Các mô hình cơ bản
TRONG PHÂN TÍCH VÀ THIẾT KẾ
HƯỚNG ĐỐI TƯỢNG

MỜI CÁC BẠN TÌM ĐỌC

1. KỸ NGHỆ PHẦN MỀM
2. LÝ THUYẾT HỆ THỐNG VÀ ĐIỀU KHIỂN HỌC
3. KỸ THUẬT PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG HƯỚNG CẤU TRÚC
4. NHẬP MÔN PHÂN TÍCH THÔNG TIN CÓ BẢO MẬT
5. SÁNG TẠO TRONG THUẬT TOÁN VÀ LẬP TRÌNH (3 TẬP)
6. GIÁO TRÌNH NGÔN NGỮ LẬP TRÌNH C/C++
7. GIÁO TRÌNH NGÔN NGỮ LẬP TRÌNH PASCAL
8. GIÁO TRÌNH MẬT MÃ HỌC VÀ HỆ THỐNG THÔNG TIN AN TOÀN
9. GIÁO TRÌNH CƠ SỞ DỮ LIỆU
10. GIÁO TRÌNH CƠ SỞ DỮ LIỆU PHÂN TÁN
11. GIÁO TRÌNH CƠ SỞ DỮ LIỆU: LÝ THUYẾT VÀ THỰC HÀNH
12. BÀI TẬP CƠ SỞ DỮ LIỆU
13. CƠ SỞ DỮ LIỆU QUAN HỆ & ỨNG DỤNG
14. CƠ SỞ DỮ LIỆU QUAN HỆ VÀ CÔNG NGHỆ PHÂN TÍCH - THIẾT KẾ

Giá: ...đ

TS. LÊ VĂN PHÙNG

CÁC MÔ HÌNH CƠ BẢN TRONG PHÂN TÍCH VÀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

TS. LÊ VĂN PHÙNG

Các mô hình cơ bản
TRONG PHÂN TÍCH VÀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

TS. LÊ VĂN PHÙNG

Các mô hình cơ bản
TRONG PHÂN TÍCH VÀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

Mã số: HT 08 HM 10

THUẬT NGỮ VÀ TỪ VIẾT TẮT

1. Tiếng Anh

ATM	Automated Teller Machine	Máy rút tiền tự động
GRASP	General Responsibility Assignment Software Pattern	Mẫu gán trách nhiệm cơ bản
ID	Identifier	Định danh
ODBMS	Object Database Management System	Hệ quản trị cơ sở dữ liệu đối tượng
OID	Object Identifiers	Định danh đối tượng
OMT	Object Modeling Technique	Kỹ thuật mô hình hóa đối tượng
OOSE	Object-Oriented Software Engineering	Công nghệ phần mềm hướng đối tượng
PC	Person Computer	Máy tính cá nhân
PIN	Personal Identification Number	Số nhận dạng cá nhân
RAD	Rapid Application Development	Phát triển ứng dụng nhanh
RDBMS	Relational Database Management System	Hệ quản trị cơ sở dữ liệu quan hệ
RUP	Rational Unified Process	Tiến trình hợp nhất Rational
UI	User Interface	Giao diện người dùng
UML	Unified Modeling Language	Ngôn ngữ mô hình hóa hợp nhất
UPC	Universal Product Code	Mã sản phẩm phổ biến

2. Tiếng Việt

CSDL	Cơ sở dữ liệu
NSD	Người sử dụng

LỜI NÓI ĐẦU

Cách đây khoảng 20 năm, để khắc phục những vấn đề tồn tại trong cách tiếp cận hướng cấu trúc, người ta đã nghiên cứu một mô hình mới thích hợp cho việc phát triển phần mềm lớn và phức tạp, đó là mô hình hướng đối tượng. Cách tiếp cận hướng đối tượng đã ngày càng trở nên phổ biến. Trong các dự án phát triển hệ thống lớn, ngôn ngữ mô hình hóa hợp nhất - UML đã được ưu tiên cho quá trình phân tích thiết kế hệ thống. Ngày nay, nó được coi là một chuẩn quốc tế được tổ chức tiêu chuẩn quốc tế ISO chấp nhận. Việc nắm vững các kiến thức cơ bản về mô hình, quá trình mô hình hóa, các kỹ thuật xây dựng mô hình là những yêu cầu bắt buộc cho bất cứ ai muốn phân tích và thiết kế một hệ thống lớn theo hướng đối tượng.

Nhằm giúp sinh viên, nghiên cứu sinh và các lập trình viên có tài liệu tham khảo tương đối hệ thống về phân tích và thiết kế theo hướng đối tượng, Nhà xuất bản Thông tin và Truyền thông trân trọng giới thiệu cuốn sách ***"Các mô hình cơ bản trong phân tích và thiết kế hướng đối tượng"*** do TS. Lê Văn Phùng (Viện Công nghệ thông tin thuộc Viện Khoa học và Công nghệ Việt Nam) biên soạn.

Nội dung cuốn sách gồm 10 chương:

Chương 1: Tổng quan về mô hình hóa phần mềm

Chương 2: Các khái niệm cơ bản trong phân tích và thiết kế hướng đối tượng

Chương 3: Yêu cầu hệ thống và mô hình nghiệp vụ

Chương 4: Mô hình phân tích đối tượng

Chương 5: Các mô hình phân tích động thái

Chương 6: Các mô hình thiết kế tương tác

Chương 7: Mô hình kiến trúc logic

Chương 8: Mô hình kiến trúc vật lý

Chương 9: Mô hình phân tích và thiết kế một ca sử dụng

Chương 10: Mô hình thiết kế đối tượng

Hy vọng cuốn sách sẽ thực sự hữu ích cho các bạn đọc yêu công nghệ thông tin, ham mê phân tích thiết kế một hệ thống thông tin, các bạn đồng nghiệp, giáo viên, sinh viên đại học, cao đẳng và học viên cao học chuyên ngành công nghệ phần mềm hoặc hệ thống thông tin,…

Nhà xuất bản xin trân trọng giới thiệu cùng bạn đọc và rất mong nhận được ý kiến đóng góp của quý vị. Mọi ý kiến đóng góp xin gửi về ***Nhà xuất bản Thông tin và Truyền thông*** - 18 Nguyễn Du, Hà Nội.

Xin trân trọng cảm ơn./.

NXB THÔNG TIN VÀ TRUYỀN THÔNG

1

TỔNG QUAN VỀ MÔ HÌNH HÓA PHẦN MỀM

1.1. TỔNG QUAN VỀ MÔ HÌNH HÓA

1.1.1. Khái niệm trừu tượng hóa

Để tìm hiểu về một thế giới phức tạp, mọi khoa học thực nghiệm đều phải vận dụng một nguyên lý cơ bản, đó là *sự trừu tượng hóa* (*Abstraction*). *Trừu tượng hóa là một nguyên lý của nhận thức, đòi hỏi phải bỏ qua những sắc thái (chi tiết của chủ điểm) không liên quan tới chủ định hiện thời, để tập trung hoàn toàn vào các sắc thái chính liên quan tới chủ định đó (từ điển Oxford).*

Theo Liberty J.,1998, *trừu tượng là nguyên lý bỏ qua những khía cạnh của chủ thể không liên quan đến mục đích hiện tại để tập trung đầy đủ hơn vào các khía cạnh còn lại. Trừu tượng hóa là đơn giản hóa thế giới thực một cách thông minh. Nó cho khả năng tổng quát hóa và ý tưởng hóa vấn đề đang xem xét. Chúng loại bỏ đi các chi tiết dư thừa mà chỉ tập trung vào các điểm chính, cơ bản.*

Trừu tượng là sự mô tả một cách khái quát một đối tượng thực và bỏ qua nhiều yếu tố, nhiều mặt không quan trọng của nó [23]. Sử dụng nguyên lý trừu tượng hóa có nghĩa là thừa nhận thế giới thực là phức tạp, thay vì cố gắng hiểu biết toàn bộ bằng lựa chọn một phần của vấn đề.

Trừu tượng bao gồm nhiều dạng: trừu tượng thủ tục, trừu tượng dữ liệu, trừu tượng điều khiển [11]. Trong đó trừu tượng dữ liệu là cơ chế mạnh, dựa trên cơ sở tổ chức suy nghĩ và đặc tả về các nhiệm vụ của hệ thống. Trừu tượng dữ liệu là nguyên tắc xác định kiểu dữ liệu cho các thao tác áp dụng cho đối tượng, với ràng buộc là các giá trị lưu trữ trong đối tượng chỉ được sửa đổi hay quan sát thông qua các thao tác đó. Người thiết kế áp dụng trừu tượng dữ liệu để xác định thuộc tính và các thao tác, xâm nhập thuộc tính thông qua thao tác.

Theo Wasserman, *"Ký pháp trừu tượng mang tính tâm lý cho phép ta tập trung vào một vấn đề ở một mức nào đó của sự khái quát, bỏ qua các chi tiết ở mức thấp ít liên quan. Việc sử dụng sự trừu tượng cũng cho phép ta làm việc với các khái niệm và thuật ngữ gần gũi trong môi trường của vấn đề đặt ra mà không phải chuyển chúng thành một cấu trúc không quen thuộc" [11].*

Nếu các mặt, các yếu tố của đối tượng được mô tả bị bỏ qua càng nhiều thì mức trừu tượng hóa càng cao. Như vậy ta có thể mô tả đối tượng thiết kế với nhiều mức trừu tượng khác nhau tùy thuộc vào sự hiểu biết, nhận thức của người phát triển và yêu cầu đặt ra đối với nó.

Có nhiều mức trừu tượng:

- *Mức cao nhất*: một giải pháp được phát biểu theo *thuật ngữ đại thể* bằng cách dùng ngôn ngữ của môi trường vấn đề.

- *Mức vừa*: lấy khuynh hướng thủ tục nhiều hơn. *Thuật ngữ hướng vấn đề* thường đi đôi với thuật ngữ hướng cài đặt trong mô tả giải pháp.

- *Mức thấp*: giải pháp được phát biểu theo *thuật ngữ chi tiết* để có thể được cài đặt trực tiếp.

Mỗi bước trong tiến trình kỹ nghệ phần mềm đều là sự làm mịn cho một mức trừu tượng của phần mềm. Trong kỹ nghệ hệ thống, phần mềm được dùng như một phần tử của hệ thống dựa trên máy tính. Trong phân tích các yêu cầu phần mềm, giải pháp phần mềm

được phát biểu dưới dạng "đó là cái quan trọng trong môi trường vấn đề". Khi chúng ta chuyển từ thiết kế sơ bộ sang thiết kế chi tiết thì mức độ trừu tượng giảm dần. Quá trình này dẫn tới mức trừu tượng thấp nhất khi sinh ra chương trình gốc.

Trừu tượng hóa là một khả năng cơ bản của con người trong việc giải quyết các vấn đề phức tạp. Nó là một cơ chế được dùng để biểu diễn một sự vật phức tạp để trở nên đơn giản hơn bằng cách dùng một số loại mô hình. Nếu sự trừu tượng được biểu diễn ở mức vật lý, chẳng hạn như một sơ đồ trên giấy hoặc một đối tượng vật lý, người ta thường dùng thuật ngữ *mô hình*.

Trong việc phân tích thiết kế hướng đối tượng, người ta sử dụng sơ đồ để đơn giản hóa hệ thống và để biểu diễn các đặc điểm chính nào đó, nghĩa là để thực hiện sự trừu tượng. Khi tạo ra một thiết kế, việc dùng sơ đồ có lợi ích chính là để trừu tượng hóa các thuộc tính của bản thiết kế. Tất nhiên, người ta phải dùng nhiều sơ đồ mới có thể thể hiện hết được các phương diện khác nhau của một đối tượng phức tạp.

Trừu tượng hóa các đặc điểm thích hợp và xây dựng mô hình chính xác là kỹ năng của người phân tích [20].

1.1.2. Khái niệm mô hình và mô hình hóa

1. Định nghĩa và ý nghĩa của mô hình

Mô hình (model) là một dạng trừu tượng hóa của một hệ thống thực. Mô hình chính là một hình ảnh (một biểu diễn) của một hệ thống thực, được diễn tả ở một mức độ trừu tượng nào đó, theo một quan điểm nào đó, theo một hình thức (hiểu được) nào đó như phương trình, bảng, đồ thị,... Mô hình có xu hướng dạng *sơ đồ* (*diagrams*) tức là đồ thị gồm các nút và cung [21].

Mô hình mô tả bản chất của một vấn đề hoặc một cấu trúc phức tạp bằng cách loại bỏ những chi tiết không quan trọng, làm cho bài toán trở nên dễ hiểu và dễ nắm bắt hơn. Để xây dựng một hệ thống

phức tạp, những người phát triển phải trừu tượng hóa những khía cạnh khác nhau của hệ thống, xây dựng các mô hình bằng cách sử dụng các ký hiệu một cách rõ ràng, cẩn thận, kiểm tra xem các mô hình đã thỏa mãn các yêu cầu của hệ thống chưa và dần dần thêm vào các chi tiết để có thể chuyển đổi từ mô hình sang một cài đặt cụ thể [18].

Trong phân tích thiết kế hệ thống, các mô hình được tạo ra để trừu tượng hóa các đặc điểm quan trọng của các hệ thống thế giới thực. Trong lĩnh vực phần mềm, mô hình là kế hoạch chi tiết của hệ thống, là bức tranh hay mô tả của vấn đề đang được cố gắng giải quyết hay biểu diễn. Mô hình còn có thể là mô tả chính giải pháp, có thể dùng biểu tượng thay cho đối tượng thực. Tiến trình phát triển phần mềm là làm giảm một số đặc trưng của đối tượng để hình thành mô hình, làm giảm độ phức tạp bằng mô hình trừu tượng.

Mọi mô hình đều phản ánh hệ thống theo một khung nhìn trừu tượng hóa nào đó. Có 2 khung nhìn chính:

+ *Khung nhìn logic*: tập trung mô tả bản chất của hệ thống và mục đích hoạt động của hệ thống, bỏ qua các yếu tố về tổ chức thực hiện, về biện pháp cài đặt. Nói cách khác, mô hình logic trả lời các câu hỏi "là gì?" (What?)- như là chức năng gì, thông tin gì, ứng xử gì, bỏ qua các câu hỏi "như thế nào?" (How?). Ở tầng này, người ta tiến hành trên 3 phương diện xử lý, dữ liệu và động thái hệ thống.

+ *Khung nhìn vật lý:* Trả lời câu hỏi "như thế nào", "ai làm", "làm ở đâu", "khi nào làm", quan tâm đến các mặt như: phương pháp, biện pháp, công cụ, tác nhân, địa điểm, thời gian, hiệu năng,... Ở tầng này yêu cầu cần làm rõ kiến trúc vật lý của hệ thống.

Việc xây dựng mô hình có một ý nghĩa rất lớn trong quá trình phát triển phần mềm:

- Mô hình giúp ta hiểu và thực hiện được sự trừu tượng, tổng quát hóa các khái niệm cơ sở để giảm thiểu độ phức tạp của hệ thống.

Qua mô hình chúng ta biết được hệ thống gồm những gì? và chúng hoạt động như thế nào.

- Mô hình giúp chúng ta quan sát được hệ thống như nó vốn có trong thực tế hoặc nó phải có như ta mong muốn. Muốn hiểu và phát triển được hệ thống phần mềm theo yêu cầu thực tế thì ta phải quan sát nó theo nhiều góc nhìn khác nhau: theo chức năng sử dụng, theo các thành phần logic.

- Mô hình thể hiện rõ các đặc tả cấu trúc và hành vi của hệ thống.

- Đồng thời, mô hình là cơ sở để trao đổi, ghi lại những quyết định đã thực hiện trong nhóm tham gia dự án phát triển phần mềm. Mọi quan sát, mọi kết quả phân tích đều được ghi lại chi tiết để phục vụ cho cả quá trình phát triển phần mềm.

2. Khái niệm về mô hình hóa

Việc dùng mô hình để nhận thức và diễn tả một hệ thống được gọi là mô hình hóa. Trong khoa học máy tính, mô hình hóa bắt đầu từ việc mô tả vấn đề, sau đó là mô tả giải pháp vấn đề. Các hoạt động này còn được gọi là *phân tích và thiết kế*. Khi khảo sát hệ thống, người ta sưu tập yêu cầu cho hệ thống, ánh xạ chúng thành yêu cầu phần mềm, từ đó phát sinh mã trình, đảm bảo yêu cầu phù hợp với mã trình và khả năng chuyển đổi mã trình ngược lại thành yêu cầu. Tiến trình đó chính là thực hiện mô hình hóa.

Người ta có thể lấy thông tin từ mô hình và hiển thị đồ họa bằng tập các phần tử đồ họa chuẩn. Tiến trình đó được gọi là mô hình hóa trực quan. Tiêu chuẩn là cốt lõi để thực hiện một trong các lợi thế của mô hình trực quan, đó là vấn đề giao tiếp. Giao tiếp giữa tất cả các người tham gia dự án là mục tiêu quan trọng nhất của mô hình hóa trực quan. Tương tác này có thể được thực hiện bằng văn bản nhưng tốt hơn là bằng đồ họa. Các nhà tin học đã rất cố gắng để hình thành ký pháp mô hình hóa trực quan. Một số ký pháp quen thuộc là của Booch, OMT và UML. Phần mềm công cụ Rational Rose 2000 đã trợ giúp tốt vấn đề này.

3. Mục đích của mô hình hóa

Mục đích của mô hình hóa là để hiểu, để làm phương tiện trao đổi, để hoàn chỉnh hệ thống, hay nói rõ hơn*:*

a) *Mô hình hóa giúp ta hiểu và thực hiện được sự trừu tượng, tổng quát hóa các khái niệm cơ sở để giảm thiểu độ phức tạp của hệ thống.* Qua mô hình chúng ta biết được hệ thống gồm những gì? và chúng hoạt động như thế nào?.

Michael B., William P. [17] từng nói: "*Hiểu tức là mô hình hóa*". Do vậy, quá trình phát triển phần mềm chẳng qua là quá trình nhận thức và mô tả lại hệ thống đó. Đó cũng là quá trình thiết lập, sử dụng và biến đổi các mô hình. Vậy, có một mô hình đúng sẽ giúp ta làm sáng tỏ những vấn đề phức tạp và cho ta cái nhìn thấu đáo về vấn đề cần giải quyết.

b) *Mô hình hóa giúp chúng ta quan sát được hệ thống như nó vốn có trong thực tế hoặc nó phải có như ta mong muốn.* Muốn hiểu và phát triển được hệ thống phần mềm theo yêu cầu thực tế thì ta phải quan sát nó theo nhiều góc nhìn khác nhau: theo chức năng sử dụng, theo các thành phần logic, theo phương diện triển khai,...

c) *Mô hình hóa cho phép chúng ta đặc tả được cấu trúc và hành vi của hệ thống:*

+ *Đảm bảo hệ thống đạt được mục đích đã xác định trước.* Mọi mô hình đều đơn giản hóa thế giới thực, nhưng phải đảm bảo sự đơn giản đó không loại bỏ đi những yếu tố quan trọng.

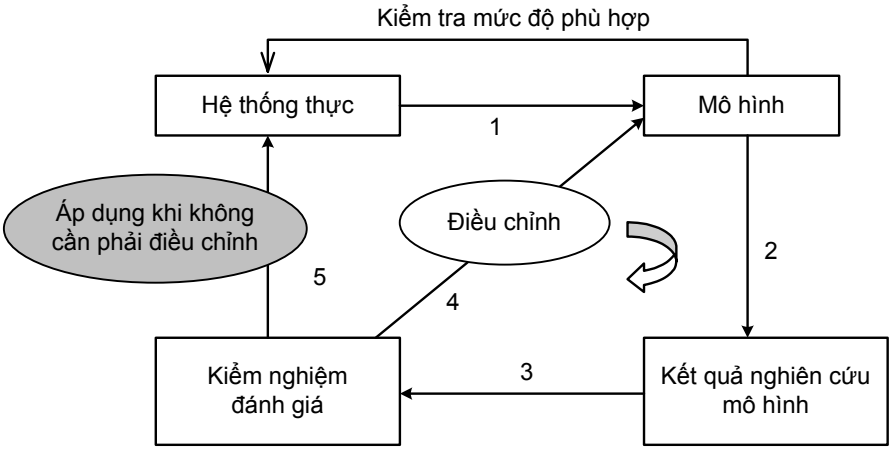
+ Kiểm tra được các quy định về cú pháp, ngữ nghĩa về tính chặt chẽ và đầy đủ của mô hình, khẳng định được tính đúng đắn của thiết kế, phù hợp với yêu cầu của khách hàng. Nghĩa là, mô hình hóa là quá trình hoàn thiện và tiến hóa liên tục.

d) *Mô hình hóa là nhằm tạo ra khuôn mẫu (template) và hướng dẫn cách xây dựng hệ thống; cho phép thử nghiệm, mô phỏng và thực hiện, hoàn thiện theo mô hình.*

1.1.3. Phương pháp mô hình hóa

Phương pháp là cách trực tiếp cấu trúc hóa sự suy nghĩ và hành động của con người. Phương pháp cho người sử dụng biết phải làm gì? làm như thế nào? khi nào? và tại sao? (mục đích của hành động). Phương pháp chứa các *mô hình (model)*, các mô hình được dùng để mô tả những gì sử dụng cho việc truyền đạt kết quả trong quá trình sử dụng phương pháp [2].

Phương pháp mô hình hóa là một trong những phương pháp quan trọng nhất để nghiên cứu hệ thống. Ý tưởng của phương pháp mô hình hóa là không nghiên cứu trực tiếp đối tượng mà thông qua việc nghiên cứu một đối tượng khác "tương tự" hay là "hình ảnh" của nó mà có thể sử dụng được các công cụ khoa học. Kết quả nghiên cứu trên mô hình được áp dụng vào cho đối tượng thực tế.



Hình 1.1. Sơ đồ nguyên tắc hoạt động của phương pháp mô hình hóa

Việc mô hình hóa thể hiện một tiến độ triển khai, bao gồm các bước đi lần lượt, các hoạt động cần làm. Mô hình hóa giữ một vai trò đặc biệt quan trọng khi nó trở thành một công cụ trợ giúp. Đó là cơ sở tạo phần mềm giúp cho việc triển khai hệ thống thực hiện đúng và nhanh.

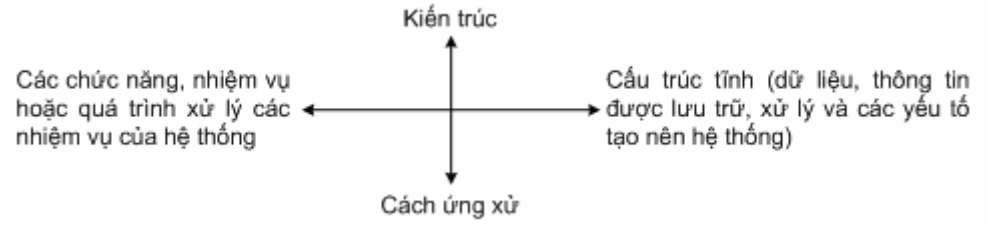
Như vậy, mô hình hóa là biểu diễn hệ thống dưới các dạng hình thức dễ hiểu như sơ đồ, đồ thị, công thức... Mô hình hóa giúp hiểu rõ bài toán, trao đổi thông tin giữa những người liên quan như khách hàng, chuyên gia, người phân tích, người thiết kế. Mô hình giúp cho việc xác định các yêu cầu tốt hơn, thiết kế rõ ràng hơn và khả năng bảo trì hệ thống cao hơn.

Mô hình hóa là phần trung tâm trong các công việc, các hoạt động để dẫn tới một phần mềm tốt. Chúng ta xây dựng mô hình để trao đổi, bàn bạc về cấu trúc và hành vi mong muốn của hệ thống. Đồng thời xây dựng mô hình để trực quan hóa và kiểm soát kiến trúc của hệ thống.

Mô hình hóa có thể mô tả các cấu trúc, nhấn mạnh về mặt tổ chức của hệ thống hoặc nó có thể mô tả các hành vi, tập trung vào mặt động của hệ thống.

Tóm lại, phương pháp mô hình hóa là phương pháp tiên tiến. Nó giúp chúng ta hiểu rõ hơn về hệ thống mà chúng ta đang xây dựng, tạo ra cơ hội để có thể đơn giản hóa và tái sử dụng. Ngoài ra phương pháp mô hình hóa còn giúp chúng ta dễ dàng kiểm soát rủi ro.

Theo Booch G., Rumbaugh J. and Jacobson I., mô hình hóa một hệ thống phải thực hiện theo cả bốn hướng [2]:



Hình 1.2. Các hướng mô hình hóa

Hướng của điểm xuất phát sẽ kéo theo phương pháp cần lựa chọn để phát triển phần mềm. Nếu ta bắt đầu từ bên trái, nghĩa là tập trung vào chức năng để phân tích thì chúng ta thực hiện, phát triển phần mềm theo cách tiếp cận hướng chức năng. Ngược lại, nếu

bắt đầu từ bên phải, nghĩa là dựa vào dữ liệu là chính thì chúng ta sử dụng phương pháp hướng đối tượng.

1.1.4. Ngôn ngữ mô hình hóa

Mô hình được biểu diễn theo một ngôn ngữ mô hình hóa. Ngôn ngữ mô hình hóa bao gồm các ký hiệu (những biểu tượng được dùng trong mô hình) và một tập các quy tắc chỉ cách sử dụng chúng. Các quy tắc này bao gồm:

- *Cú pháp (Syntactic):* cho biết hình dạng các biểu tượng và cách kết hợp chúng trong ngôn ngữ.

- *Ngữ nghĩa (Semantic):* cho biết ý nghĩa của mỗi biểu tượng, chúng được hiểu thế nào khi nằm trong hoặc không nằm trong ngữ cảnh của các biểu tượng khác.

- *Mục đích (Pragmatic):* định nghĩa ý nghĩa của biểu tượng để sao cho mục đích của mô hình được thể hiện và mọi người có thể hiểu được.

1.1.5. Nguyên tắc mô hình hóa

Thông qua mô hình hóa, chúng ta sẽ giới hạn vấn đề nghiên cứu bằng cách chỉ tập trung vào một khía cạnh của vấn đề vào một thời điểm. Mô hình hóa sẽ là tăng độ dễ hiểu của con người. Việc chọn mô hình đúng cho khả năng mô hình làm việc ở mức trừu tượng cao. Booch G., Rumbaugh J., Jacobson I., 1999, đã đưa ra 4 nguyên tắc cơ bản về mô hình hóa:

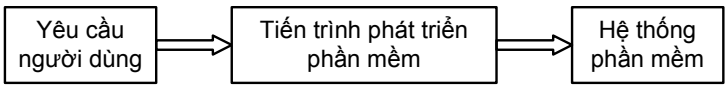
1. Việc chọn mô hình nào để tạo lập có ảnh hưởng sâu sắc đến cách giải quyết vấn đề và cách hình thành các giải pháp.
2. Mỗi mô hình biểu diễn hệ thống với mức độ chính xác khác nhau.
3. Mô hình tốt nhất phải là mô hình phù hợp với thế giới thực.
4. Không mô hình nào là đầy đủ. Mỗi hệ thống thường được tiếp cận thông qua tập mô hình gần như độc lập nhau.

Phụ thuộc vào bản chất của hệ thống mà mỗi mô hình có tầm quan trọng khác nhau. Mô hình quan sát thiết kế tĩnh sẽ quan trọng hơn trong hệ thống quản lý nhiều dữ liệu; trong khi đó mô hình ca sử dụng rất quan trọng đối với hệ thống có nhiều giao diện; còn mô hình quan sát tiến trình động rất quan trọng đối với hệ thống thời gian thực; đặc biệt mô hình triển khai và cài đặt rất quan trọng với hệ thống phân tán có ứng dụng web...

1.2. MÔ HÌNH HÓA TIẾN TRÌNH PHÁT TRIỂN PHẦN MỀM

1.2.1. Tiến trình phát triển phần mềm

Một tiến trình phát triển phần mềm là một tập của các hoạt động cần thiết để chuyển các yêu cầu người dùng thành một hệ thống phần mềm đáp ứng được các yêu cầu đặt ra [26].



Hình 1.3. Quá trình phát triển phần mềm

Vòng đời phát triển phần mềm được chia thành 4 pha: sơ bộ, soạn thảo, xây dựng và chuyển giao. Trong mỗi pha lại chia thành nhiều bước lặp nhỏ. Mỗi bước lặp đều gồm một số công việc thực hiện trọn vẹn một sản phẩm phần mềm: *lập các mô hình đặc tả nghiệp vụ, xác định yêu cầu, lập các mô hình phân tích dữ liệu, xử lý và hành vi, lập các mô hình thiết kế, triển khai và kiểm thử.*

Mô hình tiến trình phần mềm là sự mô tả tiến trình một cách đơn giản hóa khi xem xét nó từ một cách nhìn cụ thể [27].

Về bản chất, mô hình tiến trình là một sự trừu tượng hóa một lớp các tiến trình thực. Một vài mô hình tiến trình phần mềm, đã được trình bày ở trên, được nhiều người biết đến như *mô hình thác nước, mô hình xoắn ốc, mô hình làm bản mẫu.*

Năm 1996, Huff lần đầu tiên hệ thống hóa một số mô hình tiến trình phần mềm. Sommerville, 2001, nhắc tới một số loại mô hình tiêu biểu:

- *Mô hình thác nước (waterfall model)*

- *Mô hình phát triển tiến hóa (evolutionary models)*: là mô hình trình phát triển với quá trình lặp để xây dựng dần phần phềm. Mô hình loại này bao gồm mô hình làm bản mẫu, mô hình xoắn ốc, mô hình tiến trình hợp nhất Rational-RUP, mô hình phát triển tăng dần, phát triển ứng dụng nhanh-RAD (Rapid Application Development).

- *Phát triển hệ thống hình thức (formal system development)*: Một cách tiếp cận dựa trên đặc tả hệ thống bằng toán học để chứng minh hay chuyển đổi nó thành chương trình nhờ các công cụ toán học chuyên dụng.

- *Phát triển phần mềm theo hướng sử dụng lại (reuse oriented software development)*: Quá trình phát triển tập trung vào việc tích hợp các thành phần đã có để nhận được hệ thống, đáp ứng được các yêu cầu đặt ra.

1.2.2. Ngôn ngữ mô hình hóa hợp nhất (UML)

UML (Unified Modeling Language) là ngôn ngữ trực quan được dùng trong quy trình phát triển các hệ thống phần mềm. Nó là một *ngôn ngữ đặc tả hình thức (formal specification language*). UML có một tập các phần tử và một tập các quy tắc riêng. Hầu hết các phần tử của UML là các đối tượng đồ họa như đường thẳng, hình chữ nhật, hình oval,... và thường có nhãn kèm theo để tăng thông tin. Các quy tắc trong UML được mô tả trong đặc tả UML như cú pháp trừu tượng (được biểu diễn như các sơ đồ và ngôn ngữ tự nhiên), quy tắc hình thức (nằm trong ngôn ngữ ràng buộc đối tượng) và ngữ nghĩa. Quy tắc xác định cách kết hợp giữa các phần tử [20].

Do UML là ngôn ngữ mô hình hóa chuẩn, ngôn ngữ mô hình đồ họa, trực quan, vừa đặc tả vừa có cấu trúc, đồng thời lại là ngôn ngữ làm tài liệu nên đối với việc phát triển phần mềm hướng đối tượng, UML đặc biệt có khả năng sau:

- Cho phép mô tả toàn bộ các sản phẩm phân tích và thiết kế;

- Trợ giúp việc tự động hóa quá trình thiết kế trên máy tính;

- Trợ giúp việc dịch xuôi và dịch ngược các thiết kế sang mã nguồn của ngôn ngữ lập trình và CSDL.

