hiểu hệ thống cần phát triển, chúng ta có thể sử dụng những ngôn ngữ mô hình hoá hệ thống như UML (Unified Modeling Language). Mô hình phải thể hiện được cả ngữ nghĩa lẫn các ký hiệu một cách thống nhất, dễ hiểu, dễ thảo luận và dễ trao đổi giữa các nhóm phát triển phần mềm. Mô hình là một hình ảnh thực tại của bài toán mà chúng ta đang xét, được diễn đạt bằng hình thức dễ hiểu bằng văn bản, biểu đồ, đồ thị, công thức hay phương trình toán học, v.v. Ngôn ngữ mô hình hoá thống nhất UML bao gồm tập các kỹ thuật mô tả trực quan, phần lớn là các kỹ thuật đồ thị, biểu đồ để đặc tả và làm tài liệu theo hướng công nghiệp cho các hệ thống phần mềm hướng đối tượng.

Kỹ nghệ phần mềm hướng đối tượng là quá trình xây dựng một cách hệ thống các hệ thống phần mềm trên cơ sở sử dụng các đối tượng, kiểu (type) và các lớp. Cấu trúc và thiết kế hệ thống được định nghĩa chủ yếu bởi các kiểu, các lớp và mối quan hệ giữa chúng. Quá trình này gồm nhiều giai đoạn thực hiện một cách có hệ thống để xây dựng hàng loạt các mô hình các đối tượng, mỗi mô hình lại được biểu diễn bởi một số biểu đồ và các tài liệu. Mỗi mô hình mô tả về một phương diện quan trọng, nêu đặc trưng hay cách nhìn về từng lĩnh vực trong hệ thống phần mềm cần phát triển và các xây dựng các mẫu để mô phỏng và thực thi.

Bài trình bày một số khía cạnh trong mô hình phân tích, nắm bắt các yêu cầu hệ thống và mô hình phân tích hệ thống hướng đối tượng sử dụng UML. Vấn đề chuyển đổi giữa các mô hình (đối tượng và quan hệ) và giữa mô hình lớp trong UML với mô hình thực thể - liên kết (ERM) cũng được đề cập để nhằm xây dựng được những hệ thống phần mềm tích hợp ứng dụng.

1. KHÁI NIỆM ĐỐI TƯỢNG THỐNG NHẤT QUÁ TRÌNH MÔ HÌNH HÓA HỆ THỐNG

1.1. Những khái niệm cơ bản của phương pháp hướng đối tượng

Kỹ nghệ phần mềm hướng đối tượng là quá trình xây dựng một cách hệ thống các hệ thống phần mềm trên cơ sở sử dụng các đối tượng, kiểu và các lớp.

Cấu trúc và thiết kế hệ thống được định nghĩa chủ yếu bởi các kiểu, các lớp và mối quan hệ

giữa chúng. Quá trình (process) này gồm nhiều giai đoạn thực hiện một cách có hệ thống để xây dựng hàng loạt các mô hình các đối tượng, mỗi mô hình lại được biểu diễn bởi một số biểu đồ và các tài liệu.

Hệ thống phần mềm được xem như là tập các đối tượng tương tác với nhau để thực hiện các yêu cầu của người sử dụng.

Dữ liệu được biểu diễn bằng các *thuộc tính* và được sử dụng để mô tả các tính chất, đặc trưng của các đối tượng; còn *hành vi* (behavior) được thể hiện bằng các *thao tác* (operation), các *phương thức* (methods) hay hàm. Những thuộc tính này không chia sẻ với các đối tượng khác được gọi là các *thuộc tính riêng* (private), còn những thuộc tính khác có thể *công khai* (public), nghĩa là các đối tượng khác có thể quan sát được (đọc, hay làm thay đổi) chúng.

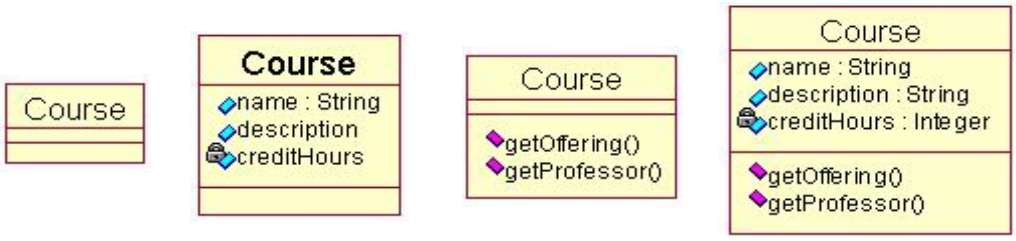
Các đối tượng trong một hệ thống chỉ liên hệ với nhau thông qua việc *gửi và nhận các thông điệp* (messages). Khi một đối tượng nhận được một thông điệp yêu cầu thực hiện những thao tác nào đó, nó có thể thao tác trên những dữ liệu riêng hoặc gửi đi thông điệp mới cho một số đối tượng khác để yêu cầu những thông tin, dữ liệu trả lời cho đối tượng gửi thông điệp.