UML được hợp nhất từ nhiều thành tựu và kinh nghiệm nghiên cứu và triển khai nhờ:

- Cách tiếp cận của Grady Booch (Booch Approach),

- Kỹ thuật mô hình hóa đối tượng (OMT: Object Modeling Technique) của James Rumbaugh,

- Công nghệ phần mềm hướng đối tượng (OOSE: Object-Oriented Software Engineering) của Ivar Jacobson,

- Thống nhất được nhiều ký pháp, khái niệm của nhiều phương pháp khác nhau. Quá trình hình thành UML bắt đầu từ ngôn ngữ Ada (Booch) trước năm 1990.

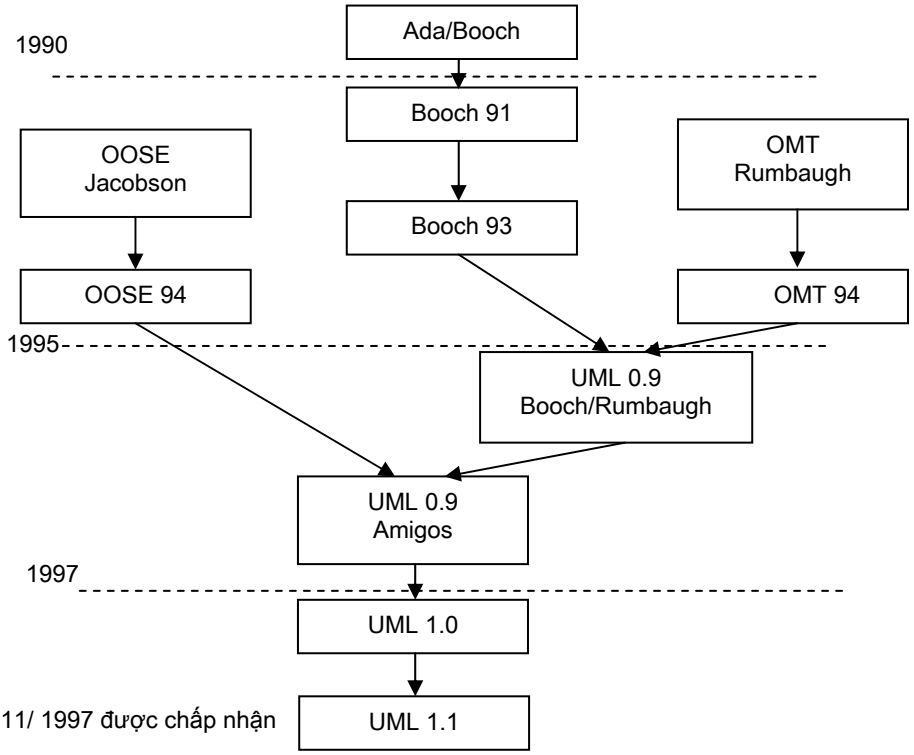
Mục đích chính của UML nhằm vào các hoạt động sau:

- Mô hình hóa được các hệ thống và sử dụng được tất cả các khái niệm hướng đối tượng một cách thống nhất.

- Cho phép đặc tả, hỗ trợ để đặc tả tường minh mối quan hệ giữa các khái niệm cơ bản trong hệ thống, đồng thời mô tả được mọi trạng thái hoạt động của hệ thống đối tượng. Nghĩa là cho phép mô tả được cả mô hình tĩnh lẫn mô hình động một cách đầy đủ và trực quan.

- Tận dụng được những khả năng sử dụng lại và kế thừa ở phạm vi diện rộng để xây dựng được những hệ thống phức tạp và nhạy cảm như: các hệ thống động, hệ thống thời gian thực, hệ thống nhúng thời gian thực...

- Tạo ra những ngôn ngữ mô hình hóa sử dụng được cho cả người lẫn máy tính.



Hình 1.4. Sự phát triển của UML

1.2.3. Quy trình phát triển phần mềm hợp nhất (USDP)

UML được phát triển để đặc tả trong quá trình phát triển phần mềm, nhằm mô hình hóa hệ thống. Quy trình phát triển phần mềm có sử dụng UML được gọi là quy trình phát triển phần mềm hợp nhất được viết tắt là USDP (Unified Software Development Proccess).

Các đặc trưng của quy trình phát triển phần mềm hợp nhất bao gồm:

- Quy trình phát triển phần mềm hợp nhất bao gồm con người, dự án, sản phẩm, quy trình và công cụ. Con người là những người

tham gia dự án để tạo ra sản phẩm phần mềm theo một quy trình với sự hỗ trợ của công cụ được cung cấp.

- Quy trình phát triển phần mềm hợp nhất là quy trình phát triển phần mềm được hướng dẫn bởi các ca sử dụng. Nghĩa là các yêu cầu của người sử dụng được mô tả trong các ca sử dụng, là chuỗi các hành động được thực hiện bởi hệ thống nhằm cung cấp các dịch vụ, các thông tin cho khách hàng. Các ca sử dụng bao gồm chuỗi các công việc được xem là nền tảng để tạo ra mô hình thiết kế và cài đặt hệ thống.

- Quy trình phát triển phần mềm hợp nhất cũng là quy trình tập trung vào kiến trúc, được lặp và phát triển tăng trưởng liên tục.

- Quy trình phát triển phần mềm hợp nhất không chỉ tạo ra một hệ thống phần mềm hoàn chỉnh mà còn tạo ra một số sản phẩm trung gian như các mô hình ca sử dụng, mô hình khái niệm, mô hình phân tích, mô hình thiết kế, mô hình triển khai, mô hình cài đặt và mô hình kiểm thử.

Quy trình phát triển phần mềm hợp nhất và ngôn ngữ mô hình hóa hợp nhất là phương pháp luận và công cụ điển hình cho công nghệ phát triển phần mềm hướng đối tượng [23]./.

2

CÁC KHÁI NIỆM CƠ BẢN TRONG PHÂN TÍCH VÀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

2.1. CÁCH TIẾP CẬN HƯỚNG ĐỐI TƯỢNG

Để khắc phục những vấn đề tồn tại trong cách tiếp cận hướng cấu trúc người ta đã nghiên cứu một mô hình mới, thích hợp cho việc phát triển phần mềm lớn và phức tạp, đó là mô hình hướng đối tượng. Quan điểm hướng đối tượng dựa trên cách tiếp cận hệ thống, các chức năng hệ thống được biểu diễn thông qua cộng tác của các thành phần.

Theo cách tiếp cận hướng đối tượng, hệ thống được nhìn nhận như một bộ các **đối tượng** (chứ không phải là một bộ chức năng). Hệ thống được phân tán, mỗi đối tượng có những thông tin trạng thái riêng của nó. Theo từ điển tiếng Việt của Viện Ngôn ngữ học Hà Nội, 1996, thì đối tượng (object) theo nghĩa thông thường là người, vật hay hiện tượng mà con người nhằm vào trong suy nghĩ, trong hành động, là bất kỳ cái gì nhìn thấy và sờ mó được. Nhưng ở đây, theo Cood P. và Yourdon E. , 1991, đối tượng là trừu tượng cái gì đó trong lĩnh vực vấn đề hay trong cài đặt của nó, phản ánh khả năng hệ thống lưu giữ thuộc tính về nó và tương tác với nó, gói các giá trị thuộc tính và các dịch vụ (phương thức hay phương pháp).

Nói rõ hơn, *đối tượng là bộ các "thuộc tính" xác định trạng thái của đối tượng đó và các phép toán thực hiện trên các thuộc tính đó.*

Mỗi đối tượng là một thể hiện cụ thể của một lớp mà lớp được xác định bởi các thuộc tính và các phép toán của nó. Lớp đối tượng được thừa kế từ một vài lớp đối tượng có mức trừu tượng cao hơn, sao cho định nghĩa nó chỉ cần nêu đủ sự khác biệt giữa nó và các lớp cao hơn nó. Các đối tượng liên lạc với nhau chỉ bằng cách trao đổi các thông báo: thực tế hầu hết các liên lạc giữa các đối tượng thực hiện bằng cách một đối tượng này gọi một thủ tục, mà thủ tục này kết hợp với một đối tượng khác.

Cách tiếp cận hướng đối tượng dựa trên ý tưởng che dấu thông tin. Thiết kế hướng đối tượng gần đây được phát triển nhiều đã tạo ra các hệ thống cấu tạo bởi nhiều thành phần độc lập và có tương tác với nhau. Che dấu thông tin là chiến lược thiết kế dấu càng nhiều thông tin trong các thành phần càng hay. Cái đó ngầm hiểu rằng việc kết hợp điều khiển logic và cấu trúc dữ liệu được thực hiện trong thiết kế càng chậm càng tốt. Liên lạc thông qua các thông tin trạng thái dùng chung (các biến tổng thể) là ít nhất, nhờ vậy khả năng hiểu được nâng lên. Thiết kế là tương đối dễ thay đổi vì sự thay đổi một thành phần không thể không dự kiến các hiệu ứng phụ trên các thành phần khác.

2.1.1. Các đặc trưng của cách tiếp cận hướng đối tượng

Cách tiếp cận hướng đối tượng có 3 đặc trưng sau:

- Không có vùng dữ liệu dùng chung. Các đối tượng liên lạc với nhau bằng cách trao đổi thông báo chứ không phải bằng các biến dùng chung.

- Các đối tượng là các thực thể độc lập, dễ thay đổi vì rằng tất cả các trạng thái và các thông tin biểu diễn chỉ ảnh hưởng trong phạm vi chính đối tượng đó thôi. Các thay đổi về biểu diễn thông tin có thể được thực hiện không cần sự tham khảo tới các đối tượng hệ thống khác.

- Các đối tượng có thể phân tán và có thể hành động tuần tự hoặc song song.

2.1.2. Các ưu khuyết điểm của thiết kế hướng đối tượng

Thiết kế hướng đối tượng đang trên đà phát triển mạnh bởi nó có được một số ưu điểm sau:

- Dễ bảo trì vì các đối tượng là độc lập. Các đối tượng có thể hiểu và cải biên như là một thực thể độc lập. Thay đổi trong thực hiện một đối tượng hoặc thêm các dịch vụ sẽ không làm ảnh hưởng tới các đối tượng hệ thống khác.

- Các đối tượng là các thành phần dùng lại được thích hợp (do tính độc lập của chúng). Một thiết kế có thể dùng lại được các đối tượng đã được thiết kế trong các bản thiết kế trước đó.

- Có một vài lớp hệ thống thể hiện phản ánh quan hệ rõ ràng giữa các thực thể có thực (chẳng hạn như các thành phần phần cứng) với các đối tượng điều khiển nó trong hệ thống. Điều này đạt được tính dễ hiểu của thiết kế.

Tuy vậy, thiết kế hướng đối tượng vẫn còng một số hạn chế sau:

- Sự tường minh các đối tượng hệ thống thích hợp là khó khăn. Cách nhìn tự nhiên nhiều hệ thống là cách nhìn chức năng và việc thích nghi với cách nhìn hướng đối tượng đôi khi là khó khăn, nhất là đối tượng trừu tượng, không phải đối tượng cụ thể.

- Phương pháp thiết kế hướng đối tượng đang trong quá trình hoàn thiện, còn nhiều tranh luận, quy trình phát triển chưa thật hoàn chỉnh như phương pháp hướng cấu trúc và thị trường ứng dụng còn rất hẹp so với hướng cấu trúc.

Sự thật, các hệ phần mềm lớn là phức tạp đến mức mà người ta đã dùng các phương pháp tiếp cận khác nhau trong việc thiết kế các thành phần khác nhau của một hệ thống. Chẳng có một chiến lược tốt nhất nào cho các dự án lớn. Các cách tiếp cận hướng chức năng và hướng đối tượng là bổ sung hỗ trợ cho nhau chứ không đối kháng nhau. Kỹ sư phần mềm sẽ chọn cách tiếp cận thích hợp nhất cho từng giai đoạn thiết kế. Nhìn ở mức tổng thể thì hệ thống như một bộ

các đối tượng (chứ không phải là bộ các chức năng), cho nên ở mức trừu tượng thì cách tiếp cận hướng đối tượng là thích hợp hơn. Đến mức chi tiết thì tự nhiên hơn là nên xem chúng là các chức năng tương tác giữa các đối tượng. Sau đó mỗi đối tượng lại được phân giải thành các thành phần, tức là lại có thể xem nó như là một hệ (con).

2.2. UML VÀ CÁC GIAI ĐOẠN PHÁT TRIỂN PHẦN MỀM

Ngôn ngữ mô hình hợp nhất UML là một ngôn ngữ trực quan giúp các nhà phân tích thiết kế hệ thống theo hướng đối tượng một cách hình dung được các hệ thống phần mềm, mô hình hóa được các hoạt động nghiệp vụ thể hiện trong đó. UML đang tiến triển như một chuẩn, chuẩn quốc tế, được *tổ chức tiêu chuẩn quốc tế ISO (International Standard Organization)* chấp nhận. Dựa vào UML, quá trình phát triển phần mềm được chia thành nhiều giai đoạn như dưới đây.

2.2.1. Giai đoạn nghiên cứu sơ bộ

UML đưa ra khái niệm Ca sử dụng (Use Case) để nắm bắt các yêu cầu của người sử dụng. UML sử dụng sơ đồ ca sử dụng để nêu bật mối quan hệ cũng như sự giao tiếp với hệ thống.

Qua phương pháp mô hình hóa ca sử dụng, các tác nhân (Actor) bên ngoài quan tâm đến hệ thống sẽ được mô hình hóa song song với chức năng mà họ đòi hỏi từ phía hệ thống. Các tác nhân và các ca sử dụng được mô hình hóa cùng các mối quan hệ và được miêu tả trong sơ đồ ca sử dụng của UML. Mỗi ca sử dụng sẽ đặc tả các yêu cầu của người dùng.

2.2.2. Giai đoạn phân tích

Giai đoạn phân tích quan tâm đến quá trình trừu tượng hóa đầu tiên (các lớp và các đối tượng) cũng như cơ chế hiện hữu trong phạm vi vấn đề. Sau khi nhà phân tích đã nhận biết được các lớp thành phần của mô hình cũng như mối quan hệ giữa chúng với nhau, các

lớp cùng các mối quan hệ đó sẽ được miêu tả bằng công cụ sơ đồ lớp của UML. Sự cộng tác giữa các lớp nhằm thực hiện các ca sử dụng cũng sẽ được miêu tả nhờ vào các mô hình động (dynamic models) của UML. Trong giai đoạn phân tích, chỉ duy nhất các lớp có tồn tại trong phạm vi các khái niệm đời thực là được mô hình hóa. Các lớp kỹ thuật định nghĩa chi tiết cũng như giải pháp trong hệ thống phần mềm, ví dụ như các lớp cho giao diện người dùng, cho ngân hàng dữ liệu, cho sự giao tiếp, trùng hợp..., chưa phải là mối quan tâm của giai đoạn này.

2.2.3. Giai đoạn thiết kế

Trong giai đoạn này, kết quả của giai đoạn phân tích sẽ được mở rộng thành một giải pháp kỹ thuật. Các lớp mới sẽ được bổ sung để tạo thành một hạ tầng cơ sở kỹ thuật: giao diện người dùng, các chức năng để lưu trữ các đối tượng trong ngân hàng dữ liệu, giao tiếp với các hệ thống khác, giao diện với các thiết bị ngoại vi và các máy móc khác trong hệ thống. Các lớp thuộc phạm vi vấn đề có từ giai đoạn phân tích sẽ được "nhúng" vào hạ tầng cơ sở kỹ thuật này, tạo ra khả năng thay đổi trong cả hai phương diện: phạm vi vấn đề và hạ tầng cơ sở. Giai đoạn thiết kế sẽ đưa ra kết quả là bản đặc tả chi tiết cho giai đoạn xây dựng hệ thống.

2.2.4. Giai đoạn lập trình

Trong giai đoạn lập trình, các lớp của giai đoạn thiết kế sẽ được mã hóa trong một ngôn ngữ lập trình hướng đối tượng cụ thể. Phụ thuộc vào khả năng của ngôn ngữ được sử dụng, đây có thể là một công việc khó khăn hay dễ dàng. Khi tạo ra các mô hình phân tích và thiết kế trong UML, tốt nhất nên cố gắng né tránh việc ngay lập tức biến đổi các mô hình này thành các dòng code. Trong những giai đoạn trước, mô hình được sử dụng để dễ hiểu, dễ giao tiếp và tạo nên cấu trúc của hệ thống. Vì vậy, vội vàng đưa ra những kết luận về việc lập trình có thể sẽ thành một trở ngại cho việc tạo ra các mô hình

chính xác và đơn giản. Giai đoạn xây dựng là một giai đoạn riêng biệt, nơi các mô hình được mã hóa.

2.2.5. Giai đoạn kiểm thử

Một hệ thống phần mềm thường được kiểm thử qua nhiều giai đoạn và với nhiều nhóm kiểm thử khác nhau. Các nhóm sử dụng nhiều loại sơ đồ UML khác nhau làm nền tảng cho công việc của mình: kiểm thử đơn vị sử dụng sơ đồ lớp và đặc tả lớp, kiểm thử tích hợp thường sử dụng sơ đồ thành phần và sơ đồ cộng tác, và giai đoạn kiểm thử hệ thống sử dụng sơ đồ ca sử dụng để đảm bảo hệ thống có đáp ứng được đúng các chức năng như đã yêu cầu hay không.

2.3. ĐẶC TRƯNG TIẾN TRÌNH PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG BẰNG UML

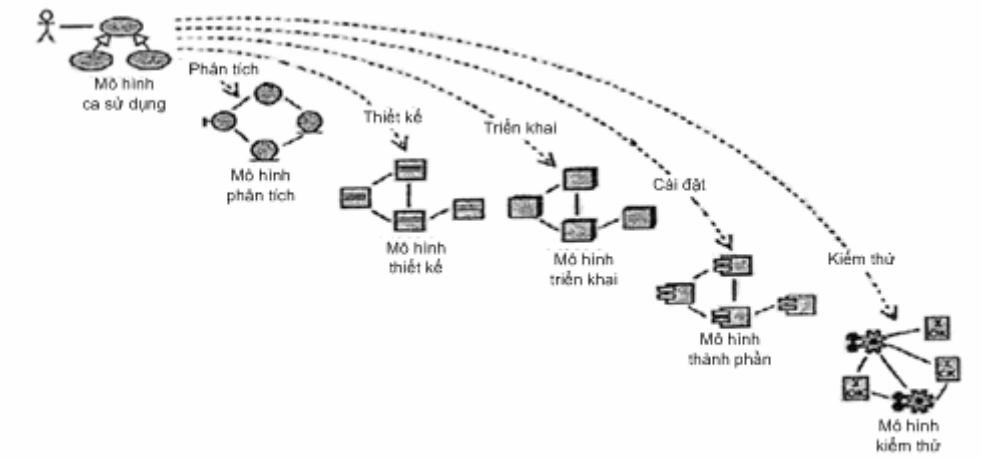
Quá trình phát triển phần mềm hướng đối tượng có thể sử dụng các công cụ khác nhau. Quá trình phát triển phần mềm có 3 đặc trưng cơ bản sau:

- Ca sử dụng điều khiển quá trình phát triển
- Lấy kiến trúc làm trung tâm
- Tiến trình phát triển là tiến trình lặp và tăng dần

2.3.1. Ca sử dụng điều khiển toàn bộ quá trình phát triển

Một hệ thống phần mềm được tạo ra là để phục vụ người dùng. Người dùng ở đây bao gồm cả người dùng hệ thống hay các hệ thống ngoài khác tương tác với hệ thống.

Một ca sử dụng là một phần chức năng của hệ thống cung cấp cho người dùng để đem lại một kết quả nào đó khi sử dụng nó. Các ca sử dụng dùng để nắm bắt các yêu cầu chức năng. Tập hợp tất cả các ca sử dụng lập thành mô hình ca sử dụng mô tả chức năng đầy đủ của hệ thống. Một đặc tả chức năng thường trả lời câu hỏi: hệ thống được dự kiến sẽ làm gì? Nhưng đối với ca sử dụng thì câu hỏi lại là: hệ thống được dự kiến sẽ làm được gì cho mỗi người sử dụng?



Hình 2.1. Ca sử dụng điều khiển quá trình phát triển phần mềm

Ca sử dụng không chỉ là một công cụ để đặc tả các yêu cầu của hệ thống mà còn điều khiển quá trình phân tích, thiết kế, cài đặt và kiểm thử. Trước hết ca sử dụng phản ánh yêu cầu của hệ thống cần phải thực hiện để đem lại dịch vụ cho những người sử dụng và kết quả là những giá trị gia tăng mà họ nhận được. Dựa trên mô hình ca sử dụng, người phát triển tạo ra một loạt các mô hình phân tích, thiết kế và cài đặt (mô hình thành phần và mô hình triển khai) nhằm vào việc thực hiện các ca sử dụng ở những mức khác nhau và xem xét để sao cho mỗi mô hình này là phù hợp với việc thực hiện mô hình ca sử dụng xây dựng được. Những người kiểm tra sẽ kiểm tra các cài đặt để đảm bảo rằng các thành phần của mô hình cài đặt thực hiện đúng các ca sử dụng. Do vậy, ta nói rằng: ca sử dụng điều khiển quá trình phát triển có nghĩa là quá trình phát triển tuân theo các luồng công việc được điều khiển bởi ca sử dụng.

2.3.2. Quá trình phát triển lấy kiến trúc làm trung tâm

Vai trò của kiến trúc hệ thống phần mềm giống một khung nền dựa trên đó phần mềm được xây dựng và phát triển đến hoàn thiện. Khái niệm kiến trúc phần mềm chứa đựng những khía cạnh tĩnh và động có ý nghĩa nhất định đối với hệ thống. Nó được phát triển dựa

theo yêu cầu của tổ chức, theo cảm nhận của người dùng và các tổ chức có liên quan khác. Mặt khác, nó cũng chịu ảnh hưởng của rất nhiều nhân tố khác, chẳng hạn như phần mềm của hệ thống, các khối xây dựng dùng lại được có sẵn, các cân nhắc về điều kiện triển khai, các hệ thống có sẵn trong môi trường tương tác với nó, và cả các yêu cầu phi chức năng. Kiến trúc là một cái nhìn thiết kế tổng thể những đặc điểm quan trọng nhất về hệ thống phần mềm khi tạm bỏ qua các chi tiết.

Mọi sản phẩm phần mềm đều bao gồm chức năng và hình thức thể hiện. Hai yếu tố này phải cân bằng với nhau để đem lại kết quả tốt nhất. Chức năng tương ứng với ca sử dụng và hình thức thể hiện tương ứng với kiến trúc. Do đó, việc lựa chọn các ca sử dụng để phát triển được định hướng theo kiến trúc và phải phù hợp với kiến trúc. Nói cách khác, kiến trúc phải cung cấp chỗ dựa cho việc thực hiện các ca sử dụng ngay khi bắt đầu tiến trình phát triển hệ thống và cả trong tương lai. Để có được bản mẫu cho kiến trúc, người phân tích phải có hiểu biết chung về các chức năng chính, đó là các ca sử dụng chính yếu. Chúng là các ca sử dụng mang ý nghĩa nhất, tạo nên các chức năng chủ yếu của hệ thống và thường ít thay đổi trong quá trình phát triển.

2.3.3. Tiến trình phát triển là quá trình lặp và tăng dần

Việc phát triển một phần mềm nói chung đòi hỏi một số lượng lớn công việc và có thể diễn ra trong một khoảng thời gian. Việc chia nhỏ toàn bộ công việc thành các phần nhỏ hoặc các dự án con là yêu cầu thiết thực. Mỗi dự án con là một bước lặp và tạo nên một sự tăng trưởng. Điều này dễ dàng thực hiện được khi phát triển phần mềm hướng đối tượng vì phần mềm được cấu thành từ các thành phần độc lập ghép nối lại với nhau. Để đạt hiệu quả nhất, các bước lặp phải được điều khiển, tức là chúng phải được lựa chọn và tiến hành theo một cách có kế hoạch từ trước.

Lựa chọn cái gì cần cài đặt trong một bước lặp dựa trên hai yếu tố sau: thứ nhất, bước lặp phải liên quan tới một nhóm các ca sử dụng để mở rộng tính khả dụng của hệ thống khi phát triển. Thứ hai, bước lặp phải giải quyết những rủi ro quan trọng nhất.

Mỗi bước lặp tiếp được xây dựng trên cơ sở các sản phẩm thiết kế từ trạng thái mà nó vừa kết thúc ở bước lặp trước. Bởi vì là một dự án con, nên từ các ca sử dụng đã dùng, nó tiếp tục mở rộng việc thực hiện các công việc phân tích, thiết kế, cài đặt và kiểm thử đối với các chức năng còn lại được nắm bắt trong các ca sử dụng tiếp theo để đưa chúng về dạng các mã nguồn thực thi được. Tuy nhiên, trong một vài bước ban đầu, người phát triển có thể chỉ thay thế một thiết kế còn sơ bộ bằng một kiến trúc khác chi tiết hơn, phức tạp hơn, vì vậy có thể chưa tạo ra sự tăng trưởng của sản phẩm thiết kế. Nhưng ở các bước sau, một sự tăng trưởng của sản phẩm nói chung là tất yếu và cần thiết.

Trong mỗi bước lặp, người thiết kế xác định các ca sử dụng liên quan, tạo lập một thiết kế dựa trên kiến trúc đã chọn, triển khai thiết kế dưới dạng các thành phần, và kiểm tra mức độ tương ứng giữa các thành phần và các ca sử dụng. Nếu một bước lặp thỏa mãn được các mục đích của nó thì có thể chuyển sang bước lặp tiếp theo. Nếu không, người thiết kế sẽ phải xem lại và thử một cách tiếp cận mới.

Một dự án gọi là thành công nếu được tiến hành mà không lệch hướng nhiều so với kế hoạch đã được định ra. Tối thiểu hóa được các vấn đề còn chưa được nhận thức chính là một trong các mục tiêu của việc giảm rủi ro. Một quá trình lặp có điều khiển sẽ mang lại rất nhiều lợi ích, đặc biệt giải quyết được những vấn đề liên quan đến các rủi ro.

Những khái niệm (ca sử dụng điều khiển tiến trình, tập trung kiến trúc lặp và tăng dần) có mức độ quan trọng như nhau. Kiến trúc cung cấp cấu trúc mà theo đó chỉ dẫn công việc trong các bước lặp.

Trong khi đó ca sử dụng xác định mục tiêu và định hướng công việc cho mỗi vòng lặp. Thiếu một trong ba khái niệm này sẽ làm giảm nghiêm trọng giá trị của quá trình phát triển. Chính quá trình lặp làm dễ dàng cho hoạt động quản lý và giảm sự phức tạp của quá trình phát triển, nhờ vậy mà giảm bớt những rủi ro.

2.4. MỘT SỐ KHÁI NIỆM CƠ BẢN TRONG UML

Cách tiếp cận hướng đối tượng để phát triển phần mềm dựa trên mô hình dữ liệu trừu tượng và khái niệm lớp nhằm chỉ ra những đặc tính chung các cấu trúc dữ liệu được sử dụng trong quá trình mô hình hóa hệ thống. Ngay từ đầu chương, chúng ta đã có cái nhìn khái quát nhất về khái niệm đối tượng và lớp, nhưng để chỉ rõ được những đặc tính chung về cấu trúc dữ liệu, trước hết chúng ta cần diễn tả đầy đủ và sâu sắc hơn về một số khái niệm cơ bản như *đối tượng, lớp, thuộc tính, thao tác và gói*.

2.4.1. Các đối tượng

Ngay từ đầu, khi nói về cách tiếp cận hướng đối tượng, chúng ta đã đưa ra khái niệm khái quát về đối tượng. Ở đây chúng ta bàn kỹ thêm để hiểu rõ hơn về đối tượng. *Đối tượng* là khái niệm cơ sở quan trọng nhất của cách tiếp cận hướng đối tượng. *Đối tượng là một khái niệm, một sự trừu tượng hóa hay một sự vật có nghĩa trong bài toán đang khảo sát.* Đối tượng là thực thể của hệ thống, của CSDL và được xác định thông qua định danh của chúng. Thông thường các đối tượng được mô tả bởi các danh từ riêng (tên gọi) hoặc được tham chiếu tới trong các mô tả của bài toán hay trong các thảo luận với người sử dụng. Có những đối tượng là những thực thể có trong thế giới thực như người, sự vật cụ thể, hoặc là những khái niệm như một công thức, hay khái niệm trừu tượng,... Có một số đối tượng được bổ sung vào hệ thống với lý do phục vụ cho việc cài đặt và có thể không có trong thực tế.

Đối tượng là những thực thể được xác định trong thời gian hệ thống hoạt động. Trong giai đoạn phân tích, ta phải đảm bảo rằng các đối tượng đều được xác định bằng các định danh. Đến giai đoạn thiết kế, ta phải lựa chọn cách thể hiện những định danh đó theo cách ghi địa chỉ bộ nhớ, gán các số hiệu, hay dùng tổ hợp một số giá trị của một số thuộc tính để biểu diễn. Theo quan điểm của *người lập trình, đối tượng được xem như là một vùng nhớ được phân chia trong máy tính để lưu trữ dữ liệu* (thuộc tính) và tập các hàm thao tác trên dữ liệu được gắn với nó. Bởi vì các vùng nhớ được phân hoạch là độc lập với nhau nên các đối tượng có thể tham gia vào nhiều chương trình khác nhau mà không ảnh hưởng lẫn nhau.

Nói gọn lại, những đặc trưng quan trọng của đối tượng là:

- *Định danh đối tượng*: dùng để phân biệt với những đối tượng khác, ngay cả khi chúng có các tính chất giống nhau. Mỗi đối tượng đều có một định danh và nó được thiết lập khi đối tượng được tạo ra trong hệ thống.

- *Tính bền vững của đối tượng*: mỗi đối tượng đều có một thời gian sống (tồn tại trong hệ thống) và điều này dẫn tới bản chất tĩnh của hệ thống. Những đối tượng bền vững là những đối tượng được lưu trữ vào trong các CSDL đối tượng.

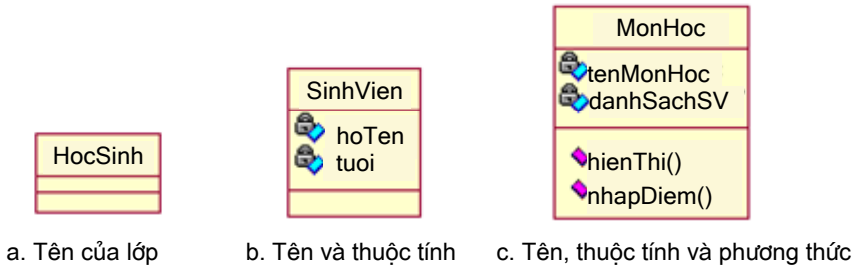
- Mỗi đối tượng phải hoặc có thể tương tác với các đối tượng khác, điều này dẫn đến bản chất động của hệ thống.

2.4.2. Lớp các đối tượng

Đối tượng là thể hiện, là một đại biểu của một lớp. Lớp là một mô tả về một nhóm các đối tượng có những tính chất (thuộc tính) giống nhau, có chung các hành vi ứng xử (thao tác gần như nhau), có cùng mối liên quan với các đối tượng của các lớp khác và có chung ngữ nghĩa trong hệ thống. Lớp chính là cơ chế được sử dụng để phân loại các đối tượng của một hệ thống. Lớp thường xuất hiện dưới dạng những danh từ chung trong các tài liệu mô tả bài toán hay trong các

thảo luận với người sử dụng. Cũng như các đối tượng, lớp có thể là những nhóm thực thể có trong thế giới thực, cũng có những lớp là khái niệm trừu tượng và có những lớp được đưa vào trong thiết kế để phục vụ cho cài đặt hệ thống,...

Lớp và mối quan hệ của chúng có thể mô tả trong các sơ đồ lớp, sơ đồ đối tượng và một số sơ đồ khác của UML. Trong sơ đồ lớp, lớp được mô tả bằng một hình hộp chữ nhật, trong đó có tên của lớp, có thể có các thuộc tính và các thao tác (hàm/phương thức).



Hình 2.2. Các ký hiệu mô tả lớp trong UML

Người ta thường đặt tên theo một quy tắc thống nhất như sau:

+ Tên của lớp thì chữ cái đầu của tất cả các từ đều viết hoa, ví dụ: SinhVien, HocSinh, KhachHang,...

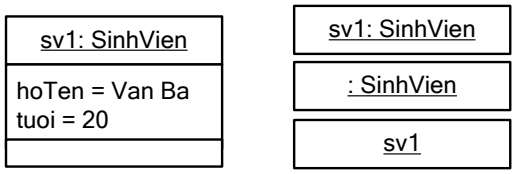
+ Tên của đối tượng, tên của thuộc tính thì viết hoa chữ cái đầu của các từ trừ từ đầu tiên, ví dụ: *hoTen, danhSachSV*,...

+ Tên của thao tác viết giống như tên của đối tượng nhưng có thêm cặp ngoặc đơn '(' và ')', ví dụ: *hienThi(), nhapDiem()*,...

Trong sơ đồ ở giai đoạn phân tích, một lớp có thể chỉ cần có tên lớp, tên và thuộc tính, hoặc có cả tên gọi, thuộc tính và các thao tác.

2.4.3. Các giá trị và các thuộc tính của đối tượng

Giá trị (value) là một phần của dữ liệu. Các giá trị thường là các số hoặc là các ký tự. *Thuộc tính của đối tượng* là thuộc tính của lớp được mô tả bởi giá trị của mỗi đối tượng trong lớp đó. Ví dụ:



Hình 2.3. Ký hiệu đối tượng trong UML

"Van Ba" và 20 là hai giá trị tương ứng với hai thuộc tính *hoTen, tuoi* của đối tượng sv1 trong lớp SinhVien.

Không nên nhầm lẫn giá trị với đối tượng. Các đối tượng có định danh chứ không phải là các giá trị. Có thể có ba sinh viên cùng tên "Van Ba", nhưng trong hệ thống các sinh viên này phải được *quản lý theo định danh để xác định duy nhất từng đối tượng*. Giá trị có thể là các giá trị của các kiểu dữ liệu nguyên thủy như các kiểu số hoặc các kiểu xâu ký tự, hoặc là tập hợp của các giá trị nguyên thủy.

Các dữ liệu thành phần của một lớp có thể được bao gói thông qua các thuộc tính quản lý quyền truy nhập để phục vụ việc che giấu thông tin của phương pháp hướng đối tượng. Trong UML ta có thể sử dụng các ký hiệu '+', '#', '-' để đặc tả các thuộc tính đó.

2.4.4. Các thao tác

Thao tác là một hàm hay thủ tục có thể áp dụng (gọi hàm) cho/bởi các đối tượng trong một lớp. Khi nói tới một thao tác là ngầm định nói tới một đối tượng đích để thực hiện thao tác đó. Ví dụ, thao tác (hàm) *hienThi()* của lớp MonHoc được gọi để hiển thị các sinh viên học một môn học cụ thể như "Lập trình hướng đối tượng" chẳng hạn.

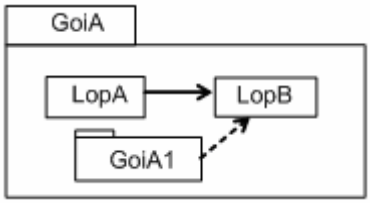
Một số thao tác có thể là *đa xạ*, *được nạp chồng*. Đa xạ nghĩa là nó có thể áp dụng cho nhiều lớp khác nhau với những nội dung thực hiện có thể khác nhau, nhưng cùng tên gọi. Ví dụ lớp ThietBi có hàm *tinhGia()*. Nạp chồng nghĩa là có nhiều phương thức (công thức) tính giá bán khác nhau tùy thuộc vào từng loại thiết bị. Tất cả các phương thức này đều thực hiện một nhiệm vụ *tinhGia()*, nhưng được cài đặt

với nội dung (các đoạn chương trình) khác nhau. Hệ thống hướng đối tượng tự động chọn phương thức tương ứng với ngữ cảnh của đối tượng đích để thực hiện.

Tương tự như các dữ liệu thành phần, các thao tác cũng được quản lý quyền truy nhập và được ký hiệu như thuộc tính.

2.4.5. Các gói

Để hiểu được những hệ thống lớn, phức tạp có nhiều lớp đối tượng, thì chúng ta phải có cách chia các lớp đó thành một số nhóm được gọi là gói. *Gói là một nhóm các phần tử của mô hình gồm các lớp, các mối quan hệ và các gói nhỏ hơn.* Cách tổ chức hệ thống thành các gói (hệ thống con) chính là cách phân hoạch mô hình thành các đơn vị nhỏ hơn để trị, dễ hiểu và dễ quản lý hơn. Gói được mô tả trong UML gồm tên của gói, có thể có các lớp, gói nhỏ khác bên trong.



Hình 2.4. Gói các lớp trong UML

Khi phân chia các lớp thành các gói, chúng ta có thể dựa vào: các lớp chính (thống trị), các mối quan hệ chính, các chức năng chính. Theo cách phân chia đó chúng ta có thể chia hệ thống thành các phân hệ phù hợp với cách phân chia trong hệ thống thực, thường là dựa trên các chức năng hoặc đặc tính kỹ thuật. Lợi thế của gói là cho khả năng sử dụng lại.

Ví dụ: hệ thống quản lý thư viện có thể tổ chức thành bốn gói: *gói giao diện, gói nghiệp vụ, gói CSDL và gói tiện ích*. Trong đó:

Gói giao diện (UI - User Interface): bao gồm các lớp giao diện với người dùng, cho các khả năng quan sát và truy nhập vào dữ liệu. Các

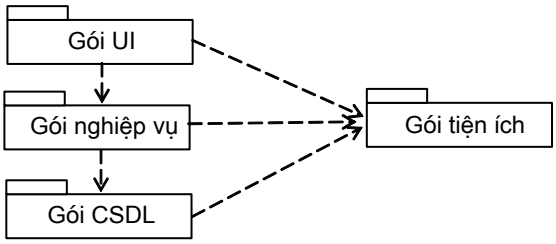
đối tượng trong gói này có thể thực hiện các thao tác trên các đối tượng tác nghiệp để truy vấn hay nhập dữ liệu.

Gói nghiệp vụ (Business Package): chứa các lớp thực thể thuộc lĩnh vực bài toán ứng dụng.

Gói CSDL (Database Package): chứa các lớp dịch vụ cho các lớp ở gói nghiệp vụ trong việc tổ chức, quản lý và lưu trữ dữ liệu.

Gói tiện ích (Utility Package): chứa các lớp dịch vụ cho các gói khác của hệ thống.

Các gói của một hệ thống thường có mối quan hệ với nhau, như quan hệ phụ thuộc.



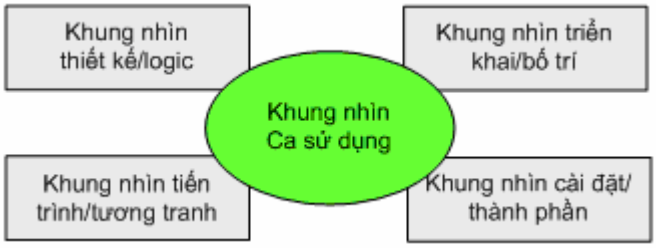
Hình 2.5. Tổ chức các gói của hệ thống thư viện

2.5. CÁC NÉT CƠ BẢN VỀ UML

2.5.1. Mô hình hóa kiến trúc hệ thống

Khi mô hình hóa hệ thống, chúng ta cần quan tâm tới một loạt các khía cạnh khác nhau như về mặt chức năng (cấu trúc tĩnh của nó cũng như các tương tác động), về mặt phi chức năng (yêu cầu về thời gian, về độ đáng tin cậy, về quá trình thực thi) cũng như về khía cạnh tổ chức (tổ chức làm việc, ánh xạ nó vào các module..). Vì vậy một hệ thống khi mô hình hóa thường được xem xét trên một loạt các phương diện khác nhau, mỗi phương diện sẽ thể hiện một bức ảnh ánh xạ của toàn bộ hệ thống và chỉ ra một khía cạnh riêng của hệ thống.

Trong UML, kiến trúc của hệ thống phần mềm chuyên sâu (cho ta một cách nhìn khái quát nhất về hệ thống phần mềm ở các góc độ khác nhau) được mô tả theo 5 loại *khung nhìn (views)* khác nhau. Mỗi khung nhìn phản ánh về một khía cạnh của tổ chức và cấu trúc của hệ thống, tập trung vào từng mặt cụ thể giúp cho ta hiểu và sử dụng hệ thống tốt nhất. Mỗi khung nhìn thường được thể hiện trong một số sơ đồ nhất định.



Hình 2.6. Mô hình hóa kiến trúc hệ thống

- *Khung nhìn ca sử dụng (Use case view)*: chứa các tác nhân, ca sử dụng, sơ đồ ca sử dụng (khía cạnh tĩnh của khung nhìn) trong hệ thống. Chúng cũng có thể bao gồm vài sơ đồ trình tự, sơ đồ cộng tác (khía cạnh động của khung nhìn) và gói. Khung nhìn ca sử dụng tập trung vào mức cao của cái hệ thống sẽ làm, không quan tâm đến hệ thống làm như thế nào.

- *Khung nhìn thiết kế logic (design view)* tập trung vào hệ thống cài đặt hành vi trong ca sử dụng như thế nào. Nó bao gồm các lớp, sơ đồ lớp, sơ đồ đối tượng (khía cạnh tĩnh của khung nhìn), sơ đồ tương tác, sơ đồ hoạt động, sơ đồ biến đổi trạng thái (khía cạnh động của khung nhìn) và các gói. Trong khung nhìn, người ta quan tâm đến hai lớp quan trọng là lớp phân tích (lớp biên, lớp điều khiển, lớp dữ liệu) và lớp thiết kế (lớp phụ thuộc vào ngôn ngữ). Khung nhìn này tập trung vào cấu trúc logic của hệ thống nên còn được gọi là *khung nhìn logic (logical view).* Khung nhìn này tập trung vào cấu trúc của hệ thống.

Nhờ nó, người ta nhận ra các bộ phận cơ bản cấu thành hệ thống thể hiện mọi quá trình trao đổi, xử lý thông tin cơ bản trong hệ thống.

- *Khung nhìn tiến trình/tương tranh (process view)* biểu diễn sự phân chia các luồng thực hiện công việc, các lớp đối tượng cho các *tiến trình* và sự đồng bộ giữa các *luồng (thread)* trong hệ thống. Khung nhìn này tập trung vào các nhiệm vụ tương tranh, tương tác với nhau trong hệ thống đa nhiệm. Nó chủ yếu diễn đạt hiệu năng, quy mô và năng lực thông qua của hệ thống.

- *Khung nhìn cài đặt/thành phần (component/implementation view)* bao gồm các thành phần (là mô-đun vật lý hay các file) để lắp ráp thành hệ thống vật lý. Khung nhìn này hướng đến việc quản lý cấu hình của hệ thống. Khung nhìn này bao gồm thành phần, sơ đồ thành phần và gói.

- *Khung nhìn triển khai/bố trí (Deployment view)* bao gồm các nút tạo nên kết cấu phần cứng mà trên đó hệ thống vận hành. Khung nhìn này chủ yếu hướng đến sự phân tán và cài đặt cụ thể của hệ thống, tức là liên quan đến triển khai vật lý của hệ thống. Khung nhìn triển khai chỉ ra các tiến trình và thiết bị trên mạng và các kết nối vật lý giữa chúng. Sơ đồ triển khai cũng hiển thị tiến trình và chỉ ra tiến trình nào chạy trên máy nào. Khung nhìn này được thể hiện trong các sơ đồ triển khai/bố trí các nút của hệ thống.

2.5.2. Các vấn đề cốt lõi trong UML

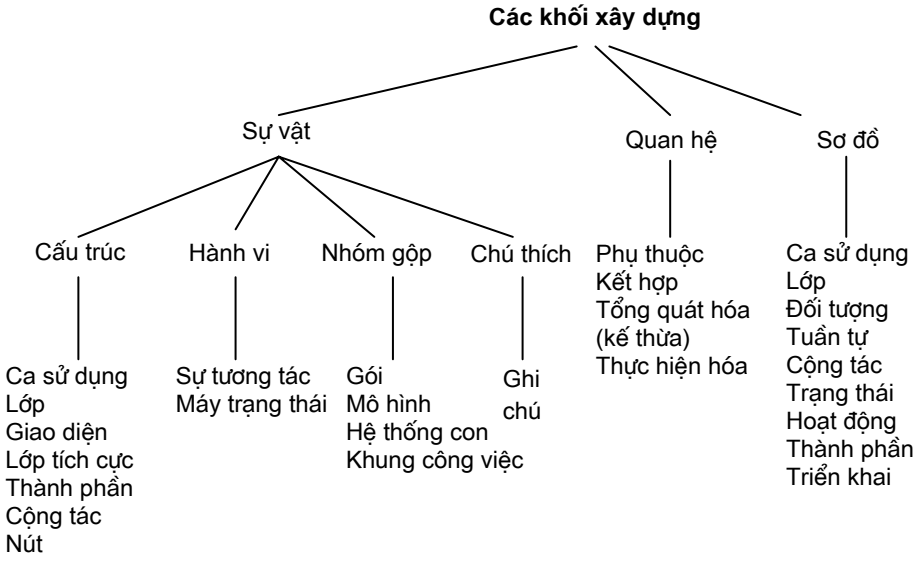
Để hiểu và sử dụng tốt UML trong phân tích, thiết kế hệ thống, đòi hỏi phải nắm bắt được ba vấn đề cốt lõi nhất:

- Các khối xây dựng cơ bản của UML

- Các quy tắc ngữ nghĩa của UML

- Một số cơ chế chung được áp dụng cho việc mô hình hóa.

2.5.2.1. Các khối xây dựng (building blocks) cơ bản của UML



Hình 2.7. Các thành phần cơ sở của UML

Các *sự vật (things)* là các trừu tượng hóa và là những phần tử lớp đầu tiên (những viên gạch) để xây dựng lên các mô hình trong UML. *Các quan hệ (relationships)* gắn kết các sự vật với nhau, *các sơ đồ (diagrams)* nhóm các sự vật được quan tâm lại tạo nên ngữ nghĩa của nó (cho một mô hình).

1. Các sự vật (things)

UML có bốn loại sự vật, đó là sự vật cấu trúc, hành vi, nhóm và chú thích.

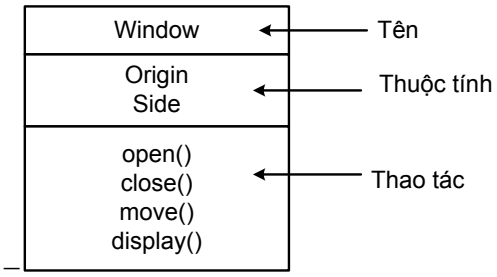
- ***Sự vật cấu trúc (structural things)****:* là các *danh từ* trong mô hình UML, biểu diễn cho các thành phần khái niệm hay vật lý của hệ thống.

UML có bảy sự vật cấu trúc:

+ *Lớp (class)*

Lớp là tập các đối tượng cùng chia sẻ các thuộc tính, thao tác, quan hệ và ngữ nghĩa. Một lớp có trách nhiệm thực hiện một hay

nhiều giao diện. Một lớp được biểu diễn bằng một hình chữ nhật bên trong có tên, các thuộc tính và các tác vụ.



Hình 2.8. Lớp

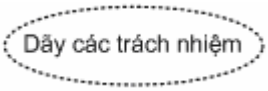
+ *Giao diện (interface)*

Giao diện là tập các thao tác làm dịch vụ cho lớp hay thành phần (mô-đun vật lý hoặc mã chương trình). Giao diện mô tả hành vi quan sát được từ bên ngoài thành phần. Giao diện chỉ khai báo các thao tác xử lý nhưng không định nghĩa nội dung thực hiện. Nó thường không đứng một mình mà thường nó được gắn với lớp hay một thành phần.

Hình 2.9. Giao diện

+ *Cộng tác (collaboration)*

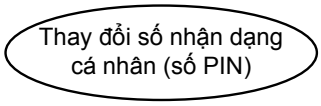
Sự cộng tác xác định các hoạt động bên trong hệ thống và là một bộ các nguyên tắc và các phần tử khác nhau cùng làm việc để cung cấp một hành vi hợp tác lớn hơn tổng hành vi của tất cả các phần tử. Nó được ký hiệu bằng hình elíp với đường đứt nét và thường chỉ gồm có tên.



Hình 2.10. Sự cộng tác

+ *Ca sử dụng (use case)*

Ca sử dụng mô tả một tập các hành động mà hệ thống sẽ thực hiện để phục vụ cho các tác nhân ngoài. *Tác nhân ngoài* là những gì bên ngoài có tương tác, trao đổi với hệ thống. Nó được ký hiệu bằng hình elíp nét liền, thường chỉ bao gồm có tên.



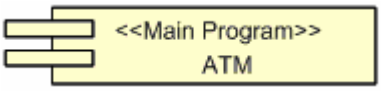
Hình 2.11. Ca sử dụng

+ *Lớp tích cực/hoạt động (active class)*

Lớp tích cực được xem như là lớp có đối tượng làm chủ một hay nhiều tiến trình, luồng hành động. Bởi vậy nó có thể khởi động hoạt động điều khiển. Nó được ký hiệu như một lớp nhưng có đường viền đậm.

+ *Thành phần (component)*

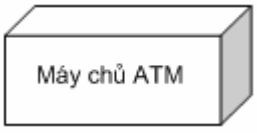
Thành phần biểu diễn vật lý mã nguồn, các tệp nhị phân trong quá trình phát triển hệ thống. Một thành phần được ký hiệu bằng một hình chữ nhật với các bảng và thường bao gồm chỉ có tên của nó.



Hình 2.12. Thành phần

+ *Nút (node)*

Nút thể hiện thành phần vật lý, tồn tại khi chương trình chạy và biểu diễn cho các tài nguyên được sử dụng trong hệ thống, thường có ít nhất bộ nhớ và khả năng xử lý. Nó được ký hiệu bằng một hình hộp thường bao gồm tên của nó.

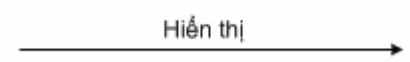


Hình 2.13. Nút

- ***Sự vật hành vi (behavioral things)****:* biểu diễn hành vi trong tương tác của các thành phần và sự biến đổi trạng thái của hệ thống. Như vậy nó mô tả hành vi của hệ thống theo thời gian và không gian. Có hai loại chính là sự tương tác và máy biến đổi trạng thái.

+ *Sự tương tác (interaction)*

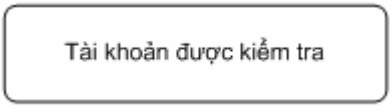
Sự tương tác là hành vi bao gồm một tập các thông điệp trao đổi giữa các đối tượng trong một ngữ cảnh cụ thể để thực hiện một ca sử dụng. Một thông điệp được ký hiệu bằng một đường thẳng có hướng, gồm tên của tác vụ.



Hình 2.14. Thông điệp/thông báo

+ *Máy biến đổi trạng thái (state machine)*

Máy biến đổi trạng thái (ôtômát hữu hạn trạng thái) chỉ ra trật tự thay đổi trạng trái khi các đối tượng hay sự tương tác sẽ phải đi qua để đáp ứng các sự kiện xảy ra. Nó gồm một số phần tử biểu diễn trạng thái, các chuyển dịch (từ trạng thái này sang trạng thái khác) và các sự kiện (kích hoạt chuyển dịch). Một trạng thái được ký hiệu bằng hình chữ nhật tròn góc trong đó có tên trạng thái.

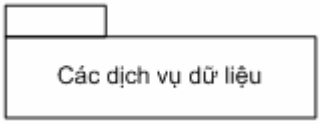


Hình 2.15. Trạng thái

- ***Các sự vật nhóm gộp (grouping things)****:* là bộ phận tổ chức của mô hình UML. Nó là công cụ để tổ chức các thành phần của một mô hình thành các nhóm. Một mô hình có thể được phân chia vào trong các gói. Một gói đơn thuần là một khái niệm. Trong ký hiệu, một gói thường có tên, đôi khi có nội dung của nó. Các sự vật nhóm có *gói (package), mô hình* và *khung công việc.*

+ *Gói (package)*

Gói là phần tử đa năng được sử dụng để tổ chức các lớp, hay một số nhóm khác vào trong một nhóm. Không giống với thành phần (*component*), phần tử gói hoàn toàn là khái niệm, có nghĩa là chúng chỉ tồn tại trong mô hình vào thời điểm phát triển hệ thống chứ không tồn tại vào thời điểm chạy chương trình. Gói giúp ta quan sát hệ thống ở mức tổng quát.



Hình 2.16. Gói

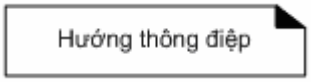
+ *Mô hình*

Mô hình là những mô tả về các đặc tính tĩnh và/hoặc động của các chủ thể trong hệ thống.

+ *Khung công việc*

Khung công việc là một tập các lớp trừu tượng hay cụ thể được sử dụng như là các khuôn mẫu để giải quyết một họ các vấn đề tương tự.

Chú thích: là bộ phận chú giải của mô hình, giải thích về các phần tử, khái niệm và cách sử dụng chúng trong mô hình.



Hình 2.17. Chú thích

2. Các mối quan hệ (relationships)

UML cho phép biểu diễn cả bốn mối quan hệ giữa các đối tượng trong các hệ thống. Đó là các quan hệ: *phụ thuộc, kết hợp, tổng quát hóa và thực hiện hóa.*

+ Quan hệ phụ thuộc (dependency)

Đây là quan hệ ngữ nghĩa giữa hai sự vật, trong đó sự thay đổi của một sự vật sẽ tác động đến ngữ nghĩa của sự vật phụ thuộc.

Hình 2.18. Quan hệ phụ thuộc

+ Quan hệ kết hợp (association)

Kết hợp là quan hệ cấu trúc xác định mối liên kết giữa các lớp đối tượng. Khi có một đối tượng của lớp này gửi/nhận thông điệp đến/từ chỗ đối tượng của lớp kia thì hai lớp đó có quan hệ kết hợp. Một dạng đặc biệt của quan hệ kết hợp là *quan hệ tụ hợp (aggregation),* biểu diễn mối quan hệ giữa toàn thể và bộ phận. Trong ký hiệu thường có nhãn và thường chứa các bài trí khác nhau giải thích vai trò của đối tượng tham gia vào liên kết và các bản số của chúng

Hình 2.19. Kết hợp

+ Quan hệ tổng quát hóa (generalization)

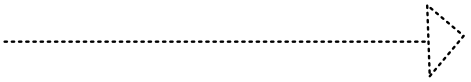
Đây là quan hệ mô tả sự khái quát hóa mà trong đó một số đối tượng cụ thể (con) sẽ được kế thừa các thuộc tính, các thao tác của các đối tượng tổng quát (cha). Nó được ký hiệu bằng đường nét liền với mũi tên rỗng chỉ về phía cha.

Hình 2.20. Tổng quát hóa

+ Quan hệ thực hiện hóa (realization)

Thực hiện hóa là quan hệ ngữ nghĩa giữa giao diện và lớp (hay thành phần) để thực hiện cài đặt các dịch vụ đã được khai báo trong các giao diện. Mối quan hệ này dựa vào 2 vị trí: giữa các giao diện và

các lớp hoặc các thành phần thực hiện nó. Một mối quan hệ thực hiện được xem như một mối quan hệ nằm giữa mối quan hệ tổng quát hóa và mối quan hệ phụ thuộc, vì thế nó được ký hiệu bằng đường nét đứt có mũi tên trống.

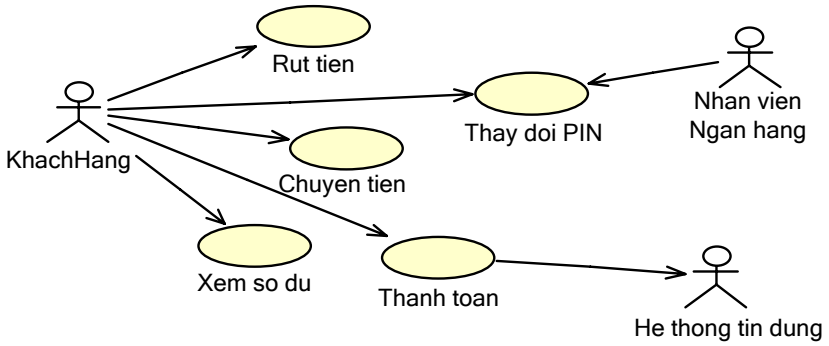


Hình 2.21. Thực hiện hóa

3. Các sơ đồ

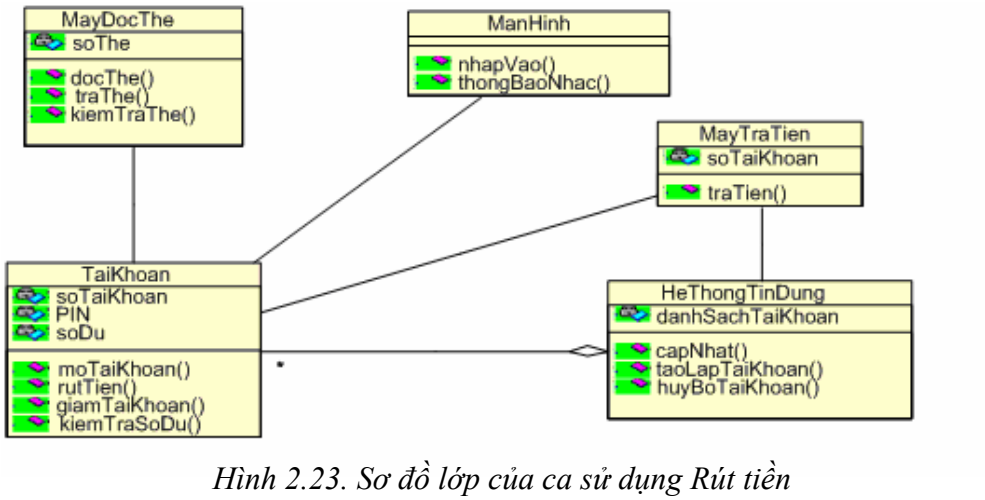
Sơ đồ (diagram) là đồ thị biểu diễn đồ họa về tập các phần tử (các từ vựng) trong mô hình và mối quan hệ của chúng (một số tác giả gọi là biểu đồ). Sơ đồ thường được thể hiện như một đồ thị liên thông với các đỉnh (là các sự vật) và các cung (là các mối quan hệ). Sơ đồ chứa đựng các nội dung của các khung nhìn dưới các góc độ khác nhau và một thành phần của hệ thống có thể xuất hiện trong một hay nhiều sơ đồ. UML cung cấp những sơ đồ trực quan để biểu diễn các khía cạnh khác nhau của hệ thống, bao gồm:

- *Sơ đồ ca sử dụng (use case diagram)* mô tả sự tương tác giữa các tác nhân ngoài và hệ thống thông qua các ca sử dụng. Các ca sử dụng là những nhiệm vụ chính, các dịch vụ, những trường hợp sử dụng cụ thể mà hệ thống cung cấp cho người sử dụng và ngược lại.



Hình 2.22. Sơ đồ ca sử dụng của máy rút tiền tự động ATM

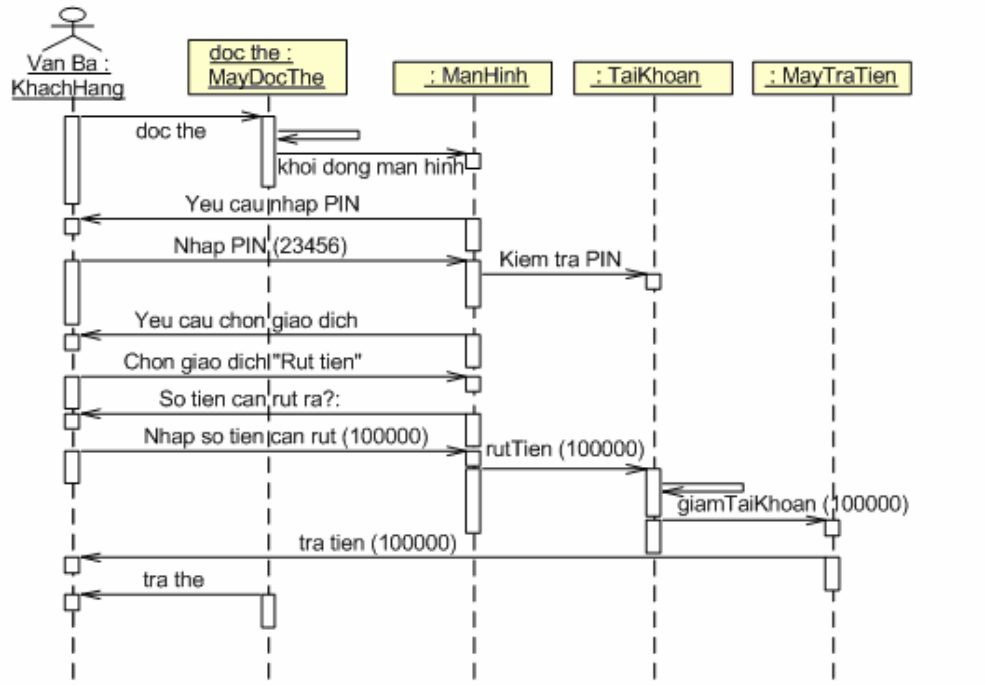
- *Sơ đồ lớp (class diagram)* mô tả cấu trúc tĩnh, mô tả mô hình khái niệm bao gồm các lớp đối tượng và các mối quan hệ của chúng trong hệ thống hướng đối tượng.