Một *đối tượng* là biểu hiện cụ thể hay còn gọi là thể hiện (instance) của một kiểu, hoặc một lớp.

Một *kiểu* là mô tả chung cho một tập hợp các đối tượng cùng chia sẻ một số thao tác, thuộc tính và một số quan hệ nhất định (xem như tầng của các lớp).

Một *lớp* là sự thể hiện của một kiểu. Lớp định nghĩa các phương thức thực hiện cài đặt các thao tác của một kiểu. Các nhà lập trình thường đồng nhất *kiểu với lớp* và trong giai đoạn thiết kế nhiều người cũng xem kiểu như là lớp.

Ví dụ, hình sau giới thiệu bốn ký pháp khác nhau của Rose [10] cho kiểu (hay lớp) trong UML



Một đặc điểm quan trọng của UML là khả năng *mở rộng, định nghĩa các kiểu mẫu (stereotype)* ([10], [12]), cho phép định nghĩa những loại kiểu mới của kiểu, lớp để các cấu trúc của các mô hình khác nhau có thể sử dụng và phân loại chúng.

Thẻ <<stereotype>> còn được sử dụng để phân biệt các loại quan hệ giữa các đối tượng, kiểu và lớp. Jacobson [6] sử dụng ba kiểu mẫu chuẩn: kiểu <<boundary>>, kiểu <<entity>> và kiểu <<control>>. Mỗi kiểu này đóng một vai trò đặc biệt trong mô hình phân tích hệ thống:

+ *Đối tượng* << *entity* >> là những đối tượng nghiệp vụ của hệ thống, tồn tại và thực hiện theo kịch bản mà chúng tham gia. *Kiểu* <<*entity*>> được sử dụng để mô hình các đối tượng nghiệp vụ, đại diện cho các "sự vật" (things), các thực thể, có liên quan đến nhiều trường hợp sử dụng.

+ *Đối tượng* <<*boundary*>> xử lý việc trao đổi của hệ thống với thế giới xung quanh. *Kiểu* <<*boundary*>> sử dụng để mô tả sự chuyển đổi các sự kiện, đối tượng từ ngoài vào hệ thống và ngược lại.

+ *Đối tượng* <<*control*>> thực hiện một trường hợp sử dụng, thường điều khiển hoặc hợp tác với một số đối tượng khác để thực hiện công việc của hệ thống. Hành vi của đối tượng thuộc kiểu này là độc lập (về các phương tiện, phương thức) với sự trao đổi của hệ thống với thế giới bên ngoài.

Các đối tượng có thể có một hoặc nhiều *mối quan hệ* với các đối tượng khác, như:

+ Một số lớp đối tượng có mối quan hệ *liên kết* (association) với nhau.

+ Một số lớp hay kiểu quan hệ với nhau vì có những đặc điển chung, gần giống nhau, có thể gộp chung thành *tổng quát hoá* (generalization) hay từ đó có thể chi tiết hơn theo quan hệ *kế thừa*

(inheritance).

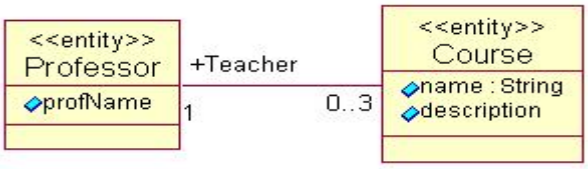
+ Một số đối tượng bao gồm hay chứa một số đối tượng khác - một trường hợp đặc biệt của liên kết là quan hệ *kết nhập* (aggregation).

+ Một số đối tượng gửi thông điệp cho hay sử dụng các đối tượng khác được thể hiện bởi *quan hệ phụ thuộc*.

Những mối quan hệ trên thể hiện sự liên kết ngữ nghĩa giữa các phần tử (kiểu, lớp, đối tượng) trong các mô hình. *Trong một số trường hợp, những mối quan hệ này có thể giao nhau*, nghĩa là một quan hệ có thể vừa là kết nhập vừa là liên kết hoặc phụ thuộc. Tuy nhiên việc phân loại được các quan hệ giúp chúng ta xây dựng được những thiết kế hợp lý, có tính cố kết và chặt chẽ về mặt logic.

1.1.a. Quan hệ liên kết

Các lớp đối tượng có thể liên hệ với nhau theo quan hệ liên kết ngữ nghĩa: một - một, một - nhiều và nhiều - nhiều giống như trong mô hình liên kết thực thể. Loại quan hệ này thường được sử dụng để chỉ ra rằng một đối tượng cần truy nhập hay tham chiếu tới một hay một số đối tượng khác. Ví dụ

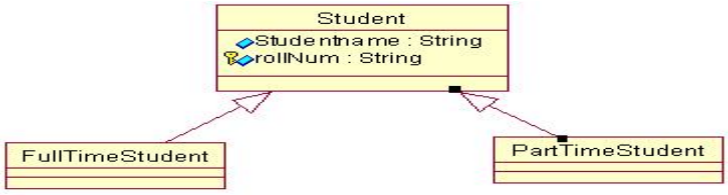


1.1.b. Quan hệ kết nhập

Một lớp đối tượng có thể hợp thành từ một số đối tượng thuộc các lớp khác, hoặc kết nhập (về mặt logic) từ một số đối tượng ở các lớp khác.

1.1.c. Quan hệ tổng quát hoá hay kế thừa

Từ một số lớp đối tượng có những đặc trưng tương tự (thuộc tính, hành vi gần giống nhau), chúng ta có thể thực hiện tổng quát hoá bằng cách gộp những nét chung của chúng lại để tạo ra lớp mới đại diện chung cho các lớp đó hoặc từ một số lớp đã xây dựng tốt, tạo ra được những lớp mới kế thừa những được tổ chức công khai của các lớp đó và bổ sung thêm một số đặc trưng mới để thi hành các yêu cầu mới.



1.1.d. Quan hệ phụ thuộc

Các đối tượng trong một hệ thống phải trao đổi với nhau bằng cách gửi và nhận các thông điệp. Khi thực hiện một công việc, đối tượng A cần một số dữ liệu và những dữ liệu này lại được một đối tượng B cung cấp. Khi đó A phải gửi cho B yêu cầu cung cấp thông tin và do vậy A sẽ phụ thuộc vào B. Một cách tổng quát, quan hệ phụ thuộc thể hiện mối quan hệ giữa các phần tử trong các mô hình, trong đó sự thay đổi của phần tử này sẽ ảnh hưởng tới các phần tử khác.

Nhận xét: Vấn đề quan trọng trong giai đoạn phân tích là xác định đầy đủ và chính xác mọi quan hệ giữa các lớp thể hiện đúng mối liên hệ của chúng trong hệ thống thực tế.

1.2. Mô hình trường hợp sử dụng: nắm bắt các yêu cầu của hệ thống

Ở mỗi giai đoạn trong quá trình thực hiện mô hình hoá hệ thống, vấn đề hữu ích là cần có những mô hình ở mức độ chi tiết khác nhau với nhiều loại đối tượng thể hiện được các nhu cầu của nhiều nhóm người tham gia phát triển hệ thống.

Mô hình thể hiện tốt nhất các yêu cầu và mối quan hệ với tất cả các phần tử có liên quan đến hệ thống là mô hình hay biểu đồ trường hợp sử dụng (use case diagram).

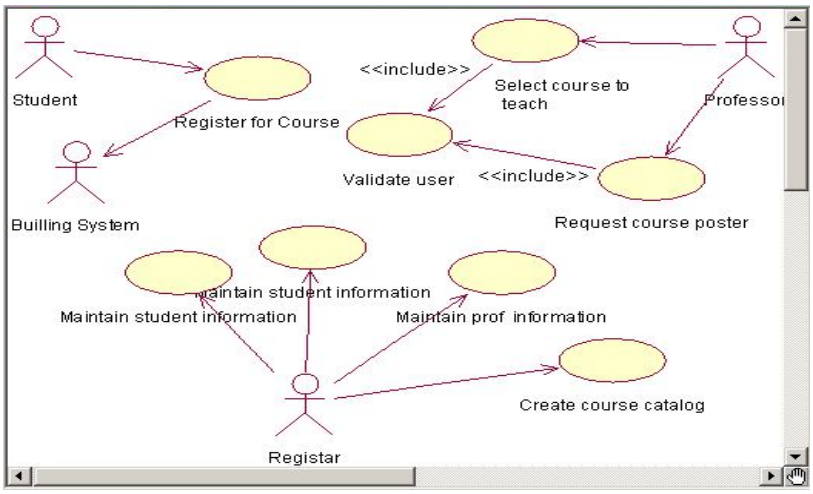
Mục đích chính của mô hình này là định nghĩa được những cái (what) mà hệ thống cần thực hiện và thể hiện được tất cả những gì mà cả những người kỹ sư phần mềm và khách hàng (người sử dụng) cùng thống nhất về yêu cầu của hệ thống [4].