Hình 2.23. Sơ đồ lớp của ca sử dụng Rút tiền

Người ta sử dụng sơ đồ lớp để mô hình hóa khung nhìn thiết kế tĩnh của hệ thống và thường sử dụng sơ đồ này để mô hình hóa bảng từ vựng của hệ thống, mô hình hóa các cộng tác đơn giản, mô hình hóa cơ sở dữ liệu logic. Người ta còn sử dụng sơ đồ đối tượng để mô hình hóa khung nhìn thiết kế tĩnh của hệ thống cũng như mô hình hóa cấu trúc của đối tượng. Sơ đồ đối tượng mô hình hóa các thể hiện của các sự vật trong một sơ đồ lớp. Nó đưa ra một tập các đối tượng và các mối quan hệ giữa chúng tại một thời điểm…

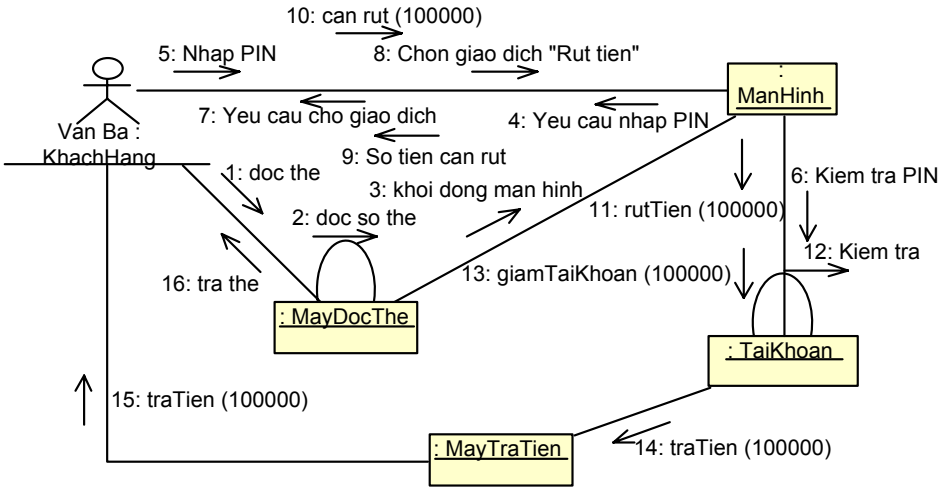
- *Sơ đồ trình tự (sequence diagram)* là một loại sơ đồ tương tác (interaction diagram) thể hiện sự tương tác của các đối tượng với nhau, chủ yếu là trình tự gửi và nhận thông điệp để thực thi các yêu cầu, các công việc *theo thời gian (hình 2.24).*



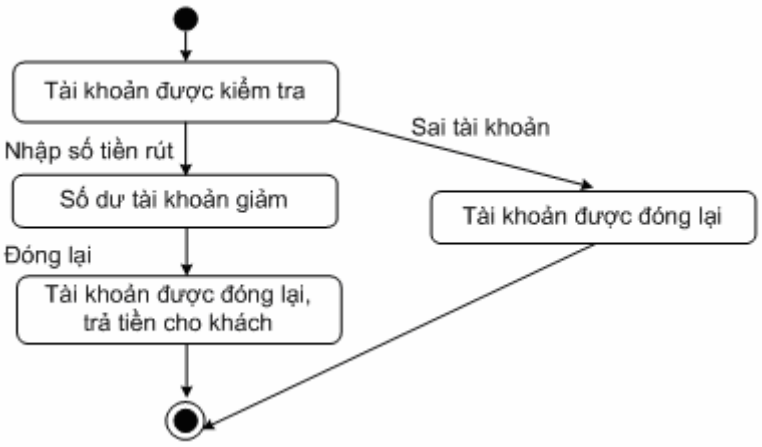
Hình 2.24. Sơ đồ trình tự của ATM mô tả ca sử dụng "Rút tiền"

- *Sơ đồ cộng tác (collaboration diagram)* cũng là một loại sơ đồ tương tác, tương tự như sơ đồ trình tự nhưng nhấn mạnh vào sự tương tác của các đối tượng trên cơ sở cộng tác với nhau bằng cách trao đổi các thông điệp để thực hiện các yêu cầu *theo ngữ cảnh công việc* (hình 2.25).

- *Sơ đồ trạng thái (statechart diagram*) cũng là một loại sơ đồ tương tác, biểu diễn một máy trạng thái. Nó biểu diễn dòng điều khiển từ trạng thái này tới trạng thái khác. Nó nhấn mạnh dòng điều khiển từ hoạt động này đến hoạt động khác đang xảy ra bên trong một máy trạng thái (hình 2.26).

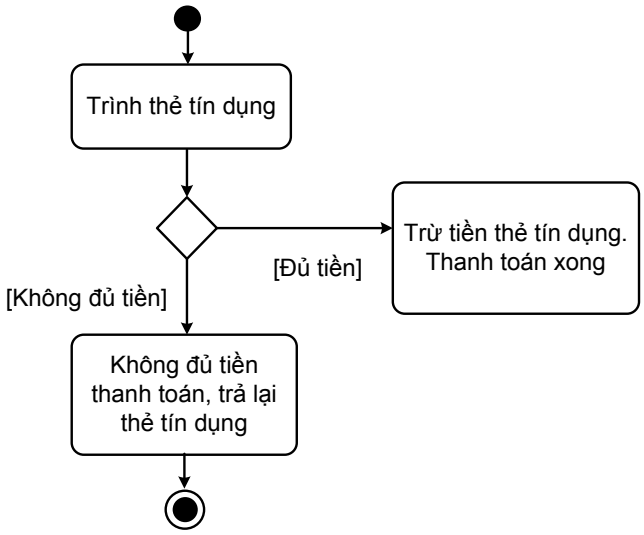


Hình 2.25. Sơ đồ cộng tác mô tả ca sử dụng "Rút tiền"



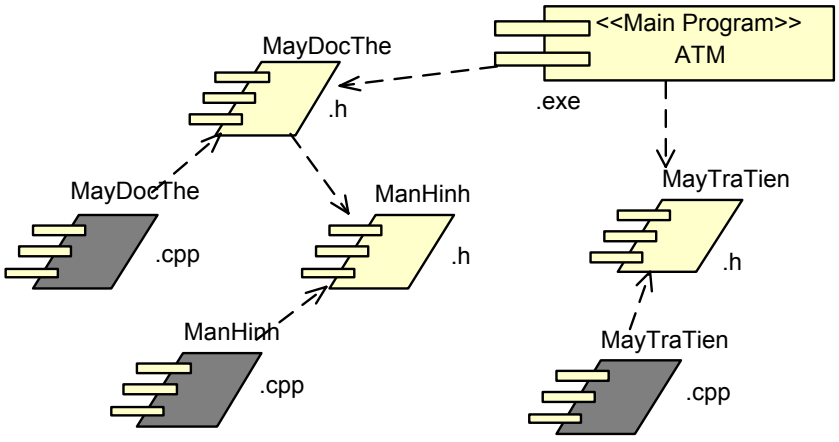
Hình 2.26. Trạng thái của tài khoản trong thực thi ca sử dụng thiết kế rút tiền

- *Sơ đồ hoạt động (activity diagram)* cũng là một loại sơ đồ tương tác, chỉ ra dòng hoạt động của hệ thống, bao gồm các trạng thái hoạt động, trong đó từ một trạng thái hoạt động sẽ chuyển sang trạng thái khác sau khi một hoạt động tương ứng được thực hiện. Nó chỉ ra trình tự các bước, tiến trình thực hiện cũng như các điểm quyết định và sự rẽ nhánh theo luồng sự kiện (hình 2.27).



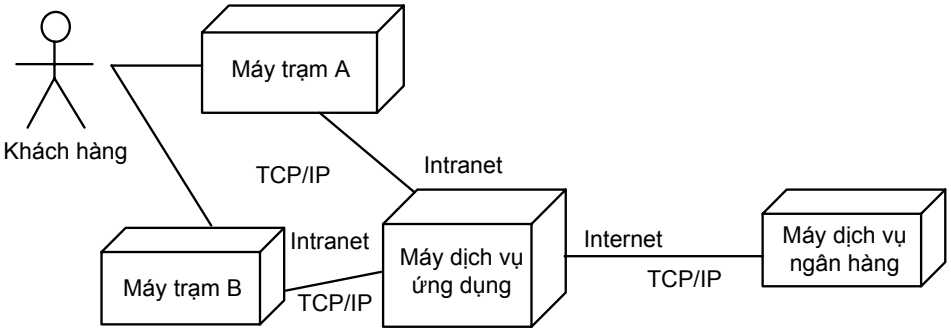
Hình 2.27. Sơ đồ hoạt động trả tiền bằng thẻ tín dụng

- *Sơ đồ thành phần (component diagram)* chỉ ra cấu trúc vật lý của các thành phần trong hệ thống (các thành phần mã nguồn, mã nhị phân, thư viện và các thành phần thực thi) tức là biểu diễn các thành phần (khả thi/thư viện) phần mềm trong hệ thống cùng với các mối quan hệ giữa chúng (hình 2.28).



Hình 2.28. Sơ đồ thành phần của máy rút tiền tự động (ATM) trên máy trạm khi cài đặt hệ thống bằng C++

- *Sơ đồ triển khai (deployment diagram)* chỉ ra cách bố trí vật lý các thành phần theo kiến trúc được thiết kế của hệ thống.



Hình 2.29. Sơ đồ triển khai của hệ thống dịch vụ ngân hàng

2.5.2.2. Các quy tắc ngữ nghĩa của UML

Mô hình hóa một hệ thống không đơn giản là lắp ghép các khối được xây dựng của UML một cách ngẫu nhiên. Cũng như mọi ngôn ngữ, UML có một số quy tắc chỉ ra rằng, một mô hình được xây dựng tốt là mô hình có ngữ nghĩa chắc chắn và đồng bộ với tất cả mô hình có quan hệ với nó.

UML đưa ra các quy tắc ngữ nghĩa:

- *Tên gọi*: là cái mà ta gọi là các sự vật, các mối quan hệ và các sơ đồ.

- *Phạm vi*: là khuôn khổ cho ý nghĩa cụ thể đối với một tên gọi.

- *Tính trực quan*: các tên gọi có thể nhìn thấy và được các phần tử khác sử dụng.

- *Tính tích hợp*: các sự vật có thể liên kết với các sự vật khác như thế nào.

- *Tính thực hiện được*: nó có ý nghĩa gì để vận hành và mô phỏng một mô hình động.

Người ta còn đưa ra những nguyên tắc khác để xây dựng tốt các mô hình như:

- *Sự lược đi*: một số phần tử được dấu đi làm đơn giản sự nhìn nhận của mô hình.

- *Sự không đầy đủ*: một số phần tử có thể tạm bỏ qua

- *Sự không chắc chắn*: sự tích hợp của mô hình là chưa đảm bảo

Trong mô hình hóa hệ thống với UML, ta có thể sử dụng *ngôn ngữ ràng buộc đối tượng (OCL)* để đặc tả chính xác các phần tử của hệ thống và các ràng buộc chặt chẽ giữa các mối quan hệ, giới hạn phạm vi của mô hình hệ thống cho phù hợp với điều kiện ràng buộc thực tế.

Trong UML có hai quy tắc chính:

- *Quy tắc ràng buộc* được sử dụng để giới hạn phạm vi của mô hình, ví dụ các quy tắc hạn chế, quy định rõ phạm trù của các mối quan hệ như kết hợp, kế thừa hay khả năng nạp chồng trong các lớp.

- *Quy tắc suy diễn* chỉ ra cách các sự vật có thể suy diễn được, ví dụ *tuổi* của một người có thể suy ra được từ *ngày/tháng/năm hiện thời* trừ đi *ngày/tháng/năm sinh*.

2.5.2.3. Các cơ chế chung trong UML

UML cung cấp 4 cơ chế chung để áp dụng trong khi mô hình hóa:

- Các đặc tả (Specification)

- Các bài trí (Adornments)

- Sự phân hoạch chung (Common divisions)

- Các cơ chế mở rộng (Extensibility mechanisms)

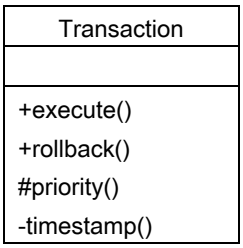
1. Các đặc tả (Specification)

UML mạnh hơn một ngôn ngữ đồ họa: vì rằng bên cạnh các ký hiệu đồ họa nó còn cung cấp một cách diễn tả vượt trội bằng văn bản theo cú pháp và ngữ nghĩa của khối đồ họa sử dụng. Ví dụ như một tập đầy đủ các thuộc tính, các tác vụ và cáchành vi mà lớp đó chứa.

Nhờ vậy ta dùng ký hiệu đồ họa để trực quan hóa hệ thống, dùng đặc tả để chỉ ra các chi tiết của hệ thống.

2. Các bài trí (Adornments)

Hầu hết các phần tử trong UML đều có ký hiệu đồ họa duy nhất, trực tiếp để cung cấp một sự thể hiện trực quan về các khía cạnh quan trọng nhất của phần tử đó. Hình 2.30 chỉ ra một lớp được bài trí cho biết lớp trừu tượng này có 2 tác vụ chung (public: +), một tác vụ được bảo vệ (protected: #) và một tác vụ riêng (private: -):

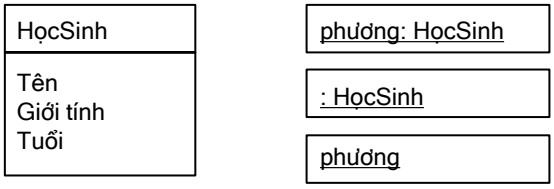


Hình 2.30. Bài trí

3. Sự phân hoạch chung (Common divisions)

Các khái niệm mô hình hóa trong UML thường được phân thành cặp theo 2 cách:

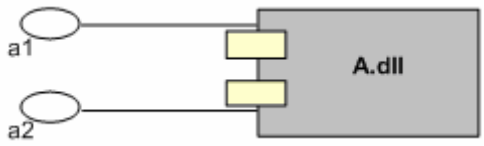
+ Phân hoạch lớp và đối tượng. Một lớp là một trừu tượng, một đối tượng là một biểu hiện cụ thể của lớp đó. UML thường phân biệt lớp và đối tượng của lớp đó bằng cách gạch chân tên của đối tượng.



Hình 2.31. Lớp và đối tượng

+ Phân hoạch giao diện và triển khai một giao diện, như khai báo một hợp đồng và triển khai một hợp đồng. Trong UML có thể mô

hình hóa cả giao diện và triển khai. Ví dụ thành phần A thực hiện giao diện a1 và a2.



Hình 2.32. Giao diện và triển khai

4. Các cơ chế mở rộng (Extensibility mechanisms)

UML cung cấp ngôn ngữ chuẩn để viết bản thiết kế phần mềm, nhưng nó vẫn chưa đủ để diễn đạt mọi sắc thái có thể của các mô hình trong mọi lĩnh vực và trong mọi thời điểm. Các cơ chế dùng để mở rộng gồm có:

- Các khuôn mẫu (stereotypes)
- Các giá trị thẻ (tagged values)
- Các ràng buộc (constraints)

Sự mở rộng từ vựng của ngôn ngữ UML cho phép tạo ra các khối xây dựng mới từ các khối có sẵn để cụ thể cho vấn đề của ta bằng cách thêm vào ký hiệu khuôn mẫu, giá trị thẻ hay ràng buộc.

2.6. QUY TRÌNH XÂY DỰNG CÁC MÔ HÌNH CƠ BẢN TRONG PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG HƯỚNG ĐỐI TƯỢNG BẰNG UML

2.6.1. Các pha chính trong quy trình phát triển phần mềm

Có năm pha chính cần thực hiện trong quy trình phát triển phần mềm:

1. Xác định các yêu cầu
2. Phân tích hệ thống
3. Thiết kế hệ thống

4. Lập trình và kiểm tra hệ thống

5. Vận hành và bảo trì hệ thống.

Trong quy trình phát triển phần mềm kể trên, hai pha cốt lõi là phân tích và thiết kế hướng đối tượng bằng UML nhằm xây dựng các mô hình đặc tả các yêu cầu, các mô hình khái niệm, logic và kiến trúc của hệ thống. Phần chi tiết trong mỗi pha có thể xem trong [23].

2.6.2. Phần mềm công cụ phục vụ phân tích và thiết kế hướng đối tượng

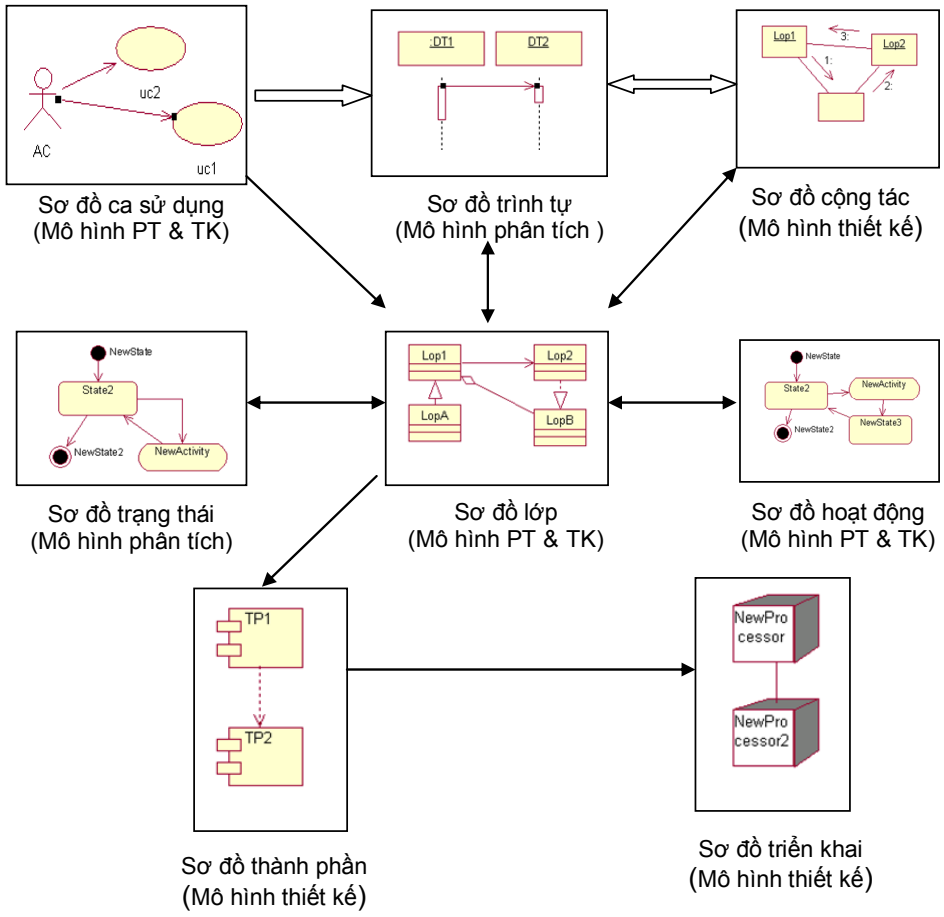
Rational Rose [12] là phần mềm công cụ mạnh hỗ trợ cho quá trình phân tích, thiết kế hệ thống hướng đối tượng. Nó giúp cho việc mô hình hóa hệ thống trước khi viết chương trình, đồng thời có khả năng kiểm tra đảm bảo tính đúng đắn, hợp lý của kiến trúc hệ thống. Rose hỗ trợ phát sinh mã khung chương trình trong nhiều ngôn ngữ lập trình khác nhau như: C++, Java, Visual Basic, Oracle 8,...

Bản thân UML không xác định quá trình phát triển phần mềm, nhưng UML và Rose hỗ trợ rất hiệu quả trong cả quá trình xây dựng phần mềm [19].

Quá trình xây dựng các mô hình đặc tả các yêu cầu, các mô hình khái niệm, logic và kiến trúc của hệ thống có thể thực hiện như trong sơ đồ (hình 2.33) [18].

Các chương tiếp theo sau sẽ trình bày chi tiết các mô hình cơ bản nhất trong quá trình phân tích và thiết kế hướng đối tượng.

Để xem chi tiết hơn về quy trình và các kỹ thuật thực hiện trong quy trình phân tích, thiết kế hướng đối tượng, bạn đọc có thể xem bổ sung trong các tài liệu [23],[25].



Hình 2.33. Quy trình xây dựng các mô hình cơ bản trong phân tích và thiết kế hệ thống hướng đối tượng bằng UML

3

YÊU CẦU HỆ THỐNG VÀ MÔ HÌNH NGHIỆP VỤ

3.1. KHÁI NIỆM YÊU CẦU

Yêu cầu cho một hệ thống phần mềm là những đòi hỏi bắt buộc những việc mà hệ thống phải làm và những ràng buộc mà hệ thống phải tuân thủ khi hoạt động. Yêu cầu có thể là *yêu cầu chức năng* (các chức năng, dịch vụ) hay *yêu cầu phi chức năng* (các ràng buộc).

Để có thể phát triển được hệ thống phần mềm, trước hết ta phải nắm vững và hiểu được yêu cầu hệ thống. Việc mô tả các yêu cầu của hệ thống đủ tốt là cần thiết để có thể đạt được sự nhất trí giữa khách hàng và những người phát triển về những gì mà hệ thống cần làm, không nên làm và những điều kiện ràng buộc đặt ra cho chúng.

3.1.1. Yêu cầu chức năng

Chức năng của hệ thống là những gì mà hệ thống được yêu cầu thực hiện.

Trong giai đoạn xác định yêu cầu chức năng, các tính chất, các yêu cầu về chất lượng hệ thống như tính hiệu quả, an toàn hệ thống chưa cần xem xét, nghĩa là chưa xét tới các đặc tính phi chức năng của hệ thống.

Các chức năng của hệ thống có thể phân loại thành các phạm trù theo các lĩnh vực chức năng hay theo những mức ưu tiên khác nhau để tránh sự lẫn lộn giữa chúng. Các chức năng hệ thống có thể chia thành hai loại:

- *Những chức năng hiển*: Những chức năng cần thực hiện và NSD có thể nhận biết, theo dõi được sự hoạt động của chúng. *Ví dụ*: khi người bán nhập các mặt hàng mà khách đã chọn mua ở trong giỏ hàng vào hệ thống thì mọi thông tin liên quan đến tên gọi sản phẩm, số lượng, giá bán,... đều phải được hiện lên màn hình và khách hàng có thể theo dõi một cách tường minh.

- *Những chức năng ẩn:* Những chức năng cần thực hiện và NSD không theo dõi được. Thường đó là những chức năng kỹ thuật như những công việc tổ chức lưu trữ, xử lý dữ liệu để đảm bảo sự bền vững dữ liệu trong các CSDL. *Ví dụ*: sau mỗi phiên bán hàng, nghĩa là sau khi khách đã trả đủ tiền mua hàng, hệ thống bán hàng phải thực hiện cập nhật lại số lượng của những mặt hàng vừa bán được. Những hoạt động này NSD không theo dõi được.

Các chức năng của hệ thống phải được chia thành các nhóm theo các mối liên hệ cố kết với nhau. Dựa vào cách phân chia các chức năng để sau này chia nhỏ hệ thống thành các gói, các hệ thống con trong quá trình phân tích và thiết kế hệ thống.

Ví dụ: Các chức năng của hệ thống bán hàng có thể chia thành hai nhóm chính: các *chức năng bán hàng* (các chức năng cơ sở) và *các chức năng thanh toán*.

Dựa trên những kết quả khảo sát bài toán bán hàng, nghiên cứu các sổ sách, tài liệu và trên cơ sở trao đổi với những người bán hàng, với khách hàng,... chúng ta xác định được các chức năng chính của hệ thống như sau [18]:

Bảng 3.1. Những chức năng bán hàng

Quy tắc	Chức năng	Loại
R1.1	Ghi nhận các mặt hàng ở trong giỏ hàng mà khách hàng đã chọn.	Hiển
R1.2	Tính tổng số tiền bán cho khách hàng đang mua.	Hiển
R1.3	Nhập các thông tin về các mặt hàng qua mã vạch bằng máy đọc mã vạch hoặc nhập mã sản phẩm phổ biến (UPC: Universal Product Code) trực tiếp từ bàn phím.	Hiển
R1.4	Cập nhật, trừ bớt số lượng đã bán sau từng phiên bán hàng.	Ẩn
R1.5	Kết thúc một phiên bán hàng.	Ẩn
R1.6	Người bán hàng (cashier) phải login để khởi động hệ thống (cho biết tên ID và password) để sử dụng hệ thống.	Hiển
R1.7	Cung cấp một cơ chế lưu trữ nhất quán, CSDL.	Ẩn
R1.8	Cung cấp cơ chế trao đổi giữa các tiến trình, trao đổi thông tin giữa các hệ thống với nhau.	Ẩn
R1.9	Hiển thị các thông tin mô tả và giá bán các mặt hàng để khách hàng có thể theo dõi được.	Hiển

Bảng 3.2. Những chức năng thực hiện thanh toán với khách hàng

Quy tắc	Chức năng	Loại
R2.1	Thu tiền mặt, nhập số tiền khách đưa và tính số dư phải trả lại cho khách hàng.	Hiển
R2.2	Thu tiền bằng thẻ tín dụng (Credit), nhập thông tin của thẻ tín dụng của khách qua máy đọc thẻ hoặc nhập trực tiếp từ bàn phím.	Hiển
R2.3	Thu tiền bằng séc, nhập số hiệu và số tiền của tờ Séc, tính số dư phải trả lại cho khách.	Hiển

Các chức năng hệ thống thường được đánh số theo các quy tắc tham chiếu (*Reference Rule)* để tiện cho việc sử dụng tham chiếu trong các mục phân tích sau này.

Chú ý: Trong đánh số các mục, phần, hay quy tắc,... chúng ta sử dụng thống nhất quy tắc đánh số dấu chấm (‘.’) như trong các tài liệu vẫn sử dụng.

3.1.2. Yêu cầu phi chức năng

Yêu cầu phi chức năng (Non-functional Requirement) là những ràng buộc, những giới hạn và những yêu cầu kỹ thuật mà người phát triển hệ thống phải tuân theo.

Ví dụ: Hệ thống bán hàng phải có các thuộc tính sau:

- *Thời gian xử lý và trả lời nhanh*, ví dụ: khi nhập vào mã từng mặt hàng (máy đọc mã vạch, hay từ bàn phím) thì các thông tin về sản phẩm, giá bán phải được hiển thị ngay tức thì (sau 5 giây chẳng hạn).

- *Dễ sử dụng với những giao diện đồ họa thân thiện* phù hợp với người bán hàng: như các window và các hộp thoại,...

- *Hệ thống thực hiện trên những hệ điều hành phổ dụng* như Microsoft Window 95, 98, 2000, NT, Unix, Linux,...

- ...

3.2. MÔ HÌNH NGHIỆP VỤ

3.2.1. Nội dung và sản phẩm của pha lập mô hình nghiệp vụ

Để hiểu đúng và đầy đủ hệ thống mà ta cần tin học hóa, sau khi xác định phạm vi và chức năng hệ thống cần nghiên cứu bằng cách liệt kê các chức năng mà hệ thống thực hiện, chỉ ra mối quan hệ của nó với môi trường thông qua việc sử dụng các chức năng của hệ thống, chúng ta cần tìm các ca sử dụng nghiệp vụ từ các chức năng của hệ thống mà qua đó con người và hệ thống khác sử dụng chúng. Mô tả nội dung của các ca sử dụng này để hiểu được nội dung các dịch vụ mà hệ thống cần cung cấp. Các công việc này được trợ giúp bằng việc xây dựng mô hình lĩnh vực và mô hình nghiệp vụ. Sản phẩm của bước này gồm:

- Mô hình lĩnh vực, mô hình nghiệp vụ của hệ thống

- Bảng các thuật ngữ sử dụng (từ điển giải thích)

- Xác định các yêu cầu bổ sung

Từ điển giải thích thường được dùng để xác định các thuật ngữ quan trọng và thông dụng mà người phát triển sử dụng để mô tả hệ thống. Nó đem lại sự thống nhất trong việc định nghĩa các khái niệm và các cách diễn đạt giữa người phát triển, khách hàng và đồng thời làm giảm được sự không nhất quán nói chung trong quá trình phát triển.

3.2.2. Xây dựng mô hình nghiệp vụ

Tùy theo yêu cầu của từng bài toán cụ thể mà ta có thể xây dựng hoặc một mô hình lĩnh vực, hoặc một mô hình nghiệp vụ hoặc cả hai hoặc không cần một mô hình nào cả.

a. Xây dựng mô hình lĩnh vực

Mô hình lĩnh vực (domain model) mô tả các khái niệm quan trọng của hệ thống qua các đối tượng của lĩnh vực nghiệp vụ và mối liên kết giữa chúng. Các đối tượng này là các "vật" hay khái niệm trong thực tế hoặc các sự kiện diễn ra trong môi trường mà hệ thống làm việc. Có 3 dạng lớp đối tượng miền điển hình:

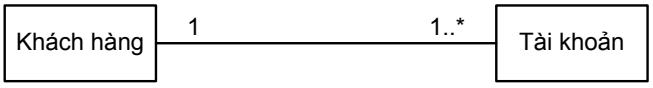
- *Các đối tượng nghiệp vụ* thể hiện những vật được quản lý trong hoạt động nghiệp vụ như đơn đặt hàng, tài khoản, hợp đồng,..

- *Các đối tượng của thế giới thực và các khái niệm* mà một hệ thống cần theo dõi, chẳng hạn giao dịch thanh toán, mua hàng, rút tiền.

- *Các sự kiện sẽ hoặc đã xuất hiện*: đưa thẻ vào máy, nhấn phím, trả tiền.

Mô hình lĩnh vực có thể mô tả bằng nhiều sơ đồ lớp của UML.

Ví dụ:



Hình 3.1. Mô hình lĩnh vực

Từ điển giải thích: ***Tài khoản*** là một loạt các thông tin dành cho một khách hàng, thường bao gồm: một mã số, tên khách hàng, số dư tiền gửi,...

b. Xây dựng mô hình nghiệp vụ

Mô hình ca sử dụng nghiệp vụ mô tả các quá trình nghiệp vụ của một tổ chức dưới dạng *các ca sử dụng nghiệp vụ* và *các tác nhân nghiệp vụ* tương ứng liên kết với nhau. Mô hình này xem xét một hệ thống nghiệp vụ từ quan điểm NSD và chỉ ra cách thức làm thế nào để hệ thống cung cấp được một giá trị gia tăng cho tác nhân nghiệp vụ của nó.

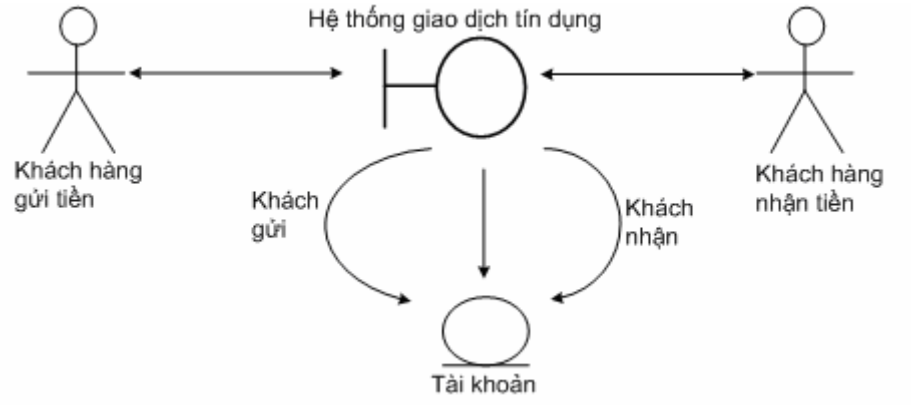
Mô hình ca sử dụng nghiệp vụ được mô tả bằng các sơ đồ ca sử dụng.

Chúng ta xây dựng mô hình nghiệp vụ theo 2 bước:

- Xây dựng một mô hình ca sử dụng nghiệp vụ bao gồm việc xác định các tác nhân nghiệp vụ và các ca sử dụng nghiệp vụ mà các tác nhân sử dụng.

- Phát triển một mô hình đối tượng nghiệp vụ gồm những người tham gia nghiệp vụ, các thực thể nghiệp vụ và các đơn vị công việc cùng nhau thực hiện các ca sử dụng nghiệp vụ. Các quy tắc nghiệp vụ và các điều chỉnh khác trong nghiệp vụ được đặt ra và có liên kết với các đối tượng khác là một bộ phận của mô hình.