Mô hình trường hợp sử dụng có các kiểu chính là: kiểu "Tác nhân" (Actor) và kiểu "Trường hợp sử dụng" (Use case type).

+ Những người sử dụng và các phần tử hay hệ thống khác có tương tác (sử dụng hay cung cấp thông tin) với hệ thống, được xem như là các tác nhân (ngoài). Nói chung, *tác nhân là bất kỳ cái gì đó ở bên ngoài và có tương tác với hệ thống* (làm thay đổi dữ liệu và các sự kiện xảy ra trong hệ thống). Mỗi tác nhân đều có một vai trò nhất định với hệ thống, có thể gửi hay nhận thông điệp từ hệ thống.

+ *Hành vi (các chức năng) của hệ thống được mô tả bởi các trường hợp sử dụng* và tất cả các trường hợp sử dụng đều được phát triển theo yêu cầu của các tác nhân. Nói cách khác, *hệ thống sẽ làm tất cả những gì mà các tác nhân yêu cầu và mọi trường hợp sử dụng đều phải được khởi động bởi các tác nhân.*

Ví dụ, biểu đồ trường hợp sử dụng của hệ "Đăng ký môn học theo học phần" được xây dựng như sau:



1.3. Mô hình phân tích: biểu đồ lớp đặc tả hệ thống

Mô hình phân tích là mô hình các thành phần hệ thống ở mức cao, bỏ qua những chi tiết ở mức thấp của môi trường cài đặt.

Đối tượng phân tích là những tình huống mà ở đó các đối tượng hoạt động được trong hệ thống

như mong muốn. Nhiều yếu tố như hệ quản trị CSDL, ngôn ngữ lập trình, môi trường phân tán, đánh giá hiệu năng của hệ thống, v.v., chưa được đề cập đến trong mô hình phân tích mà sẽ xác định cụ thể ở pha thiết kế.

Nhiệm vụ chính của pha phân tích là tìm cách ánh xạ các trường hợp sử dụng (yêu cầu) sang mô hình phân tích (biểu đồ lớp, biểu đồ trình tự, hợp tác, v.v.). Để thực hiện được điều đó thì người phân tích phải phân tích, nghiên cứu các mô tả về các yêu cầu, các trường hợp sử dụng và tìm ra những phần tử sẽ là các kiểu trong mô hình phân tích.

Hai vấn đề quan trọng nhất trong việc xây dựng mô hình phân tích là: xác định đầy đủ, chính xác các lớp đối tượng (kiểu) và mối quan hệ giữa chúng trong hệ thống.

1.3.a. Các phương pháp xác định các lớp đối tượng

Chúng ta có thể dựa vào các phương pháp sau để tìm ra được các lớp đối tượng:

1. Dựa vào mô tả các yêu cầu, các trường hợp sử dụng và các kịch bản, *các danh từ chung có thể là đại biểu các lớp.*
2. Dựa vào *kinh nghiệm và kiến thức* của người phân tích để xác định thêm những lớp cần bổ sung hoặc loại bỏ đi những danh từ chung không phải là lớp.
3. Dựa vào *hồ sơ, tài liệu của những hệ thống tương tự hoặc có liên quan* để tham khảo và tìm ra những lớp đối tượng cho mô hình.
4. Trao đổi với *các chuyên gia hệ thống.*
5. Ngoài ra còn có thể dựa vào sự *phân loại phạm trù các khái niệm* như: khái niệm lôgic, vật lý, nơi chốn, danh mục, giao dịch, sự kiện, v.v.

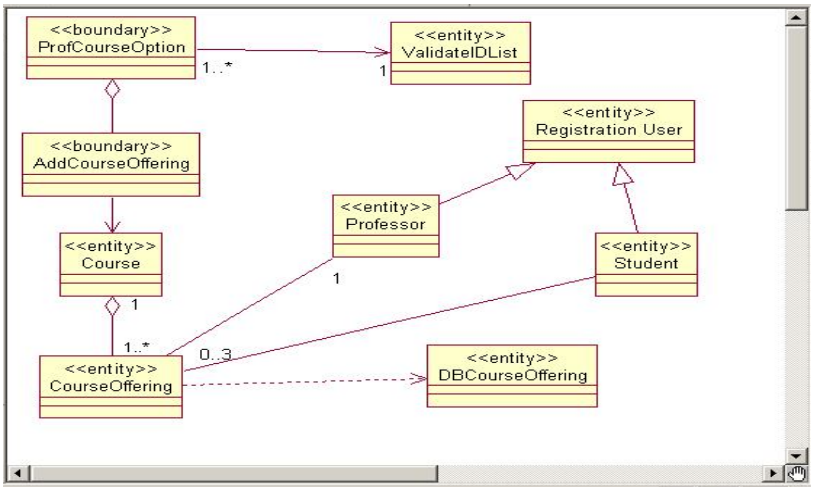
1.3.b. Cách xác định mối quan hệ giữa các lớp

Các đối tượng tương tác với nhau để thực hiện các trường hợp sử dụng, mỗi đối tượng giữ một hay một số vai trò nhất định trong ít nhất một trường hợp sử dụng. Phương pháp chung để thực hiện là:

1. Phân tích mối quan hệ, tương tác giữa các thành phần trong hệ thống,
2. Tìm các mối liên kết chung giữa chúng,
3. Xác lập quan hệ tổng quát (kế thừa) giữa một số lớp,
4. Xác định mối quan hệ kết nhập (giữa tổng thể và bộ phận) của một số lớp.

Tương tự như trên, chúng ta có thể dựa vào việc phân loại phạm trù liên kết như bộ phận logic hoặc vật lý, "chứa", "bao gồm", "trao đổi", " liên quan", "có", "sở hữu", v.v. để xác định các quan hệ giữa các lớp đối tượng.

Biểu đồ lớp của UML là công cụ tốt nhất để xây dựng mô hình khái niệm cho hệ thống. Ví dụ, mô hình khái niệm của hệ thống "Đăng ký học theo môn học phần" được mô tả bằng biểu đồ lớp như sau:



Để hiểu và thực hiện đúng được các trường hợp sử dụng thì chúng ta phải mô hình chúng theo nhiều phương diện khác nhau, nghĩa là cần phải xây dựng thêm các biểu đồ mô tả sự tương tác giữa các đối tượng tham gia vào từng trường hợp sử dụng như: biểu đồ trình tự (sequence diagram), biểu đồ cộng tác (collaboration diagram) và biểu đồ trạng thái (state diagram), v.v.

2. MỘT SỐ VẤN ĐỀ TRONG VIỆC PHÁT TRIỂN HỆ THỐNG TÍCH HỢP ỨNG DỤNG

Xu thế phát triển hiện nay của CNTT là phát triển những hệ thống phần mềm tích hợp được nhiều mô hình, nhiều CSDL và nhiều công nghệ trong các môi trường khác nhau nhằm đáp ứng mọi yêu cầu của nhiều loại khách hàng khác nhau. Để tận dụng được những gì đã có và phát triển ứng dụng được các công nghệ mới đòi hỏi phải thực hiện các phép biến đổi tương đương ngữ nghĩa giữa các mô hình với nhau.

2.1. Chuyển đổi giữa hai mô hình đối tượng và quan hệ

Các hệ thống phần mềm được phát triển đều dựa trên một trong hai cách tiếp cận: hướng chức năng và hướng đối tượng. Vấn đề quan trọng đặt ra là hai mô hình này có tương thích với nhau không, có thể truy vấn vào được những hệ CSDL mà không cần quan tâm tới mô hình của chúng hay không?

Trước tiên chúng ta nhận thấy có những "trở ngại" nhất định trong việc đối sánh hai mô hình trên. Mô hình đối tượng dựa chủ yếu trên các nguyên lý của kỹ nghệ phần mềm như các nguyên lý cặp bộ (coupling), cố kết (cohesion), bao bọc (encapsulation), và kế thừa (inheritance) còn mô hình quan hệ lại dựa vào các nguyên lý của toán học, đó chính là lý thuyết tập hợp. Mô hình đối tượng tập trung vào việc phát triển những ứng dụng với các đối tượng - một cấu trúc gộp chung cả dữ liệu với các hành vi là trọng tâm, còn mô hình quan hệ đặt trọng tâm vào việc tổ chức lưu trữ dữ liệu. Những khác biệt đó kéo theo cả những điểm mạnh, yếu khác nhau của mỗi mô hình.