Ví dụ:



Hình 3.2. Mô hình nghiệp vụ chuyển tiền từ người gửi sang người nhận

3.2.3. Xác định các yêu cầu bổ sung

Các yêu cầu bổ sung là các yêu cầu phi chức năng chủ yếu mà không thể liên kết với một ca sử dụng cụ thể nào. Ví dụ như các yêu cầu về thể hiện, yêu cầu về giao diện (yêu cầu phần cứng), thiết kế vật lý và kiến trúc, các ràng buộc về thiết kế (mở rộng, bảo trì, sử dụng lại) và cài đặt (mã hóa, chuẩn sử dụng, môi trường, tài nguyên, tích hợp dữ liệu).

Ví dụ: Đối với ca sử dụng rút tiền:

- Những yêu cầu về thực hiện: khi một khách hàng đưa thẻ tín dụng vào để rút tiền, hệ thống cần hồi đáp việc xác minh yêu cầu của khách hàng có hợp lệ hay không trong vòng 1 giây đối với 90% trường hợp. Thời gian cho việc xác minh đó không quá 10 giây nếu như máy không có gì trục trặc. Nếu có trục trặc phải có thông báo.

- Yêu cầu phần cứng: cần có máy chủ loại PC pentium, máy khách PC có bộ vi xử lý tối thiểu là Intel 486.

- Yêu cầu bảo mật: việc giao dịch phải được bảo mật, có nghĩa là chỉ có khách có thẻ mới có thể tiếp cận được với thông tin về thay đổi tài khoản.

3.3. XÁC ĐỊNH YÊU CẦU HỆ THỐNG VÀ MÔ HÌNH CA SỬ DỤNG

Nhiệm vụ chính trong xác định yêu cầu là phát triển một mô hình của hệ thống cần xây dựng bằng cách dùng các ca sử dụng. Điều này là tự nhiên, bởi vì *các yêu cầu về chức năng* được cấu trúc thành các ca sử dụng và do phần lớn các yêu cầu phi chức năng đều là riêng đối với một ca sử dụng đơn nên chúng cũng được xử lý trong ca sử dụng đó. *Các yêu cầu phi chức năng* còn lại mang tính chung cho nhiều hoặc toàn bộ các ca sử dụng thì được đưa vào một tài liệu riêng và được gọi là *các* yêu cầu bổ sung.

3.3.1. Nội dung và sản phẩm của pha xác định yêu cầu

Trong pha xác định yêu cầu, chúng ta cần tìm:

- Các tác nhân và các ca sử dụng để chuẩn bị cho phiên bản đầu tiên của ca sử dụng

- Xác định các ca sử dụng có ý nghĩa về mặt kiến trúc và sắp thứ tự ưu tiên các ca sử dụng

- Mô tả mọi ca sử dụng

- Đưa ra các yếu tố giao diện người dùng thích hợp cho mỗi tác nhân tương tác với các ca sử dụng

- Tái cấu trúc mô hình ca sử dụng bằng cách xác định và đưa ra các tổng quát hóa, các mở rộng hay bao gồm của các ca sử dụng để làm cho mô hình càng rõ ràng và dễ hiểu hơn.

Sản phẩm của pha này gồm có:

- Một phiên bản đầu tiên của mô hình ca sử dụng (sơ bộ và tổng thể)

- Mô tả về ca sử dụng

- Các bản mẫu giao diện người dùng

Các phiên bản của các chế tác trên được mịn dần qua quá trình lặp.

3.3.2. Mô hình ca sử dụng

1. Một số khái niệm

a. Ca sử dụng

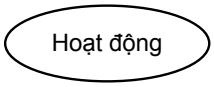
Khái niệm *ca sử dụng, hay trường hợp sử dụng (Use Case)* được Ivan Jacobson đề xuất từ năm 1994 nhằm đặc tả các yêu cầu dịch vụ của hệ thống cho khách hàng và xác định mối quan hệ tương tác giữa hệ thống phần mềm với NSD trong nghiệp vụ. Một cách hình thức hơn, *ca sử dụng mô tả tập các hoạt động của hệ thống theo quan*

điểm của các tác nhân (Actor). Nó đặc tả các yêu cầu của hệ thống và trả lời cho câu hỏi:

Hệ thống phải làm *cái gì* (What?).

Ca sử dụng mô tả một quá trình từ bắt đầu cho đến khi kết thúc, gồm dãy các thao tác, các giao dịch cần thiết để sản sinh ra cái gì đó (giá trị, thông tin) theo yêu cầu của một tổ chức, của tác nhân,...

Ca sử dụng được ký hiệu:



Hình 3.3. Ký hiệu của ca sử dụng

Trong đó, "*Hoạt động"* là các chức năng, nhiệm vụ hay gọi chung là dịch vụ của hệ thống và nó thường được mô tả bằng các *động từ*, hay *mệnh đề động từ đơn,* ví dụ: *bán hàng, thanh toán, khởi động hệ thống,...*

Những ca sử dụng phức tạp sẽ được mô tả chi tiết thông qua các *kịch bản.*

b. Tác nhân ngoài

Tác nhân ngoài, hay gọi ngắn gọn là tác nhân là những thực thể bên ngoài có tương tác với hệ thống, bao gồm người, vật, thiết bị hay các hệ thống khác có trao đổi thông tin với hệ thống. Nói cách khác, tác nhân đại diện cho người hay một bộ phận của tổ chức mong muốn nhận được các thông tin (dữ liệu) hoặc các câu trả lời từ những ca sử dụng tương ứng.

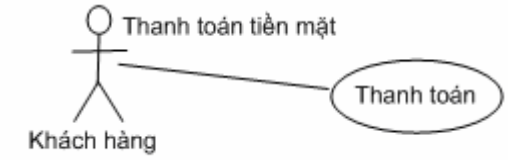
Ví dụ: Khách mua hàng, người bán hàng là hai tác nhân của hệ thống bán hàng.

Ký hiệu của tác nhân là *hình nhân cùng với tên gọi* như hình 3.4.

Hình 3.4. Ký hiệu tác nhân Khách hàng

Tên gọi của tác nhân được mô tả bằng các danh từ (chung) và thường phải đặc tả được vai trò của nó đối với hệ thống.

Tác nhân trao đổi với hệ thống thông qua việc tương tác, sử dụng các dịch vụ của hệ thống là các ca sử dụng bằng cách trao đổi các thông điệp. Như vậy, tác nhân sẽ cung cấp hoặc sử dụng các thông tin của hệ thống thông qua các ca sử dụng.



Hình 3.5. Tác nhân Khách hàng tương tác với ca sử dụng Thanh toán

Thông thường trong mỗi hệ thống: khách hàng, NSD, người quản lý, người phục vụ,... có thể xem như là các tác nhân của hệ thống đó. Chúng ta cũng dễ nhận thấy, một ca sử dụng thì phải được khởi động *bởi/phục vụ cho* một hay nhiều tác nhân.

c. Luồng sự kiện

Ca sử dụng dùng để mô tả cái mà hệ thống sẽ làm. Để tăng độ dễ hiểu, khi xây dựng hệ thống chúng ta cần mô tả chi tiết hơn hành vi của ca sử dụng trong tài liệu văn bản *luồng sự kiện (flow of events)*. Mục đích của luồng sự kiện là làm tài liệu luồng logic đi qua các ca sử dụng. Tài liệu luồng sự kiện mô tả chi tiết người sử dụng sẽ làm gì và hệ thống sẽ làm gì. Tuy là chi tiết nhưng luồng sự kiện vẫn độc lập ngôn ngữ (chưa đề cập đến loại ngôn ngữ lập trình nào được chọn

để cài đặt). Mỗi ca sử dụng có nhiều luồng sự kiện (luồng chính, luồng phụ/rẽ nhánh). Không thể thể hiện mọi chi tiết của ca sử dụng trong một luồng sự kiện.

Kịch bản (scenario) chỉ ra luồng sự kiện trong một *thể hiện (instance)* cụ thể của ca sử dụng. Nó là trình tự hành động cụ thể để mô tả hành vi. Kịch bản đi xuyên suốt ca sử dụng theo nhánh chính hay các nhánh phụ hoặc nhánh đặc biệt của đường đi. Kịch bản là một thể hiện của ca sử dụng, tương tự đối tượng là thể hiện của lớp, nó cho biết cách sử dụng hệ thống. Khách hàng hiểu hệ thống phức tạp thông qua tập kịch bản mô tả hành vi của nó.

Mọi chi tiết của ca sử dụng được mô tả trong các luồng sự kiện chính và luồng rẽ nhánh. Nó mô tả từng bước cái gì sẽ xảy ra để thực hiện các chức năng của ca sử dụng. Luồng sự kiện tập trung vào cái hệ thống sẽ làm chứ không tập trung vào nó làm như thế nào và được mô tả từ góc nhìn của người sử dụng. Luồng sự kiện chính và rẽ nhánh bao gồm:

- Ca sử dụng khởi động như thế nào
- Các đường đi xuyên qua các ca sử dụng
- Luồng chính thông qua ca sử dụng
- Luồng rẽ nhánh thông qua ca sử dụng
- Các luồng lỗi
- Ca sử dụng kết thúc như thế nào.

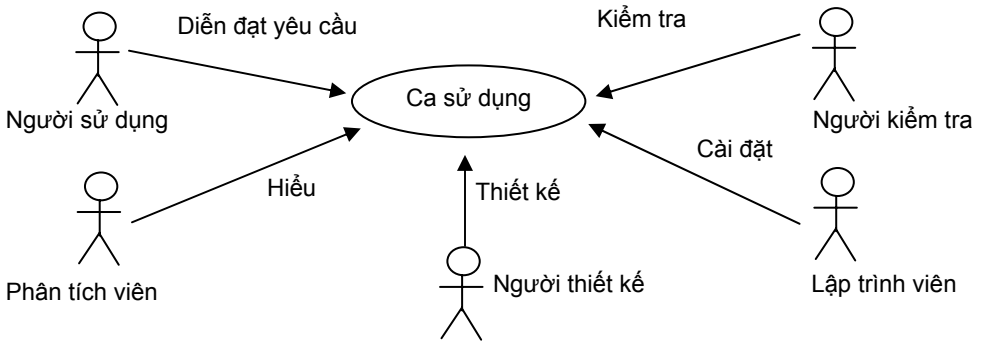
Khi làm tài liệu luồng sự kiện có thể sử dụng các hình thức khác nhau như *mô tả văn bản, biểu đồ luồng (flow charts)*... Bằng cách nào đi nữa thì luồng sự kiện phải phù hợp với yêu cầu của hệ thống. Khách hàng có tài liệu này để khẳng định tính chính xác cái mà họ mong đợi.

2. Mục tiêu và vai trò của ca sử dụng trong cả quá trình phát triển phần mềm

Ca sử dụng dùng để:

- Mô tả các yêu cầu chức năng của hệ thống, là kết quả của quá trình khảo sát, nghiên cứu các yêu cầu của bài toán và những thỏa thuận giữa khách hàng, NSD hệ thống với người phát triển phần mềm.
- Làm cơ sở để người phân tích viên hiểu, người thiết kế xây dựng các kiến trúc, người lập trình cài đặt các chức năng của hệ thống,
- Cung cấp các cơ sở để kiểm duyệt, thử nghiệm hệ thống.

Các ca sử dụng đóng vai trò rất quan trọng trong cả quá trình phát triển phần mềm, tất cả các pha phân tích, thiết kế sau này đều dựa vào các ca sử dụng. Như vậy, quá trình được hướng dẫn bởi ca sử dụng là một cách hữu hiệu để mô hình hóa hệ thống với UML. Hình 3.6 chỉ cho chúng ta thấy ai sẽ cần đến ca sử dụng và cần để làm gì. Người sử dụng phải nêu được các yêu cầu của hệ thống, phân tích viên phải hiểu được các công việc của hệ thống, người thiết kế (kiến trúc sư) phải đưa ra được các thành phần để thực hiện các ca sử dụng, người lập trình thực hiện cài đặt chúng và cuối cùng nhân viên kiểm tra hệ thống dựa vào những ca sử dụng đó.



Hình 3.6. Vai trò của ca sử dụng trong quá trình phát triển phần mềm

3. Tìm các tác nhân

Tác nhân là một bộ phận bên ngoài hệ thống nhưng cộng tác chặt chẽ với hệ thống. Nó chính là đối tượng mà hệ thống phục vụ hoặc cần có để cung cấp dữ liệu. Do đó, nhiệm vụ trước tiên của người phân tích là xác định các tác nhân.

Một trong các kỹ thuật hỗ trợ để xác định các tác nhân là dựa trên các câu trả lời những câu hỏi sau:

- ♦ Ai sẽ sử dụng các chức năng chính của hệ thống?
- ♦ Ai cần sự hỗ trợ của hệ thống để thực hiện các công việc hàng ngày?
- ♦ Ai quản trị, bảo dưỡng để đảm bảo cho hệ thống hoạt động thường xuyên?
- ♦ Hệ thống quản lý, sử dụng những thiết bị nào?
- ♦ Hệ thống cần tương tác với những bộ phận, hệ thống nào khác?
- ♦ Ai hay cái gì quan tâm đến kết quả xử lý của hệ thống?

4. Tìm các ca sử dụng

Có hai phương pháp chính hỗ trợ giúp ta xác định các ca sử dụng:

Phương pháp thứ nhất là dựa vào các tác nhân:

+ Xác định những tác nhân liên quan đến một hệ thống hoặc đến một tổ chức, nghĩa là tìm và xác định những tác nhân là NSD hay những hệ thống khác tương tác với hệ thống cần xây dựng.

+ Với mỗi tác nhân, tìm những tiến trình (chức năng) được khởi đầu, hay giúp các tác nhân thực hiện, giao tiếp/tương tác với hệ thống.

Phương pháp thứ hai để tìm các ca sử dụng là dựa vào các sự kiện kích hoạt ca sử dụng.

+ Xác định những sự kiện bên ngoài có tác động đến hệ thống hay hệ thống phải trả lời.

+ Tìm mối liên quan giữa các sự kiện và các ca sử dụng.

Tương tự như trên, hãy trả lời những câu hỏi sau đây để tìm ra các ca sử dụng:

- Nhiệm vụ chính của các tác nhân là gì?

- Tác nhân cần phải đọc, ghi, sửa đổi, cập nhật, hay lưu trữ thông tin hay không?

- Những thay đổi bên ngoài hệ thống thì tác nhân có cần phải thông báo cho hệ thống hay không?

- Những tác nhân nào cần được thông báo về những thay đổi của hệ thống?

- Hệ thống cần có những đầu vào/ra nào?, từ đâu và đến đâu?

Ví dụ: Dựa vào các phương pháp nêu trên, chúng ta xác định được các tác nhân và các ca sử dụng của hệ thống bán hàng. Các tác nhân gồm có:

+ *Khách hàng (Customer)*: là những người được hệ thống bán hàng phục vụ.

+ *Người bán hàng (Cashier):* những người cần sử dụng chức năng bán hàng của hệ thống để thực hiện nhiệm vụ của mình.

+ *Người quản lý (Manager):* những người được phép khởi động (*Start Up*) hay kết thúc cả hệ thống (*Shut Down*) tại các điểm bán hàng đầu cuối.

+ *Người quản trị hệ thống (System Administrator):* có thể bổ sung, thay đổi những NSD.

+ *Bộ phận kiểm duyệt séc và thẻ tín dụng*: thực hiện nhiệm vụ kiểm duyệt séc/thẻ khi thanh toán.

Các ca sử dụng gồm có:

- *Bán hàng, mua hàng (Buy Items)* là nhiệm vụ của hệ thống bán hàng liên quan trực tiếp tới *khách hàng và người bán hàng.* Trong trường hợp này, hai chức năng *bán hàng* và *mua hàng* là đồng nghĩa, nên có thể chọn một trong hai chức năng đó. Ca sử dụng này liên quan đến cả *người bán hàng* và *khách hàng.*

- *Thanh toán, trả tiền mua hàng hay thu tiền (Refund Items, Cash Out):* là chức năng mà hệ thống phải thực hiện để thanh toán với khách hàng bằng phương thức mà họ lựa chọn: trả tiền mặt, thẻ tín dụng, hay trả bằng séc. Ca sử dụng này cũng liên quan đến cả *người bán hàng* và *khách hàng.*

- *Đăng nhập hệ thống (Log In):* Người bán hàng cần sử dụng để nhập vào hệ thống và sử dụng nó để bán hàng.

- *Khởi động (Start Up), Đóng hệ thống (Shut Down):* Người quản lý thực hiện để khởi động hay kết thúc hoạt động của hệ thống.

- *Bổ sung NSD mới (Add New Users), Loại bỏ NSD (Remove User):* Người quản trị hệ thống có thể bổ sung thêm người sử dụng mới hay loại bỏ những NSD không còn cần sử dụng hệ thống.

Sau khi xác định được các tác nhân và các ca sử dụng thì phải đặt lại tên cho chúng. Tên của các tác nhân và ca sử dụng phải đơn giản, rõ nghĩa và phù hợp với lĩnh vực của bài toán ứng dụng.

- Tên của tác nhân phải là *danh từ chung* và biểu hiện được vai trò của nó trong các mối quan hệ với hệ thống.
- Tên của ca sử dụng phải *bắt đầu bằng động từ, là mệnh đề đơn,* ngắn gọn và mô tả đúng nhiệm vụ mà hệ thống cần thực hiện.

Bảng 3.3. Danh sách các tác nhân và ca sử dụng trong hệ thống bán hàng

Tác nhân	Ca sử dụng
Người bán hàng (Cashier)	Đăng nhập hệ thống (Log In) Thu tiền bán hàng (Cash Out)
Khách hàng (Customer)	Mua hàng (Buy Items) Thu tiền, thanh toán (Refund Items)
Người quản lý (Manager) Hay gian hàng trưởng	Khởi động hệ thống (Start Up) để người bán hàng có thể sử dụng để bán hàng. Đóng hệ thống khi hết giờ (Shut Down)
Quản trị hệ thống (System Administrator)	Bổ sung NSD (Add New Users) Loại bỏ NSD (Delete User)

Lưu ý:

Các chức năng *mua hàng* và *bán hàng* tương ứng với *khách hàng* hay *người bán*, nhưng trong hệ thống bán hàng ta có thể sử dụng một tên gọi chung là *bán hàng.* Tương tự, *Thu tiền bán hàng* và *Thu tiền* có thể đồng nhất là *Thu tiền hoặc Thanh toán.*

Sau khi đã xác định xong các ca sử dụng cho hệ thống thì phải kiểm tra lại xem chúng đã được phát hiện hết chưa. Công việc này được thực hiện bằng cách trả lời cho các câu hỏi sau:

- Mỗi yêu cầu chức năng của hệ thống đã có trong ít nhất một ca sử dụng chưa? Những chức năng không được mô tả trong các ca sử dụng sẽ không được cài đặt sau này.

- Các mối tương tác giữa các tác nhân và hệ thống đã xác định hết chưa?

- Tác nhân cung cấp những gì cho hệ thống?

- Tác nhân nhận được những gì từ hệ thống?

- Đã nhận biết được tất cả các hệ thống bên ngoài có tương tác với hệ thống chưa?

- Những thông tin nào mà hệ thống ngoài gửi tới hoặc nhận được từ hệ thống?

Một lần nữa cần duyệt lại xem đã xác định đầy đủ, chính xác các tác nhân, ca sử dụng và mối quan hệ của chúng.

5. Mô tả ca sử dụng

a. Mô tả mỗi ca sử dụng theo khuôn dạng hoặc kịch bản

* Đối với những ca sử dụng đơn giản, có thể mô tả chúng theo các *mẫu khuôn dạng đặc tả*.

Khuôn dạng (*Format*) đặc tả ca sử dụng có các thành phần:

- *Ca sử dụng*: Tên của ca sử dụng bắt đầu bằng động từ.

- *Các tác nhân:* Danh sách các tác nhân liên quan đến ca sử dụng, chỉ rõ ai bắt đầu với ca sử dụng này.

- *Mục tiêu:* nêu rõ chức năng hệ thống mà ca sử dụng đảm nhận

- *Mô tả:* Mô tả tóm tắt tiến trình xử lý công việc cần thực hiện.

- *Tiền điều kiện (pre-conditions):* các điều kiện cần được thực hiện trước khi ca sử dụng khởi động. Không phải ca sử dụng nào cũng có tiền điều kiện.

- *Hậu điều kiện (post-conditions):* các điều kiện được thực hiện ngay sau khi ca sử dụng kết thúc. Nó mô tả trạng thái hệ thống hay tác nhân sau khi kết thúc ca sử dụng. Không phải ca sử dụng nào cũng có hậu điều kiện.

- *Tham chiếu:* Các chức năng, ca sử dụng và những hệ thống liên quan.

- *Yêu cầu đặc biệt* (hiệu năng của hệ thống cần có): các yêu cầu phi chức năng cụ thể.

Ví dụ: ***Ca sử dụng: Mua hay bán hàng***

Tác nhân: Khách hàng, người bán hàng

Mục tiêu: Thực hiện được một giao dịch là mua hàng

Mô tả: Khách hàng sau khi đã chọn đủ các mặt hàng cần mua để ở trong giỏ hàng thì đưa hàng đến quầy thu tiền. Người bán hàng lần lượt ghi nhận các mặt hàng trong giỏ hàng của khách và thu tiền. Sau khi thanh toán xong khách hàng được mang số hàng đã mua đi ra khỏi cửa hàng.

Tham chiếu tới: Các chức năng R1.1, R1.2, R1.3, R1.6, R1.7, R1.8, R2.1, R2.2, R2.3.

Tiền điều kiện: khách hàng đã quyết định mua những mặt hàng đã bỏ vào giỏ hàng và các mặt hàng mà khách đã chọn đều được xác định giá.

Hậu điều kiện: thể hiện giao dịch mua hàng kết thúc. Số lượng tồn kho các loại mặt hàng khách lấy giảm đi tương ứng với lượng hàng khách đã lấy, lượng tiền hệ thống thu được trong ngày tăng lên tương ứng với lượng tiền khách đã trả.

Yêu cầu đặc biệt: thành tiền mỗi loại mặt hàng thanh toán cũng như tổng tiền phải hiện sau 5 giây mỗi khi người bán hàng "chuyển dòng" hoặc bấm phím "kết thúc"

* Đối với những ca sử dụng phức tạp, ta sẽ mô tả chi tiết chúng thông qua các *kịch bản* nhằm mô tả những *con đường cơ bản (luồng công việc chính)*. Sau đó mô tả những luồng ngoại lệ/rẽ nhánh.

Ví dụ: ***Luồng sự kiện của ca sử dụng rút tiền trong hệ thống rút tiền tự động.***

+ Luồng chính:

1. Ca sử dụng bắt đầu khi khách hàng đặt thẻ tín dụng vào máy ATM.

2. ATM hiển thị thông báo và chờ khách hàng nhập số PIN

3. Khách hàng nhập PIN

4. ATM khẳng định PIN hợp lệ? Nếu PIN không hợp lệ thì thực hiện luồng nhánh A1

5. ATM hiển thị các lựa chọn:

+ Gửi tiền vào tài khoản

+ Rút tiền

+ Chuyển tiền sang tài khoản khác

6. Khách hàng chọn rút tiền

7. ATM hiển thị câu hỏi số tiền sẽ rút

8. Khách hàng nhập số tiền muốn rút

9. ATM xác định số dư còn đủ? Nếu không đủ tiền: thực hiện luồng nhánh A2. Nếu phát hiện lỗi khi kiểm tra số dư: thực hiện luồng nhánh A3.

10. ATM trừ số tiền trong tài khoản của khách hàng

11. ATM chuyển tiền cho khách hàng

12. ATM in biên nhận

13. ATM trả lại thẻ tín dụng

14. Ca sử dụng kết thúc.

Luồng nhánh A1- Số PIN không hợp lệ:

1. ATM thông báo số PIN không hợp lệ

2. ATM trả lại thẻ tín dụng

3. Kết thúc ca sử dụng.

Luồng nhánh A2: không đủ tiền trong tài khoản để rút:

1. ATM thông báo không đủ tiền

2. ATM trả lại thẻ tín dụng

3. Kết thúc ca sử dụng.

Luồng lỗi A3: lỗi xảy ra khi kiểm tra số dư không đủ

1. ATM thông báo lỗi

2. ATM ghi thông báo này vào tệp chứa lỗi (error log)

3. ATM trả lại thẻ tín dụng

4. Kết thúc ca sử dụng.

Có thể xây dựng những kịch bản khác hoặc những kịch bản con để hiểu và nắm bắt đầy đủ được mọi yêu cầu của hệ thống.

Chúng ta xây dựng các *kịch bản* hành động để mô tả các sự kiện xảy ra trong hệ thống. Mỗi kịch bản có thể mô tả theo hai luồng: luồng thực hiện của các tác nhân và luồng ứng xử của hệ thống.

Ví dụ:

Bảng 3.4. Kịch bản thực hiện đối với ca sử dụng "Bán hàng"

Hành động của các tác nhân	Hành động của hệ thống
1. Khách hàng sau khi chọn đủ số hàng cần thiết thì đưa hàng đã chọn đến cho quầy thu tiền	
2. Người bán ghi nhận từng mặt hàng. Nếu một mặt hàng mua với số lượng nhiều hơn thì người bán nhập số lượng đó vào từ bàn phím.	3. Xác định giá và các thông tin về sản phẩm được hiển thị.
4. Khi đã nhập xong các mặt hàng của khách đã chọn mua thì người bán phải chỉ cho hệ thống biết là đã kết thúc phiên bán hàng bằng cách nhấn phím Enter hoặc nhấn nút "Kết thúc" phiên bán hàng (EndSale).	
	5. Tính và hiển thị tổng số tiền bán hàng.
6. Người bán thông báo cho khách hàng biết tổng số tiền phải trả.	
7. Khách hàng chọn phương thức thanh toán: - Nếu chọn trả tiền mặt: xem tiếp kịch bản con (Sub_scenario) *Thanh toán tiền mặt*. - Nếu trả bằng thẻ tín dụng: xem kịch bản con *Thanh toán bằng thẻ tín dụng*. - Nếu trả tiền séc: xem kịch bản con *Thanh toán bằng Séc*.	8. Hiển thị số tiền dư phải trả cho khách hàng
	9. Kết thúc một phiên giao dịch bán hàng.
	10. Cập nhật lại các hàng trong cửa hàng.
	11. Phát sinh phiếu bán hàng (hóa đơn).
12. Người bán trả tiền thừa và đưa phiếu bán hàng cho khách hàng.	
13. Khách hàng ra khỏi cửa hàng với hàng đã thanh toán.	

Kèm theo mỗi kịch bản, chúng ta mô tả cả các luồng rẽ nhánh:

- Nếu khách hàng không trả đủ tiền, hoặc khi thẻ tín dụng, séc không hợp lệ thì hủy bỏ phiên giao dịch đó.

- Nếu tất cả các loại hàng trong giỏ không xác định được giá thì phiên giao dịch đó cũng bị loại bỏ.

b. Mô tả mô hình ca sử dụng tổng thể

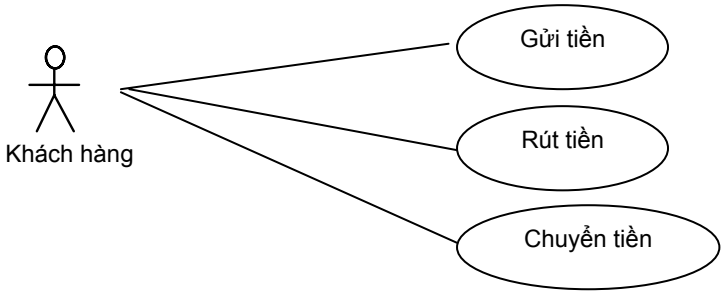
Chúng ta sử dụng sơ đồ để mô tả mô hình ca sử dụng tổng thể. Mô hình ca sử dụng có thể tổ chức thành từng cụm ca sử dụng gọi là *gói ca sử dụng*.

Để đảm bảo tính nhất quán khi mô tả nhiều ca sử dụng, cần xây dựng *từ điển thuật ngữ*. Kết quả của bước này là một *mô tả tổng quan của mô hình ca sử dụng*. Nó mô tả các tác nhân, các ca sử dụng tương tác với nhau như thế nào và các ca sử dụng liên kết với nhau như thế nào. Sau khi hoàn tất mô hình ca sử dụng, cần thẩm định nó theo các tiêu chí sau:

- Mọi yêu cầu cần thiết về chức năng đã được nắm bắt thành các ca sử dụng chưa?

- Chuỗi các hành động là đúng dần, đầy đủ và có thể hiểu được đối với mỗi ca sử dụng chưa?

Bất kỳ các ca sử dụng nào cũng đều phải cung cấp ít nhất một giá trị gia tăng cho tác nhân. Nếu không thì các ca sử dụng này đều phải xem lại.



Hình 3.7. Mô hình ca sử dụng tổng thể

Việc mô tả tổng quan của các ca sử dụng có thể ở dạng văn bản hay dạng bảng.

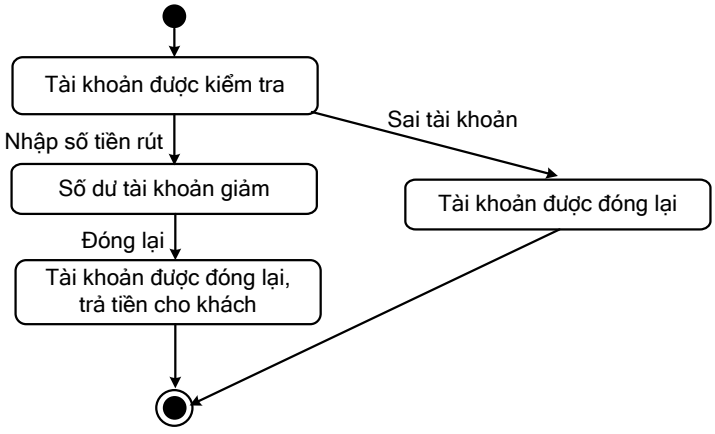
Bảng 3.5. Mô tả tổng quan các ca sử dụng

Các tác nhân	Khách hàng
Các ca sử dụng	3 ca sử dụng: gửi tiền, rút tiền và chuyển tiền
Mô tả nội dung	Nội dung
A. Ca sử dụng gửi tiền	Khách hàng có thẻ tín dụng đưa thẻ vào kích hoạt hệ thống. Hệ thống hiện ra cửa sổ cho phép khách hàng nhập vào định danh của mình. Nếu định danh đúng, khách hàng chọn chức năng gửi tiền, nhập vào số tài khoản và số tiền muốn gửi làm kích hoạt máy nhận tiền. Khách hàng đưa số tiền muốn gửi vào máy. Máy nhận hết tiền và trả lại thẻ.
B. Ca sử dụng rút tiền	Giống như ca sử dụng gửi tiền. Chỉ khác ở chỗ chọn chức năng rút tiền và thay bằng việc đưa tiền vào thì hệ thống đưa tiền ra cho khách.
C. Ca sử dụng chuyển tiền	Giống như ca sử dụng gửi tiền. Chỉ khác ở chỗ chọn chức năng chuyển tiền là xong, sau đó không cần làm gì thêm.

c. Hình thức hóa mô tả ca sử dụng

Đối với ca sử dụng phức tạp, có nhiều trạng thái và sự chuyển tiếp, các mô tả ca sử dụng và bảng gặp khó khăn, thì người ta có thể sử dụng các sơ đồ khác như sơ đồ trạng thái, sơ đồ hoạt động, sơ đồ tương tác để biểu diễn trực quan. Đôi khi người ta dùng cả hai cách để bổ sung cho nhau.

Ví dụ: Sử dụng sơ đồ trạng thái để mô tả hình thức ca sử dụng



Hình 3.8. Sơ đồ trạng thái của tài khoản cho ca sử dụng ***rút tiền***

6. Sắp thứ tự ưu tiên các ca sử dụng

Cần xác định các ca sử dụng quan trọng nhất, mang ý nghĩa về mặt kiến trúc để xây dựng một khung kiến trúc sơ bộ, từ đó quyết định xem cái nào cần được triển khai ngay (tức là được phân tích, được thiết kế, được thực thi,..) trong những lần lặp đầu và cái nào có thể để lại bố trí trong những lần lặp sau.

Ví dụ: Trong hệ thống giao dịch tín dụng, rút tiền là một ca sử dụng quan trọng vì thiếu nó thì hệ thống sẽ mất tính thực. Các ca sử dụng chuyển tiền và gửi tiền mang tính ít quan trọng hơn đối với một khách hàng, vì nhu cầu thường xuyên là rút tiền. Do vậy để xác định kiến trúc, ca sử dụng rút tiền sẽ được cài đặt đầy đủ ở giai đoạn khởi thảo.

7. Thực hiện ca sử dụng hiệu quả - tạo bản mẫu giao diện người dùng

Mục đích của hoạt động này là xây dựng được bản mẫu giao diện người dùng. Đây là công việc thiết kế giao diện người dùng logic. Sau này chúng ta còn phải thiết kế giao diện người dùng thực và phát triển các bản mẫu để minh họa cách thức mà người dùng có thể sử dụng hệ thống.

Khi các tác nhân tương tác với hệ thống, họ sẽ dùng và thao tác qua *yếu tố giao diện người dùng* mà thể hiện ra qua các thuộc tính (của các ca sử dụng). Vì vậy ta cần xác định và đặc tả các yếu tố giao diện người dùng này.

Để xác định các yếu tố giao diện người dùng cần có trong mỗi ca sử dụng mà tác nhân có thể truy nhập được, ta có thể đặt các câu hỏi sau:

- Cái nào trong số các lớp lĩnh vực, các thực thể nghiệp vụ, hoặc các đơn vị công việc là thích hợp để làm các yếu tố giao diện người dùng cho ca sử dụng này?

- Những yếu tố giao diện người dùng mà tác nhân phải làm việc với chúng?

- Những hành động nào mà tác nhân có thể gọi đến và những quyết định nào tác nhân có thể tạo ra?

- Hướng dẫn nào và các thông tin nào mà tác nhân cần, trước khi gọi đến mỗi hành động trong ca sử dụng

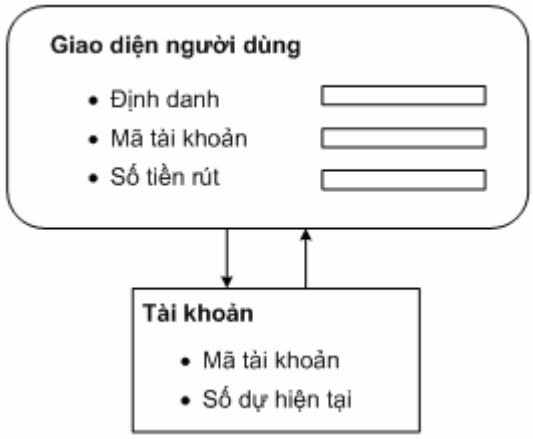
- Thông tin gì mà tác nhân cần cung cấp cho hệ thống

- Thông tin gì mà hệ thống cần cung cấp cho tác nhân

- Các giá trị trung bình đối với mọi tham số đầu vào/đầu ra là gì?

Như vậy, việc thiết kế giao diện người dùng đảm bảo rằng người dùng có thể truy cập vào mỗi ca sử dụng thông qua các yếu tố giao diện người dùng của nó và cũng cần đảm bảo rằng, một tác nhân phải có một giao diện người dùng được tích hợp tốt, dễ sử dụng và nhất quán để truy nhập đến toàn bộ các ca sử dụng cần thiết.

Ví dụ:



*Hình 3.9. Các yếu tố giao diện người dùng ứng với lớp **Tài khoản** trong ca sử dụng **rút tiền***

8. Cấu trúc mô hình ca sử dụng

Mục đích cấu trúc mô hình ca sử dụng là tìm ra các mô tả chức năng có đặc tính chung hay được chia sẻ trong nhiều ca sử dụng để tách ra mô tả riêng, tìm ra các chức năng phụ hoặc tùy chọn để mở rộng mô tả chức năng thành các chức năng mới. Bằng cách cấu trúc

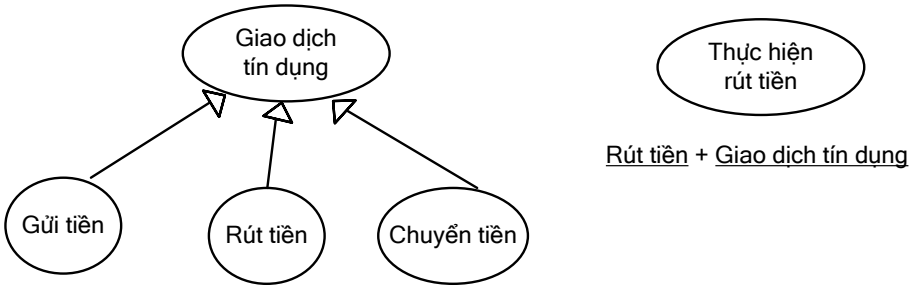
lại toàn bộ các ca sử dụng với những thao tác nêu trên làm cho mô hình trở nên dễ hiểu và dễ làm việc với nó.

a. Mô tả chức năng chung

Khi nhận dạng và phác thảo các hành động của mỗi ca sử dụng, chúng ta cần phải tìm các hành động và các phần của hành động mà là chung hoặc được chia sẻ cho nhiều ca sử dụng. Phần chung này có thể được tách ra và được mô tả trong một ca sử dụng riêng biệt để có thể sử dụng lại. Chúng ta chỉ ra mối quan hệ tái sử dụng này bằng một sự tổng quát hóa.

Ca sử dụng tổng quát hóa là ca sử dụng trừu tượng, tức là nó không thể tự hoạt động, nhưng một thể hiện của ca sử dụng cụ thể có thể biểu diễn hành vi do các ca sử dụng trừu tượng đặc tả mà nó (tái) sử dụng. Ta gọi thể hiện này là ca sử dụng "thực" mà tác nhân nhận thức được khi tương tác với hệ thống. Các ca sử dụng "thực" có được nhờ việc áp dụng các quan hệ tổng quát hóa và mở rộng (hay bao gồm) đối với các ca sử dụng trong mô hình. Các ca sử dụng mới đem lại giá trị cho người dùng.

Ví dụ:



Hình 3.10. Quan hệ tổng quát hóa (bên trái) và Ca sử dụng "thực" được tạo thành từ các ca sử dụng giao dịch tín dụng và ca sử dụng rút tiền (bên phải)

b. Mô tả mối quan hệ giữa các ca sử dụng

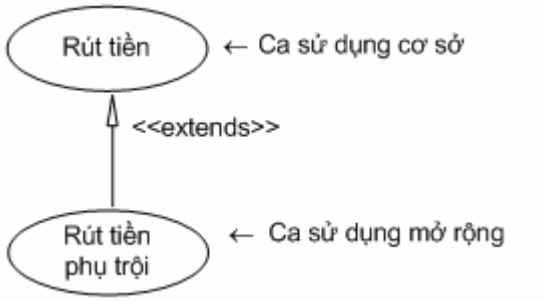
Sơ đồ ca sử dụng chỉ ra mối quan hệ giữa các tác nhân và các ca sử dụng trong hệ thống. Mỗi ca sử dụng cần phải biểu diễn trọn vẹn một giao dịch giữa NSD và hệ thống. Giữa các ca sử dụng có ba quan

hệ chính: quan hệ "mở rộng", quan hệ "sử dụng" và quan hệ theo nhóm hay theo "gói".

+ *Quan hệ mở rộng*

Trong khi xây dựng những ca sử dụng, ta nhận thấy có những ca sử dụng lại được sử dụng như là một phần của chức năng khác. Trong những trường hợp như thế ta nên xây dựng một ca sử dụng mới mở rộng một hay nhiều ca sử dụng đã xác định trước. Ca sử dụng mới được gọi là *ca sử dụng mở rộng* của những ca sử dụng cũ.

Quan hệ mở rộng bao gồm: một điều kiện cho việc mở rộng, một tham chiếu tới điểm cần mở rộng trong ca sử dụng đích (điểm mà mở rộng cần được thực hiện). Mối quan hệ mở rộng giữa các ca sử dụng được mô tả và ký hiệu giống như quan hệ tổng quát hóa với nhãn của quan hệ là <<extends>>. Ví dụ:



Hình 3.11. Mối quan hệ mở rộng giữa các ca sử dụng

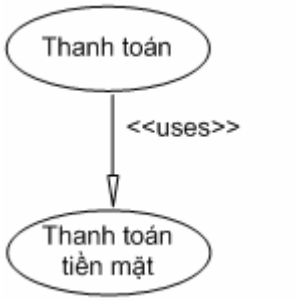
Điều kiện ở đây là: số tiền rút vượt quá số dư tài khoản. Ca sử dụng này tham chiếu đến trường hợp trong ca sử dụng ***rút tiền*** đã mô tả luồng sự kiện về rút quá tiền có trong tài khoản (hệ thống gợi ý chỉ được rút số tiền tối đa có thể là bao nhiêu).

Ca sử dụng *B mở rộng <<extends>> A* nếu: B là biến thể của A, nó chứa thêm một số sự kiện cho những điều kiện nào đó.

+ *Quan hệ sử dụng*

Khi một số ca sử dụng có một hành vi chung thì hành vi này có thể xây dựng thành một ca sử dụng để có thể được sử dụng trong

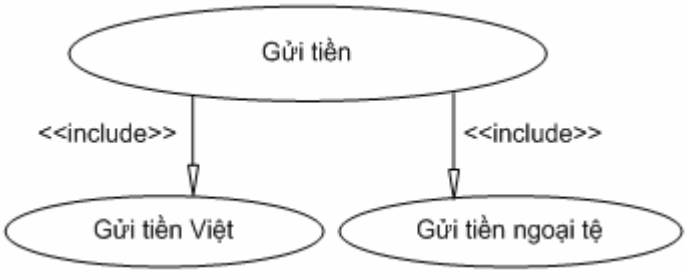
những ca sử dụng khác. Mối quan hệ như thế được gọi là *mối quan hệ sử dụng*. Nói cách khác, trong mối quan hệ sử dụng, có một ca sử dụng dùng ca sử dụng khác để chỉ ra dạng chuyên biệt hóa ca sử dụng cơ sở. Giống như mối quan hệ mở rộng, mối quan hệ sử dụng cũng sử dụng ký hiệu tổng quát hóa để thể hiện với nhãn <<uses>>. Ví dụ: hình 3.12 minh họa về quan hệ sử dụng giữa các ca sử dụng.



Hình 3.12. Quan hệ sử dụng giữa các ca sử dụng

Để thực hiện được ca sử dụng *Thanh toán* thì phải gọi tới ca sử dụng *Thanh toán tiền mặt* khi khách hàng chọn phương thức trả bằng tiền mặt.

Trong UML 1.3, quan hệ sử dụng <<uses>> được gọi là quan hệ *bao gồm* <<includes>>.



Hình 3.13. Quan hệ "bao gồm"

+ *Mối quan hệ gộp nhóm*

Khi có một số ca sử dụng cùng xử lý những chức năng giống nhau hoặc có quan hệ với ca sử dụng khác theo cùng một cách thì tốt nhất là nhóm chúng lại thành từng *gói chức năng*.

Ví dụ: Hệ thống bán hàng có thể chia các ca sử dụng thành các gói *Bán hàng, gói Thanh toán* và *gói Quản lý hệ thống,...*

9. Xây dựng sơ đồ ca sử dụng

a. Một số chú ý trước khi xây dựng sơ đồ ca sử dụng:

- Mỗi ca sử dụng luôn được ít nhất một tác nhân kích hoạt một cách trực tiếp hay gián tiếp, nghĩa là giữa tác nhân và ca sử dụng thường có mối quan hệ giao tiếp (kết hợp).
- Ca sử dụng cung cấp các thông tin cần thiết cho tác nhân và ngược lại, tác nhân cung cấp dữ liệu đầu vào cho ca sử dụng thực hiện.

b. Các bước xây dựng sơ đồ ca sử dụng:

Bước 1. Xác định các tác nhân ngoài

Bước 2. Xác định các ca sử dụng

Bước 3. Xác định các mối quan hệ giữa các ca sử dụng

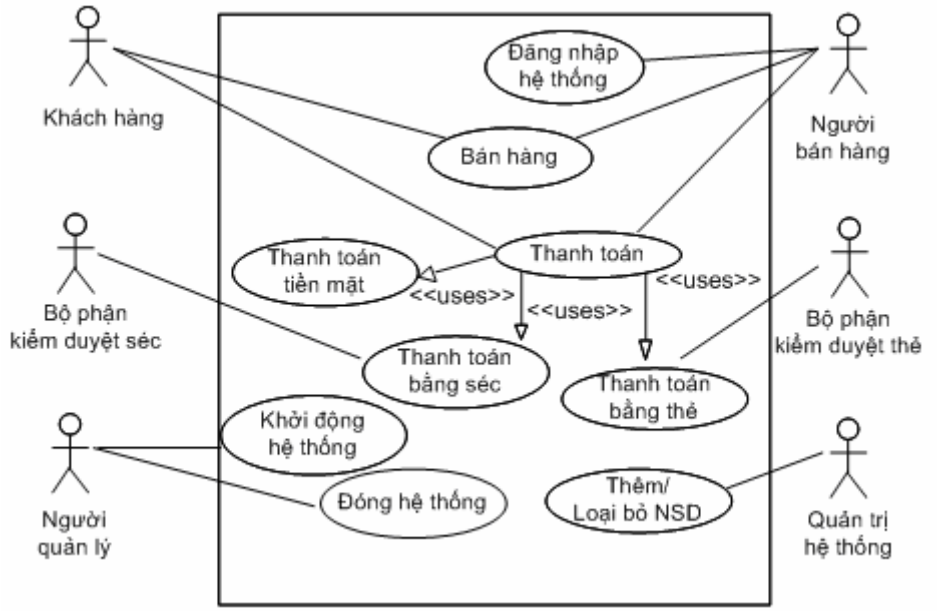
Bước 4. Vẽ sơ đồ ca sử dụng theo cách vẽ các tác nhân ngoài ở xung quanh, các ca sử dụng ở trung tâm cùng với các mối quan hệ giữa các ca sử dụng đó, vẽ đường nối thể hiện tương tác giữa các tác nhân ngoài và các ca sử dụng.

c. Một số điểm cần chú ý khi xây dựng sơ đồ ca sử dụng:

- Giữa các tác nhân với nhau không có quan hệ giao tiếp với nhau. Tác nhân nằm ngoài hệ thống, do vậy giao tiếp giữa các tác nhân cũng nằm ngoài hệ thống cần xây dựng.
- Giữa các ca sử dụng không có quan hệ trực tiếp, trừ mối quan hệ mở rộng *<<extends>>* hoặc *<<uses>>* như đã nói ở trên.
- Mỗi ca sử dụng phải được một tác nhân khởi động hoặc phục vụ cho một tác nhân, ngoại trừ trường hợp những ca sử dụng mở rộng hay ca sử dụng theo quan hệ sử dụng.

- Mỗi phần tử của sơ đồ ca sử dụng (ca sử dụng và tác nhân) khi cần thiết có thể mô tả chi tiết trong phần đặc tả (Documentation của Rose) hay mô tả trong một tệp riêng sau đó gán tương ứng cho phần tử đó trong mô hình ca sử dụng.
- Khi hệ thống lớn, phức tạp thì nên chia các ca sử dụng thành các *gói,* trong đó nhóm những ca sử dụng và các tác nhân có quan hệ với nhau để tạo ra một hệ thống con.

Ví dụ:



Hình 3.14. Sơ đồ ca sử dụng của hệ thống bán hàng

4

MÔ HÌNH PHÂN TÍCH ĐỐI TƯỢNG

Sự khác biệt chính giữa phân tích hướng đối tượng và phân tích có cấu trúc là sự phân rã hệ thống thành các khái niệm (đối tượng) thay vì phân rã thành các chức năng. Điểm mấu chốt trong cách phân rã này là *phải tìm những cái gì (thực thể) hoạt động trong hệ thống* để sau đó quyết định thiết kế và cài đặt chúng nhằm thực hiện các chức năng (yêu cầu) của bài toán. Các đối tượng trong hệ thống cần phải được phân biệt với nhau và thuộc tính để phân biệt các đối tượng trong hệ thống chính là *định danh* (*Identity*).

Trong chương 2, chúng ta đã biết đối tượng là cái gì đó tồn tại trong thế giới thực và cũng có thể chỉ là những khái niệm trừu tượng nhưng điều quan trọng là chúng đều có liên quan đến hiểu biết của chúng ta về thế giới thực. Đối tượng là thể hiện, là một đại biểu của một lớp. Lớp là một mô tả về một kiểu (một tập) các đối tượng có những thuộc tính giống nhau, có chung các hành vi ứng xử, có cùng mối liên quan với các đối tượng của các lớp khác và có chung ngữ nghĩa trong hệ thống.

Chúng ta cũng đã biết, quá trình phát triển phần mềm hướng đối tượng với UML được hướng dẫn bởi các ca sử dụng, nên các lớp đối tượng chủ yếu cũng sẽ được xác định dựa trên cơ sở phân tích các ca sử dụng của hệ thống.

4.1. XÁC ĐỊNH CÁC PHẦN TỬ TRONG MÔ HÌNH KHÁI NIỆM

Trong UML, *mô hình khái niệm của một hệ thống* biểu diễn các thành phần (các đối tượng) của hệ thống và các mối quan hệ giữa chúng. Nó được mô tả bởi sơ đồ lớp và còn được gọi là *mô hình đối tượng*. Vấn đề quan trọng nhưng cũng rất khó khăn là xác định đầy đủ và chính xác các lớp đối tượng và mối quan hệ của chúng trong hệ thống cần phát triển.

4.1.1. Xác định đối tượng

Cách tốt nhất để tìm ra đối tượng là khảo sát danh từ trong luồng sự kiện hay tìm trong tài liệu kịch bản. Kịch bản là phiên bản cụ thể của luồng sự kiện.

Ví dụ: Một luồng sự kiện của ca sử dụng rút tiền là ai đó rút tiền. Một kịch bản của nó có thể là ông Lê rút 1 triệu đồng. Kịch bản khác có thể là bà Trần rút 2 triệu đồng nhưng lại nhập nhầm mật mã (PIN). Kịch bản khác có thể là chị Hà rút 3 triệu nhưng số dư tài khoản của chị chỉ còn 2 triệu. Thông thường có nhiều kịch bản cho một luồng sự kiện. Một sơ đồ trình tự hay sơ đồ cộng tác sẽ mô tả một trong các kịch bản này. Nếu tìm ra danh từ trong kịch bản thì một số trong đó là tác nhân, một số khác có thể là đối tượng, một số khác có thể là thuộc tính của đối tượng.

Khi xây dựng sơ đồ tương tác thì các danh từ sẽ gợi ý cho biết cái nào là đối tượng. Nếu còn do dự cái nào là đối tượng, cái nào là thuộc tính thì hãy xét xem nó có hành vi hay không. Nếu nó chỉ cho thông tin thì nó là thuộc tính, còn nếu có hành vi thì nó là đối tượng.

Chú ý rằng, không nhất thiết mọi đối tượng có mặt đủ ở trong luồng các sự kiện. Ví dụ như mẫu báo cáo (forms) không trong luồng sự kiện, nhưng nó xuất hiện trong sơ đồ để tác nhân có thể nhập hay xem dữ liệu. Các đối tượng điều khiển cũng không có trong luồng các sự kiện. Đối tượng điều khiển là đối tượng điều khiển trình tự luồng xuyên qua ca sử dụng.

Đặc biệt, có rất nhiều khái niệm khi phân tích sẽ chuyển sang đối tượng khi thiết kế. Một số đối tượng trong thiết kế được ủy quyền từ các tác nhân hay khái niệm trừu tượng trong lĩnh vực vấn đề. Ví dụ trong quản lý thư viện, ta phải tách khái niệm sách và bản sách. Bản sách là một quyển sách cụ thể, còn sách là trừu tượng, không ứng với đối tượng nào trong thế giới thực.

4.1.2. Xác định thuộc tính của lớp

Thuộc tính của lớp mô tả các đặc tính của các đối tượng trong lớp đó. Ví dụ, *mỗi khách hàng* của lớp KhachHang có *họ và tên*, *địa chỉ*, *số tài khoản*,... Ở mỗi thời điểm thuộc tính của một đối tượng của một lớp là giá trị dữ liệu logic biểu diễn cho tính chất tương ứng của đối tượng, được gọi là *giá trị thuộc tính* của đối tượng tại thời điểm đó.

Mỗi thuộc tính có:

+ Tên của thuộc tính,

+ Kiểu xác định các loại giá trị mà thuộc tính mô tả,

+ Giá trị mặc định (khởi đầu) cho mỗi thuộc tính. Việc gán giá trị khởi đầu cho thuộc tính là tùy chọn, không bắt buộc.

1. Kiểu của thuộc tính

Kiểu của thuộc tính là kiểu dữ liệu cơ sở hoặc là các lớp đã xây dựng trước. Kiểu thuộc tính nói cho ta biết về loại giá trị kiểu *số nguyên* (Integer, int), *số thực* (Real, Float), *giá trị logic* (Boolean), *ký tự* (Character), *thời gian* (Time),... được gọi là các *kiểu nguyên thủy* (primitive type). Ngoài các kiểu nguyên thủy, người phát triển hệ thống có thể tạo ra những kiểu mới tùy ý.

2. Phạm vi của thuộc tính

Thuộc tính của lớp còn có thêm đặc tính để thể hiện khả năng nhìn thấy được hay đặc tính quản lý khả năng truy cập của thuộc tính đối với các đối tượng khác, gọi chung là *phạm vi quan sát* (gọi tắt là

phạm vi) của thuộc tính nói riêng và của các thành phần trong lớp nói chung. Đó là các đặc tính được khai báo trong lớp bằng các ký hiệu:

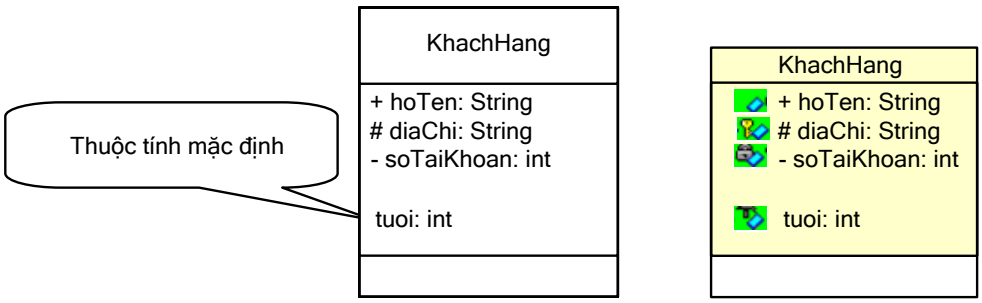
'+' đứng trước tên thuộc tính trong UML, hoặc biểu tượng ổ khóa nhưng không bị khóa trong Rose , để thể hiện thuộc tính này là công khai (*public)*, mọi đối tượng đều nhìn thấy được. Bất kỳ đối tượng nào trong hệ thống cũng nhìn thấy được và được quyền truy cập từ mọi lớp khác.

'#' đứng trước tên thuộc tính trong UML, hoặc biểu tượng ổ khóa và có chìa để bên cạnh trong Rose , để thể hiện thuộc tính này là được bảo vệ (*protected)*, những đối tượng có quan hệ kế thừa có thể nhìn thấy được.

'-' đứng trước tên thuộc tính trong UML, hoặc biểu tượng khóa bị khóa và chìa bị cất đi trong Rose , để thể hiện thuộc tính này là sở hữu riêng (*private*), chỉ bản thân đối tượng của một lớp nhìn thấy được, những đối tượng khác không được phép truy cập.

Mặc định: trường hợp những thuộc tính không có ký hiệu đặc tính phạm vi nào đứng trước thì được xem là *mặc định*, biểu tượng mặc định trong Rose là , nghĩa là những thuộc tính có thể quan sát được đối với các lớp trong cùng gói. Phạm vi mặc định còn được gọi là *phạm vi gói* hay *phạm vi cài đặt (Implementation).*

Ví dụ:



Hình 4.1. Các thuộc tính của lớp trong UML và trong Rose

Thuộc tính *hoTen* có kiểu *String* và là công khai, *soTaiKhoan* có kiểu int (các số nguyên) *là riêng* còn *diaChi* là được bảo vệ, có kiểu *String.*

Lưu ý: Khi cài đặt chương trình thì phạm vi quan sát của thuộc tính còn tùy thuộc vào những quy định khác nhau của ngôn ngữ lập trình được lựa chọn. Ví dụ đặc tính mặc định, hay được bảo vệ của thuộc tính trong C++ và Java là khác nhau.

3. Tính chất lưu trữ của thuộc tính

Các thuộc tính trong lớp nếu được xác định thì có thể lưu trữ theo *giá trị*, hoặc *theo tham chiếu,* ngược lại *chưa xác định.*

- *Giá trị (By value):* thuộc tính được gán tính chất *By value* trong mô hình thì nó sẽ được lưu trong lớp bền vững chứa thuộc tính đó.

- *Tham chiếu (Reference):* thuộc tính được gán giá trị này cho biết nó sẽ được lưu trữ ở bên ngoài lớp, nhưng có con trỏ tham chiếu đến nó.

- *Chưa xác định (Unspecified):* thuộc tính được gán tính chất này cho biết cách lưu trữ là chưa xác định, nhưng khi phát sinh mã chương trình thì Rose coi là *By value.*

4. Thuộc tính tĩnh (static)

Thông thường mỗi thuộc tính có một bản sao dữ liệu riêng cho từng đối tượng của một lớp. Ví dụ, lớp NguoiBanHang của một Công ty có thuộc tính *tenGoi* và *taiKhoan.* Khi thực hiện hệ thống tạo ra hai đối tượng là hai người bán hàng cụ thể và mỗi đối tượng có một bản sao *tenGoi* và *taiKhoan* riêng, như vậy mỗi người sử dụng một tài khoản riêng. Việc này không thật phù hợp với thực tế, vì tất cả những người bán hàng chỉ có chung một tài khoản, đó là tài khoản của Công ty. Việc này có thể thực hiện được bằng cách sử dụng thuộc tính *static* cho *taiKhoan.*

Những thuộc tính khai báo static trong một lớp là những thuộc tính được chia sẻ đối với tất cả các đối tượng, nghĩa là chỉ có một bản sao dữ liệu cho tất cả các đối tượng của lớp đó.

UML biểu diễn thuộc tính *static bằng dấu '$'* trước tên thuộc tính.

5. Thuộc tính suy dẫn (derived)

Một số thuộc tính có thể được suy dẫn từ những thuộc tính khác. Ví dụ, lớp HinhChuNhat có thuộc tính *chieuCao, chieuRong*, nếu có thêm thuộc tính *dienTich* thì tất nhiên *dienTich* là có thể suy ra từ tích của *chieuCao * chieuRong*.

Trong UML, thuộc tính suy dẫn được ký hiệu bằng dấu '/' trước tên thuộc tính.

6. Đặc tính lưu trữ đối tượng trong lớp

Trong các hệ thống đối tượng, có những đối tượng cần phải được lưu trữ vào CSDL được gọi là *đối tượng duy trì*, những đối tượng không cần lưu trữ ở bộ nhớ ngoài được gọi là *đối tượng không bền vững*. Trong Rose có thể gán đặc tính lưu trữ cho lớp, các đặc tính này có thể là:

- *Lưu trữ (persistent):* đối tượng lưu trữ là đối tượng sẽ được lưu trong CSDL, hay lưu trữ dưới khuôn dạng khác, nghĩa là chúng vẫn tồn tại khi hệ thống không thực hiện.

- *Tĩnh (static):* đối tượng tồn tại trong bộ nhớ cho đến khi chương trình kết thúc.

- *Tạm thời (transient):* đối tượng chỉ tồn tại trong bộ nhớ trong khoảng thời gian ngắn, nghĩa là các đối tượng của nó không cần lưu trữ vào bộ nhớ ngoài.

- *Trừu tượng (abstract):* cho những lớp trừu tượng, những lớp không có đối tượng thể hiện trong thực tế. Thường lớp trừu tượng được xây dựng làm các lớp cơ sở cho các lớp khác kế thừa.

7. Tìm kiếm các thuộc tính

Với mỗi lớp được tạo ra trong sơ đồ lớp, chúng ta mong muốn tìm được những thuộc tính sao cho:

- *Đầy đủ:* chứa đựng tất cả các thông tin về đối tượng của lớp,
- *Tách biệt hoàn toàn:* mỗi thuộc tính thể hiện được một đặc tính khác nhau của đối tượng,
- *Độc lập với nhau:* đối với mỗi đối tượng, các giá trị của các thuộc tính là độc lập với đối tượng khác, tốt nhất là loại bỏ những thuộc tính có thể được suy dẫn từ những thuộc tính khác.
- *Liên quan đến các yêu cầu và các ca sử dụng:* các thuộc tính được đưa vào lớp phải được xác định trên cơ sở các yêu cầu thực hiện công việc hoặc cần để tổ chức, lưu trữ, trao đổi các thông tin về đối tượng.

Trong mô hình hóa các khái niệm, có thể có những nhầm lẫn giống nhau là nhiều khi ta sử dụng thuộc tính để biểu diễn cho những cái mà đáng lý ra phải sử dụng khái niệm lớp hay các mối quan hệ liên kết để thể hiện chúng.

Để đáp ứng các nguyên tắc cơ bản và tránh được những sai sót trên, cần lưu ý:

a. Đảm bảo các thuộc tính đơn giản

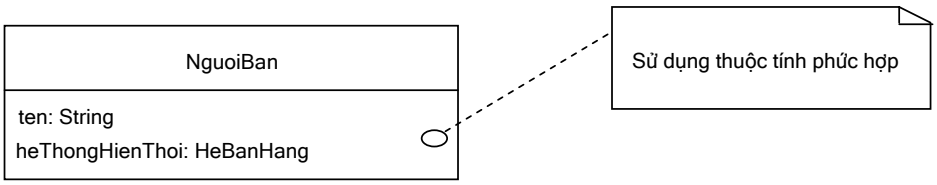
Các thuộc tính trong mô hình lớp phải là những thuộc tính đơn giản hoặc là những *kiểu dữ liệu thuần túy*:

+ *Một cách trực quan*, thuộc tính đơn giản nhất là những thuộc tính có giá trị kiểu dữ liệu nguyên thủy*: Boolean, Number, String (Text), Time,...*

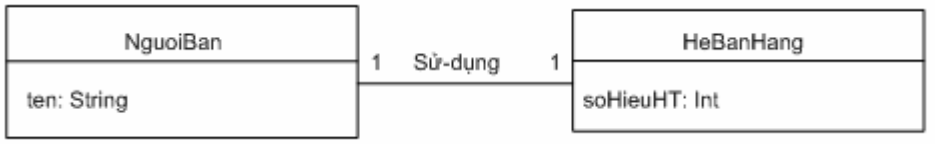
+ Một số các kiểu phổ dụng bao gồm: *DiaChi(Address), MauSac (Color),HinhHoc(Geometrics),SoDienThoai(PhoneNumber),SoBaoHiem (SocialSecurityNumber),MaSanPham(Universal ProductCode),các kiểu liệt kê (EnumeratedTypes),...*

Thông thường, nên cố tránh những thuộc tính phức hợp và nên thay vào đó là những quan hệ liên kết giữa các lớp.