Như vậy, để phát triển được những hệ thống ứng dụng tích hợp và tận dụng được sự tiến bộ của khoa học công nghệ hướng đối tượng thì chúng ta phải chọn một trong các giải pháp sau:

1) Xây dựng CSDL hướng đối tượng và giao diện ứng dụng,
2) Tái kỹ nghệ lại (Reengineering) CSDL thành hướng đối tượng từ CSDL quan hệ,
3) Phát triển hệ thống đa CSDL (multi database system).

Trong cả ba cách giải quyết trên đều đòi hỏi phải thực hiện các phép chuyển đổi, ánh xạ giữa hai mô hình đó.

2.1.a. Chuyển đổi các đối tượng sang các CSDL quan hệ

Dựa vào kỹ thuật ánh xạ của Scott [9], chúng ta có thể xây dựng thuật toán để chuyển các đối tượng sang các CSDL quan hệ như sau:

1) *Ánh xạ các thuộc tính sang các cột của các bảng quan hệ.* Mỗi thuộc tính của lớp có thể ánh xạ sang không hoặc một số cột trong CSDL quan hệ và ngược lại một số thuộc tính lại có thể ánh xạ vào một cột của bảng dữ liệu.
2) *Ánh xạ các quan hệ, liên kết và kết nhập.* Để thực hiện được các phép chuyển đổi hiệu quả thì vấn đề trước tiên là chúng ta phải hiểu và phân biệt được các quan hệ đó. Trên cơ sở phân tích các mối ràng buộc trong CSDL quan hệ, mối quan hệ được thiết lập thông qua các khoá ngoại (foreign key), chúng ta có thể cài đặt các mối quan hệ liên kết (một - một, một- nhiều và cả nhiều-nhiều) giữa các đối tượng trong CSDL quan hệ bằng cách sử dụng các định danh đối tượng (OID) làm các khoá ngoại.
3) *Cài đặt quan hệ kế thừa.* Quan hệ kế thừa là một cơ chế rất ưu việt của phương pháp hướng đối tượng, song nó cũng gây ra không ít khó khăn, trở ngại trong việc đảm bảo tính nhất quán dữ liệu trong các hệ thống và trong việc tổ chức lưu trữ và tìm kiếm. Hiện nay có *ba giải pháp để ánh xạ quan hệ kế thừa vào CSDL quan hệ*:

• *Sử dụng một bảng thực thể cho cả cấu trúc phân cấp lớp* (hierarchy). Những lớp có quan hệ kế thừa đơn với nhau sẽ tạo ra một cấu trúc phân cấp có thứ bậc. Để cài đặt cấu trúc đó, chúng ta có thể tạo ra một bảng thực thể có các cột thuộc tính chính là các thuộc tính của tất cả các lớp trong cấu trúc.

• *Sử dụng mỗi bảng thực thể cho các lớp cụ thể.* Có hai loại lớp trong mô hình hướng đối tượng: lớp trừu tượng, lớp không có các đại diện (thể hiện) trong cụ thể và lớp cụ thể.

• *Sử dụng một bảng dữ liệu cho mỗi lớp.* Với mỗi lớp tạo ra một bảng bao gồm OID và các thuộc tính xác định lớp đó.

2.1.b. Chuyển đổi CSDL quan hệ sang CSDL đối tượng

Để chuyển đổi dữ liệu từ mô hình quan hệ sang đối tượng, điều kiện tiên quyết là biên dịch các lược đồ quan hệ sang biểu đồ đối tượng mà vẫn đảm bảo được ngữ nghĩa và các ràng buộc của dữ liệu [4]. Ngoài chuyển đổi mô hình thì vấn đề chuyển đổi dữ liệu cũng cần phải được thực hiện, bao gồm việc chuyển các bộ trong quan hệ vào các tệp có cấu trúc thích hợp và tái nạp chúng vào các tệp được tổ chức theo các đối tượng ([5],[11]). Các phương pháp này bảo toàn được các ràng buộc của CSDL quan hệ thông qua ánh xạ các phụ thuộc dữ liệu tương đương.

Một vấn đề quan trọng nữa là các phương pháp, ngôn ngữ truy vấn đối với các mô hình dữ liệu khác nhau cũng rất khác nhau. Vấn đề nghiên cứu các phương pháp chuyển đổi và tối ưu hoá các câu truy vấn từ mô hình quan hệ sang mô hình đối tượng hay ngược lại cũng được nhiều người quan tâm. Trong [3] các tác giả đã sử dụng các phép biến đổi đại số để chuyển đổi từng phép toán quan hệ sang biểu thức đại số đối tượng tương ứng.

2.2. Chuyển đổi giữa hai mô hình UML với mô hình thực thể liên kết ERM

Mô hình thực thể liên kết (Entity-Relationship Model), viết tắt là ERM, rất quan trọng và được sử dụng nhiều để mô hình dữ liệu và thiết kế CSDL.

Mặt khác UML ngày càng trở nên phổ biến, được sử dụng nhiều trong phân tích, thiết kế hướng đối tượng (PT/TKHĐT - OOA/OOD) và hiện nay nhiều hãng sản xuất phần mềm lớn sử dụng nó như là chuẩn công nghiệp để mô hình hoá các hệ thống phần mềm.

Một thiết kế CSDL cũng có thể được dẫn xuất ra từ những phép chuyển đổi các đối tượng sang các thực thể. Các lớp đối tượng trong một ứng dụng có thể được lưu trữ trong CSDL hướng đối tượng hay CSDL quan hệ. Trường hợp sau đòi hỏi phải chuyển đổi mô hình đối tượng sang mô hình quan hệ.

Mặt khác, ERM là lựa chọn tốt nhất để biểu diễn mô hình dữ liệu trực quan và nhiều nhà thiết kế hệ thống đã quen sử dụng ERM để thiết kế CSDL. Ngoài ra, có thể có một số phần đã được thiết kế bằng ERM lại cần đưa vào biểu đồ lớp, khi đó cần thiết phải tích hợp hệ thống và tích hợp các CSDL. Trong các trường hợp trên đều đòi hỏi chuyển đổi tự động giữa các mô hình, biểu đồ lớp UML với ERM.

Mục đích chính của chúng ta là phát triển một môi trường (platform) đa mô hình, đa công cụ thiết kế, cho phép nhiều công cụ phân tích, thiết kế và lập trình khác nhau, có khả năng dựa trên những mô hình khác nhau (như UML, ERM, v.v.).

2.2.a. Ánh xạ từ UML sang ERM

Biểu đồ lớp là trọng tâm của mô hình UML, thể hiện các thành phần và mối quan hệ logic giữa các thành phần trong hệ thống. Trong ERM có các khái niệm như: các thực thể và các thuộc tính, thuộc tính khoá, và các mối quan hệ liên kết giữa các thực thể được thể hiện bằng các ký pháp đồ họa thành biểu đồ thực thể liên kết (ERD) tương ứng.

Dựa vào phương pháp của Yongzhen Ou [12], chúng ta có thể xây dựng thuật toán chuyển đổi biểu đồ lớp sang biểu đồ thực thể liên kết được thực hiện theo các bước sau:

1) Với mỗi biểu đồ lớp chính tắc (lớp không liên kết) trong biểu đồ lớp tạo ra một kiểu thực thể (entity type) tương ứng,
   + Tên của kiểu thực thể được xác định từ tên của lớp,

+ Các thuộc tính của lớp chuyển sang các thuộc tính của các thực thể,

+ Những thuộc tính có tên là tên một lớp thì bổ sung thêm "_ID" vào thành tên của thuộc tính trong kiểu thực thể và đánh dấu là khoá nguyên thuỷ (primary key).

2) Với mỗi quan hệ kế thừa giữa hai lớp, tạo ra một kiểu quan hệ "is-a" (là kiểu con) giữa hai kiểu thực thể tương ứng.

3) Với mỗi quan hệ kết nhập, với mỗi quan hệ liên kết, tạo ra một kiểu thực thể tương ứng với những lớp có liên quan, trong đó:

+ Nếu có các tên của vai diễn ở mỗi phía của quan hệ thì chuyển chúng sang kiểu quan hệ kết quả tương ứng.

+ Nếu có các bản số (multiciple) xác định số các đối tượng tham gia ở mỗi phía của quan hệ thì chuyển tương ứng sang kiểu quan hệ kết quả với các vị trí đổi nhau.

Kết quả của thuộc tính trên là biểu đồ thực thể liên kết tương đương với biểu đồ lớp trong UML.

2.2.b Ánh xạ từ ERM sang UML

Rumbaugh đã khẳng định, kỹ thuật mô hình đối tượng OMT (Object Modeling Technique) là dạng phát triển của ERM với việc bổ sung thêm một số khái niệm mới. Tiếp theo UML lại là dạng phát triển mở rộng của OMT [1].