Ví dụ: Xét lớp NguoiBan, có thể đưa thuộc tính *heThongHienThoi* vào để thể hiện là người bán hàng đang sử dụng hệ thống bán hàng nào đó. Bởi vì HeBanHang là lớp, nên nó là kiểu phức tạp. Do vậy, cách làm như thế không phải là tốt (hình 4.2 (a)). Để thể hiện được mối quan hệ này, ta có thể sử dụng mối quan hệ kết hợp như hình 4.2 (b).



a) Trường hợp thiết kế lớp không tốt



b) Tốt hơn là chuyển thuộc tính phức hợp thành quan hệ kết hợp

Hình 4.2.

Quy tắc hướng dẫn đầu tiên: Liên kết các khái niệm với nhau bằng quan hệ kết hợp, không bằng các thuộc tính phức hợp.

b. Sử dụng giá trị dữ liệu thuần túy

Nói một cách tổng quát, các thuộc tính phải có giá trị dữ liệu thuần túy hoặc kiểu Data Type trong UML, trong đó việc xác định duy nhất không có nhiều ý nghĩa trong ngữ cảnh của mô hình hệ thống. Ví dụ, đối với những kiểu dữ liệu nguyên thủy thì:

+ Không cần tách biệt các giá trị số giống nhau,

+ Không cần tách biệt các thể hiện của soDienThoai mà chúng có thể có cùng số,

+ Không cần tách biệt hai địa chỉ vì chúng có thể giống nhau,...

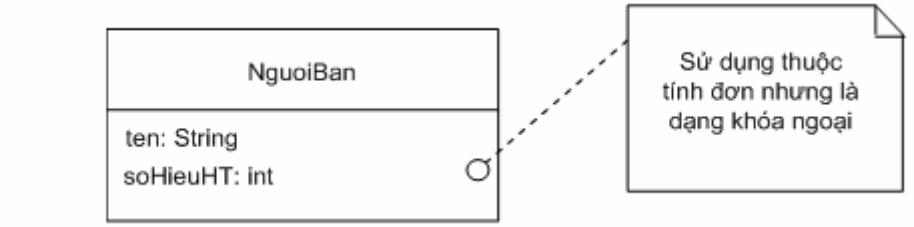
Song, đối với các đối tượng thì việc phân biệt chúng (thông qua định danh) lại *có ý nghĩa và bắt buộc.* Ví dụ, hai khách hàng có thể cùng tên "*VânAnh*", nhưng trong hệ thống phải được xác định riêng biệt bằng định danh.

Quy tắc hướng dẫn thứ hai: *Phần tử của kiểu dữ liệu thuần túy có thể được xác định trong một thuộc tính của một lớp, mặc dù nó cũng có thể được sử dụng như một khái niệm (lớp) tách biệt trong mô hình.*

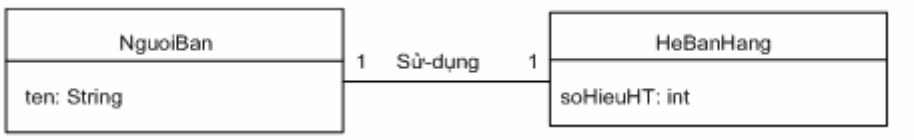
c. Không sử dụng thuộc tính như khóa ngoại

Các thuộc tính không nên sử dụng để liên kết các khái niệm lại với nhau trong mô hình khái niệm, mà chỉ được sử dụng để lưu giữ các thông tin chính các đối tượng của lớp có các thuộc tính đó. Trong thiết kế mô hình CSDL quan hệ thì nguyên lý này bị vi phạm, vì người ta thường sử dụng thuộc tính là khóa ngoại để kết nối hai kiểu thực thể với nhau.

Ví dụ: trong hình 4.3 (a), lớp NguoiBan có thuộc tính *soHieuHT* được xem như là khóa ngoại để thể hiện một người bán hàng sử dụng hệ thống bán hàng có *số hiệu xác định.* Mặc dù thuộc tính này là thuộc tính đơn (không vi phạm hướng dẫn 1), nhưng nó là khóa ngoại vì thế vẫn không tốt và do đó nên thay bằng quan hệ kết hợp như hình 4.3 (b).



a) Trường hợp sử dụng khóa ngoại



b) Chuyển thuộc tính phức thành quan hệ kết hợp

Hình 4.3.

Từ đó ta có *quy tắc hướng dẫn thứ ba*: Nên kết nối các khái niệm với nhau bằng các quan hệ kết hợp, không sử dụng thuộc tính (khóa ngoại).

Tóm lại, để tìm thuộc tính, người ta thực hiện các công việc sau:

+ Đọc kỹ các mô tả bài toán, nghiên cứu hồ sơ các chức năng hệ thống, các đặc tả ca sử dụng, các kịch bản để *tìm tất cả những thông tin, dữ liệu cần phải lưu trữ, xử lý và cập nhật.* Các mục này thường là các *danh từ, hoặc mệnh đề danh từ đơn, được xem như là đại biểu của các thuộc tính. Ví dụ:* khi xem xét luồng các sự kiện: "Đối với mỗi mặt hàng, người bán nhập vào mã sản phẩm thông qua máy đọc thẻ và số lượng hàng mà khách hàng chọn mua". Như vậy, trong lớp MatHang tất nhiên phải có thuộc tính *maSanPham,* và qua nó có thể xác định được *tên gọi, chủng loại, giá bán,...* là những thuộc tính của moTaMatHang.

+ *Sử dụng các quy tắc hướng dẫn* nêu trên để xác định chính xác các thuộc tính: đặc tính xác định phạm vi quan sát, tên gọi, kiểu và giá trị khởi đầu (nếu có) của mỗi thuộc tính.

+ *Đọc những giả thiết, sự phân loại hay những quy ước cần áp dụng cho hệ thống hiện thời* để khẳng định lại những thuộc tính của từng lớp.

+ *Gán các thuộc tính cho các lớp đối tượng* trong sơ đồ lớp.

Sơ đồ các lớp với các thuộc tính mô tả *cấu trúc tĩnh của hệ thống.* Nó mô tả mối liên kết có cấu trúc giữa các mục dữ liệu được thao tác, xử lý trong hệ thống. Nó cũng mô tả cách các thông tin được phân chia thành từng phần cho các đối tượng, chỉ ra cách các đối tượng được chia thành các lớp và thể hiện mối quan hệ giữa các đối tượng là gì.

4.1.3. Xác định các thao tác của lớp

1. Ký pháp của thao tác

Thao tác của lớp mô tả các hành vi và các mối quan hệ của các đối tượng trong hệ thống. Trên sơ đồ lớp, mỗi thao tác được mô tả bởi *phạm vi, tên gọi, danh sách các tham số* và *kiểu trả lại giá trị.*

visibility name (arg1: DataType1, arg2: DataType2,..., argn: DataTypen): ReturnType

Trong đó:

- *visibility* khai báo phạm vi quan sát của thao tác trong lớp. *visibility* có thể là một trong các đặc tính: *public, protected, private,* hay *mặc định* giống như đối với các thuộc tính như đã nêu ở trên.

- *name* là tên gọi của thao tác, theo quy ước thường là các động từ (cụm động từ), viết hoa các chữ cái đầu tiên của các từ trừ từ đầu tiên.

- *argk là tên của tham số hình thức thứ k* trong danh sách. Chúng là các đối số mà thao tác nhận ở đầu vào.

- *DataTypek là các kiểu thuộc tính*, thường đó là các kiểu của các thuộc tính đã khai báo trong lớp.

- *ReturnType là kiểu dữ liệu trả lại sau khi thao tác kết thúc.* Những thao tác không có kiểu trả lại (các thủ tục) thì sử dụng Void (void).

Ví dụ: trong lớp *ThanhToan* có thao tác:

+ makePayment(upc: UPC, n: Integer): Boolean

Lưu ý:

trong bảng thiết kế lớp, nhiều khi các thao tác chỉ cần hiển thị ngắn gọn tên và danh sách các tham số cho đơn giản. Ví dụ, thao tác trên có thể viết ngắn gọn:

+ *makePayment(upc, n)*, trong đó *upc* là mã sản phẩm,
n là số lượng mặt hàng mà khách hàng chọn mua.

2. Phân loại thao tác

Nói chung có năm loại thao tác: nghiệp vụ, quản lý, truy cập, hiển thị (trao đổi) và trợ giúp.

- *Thao tác nghiệp vụ/cài đặt (implementor):* Thao tác loại này đảm nhận một chức năng tác nghiệp mà lớp đối tượng cần thực hiện.

Chúng được tìm ra từ những *thông điệp* được gửi tới cho đối tượng của lớp trong các sơ đồ tương tác.

Ví dụ: khi đối tượng :ThanhToan nhận được thông điệp *makePayment(soTien)* trong sơ đồ cộng tác (hay sơ đồ trình tự) thì lớp ThanhToan sẽ có thao tác *makePayment(soTien)*.

- *Thao tác quản lý (manager):* thao tác quản lý các đối tượng của lớp, làm nhiệm vụ tạo lập, hủy bỏ đối tượng. Ví dụ, các toán tử tạo lập và các hủy tử là thuộc nhóm này.

- *Thao tác truy cập (access):* Theo quy định, những thuộc tính khai báo *private* hay *protected* trong lớp là nhằm hạn chế việc truy cập của các đối tượng khác. Tuy nhiên, để trao đổi được với nhau thì nhiều đối tượng của các lớp khác lại cần truy cập đến những dữ liệu đó. Việc này thực hiện được thông qua những thao tác truy cập để đọc hay ghi (thay đổi) dữ liệu thành phần của các lớp.

Ví dụ: lớp KhachHang có thuộc tính *taiKhoan* được khai báo là *private* để không cho các đối tượng khác truy cập tự do. Nhưng để xử lý được việc thanh toán với khách hàng thì hệ thống cũng phải có cách để truy cập được *taiKhoan* của người mua hàng, do vậy phải có những thao tác như *getTaiKhoan()* để đọc, hay *setTaiKhoan()* để cập nhật số tiền của khách hàng. Những thao tác loại này thường khai báo là *public.* Thông thường thao tác Get và Set được hình thành cho mỗi thuộc tính trong lớp.

- *Thao tác trợ giúp (helper):* những thao tác mà chính các lớp chứa nó cần thực hiện những công việc được phân công. Đó thường là những thao tác có tính chất *private, protected.* Phương thức này được hình thành từ các thông điệp bản thân trong sơ đồ trình tự và cộng tác.

- *Thao tác hiển thị, trao đổi thông tin:* những thao tác đảm nhận việc hiển thị thông tin ra các thiết bị ngoại vi như máy in, máy vẽ, màn hình,... hay chuyển ra các kênh truyền thông trên mạng.

3. Xác định các thao tác

Để nhận diện được các thao tác của lớp, chúng ta phải:

- Khảo sát sơ đồ tương tác. Phần lớn các thông điệp của sơ đồ này sẽ trở thành thao tác nghiệp vụ, các thông điệp phản thân sẽ trở thành thao tác giúp đỡ.

- Các thao tác quản lý: bổ sung cấu tử và hủy tử.

- Các thao tác xâm nhập: tạo lập các thao tác *Get* và *Set* cho các thuộc tính cần được xem xét trong lớp khác hay lớp khác cần làm thay đổi giá trị của chúng.

Thao tác xác định trách nhiệm của lớp. Sau khi đã nhận biết thao tác thì xem xét lại lớp chứa chúng, chú ý rằng:

- Nếu lớp chỉ có 1 hay 2 thao tác thì nên gộp vào lớp khác.

- Nếu lớp không có thao tác nào thì nên mô hình hóa nó như thuộc tính.

- Nếu lớp có quá nhiều thao tác thì nên chia sẻ chúng cho các lớp khác.

4.2. XÁC ĐỊNH MỐI QUAN HỆ GIỮA CÁC LỚP

Như chúng ta đã biết, hệ thống là một thể thống nhất, các phần tử của hệ thống phải có quan hệ tương tác với nhau. Do vậy, trong mô hình khái niệm, ngoài sự hiện diện những khái niệm (các lớp đối tượng) khác nhau còn có cả các mối quan hệ giữa các lớp đối tượng đó. Nhiệm vụ khó khăn nhưng không thể thiếu của chúng ta là đi tìm các mối quan hệ đó.

Hệ thống hướng đối tượng là tập các đối tượng tương tác với nhau để thực hiện công việc theo yêu cầu. Quan hệ là sự kết nối ngữ nghĩa giữa các lớp đối tượng, trong đó thể hiện mối liên quan về các thuộc tính, các thao tác của chúng với nhau trong hệ thống. Các quan hệ này được thể hiện chính trong sơ đồ lớp.

Giữa các lớp có năm quan hệ cơ bản:

- Quan hệ kết hợp (association),
- Quan hệ tụ hợp - dạng đặc biệt của quan hệ kết hợp (aggregation),
- Quan hệ tổng quát hóa, kế thừa (generalization),
- Quan hệ phụ thuộc (dependencies),
- Quan hệ thực hiện hóa (realisation).

4.2.1. Quan hệ kết hợp

1. Khái niệm về sự liên kết giữa các đối tượng và sự kết hợp giữa các lớp

Trước tiên chúng ta cần phân biệt các mối quan hệ giữa các lớp và giữa các đối tượng với nhau.

Một liên kết là một sự kết nối vật lý hoặc logic giữa các đối tượng với nhau. Phần lớn các liên kết là sự kết nối giữa hai đối tượng với nhau. Tuy nhiên cũng có những liên kết giữa ba hoặc nhiều hơn ba đối tượng. Nhưng các ngôn ngữ lập trình hiện nay hầu như chỉ cài đặt những liên kết (phép toán) nhiều nhất là ba ngôi, ví dụ phép lấy thành phần (.)

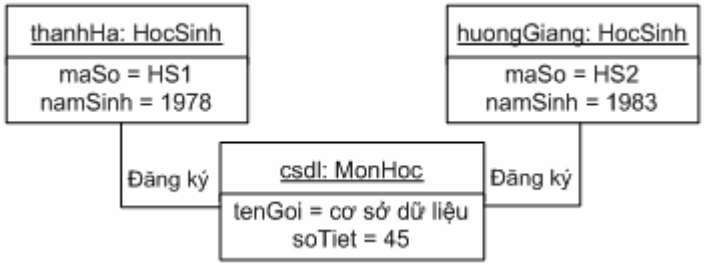
Booch đã mô tả vai trò của mối liên kết (link) giữa các đối tượng như sau: "*Một liên kết chỉ rõ sự kết hợp mà qua đó, một đối tượng được một đối tượng khác phục vụ hoặc một đối tượng có thể điều khiển đối tượng kia*".

Trong UML, sự kết hợp (*Association*) là quan hệ giữa hai lớp, nó xác định cách các đối tượng của các lớp có thể liên kết với nhau để thực hiện công việc như thế nào. Tương tự như đối với lớp, thể hiện của lớp là các đối tượng, đối với mối quan hệ kết hợp, thể hiện của nó là sự *liên kết (link)* giữa các đối tượng của hai lớp. Nghĩa là các đối tượng của một lớp cùng chia sẻ với nhau các mối quan hệ.

Sự kết hợp là một mô tả về một nhóm các liên kết có chung cấu trúc và ngữ nghĩa như nhau. Vậy, liên kết là một thể hiện kết hợp

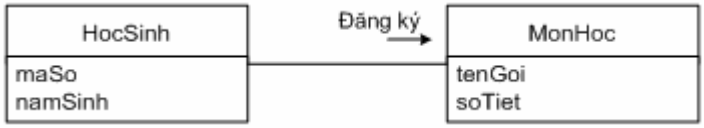
của lớp. Liên kết và kết hợp thường xuất hiện ở dạng các động từ trong các tài liệu mô tả bài toán ứng dụng.

Các hình dưới đây mô tả các quan hệ liên kết (cho đối tượng) và kết hợp (cho lớp).



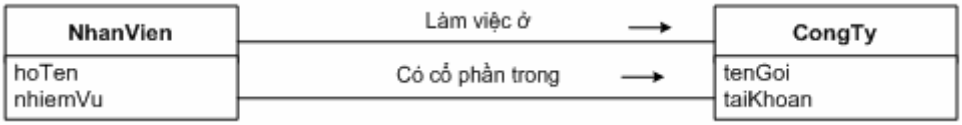
Hình 4.4. Liên kết giữa các đối tượng

Quan hệ liên kết giữa hai đối tượng được biểu diễn bằng đoạn thẳng nối chúng với nhau và có tên gọi (nhãn của quan hệ). Nhãn của các quan hệ thường là các động từ. Khi muốn biểu diễn cho quan hệ một chiều thì sử dụng mũi tên để chỉ rõ hướng của quan hệ. Quan hệ kết hợp giữa hai lớp cũng được biểu diễn như vậy.



Hình 4.5. Quan hệ kết hợp giữa các lớp

Khi mô hình không có sự nhập nhằng thì tên của kết hợp là tùy chọn. Sự nhập nhằng sẽ xuất hiện khi giữa hai lớp có nhiều quan hệ kết hợp, ví dụ: giữa lớp NhanVien và lớp CongTy có hai quan hệ *làm việc ở* và *có cổ phần trong.* Khi có nhiều quan hệ như thế thì nên gán tên vào mỗi đường nối hoặc *tên của vai trò* ở mỗi đầu của quan hệ để tránh sự nhập nhằng.



Hình 4.6. Quan hệ kết hợp giữa các lớp

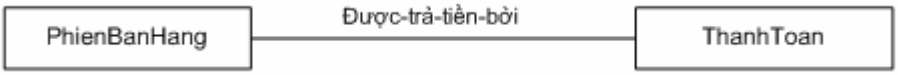
Lưu ý:

Không nên nhầm lẫn liên kết với giá trị. Giá trị là dữ liệu nguyên thủy như là dữ liệu số hoặc xâu ký tự. Liên kết là mối liên quan giữa hai đối tượng. Trong giai đoạn phân tích ta phải mô hình (xác định) tất cả các tham chiếu tới các đối tượng thông qua các liên kết và nhận dạng được các nhóm liên kết tương tự thông qua các quan hệ kết hợp. Sau này, đến giai đoạn thiết kế ta có thể chọn cấu trúc con trỏ, khóa ngoại, hoặc một số cách khác để cài đặt những quan hệ đó.

Các đối tượng có thể có nhiều loại quan hệ và vì thế, các lớp (khái niệm) cũng phải có tất cả các loại quan hệ đó trong lĩnh vực ứng dụng.

- Quan hệ kết hợp giữa hai lớp là *sự kết nối vật lý hay khái niệm* giữa các đối tượng của hai lớp đó.
- Chỉ những đối tượng có mối quan hệ kết hợp với nhau mới có thể cộng tác với nhau theo các đường liên kết được thiết lập giữa các lớp.

Ví dụ: PhienBanHang và ThanhToan là hai lớp đã phân tích ở trên trong hệ thống bán hàng là có quan hệ kết hợp với nhau, mỗi lần thanh toán là để trả tiền cho một lần mua hàng. Quan hệ này được mô tả như hình 4.7:



Hình 4.7. Mối quan hệ kết hợp giữa hai lớp

Trong đó, "được-trả-tiền-bởi" là tên của quan hệ kết hợp. Tên của các mối quan hệ kết hợp được đặt theo cách sau:

- Tên của quan hệ kết hợp thường là *mệnh đề động từ đơn* dễ đọc và có nghĩa trong ngữ cảnh của mô hình, thể hiện được mối liên hệ giữa các lớp.

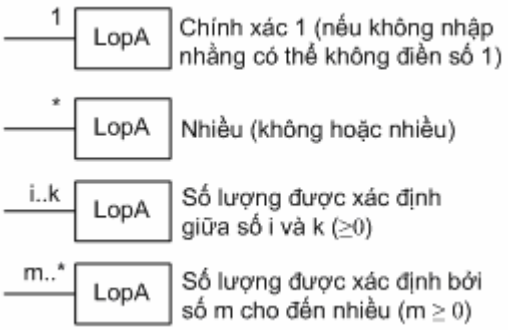
- Tên của quan hệ kết hợp thường *bắt đầu bằng động từ*, hay tính động từ với chữ đầu được viết hoa.
- Giữa các từ trong tên của quan hệ được nối với nhau bằng '-'.

Ví dụ: Tên quan hệ giữa hai lớp PhienBanHang và ThanhToan là *Được-trả-tiền-bởi* như ở hình 4.7.

Vấn đề quan trọng đặt ra là làm thế nào để xác định chính xác các mối quan hệ giữa các lớp trong hệ thống.

2. Khái niệm "bản số" và "vai trò" trong quan hệ

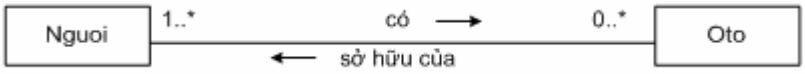
Quan hệ kết hợp thường là quan hệ hai chiều: một đối tượng liên kết với một số đối tượng của lớp khác và ngược lại. Để xác định số các đối tượng có thể tham gia vào mỗi đầu của mối quan hệ ta có thể sử dụng khái niệm *bản số* (*cardinal number*). *Bản số* xác định số lượng các thể hiện (đối tượng) của một lớp trong quan hệ kết hợp với lớp khác. Cũng cần phân biệt *bản số* với *lực lượng*. Bản số là ràng buộc về kích cỡ của một tuyển tập, còn lực lượng là đếm số phần tử của tuyển tập đó. Do đó, *bản số là sự ràng buộc về lực lượng của các phần tử trong một lớp tham gia vào quan hệ xác định trước*. Trong UML, các bản số được biểu diễn như hình 4.8:



Hình 4.8. Cách biểu diễn bản số

Để phân biệt chiều của quan hệ kết hợp ta có thể sử dụng mũi tên chỉ chiều kết hợp.

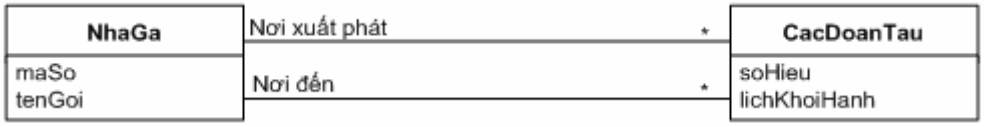
Ví dụ:



Hình 4.9. Quan hệ kết hợp hai chiều giữa hai lớp

Hình 4.9 mô tả như sau: mỗi người trong lớp Nguoi có thể không có hoặc có nhiều ô tô (thuộc lớp Oto) và ngược lại một ô tô phải là sở hữu của ít nhất một người nào đó thuộc lớp Nguoi.

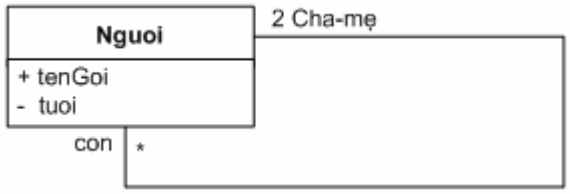
Vai trò là tên gọi một nhiệm vụ, thường là một danh từ được gán cho một đầu của quan hệ kết hợp. Hình 4.10 mô tả hai lớp NhaGa và lớp CacDoanTau có quan hệ kết hợp với nhau. Một nhà ga có thể là điểm đến của đoàn tàu này và lại có thể là điểm xuất phát của đoàn tàu khác. Ngược lại một đoàn tàu bao giờ cũng phải khởi hành từ một nhà ga này tới một nhà ga khác [18].



Hình 4.10. Vai trò trong các quan hệ kết hợp giữa 2 lớp

Khi mô hình không có sự nhập nhằng thì tên của vai trò là tùy chọn. Sự nhập nhằng sẽ xuất hiện khi giữa hai lớp có nhiều quan hệ, hoặc khi một lớp lại có quan hệ với chính nó (quan hệ đệ quy). Ví dụ:

Một người có thể là con của hai người (cha-mẹ) và hai cha-mẹ lại có thể có nhiều con.



Hình 4.11. Vai trò trong các quan hệ kết hợp

3. Các phương pháp xác định mối quan hệ kết hợp

Có hai phương pháp chính để xác định các mối quan hệ giữa các lớp trong hệ thống:

- Dựa vào nguyên lý "Cần để biết" vì mối quan hệ kết hợp giữa các lớp đối tượng là cần để biết về những thông tin liên quan đến các lớp đó.
- Dựa vào sự phân loại phạm trù giữa các quan hệ trong hệ thống.

a. Xác định quan hệ kết hợp theo nguyên lý "Cần để biết"

Khi phân tích các yêu cầu, ta cần phải tuân theo các nguyên lý sau:

- Quan hệ kết hợp hữu ích thường cho ta sự hiểu biết về một mối quan hệ cần được duy trì trong một thời khoảng nào đó, được gọi là sự kết hợp "*cần để biết" (Need-to-know).*
- Sự liên kết quan trọng giữa hai đối tượng phải thể hiện được vai trò của sự cộng tác hay sự tương tác giữa các đối tượng đó.

Ví dụ: Áp dụng nguyên lý "*Cần để biết"* vào ca sử dụng *"Mua hàng bằng tiền mặt",* chúng ta nhận diện được những quan hệ sau giữa các lớp đối tượng:

- HeBanHang *Xử-lý* PhienBanHang để biết về lần bán hàng hiện thời, biết tổng số tiền khách hàng phải trả và để in phiếu bán hàng giao cho khách.
- PhienBanHang *Được-trả-tiền-bởi* ThanhToan để biết xem hàng vừa bán đã được trả tiền hay chưa, nó cũng liên quan đến số tiền mà khách đưa, hệ thống phải trả lại tiền dư và vấn đề in phiếu bán hàng.
- DanhMucMatHang *Ghi-lại* MoTaMatHang để tìm các thông tin mô tả về các mặt hàng như: chủng loại, giá cả, chất lượng,... khi biết mã sản phẩm.

b. Xác định mối quan hệ kết hợp dựa vào việc phân loại phạm trù quan hệ

Việc tìm các quan hệ kết hợp cũng giống như việc tìm kiếm đối với các lớp, chúng ta có thể dựa vào danh sách các phạm trù kết hợp để xác định.

Ví dụ: Xét *Hệ thống bán hàng*, những phạm trù quan hệ trong bảng 4.1 cần xem xét để tìm kiếm các mối quan hệ kết hợp.

Bảng 4.1. Những phạm trù quan hệ trong hệ thống bán hàng

Các phạm trù kết hợp	Ví dụ
A là một bộ phận logic của B	DongBanHang - PhienBanHang
A là một loại/lớp con/kiểu con của B	ThanhToanTienMat - ThanhToan
A được chứa (vật lý) trong/trên B	HeBanHang - CuaHang
A được chứa (logic) trong B	MoTaMatHang - DanhMucMatHang
A là một mô tả của B	MoTaMatHang - MatHang
A là một mục trong một giao dịch	DongBanHang - PhienBanHang
A là thành viên của B	NguoiBan - CuaHang
A sử dụng hoặc quản lý B	NguoiBan - HeBanHang
A trao đổi với B	KhachHang - NguoiBan
Giao dịch A có quan hệ với giao dịch B	ThanhToan - PhienBanHang
A là sở hữu của B	HeBanHang - CuaHang

Tóm lại, dựa vào việc phân loại các phạm trù kết hợp như trên, dựa vào kinh nghiệm, kiến thức về hệ thống và dựa vào các kết quả khảo sát hệ thống trong thực tế để liệt kê tất cả các mối quan hệ kết hợp thực giữa các lớp trong hệ thống.

Lưu ý: Trong giai đoạn phân tích, quan hệ kết hợp không cần phải mô tả về các dòng dữ liệu, các biến thể hiện, hoặc các mối kết nối của các đối tượng như trong lời giải cụ thể mà chỉ cần thể hiện được những mối liên hệ có nghĩa trong thế giới thực. Trong các pha sau, pha thiết kế và cài đặt, những mối liên hệ này sẽ được cài đặt như là các đường dẫn thông tin liên kết giữa các lớp thể hiện được khả năng quan sát giữa các đối tượng trong hệ thống khi nó thực hiện.

4.2.2. Quan hệ tụ hợp

Tụ hợp (aggregation) là một loại của quan hệ kết hợp, tập trung *thể hiện quan hệ giữa tổng thể và bộ phận*. Tụ hợp thường biểu diễn cho quan hệ "*có một*", "*là bộ phận của*", hoặc "*bao gồm*",... thể hiện mối quan hệ một lớp tổng thể có, gồm, chứa hay liên kết với một hoặc nhiều lớp thành phần.

Có ba loại quan hệ tụ hợp:

- Tụ hợp thông thường/tụ hợp
- Tụ hợp chia sẻ
- Tụ hợp hợp thành/hợp thành.

a. Quan hệ tụ hợp

Quan hệ tụ hợp thông thường, gọi tắt là tụ hợp thể hiện mối kết hợp giữa hai lớp, trong đó đối tượng của lớp này bao gồm một số đối tượng của lớp kia, song không tồn tại trong nội tại của lớp đó. Lớp phía bộ phận cũng chỉ là một bộ phận logic của phía tổng thể và chúng không được chia sẻ với các lớp khác.

UML sử dụng ký hiệu: để biểu diễn quan hệ tụ hợp và luôn được gắn với phía tổng thể.

Ví dụ:

Hình 4.12. Quan hệ tụ hợp

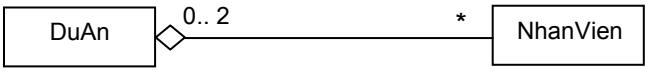
Trong quan hệ này, việc quản lý các đối tượng của các lớp liên quan là khác nhau. Ta có thể loại bỏ một số *học sinh* của một *nhóm thực tập* sao cho số còn lại ít nhất là 2, tương tự có thể bổ sung vào một số *học sinh* sao cho không quá 5. Nhưng khi đã loại bỏ một *nhóm* thì phải loại bỏ tất cả các học sinh của *nhóm* đó vì mỗi học sinh chỉ thuộc một *nhóm*.

b. Quan hệ tụ hợp chia sẻ

Quan hệ *Tụ hợp chia sẻ* là loại tụ hợp, trong đó phía bộ phận có thể tham gia vào nhiều phía tổng thể. Ví dụ: một *dự án* của lớp DuAn có nhiều *nhân viên* của lớp NhanVien tham gia và một nhân viên có thể tham gia vào nhiều (hai) dự án.

UML sử dụng ký hiệu: ◇— để biểu diễn quan hệ tụ hợp chia sẻ và luôn được gắn với phía tổng thể.