Thuật toán thực hiện phép chuyển đổi này gồm các bước sau:

1) Với mỗi kiểu thực thể trong biểu đồ thực thể liên kết, tạo ra một lớp tương ứng trong UML có cùng tên và có cùng tập các thuộc tính như nhau;

2) Với mỗi kiểu quan kệ:

+ Nếu là dạng quan hệ "is-a", tạo ra tương ứng quan hệ kế thừa giữa hai lớp biểu diễn cho hai kiểu thực thể tương ứng,

+ Nếu là quan hệ xác định (identifying) thì tạo ra liên kết một chiều giữa các lớp tương ứng, ngược lại tạo ra quan hệ liên kết (hai chiều) giữa chúng,

+ Nếu có các tên gọi của vai diễn của mỗi đầu của quan hệ thì chuyển chúng sang các quan hệ kết quả,

+ Nếu có các bản số trên mỗi đầu của quan hệ thì chuyển sang quan hệ kết quả với chiều đổi nhau.

Hai thuật toán nêu trên giúp chúng ta chuyển đổi gần như "tương đương" ngữ nghĩa giữa hai mô hình.

3. KẾT LUẬN

UML là ngôn ngữ mô hình hoá trực quan, chuẩn hoá, rất mạnh và phong phú về ngữ nghĩa, hỗ trợ để đặc tả và làm tư liệu phần mềm theo hướng công nghiệp hoá cho các hệ thống phần mềm hướng đối tượng. Những vấn đề cơ bản của quá trình phát triển phần mềm như: phân tích các yêu cầu và các lớp của hệ thống phần mềm được đề cập trong báo cáo thông qua việc phân tích hệ thống "Đăng ký môn học theo học phần". Bài báo cũng giới thiệu một số thuật toán chuyển đổi tương đương về ngữ nghĩa giữa mô hình quan hệ với mô hình đối tượng; mô hình hướng đối tượng trong UML với mô hình thực thể liên kết (ERM). Những thuật toán này có thể sử dụng để xây dựng được những hệ thống phần mềm tích hợp ứng dụng, những hệ thống đa cơ sở dữ liệu.

TÀI LIỆU THAM KHẢO

[1] Booch G., Rumbaugh J. and Jacobson I., *The Unified Software Development Process*, Addision - Wesley, 1998.

[2] Đoàn Văn Ban, *Phân tích, thiết kế và lập trình hướng đối tượng*, NXB Thống kê 1997.

[3] Đoàn Văn Ban, Hoàng Quang, *Chuyển đổi các biểu thức đại số quan hệ thành câu truy vấn trong mô hình dữ liệu hướng đối tượng,* Báo cáo toàn văn Hội nghị KH 25 Viện Công nghệthông tin, Hà Nội, 2001.

[4] Fong J., *Converting Relational to Object-Oriented DataBases, SIGMOD Record* 26 (1)
[5] Hoàng Bảo Hùng, Lê Mạnh Thạnh, Một phương pháp chuyển đổi dữ liệu từ cơ sở dữ liệu quan hệ sang CSDL hướng đối tượng, *Tạp chí Khoa học Huế*, (11) 2002.
[6] Jacobson I., Martin Griss, Patrik Jonsson, *Software Reuse*, Addision - Wesley, 1997.
[7] Kafura D., *Object-Oriented Software Design And Construction With Java*, Prentice Hall, 2000.
[8] Larman C., *Applying UML and Patterrns: An Instruction to Object-Oriented Analysis and Design*, Prentice Hall, 1997.
[9] Scott W. Ambler, *Mapping Objects to Relational DataBase, What you need to know and why?*, http://www-4.ibm.com/software/developer/library/mapping-to-rdb/index.html, 2000.
[10] Quatrani T., *Visual Modeling With Rational Rose and UML*, Addison-Wesley, 2000.
[11] Siu B., Cheung T. Y., *Towards a Method for Schema Translation from Relational to Object-Oriented DataBases, Multimedia, Knowledge-Based & Object-Oriented DataBase*, Springer, 1996.
[12] Yongzhen Ou, *On Mapping between UML and Entity-Relationalship Model*, http://citeser.nj.nec.com

Nhận bài ngày 10-5-2002

Hoàng Quang - Trường Đại học Khoa học Huế
Đoàn Văn Ban, Doãn Ngọc Liên -Viện Công nghệ Thông tin