Ví dụ:



Hình 4.13. Quan hệ tụ hợp chia sẻ

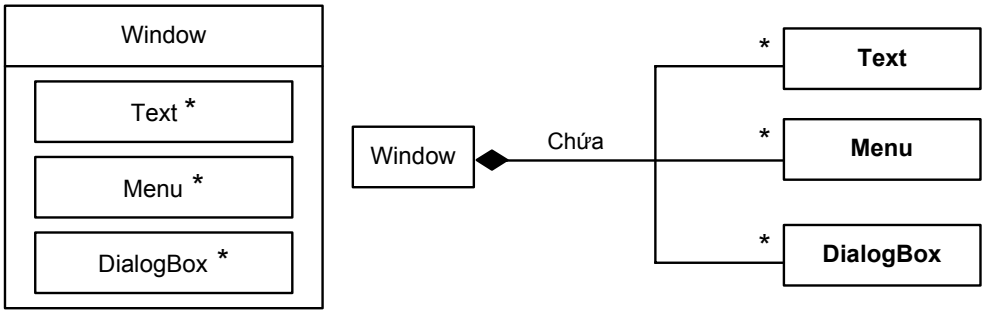
Mỗi dự án có thể có nhiều người tham gia và mỗi người lại có thể tham gia nhiều nhất là hai dự án. Trong quan hệ này, ta có thể loại bỏ, hay thành lập một dự án (phía tổng thể) nhưng không nhất thiết phải loại bỏ, hay phải tuyển thêm những người tham gia (phía bộ phận) vào dự án như kiểu tụ hợp ở trên. Tuy nhiên khi xử lý các mối quan hệ đó thì phải cập nhật lại các mối liên kết của các nhân viên tham gia vào các dự án tương ứng.

c. Quan hê hợp thành (composition)

Quan hệ tụ hợp hợp thành, gọi tắt là hợp thành chỉ ra một vật có chứa một số bộ phận và các bộ phận đó tồn tại vật lý bên trong vật tổng thể. Do vậy khi thực hiện hủy bỏ, hay thiết lập mới bên tổng thể thì các bộ phận bên thành phần cũng sẽ bị hủy bỏ hoặc phải được bổ sung.

UML sử dụng ký hiệu: ◆— để biểu diễn quan hệ hợp thành và luôn được gắn với phía tổng thể.

Ví dụ: Lớp Window chứa các lớp Text, Menu và DialogBox. Trong UML có hai cách biểu diễn quan hệ hợp thành như sau:



Hình 4.14. Quan hệ hợp thành

4.2.3. Quan hệ tổng quát hóa

Tổng quát hóa và chuyên biệt hóa là hai cách nhìn dưới/lên và trên/xuống về sự phân cấp các lớp, mô tả khả năng quản lý cấp độ phức tạp của hệ thống bằng cách trừu tượng hóa các lớp.

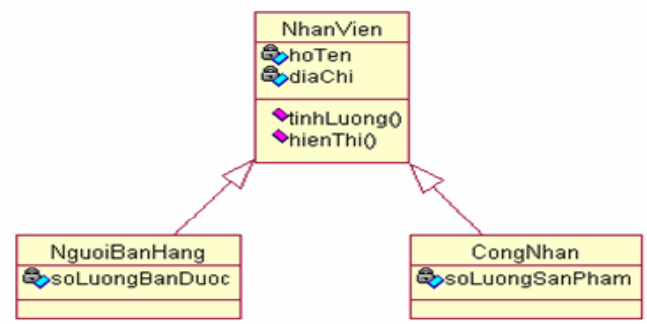
Tổng quát hóa là đi từ các lớp dưới lên sau đó hình thành lớp tổng quát (lớp trên, lớp cha), tức là cây cấu trúc các lớp từ lá đến gốc.

Chuyên biệt hóa là quá trình ngược lại của tổng quát hóa, nó cho phép tạo ra các lớp dưới (lớp con) khác nhau của lớp cha.

Trong UML, tổng quát hóa chính là quan hệ kế thừa giữa hai lớp. Nó cho phép lớp con (lớp dưới, lớp kế thừa, hay lớp dẫn xuất) kế thừa trực tiếp các thuộc tính và các hàm thuộc loại công khai, hay được bảo vệ (*protected*) của lớp cha (lớp cơ sở, lớp trên). Trong quan hệ tổng quát hóa có hai loại lớp: *lớp cụ thể* và *lớp trừu tượng*.

Lớp cụ thể là lớp có các đại diện, các thể hiện cụ thể. Ngược lại, *lớp trừu tượng* là lớp không có thể hiện (đối tượng) cụ thể trong hệ thống thực. Các lớp con cháu của lớp trừu tượng có thể là lớp trừu tượng, tuy nhiên trong cấu trúc phân cấp theo quan hệ tổng quát hóa thì mọi nhánh phải kết thúc (lớp lá) bằng các lớp cụ thể.

Ví dụ:



Hình 4.15. Lớp trừu tượng và cụ thể trong quan hệ tổng quát hóa

Lưu ý:

+ Quan hệ tổng quát và kết hợp là hai quan hệ liên quan đến hai lớp, nhưng chúng có những điểm khác nhau. Quan hệ kết hợp mô tả mối liên kết giữa hai hoặc nhiều hơn đối tượng còn quan hệ khái quát mô tả các phương diện khác nhau của cùng một thể hiện.

+ Trong giai đoạn phân tích, các quan hệ kết hợp là quan trọng hơn quan hệ tổng quát hóa. Kết hợp bổ sung thêm các thông tin cho các lớp. Ngược lại, tổng quát hóa là loại bỏ những thông tin bị chia sẻ ở các lớp con cháu vì chúng được kế thừa từ lớp cha.

+ Trong giai đoạn thiết kế thì tổng quát hóa lại quan trọng hơn. Người phát triển hệ thống quan tâm để phát hiện ra những cấu trúc dữ liệu ở khâu phân tích và phát hiện ra các hành vi ở khâu thiết kế. Tổng quát hóa cung cấp cơ chế sử dụng lại để thể hiện chính xác các hành vi và mã hóa của các thư viện trong các lớp.

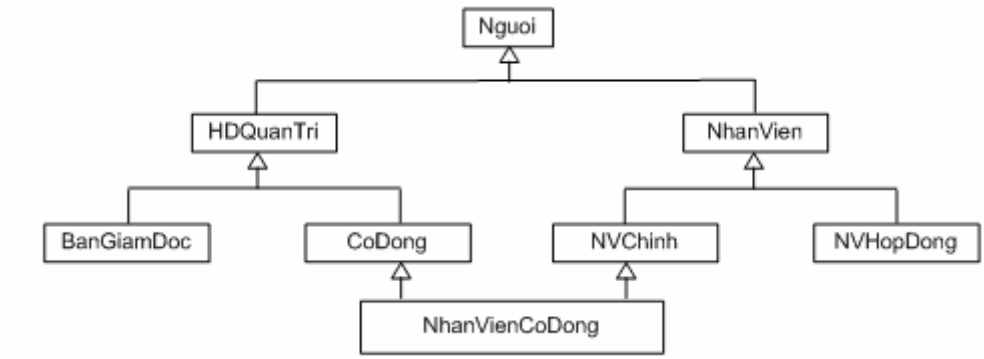
+ Quan hệ tụ hợp và tổng quát hóa cũng khác nhau. Cả hai đều làm xuất hiện cấu trúc cây thông qua bao đóng bắc cầu của quan hệ cơ sở, nhưng quan hệ tổng quát là mối quan hệ "hoặc" (OR) còn quan hệ tụ hợp là mối quan hệ "và" (AND).

4.2.4. Quan hệ kế thừa

Kế thừa bội cho phép một lớp (con) được kế thừa các thuộc tính, các thao tác và các quan hệ kết hợp từ nhiều lớp tổng quát (cha). Trong quan hệ kế thừa bội có thể dẫn đến sự pha trộn thông tin từ nhiều nguồn dữ liệu khác nhau từ các lớp được kế thừa. *Quan hệ kế thừa đơn*, một lớp được kế thừa từ một lớp trên, thường tạo ra cấu trúc cây, còn *kế thừa bội lại tổ chức các lớp thành đồ thị định hướng phi chu trình*. Kế thừa bội là cơ chế mạnh trong mô hình hệ thống, nhưng đồng thời cũng sẽ tạo ra nhiều sự phức tạp về tính nhập nhằng, không nhất quán dữ liệu.

a. Kế thừa bội từ những lớp phân biệt

Một lớp có thể kế thừa từ nhiều lớp tổng quát khác nhau. Ví dụ:

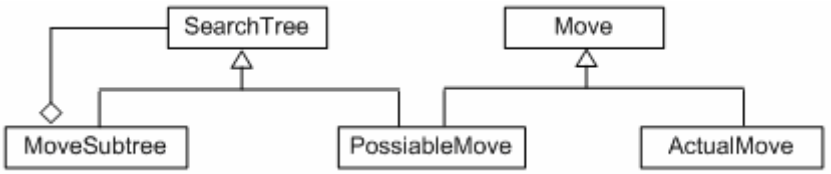


Hình 4.16. Kế thừa bội từ hai lớp khác nhau và có lớp tổng quát chung (lớp người)

b. Kế thừa bội không có lớp tổng quát chung

Kế thừa bội như trên là kế thừa có lớp tổng quát chung (lớp Nguoi). Chúng ta có thể tạo ra lớp kế thừa bội từ nhiều lớp mà chúng lại không có lớp tổng quát chung. Loại kế thừa này thường xuất hiện khi ta muốn pha trộn một số chức năng của các lớp thư viện khác nhau.

Ví dụ:



Hình 4.17. Kế thừa bội không có lớp tổng quát chung

4.2.5. Quan hệ phụ thuộc

Sự phụ thuộc là một loại quan hệ liên kết giữa hai phần tử trong mô hình, trong đó thể hiện sự thay đổi trong một phần tử sẽ kéo theo sự thay đổi của phần tử kia. Quan hệ kết hợp thường là quan hệ hai chiều, nhưng quan hệ phụ thuộc lại thường là quan hệ một chiều, thể hiện một lớp phụ thuộc vào lớp khác. Lớp A phụ thuộc vào lớp B khi:

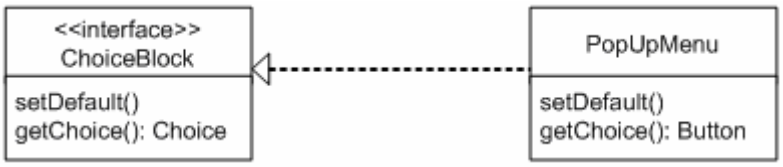
- Lớp A sử dụng một đối tượng của lớp B như là tham số trong các thao tác,
- Trong các thao tác của lớp A có truy cập tới các đối tượng của lớp B,
- Khi thực hiện một thao tác nào đó trong lớp A lại phải tham chiếu tới miền xác định của lớp B.
- Lớp A sử dụng các *giao diện* của lớp B.

Tương tự, hai *gói* có thể phụ thuộc vào nhau khi có một lớp ở một gói phụ thuộc vào lớp của gói kia. Trong UML, quan hệ phụ thuộc được thể hiện bằng mũi tên đứt nét. Ví dụ sau mô tả quan hệ phụ thuộc giữa hai lớp và phụ thuộc của hai gói.

Hình 4.18. Quan hệ phụ thuộc giữa các lớp và các gói

4.2.6. Quan hệ thực hiện hóa

Quan hệ *thực hiện hóa* thể hiện sự kết nối giữa các lớp và giao diện. Quan hệ này thường được sử dụng với các giao diện và những lớp làm nhiệm vụ cài đặt các dịch vụ (thao tác) đã được khai báo trong các giao diện. Quan hệ thực hiện hóa được ký hiệu bằng mũi tên đứt nét như sau:



Hình 4.19. Quan hệ thực hiện hóa

4.3. PHÁT TRIỂN MÔ HÌNH ĐỐI TƯỢNG

Ngay từ đầu chương, chúng ta đã biết mô hình khái niệm của một hệ thống được mô tả bởi sơ đồ lớp. *Sơ đồ lớp mô tả quan sát tĩnh của hệ thống thông qua các lớp và các mối quan hệ của chúng*. Nó được sử dụng để hiển thị các lớp và gói các lớp cùng các mối quan hệ của chúng. Sơ đồ lớp giúp người phát triển phần mềm quan sát và lập kế hoạch cấu trúc hệ thống trước khi lập trình. Nó đảm bảo rằng hệ thống được thiết kế tốt ngay từ đầu.

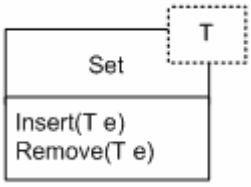
4.3.1. Các loại lớp trong sơ đồ lớp

Sơ đồ lớp có thể chứa nhiều loại lớp khác nhau, chúng có thể là những lớp thông thường, lớp tham số hóa, lớp hiện thực, lớp tiện ích, và lớp metaclass (siêu lớp).

1. Lớp tham số hóa

Lớp tham số hóa (Parameterized Class) là lớp được sử dụng để tạo ra một họ các lớp khác. Trong những ngôn ngữ lập trình có kiểu mạnh như C++, lớp tham số hóa chính là *lớp mẫu* (*template class*). Trong UML, có thể khai báo lớp tham số hóa (lớp mẫu) Set cho họ các

lớp có các phần tử là kiểu T bất kỳ, được xem như là tham số như hình 4.20:

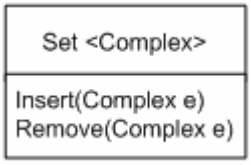


Hình 4.20. Lớp được tham số hóa

Lớp tham số hóa có thể sử dụng để thể hiện quyết định thiết kế về các giao thức trao đổi giữa các lớp. Lớp tham số hóa ít được sử dụng trong mô hình khái niệm mà chủ yếu được sử dụng trong các mô hình cài đặt, nhưng cũng chỉ khi ngôn ngữ lập trình được chọn để lập trình có hỗ trợ cơ chế lớp mẫu như C++ chẳng hạn. Cũng cần lưu ý là không phải tất cả các ngôn ngữ lập trình hướng đối tượng đều hỗ trợ kiểu lớp mẫu, ví dụ Java không hỗ trợ, nhưng tất cả các lớp trong Java lại tạo ra cấu trúc cây phân cấp có gốc là lớp Object. Do tính chất phân cấp của cây và nguyên lý chung bảo toàn mối quan hệ giữa các lớp con với lớp cha (nguyên lý thành viên và nguyên lý 100%) mà ta vẫn có thể tạo ra được những cấu trúc tổng quát, không thuần nhất tương tự như lớp mẫu.

2. Lớp hiện thực

Lớp hiện thực (Instantiated Class) là loại lớp tham số hóa mà đối số của nó là một kiểu trị cụ thể. Như vậy, lớp tham số hóa là khuôn để tạo ra các lớp hiện thực. Ví dụ, lớp **Set<Complex>** tập các số phức (*Complex)* là lớp hiện thực được biểu diễn trong UML như hình 4.21:



Hình 4.21. Lớp thực hiện hóa

3. Lớp tiện ích

Lớp tiện ích (Class Utility) là tập hợp các thao tác được sử dụng nhiều nơi trong hệ thống, chúng được tổ chức thành lớp tiện ích để các lớp khác có thể cùng sử dụng. Trong sơ đồ, lớp tiện ích được thể hiện bằng lớp có đường viền bóng như hình 4.24 (a).

Hình 4.22.

4. Giao diện

Giao diện (Interface) là tập những thao tác quan sát được từ bên ngoài của một lớp và/hoặc một thành phần, và không có nội dung cài đặt của riêng lớp đó. *Giao diện* thuộc quan sát logic và có thể xuất hiện trong cả sơ đồ lớp và sơ đồ thành phần với ký hiệu đồ họa như hình 4.22 (b).

5. Siêu lớp

Siêu lớp (MetaClass) là lớp để tạo ra các lớp khác, nghĩa là thể hiện của nó là lớp chứ không phải là đối tượng. Lớp tham số hóa chính là một loại siêu lớp.

4.3.2. Khuôn mẫu của các lớp

Khuôn mẫu (Stereotype) là cơ chế mở rộng các phần tử của mô hình để tạo ra những phần tử mới. Nó cho phép dễ dàng bổ sung thêm các thông tin cho các phần tử của mô hình và những phần tử này được đặc tả trong các dự án hay trong quá trình phát triển phần mềm. Nó được sử dụng để phân loại các lớp đối tượng.

Trong sơ đồ lớp, *stereotype* là cơ chế để phân nhóm lớp. Ví dụ: để nhanh chóng tìm biểu mẫu trong mô hình, ta tạo *stereotype* với tên biểu mẫu (form) rồi gán nó cho mọi lớp tương ứng.

Rational Rose đã xây dựng một số *stereotype* như <<boundary>>, <<entity>>, <<control>>, <<interface>>,..., ngoài ra chúng ta có thể định nghĩa những loại kiểu mới cho mô hình hệ thống.

1. Lớp biên

Lớp biên (Boundary Class) là lớp nằm trên đường biên của hệ thống với phần thế giới bên ngoài. Nó có thể là biểu mẫu (*form*), báo cáo (*report*), giao diện với các thiết bị phần cứng như máy in, máy đọc ảnh (*Scanner*),... hoặc là giao diện với các hệ thống khác. Trong UML, lớp biên được ký hiệu như hình 4.23(a).

Hình 4.23(a). Lớp biên

Để tìm lớp biên hãy khảo sát sơ đồ ca sử dụng, một tác nhân có thể xác định ít nhất một lớp biên. Nếu có hai tác nhân cùng kích hoạt một ca sử dụng thì chỉ cần tạo ra một lớp biên cho cả hai.

2. Lớp thực thể

Lớp thực thể (Entity Class) là lớp lưu giữ các thông tin mà nó được ghi vào bộ nhớ ngoài. Ví dụ lớp SinhVien là lớp thực thể. Trong UML, lớp thực thể được ký hiệu như hình 4.23(b).

Hình 4.23(b). Lớp thực thể

Lớp thực thể có thể tìm thấy trong các luồng sự kiện và sơ đồ tương tác. Thông thường phải tạo ra các bảng dữ liệu trong CSDL cho mỗi lớp thực thể. Mỗi thuộc tính của lớp thực thể trở thành trường dữ liệu trong bảng dữ liệu.

3. Lớp điều khiển

Lớp điều khiển (Control Class) là lớp làm nhiệm vụ điều phối hoạt động của các lớp khác. Thông thường mỗi ca sử dụng có một lớp điều khiển để điều khiển trình tự các sự kiện xảy ra trong nó. Chú ý, lớp điều khiển không tự thực hiện các chức năng nghiệp vụ của hệ thống, nhưng chúng lại gửi nhiều thông điệp cho những lớp có liên quan, do vậy còn được gọi là lớp quản lý.

Hình 4.23(c). Lớp điều khiển

4.3.3. Quy trình phát triển các lớp đối tượng và sơ đồ lớp có mối quan hệ chủ yếu

Có sáu cách chính để tìm các lớp đối tượng [18]:

- Dựa vào văn bản, kịch bản mô tả bài toán: các danh từ có thể là các đại biểu của lớp.
- Dựa vào danh sách phân loại các phạm trù khái niệm.
- Dựa vào mục đích của các ca sử dụng,
- Dựa vào kinh nghiệm và kiến thức của người phân tích,
- Dựa vào hồ sơ tài liệu những hệ thống có liên quan,
- Dựa vào ý kiến tham khảo với các chuyên gia hệ thống.

Trong quá trình phân tích, thường phải kết hợp các cách trên để tìm ra các lớp của một hệ thống. Ở đây chúng ta tập trung tìm hiểu cách thứ ba để xác định các lớp đối tượng trong quá trình phân tích một hệ thống bởi vì: các lớp đối tượng chính là các nhóm thực thể tham gia thực hiện để đạt được mục đích của hệ thống trong thế giới thực. Hơn nữa, các thực thể và các mục đích phải được mô tả bởi các khách hàng (NSD), chứ không phải bởi các nhà phân tích.

Việc phân tích dựa vào mục đích của các ca sử dụng để xác định lớp các đối tượng được tiến hành theo năm bước:

Bước 1: Xác định mục đích của mỗi ca sử dụng

Ca sử dụng thể hiện những chức năng, nhiệm vụ của hệ thống mà NSD yêu cầu. Mục đích của NSD cũng chính là mục đích của ca sử dụng. Tác nhân của ca sử dụng có mục tiêu là sử dụng ca sử dụng để đạt được những điều mà họ muốn.

Mục đích của ca sử dụng là những mục tiêu mà hệ thống cần thực hiện. Mục đích thường được thể hiện dưới dạng các giá trị (dữ liệu hoặc thông tin) mà ca sử dụng cung cấp cho tác nhân hoặc những đáp ứng của ca sử dụng đó. Như vậy, mục đích là duy nhất ứng với mỗi ca sử dụng.

Để xác định các mục đích, chúng ta phải tìm cách trả lời những câu hỏi sau:

- Mục tiêu của ca sử dụng là gì?

- Ca sử dụng này cung cấp những dịch vụ nào?

- Những giá trị hay những đáp ứng nào mà ca sử dụng có thể cung cấp.

Ví dụ: Sơ đồ ở hình 4.24 mô tả ca sử dụng "*Đăng ký môn học*" và hai tác nhân: *Bộ phận tiếp nhận* và *Phòng đào tạo.* Mục đích của *Bộ phận tiếp nhận* là nhận được *phiếu đăng ký môn học mới* của sinh viên còn mục đích của *Phòng đào tạo* là có *bản sao của phiếu đăng ký* đó.

Hình 4.24. Sơ đồ ca sử dụng "Đăng ký môn học"

Bước 2: Dựa vào các mục đích để xác định các thực thể (lớp)

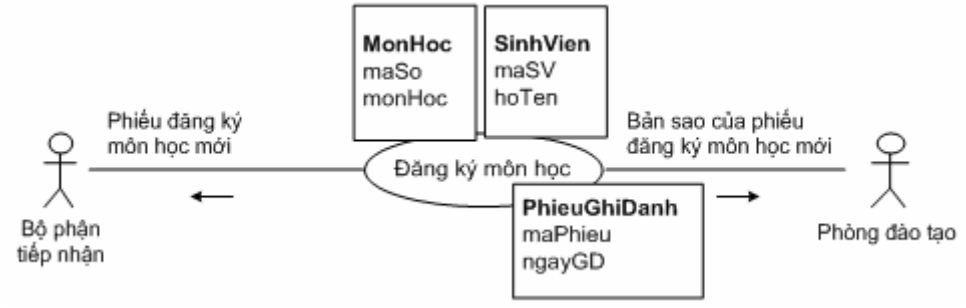
Để thực hiện được những mục đích của ca sử dụng thì bao giờ cũng phải có các thực thể tham gia. *Thực thể* là người nào đó, cái gì đó đóng một vai trò nhất định trong ca sử dụng để thực hiện được những mục đích của ca sử dụng. Một thực thể có thể tham gia vào nhiều ca sử dụng với những vai trò khác nhau. Nói chung, mỗi thực thể đều có những đặc tính (thuộc tính) và những hành vi ứng xử (thao tác xử lý) để phân biệt với những thực thể khác. Mặt khác, những thực thể này lại cộng tác với nhau để cùng nhau đạt được mục đích của ca sử dụng.

Dựa vào các câu hỏi sau để xác định các thực thể/lớp cho mỗi ca sử dụng:

- Những thực thể nào là cần thiết và quan trọng để thực hiện được mục đích của ca sử dụng?

- Những thuộc tính nào của thực thể là cần thiết và quan trọng để thực hiện được mục đích của ca sử dụng?

Ví dụ: Để có phiếu đăng ký môn học thì phải có sự tham gia của các thực thể: môn học, sinh viên và phiếu ghi danh (trả lời câu hỏi thứ nhất). Môn học phải có thuộc tính: mã số của khóa học, tên môn học, sinh viên phải có thuộc tính: mã SV, họ tên, và phiếu ghi danh có: mã phiếu, ngày ghi danh,... (trả lời câu hỏi thứ hai). Từ các mục đích, chúng ta xác định được các lớp và các thuộc tính tương ứng như hình 4.25.



Hình 4.25. Ca sử dụng "Đăng ký môn học" và các thực thể liên quan

Bước 3: Xác định các mối quan hệ chủ yếu (kết hợp) giữa các lớp

Một ca sử dụng có thể liên quan tới nhiều thực thể như hình 4.25, những thực thể đó kết hợp với nhau để thực hiện được mục đích của ca sử dụng.

Thông qua các câu hỏi tác nhân để xác định được mối liên quan giữa các lớp:

- Với mỗi thực thể, nó được sinh ra dựa vào hay bị phụ thuộc vào những thực thể khác? Nếu có, nó phải tham chiếu tới những thực thể đó.

- Với mỗi thực thể, nó có thể tác động vào hay bị tác động bởi những thực thể khác? Nếu có, nó phải tham chiếu tới những thực thể đó.

Ví dụ: Phiếu ghi danh ở hình 4.25 được sinh ra dựa vào sự lựa chọn *môn học* của *sinh viên*. Một *sinh viên* lại có thể *ghi danh nhiều môn học,* một *môn học* được nhiều *sinh viên ghi danh.* Sự tham chiếu giữa các lớp được biểu diễn bằng quan hệ kết hợp trong sơ đồ lớp như hình 4.26.

Hình 4.26. Mối quan hệ giữa các lớp

Bước 4: Xác định các thao tác/hàm thành phần thể hiện sự cộng tác của các lớp trong ca sử dụng

Những thực thể liên quan đến một ca sử dụng thì chúng cộng tác với nhau để thực hiện một số công việc nhằm đạt được mục đích của ca sử dụng. Những công việc đó chính là các hàm thành phần của lớp.

Có thể xác định được các hàm thành phần của lớp thông qua các câu hỏi các tác nhân như sau:

- Ca sử dụng này cần làm gì với mỗi thực thể liên quan với nó?

- Ca sử dụng này cần biết gì về mỗi thực thể liên quan với nó?

- Mỗi thực thể liên quan có thể đóng góp được gì trong ca sử dụng này?

Ví dụ:

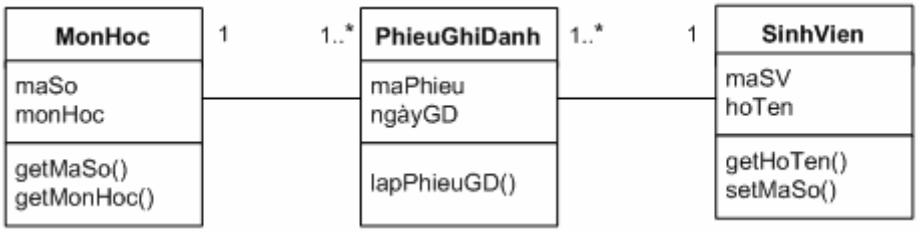
Xét các thực thể ở hình 4.26.

- Ca sử dụng gán *maSV* cho *sinh viên,* nó cần biết về họ tên của sinh viên. *Sinh viên* phải cho ca sử dụng biết họ và tên.

- Ca sử dụng cần phải biết mã số và tên môn học.

- Ca sử dụng tạo lập *phiếu ghi danh mới* với *ngày ghi danh* của sinh viên và với số ghi danh (mã phiếu) duy nhất.

Từ những phân tích như trên chúng ta sẽ xác định được sơ đồ lớp biểu diễn mối quan hệ kết hợp giữa các lớp với các hàm thành phần được bổ sung.



Hình 4.27. Sơ đồ lớp

Bước 5: Kiểm tra các sơ đồ ca sử dụng

Chúng ta có thể kiểm tra các sơ đồ được xây dựng theo các quy tắc sau:

- Kiểm tra các yêu cầu chức năng xem:

+ Tất cả các ca sử dụng có thực hiện được hết các yêu cầu chưa?

+ Mục đích của mỗi ca sử dụng có đúng như các tác nhân yêu cầu không?

- Kiểm tra các thực thể của các ca sử dụng:

+ Các thực thể trong sơ đồ lớp có cần và đủ để thực hiện các mục đích của mçi ca sử dụng hay không?

+ Các thuộc tính của mỗi thực thể có phải là những cái mà ca sử dụng cần biết hay không?

+ Các hàm thành phần của lớp có cần và đủ để thực hiện các mục đích của mỗi ca sử dụng hay không?

Sau này, việc kiểm tra sơ đồ lớp được xây dựng tương ứng với một ca sử dụng sẽ đỡ phức tạp hơn là kiểm tra cả hệ thống.

Chú ý:

Bằng phương pháp nêu trên chúng ta có được danh sách các đại biểu lớp, nhưng không phải tất cả đều có thể là đại biểu lớp. Hãy loại bỏ những đại biểu dư thừa.

- *Lớp dư thừa:* khi có 2 lớp trở lên định nghĩa cho cùng một thực thể thì chỉ cần giữ lại một.

- *Lớp biệt lập:* những lớp không có quan hệ với các lớp khác trong hệ thống thì cần phải loại bỏ.

- *Lớp mơ hồ:* những lớp không có chức năng rõ ràng thì phải định nghĩa lại chính xác hoặc loại bỏ đi./.

5

CÁC MÔ HÌNH PHÂN TÍCH ĐỘNG THÁI

Trong UML, các đối tượng trao đổi với nhau bằng cách gửi các thông điệp cho nhau. Sự trao đổi này thể hiện *sự tương tác* giữa các đối tượng trong hệ thống. Mô hình động thái hệ thống được biểu diễn bằng các sơ đồ tương tác *(Interaction Diagram)*:

- *Sơ đồ trình tự (Sequence Diagram):* mô tả sự trao đổi, tương tác của các đối tượng với nhau theo *trình tự thời gian.* Sơ đồ trình tự bao gồm các phần tử biểu diễn cho các đối tượng, các thông điệp được gửi và nhận trình tự theo thời gian để thực hiện các ca sử dụng của hệ thống.

- *Sơ đồ trạng thái (State Diagram):* mô tả các trạng thái, hành vi của các đối tượng. Sơ đồ trạng thái bao gồm những thông tin về những trạng thái khác nhau của các đối tượng, thể hiện các đối tượng chuyển từ trạng thái này sang trạng thái khác như thế nào, hành vi ứng xử của mỗi đối tượng khi có các sự kiện xảy ra để làm thay đổi trạng thái.

- *Sơ đồ hoạt động (Activity Diagram):* mô tả cách các đối tượng tương tác với nhau nhưng nhấn mạnh về công việc, xác định các hoạt động và thứ tự thực hiện những hoạt động đó.

- *Sơ đồ cộng tác (Collaboration Diagram):* mô tả sự tương tác của các đối tượng với nhau theo ngữ cảnh và không gian công việc.

Việc xây dựng sơ đồ tương tác nhằm mục đích mô hình hóa khía cạnh động của hệ thống (hệ thống đang chạy), thực hiện việc gán trách nhiệm cho các đối tượng. Sơ đồ tương tác chỉ ra các tương tác, bao gồm tập đối tượng, quan hệ và các thông điệp trao đổi giữa chúng. Sơ đồ trình tự và sơ đồ tương tác chỉ ra từng bước của một luồng điều khiển cụ thể trong ca sử dụng. Từ sơ đồ tương tác, người thiết kế có thể phát hiện thêm các lớp, các thao tác cần thực hiện của mỗi lớp,... Do vậy, sơ đồ tương tác trở thành nền tảng cho các bước còn lại của quá trình phát triển phần mềm.

Không phải tất cả các hệ thống đều cần cả bốn sơ đồ trên để mô tả hành vi ứng xử của các đối tượng trong các ca sử dụng. Số các sơ đồ tương tác cần xây dựng hoàn toàn phụ thuộc vào mức độ khó, phức tạp của bài toán ứng dụng. Một số người sử dụng sơ đồ trình tự, sơ đồ trạng thái trong pha phân tích để mô tả hoạt động của hệ thống, sau đó xây dựng sơ đồ cộng tác, sơ đồ hoạt động để phục vụ cho việc thiết kế chi tiết các thành phần của hệ thống. Đối với những hệ thống tương đối đơn giản thì chỉ cần sơ đồ trình tự và sơ đồ trạng thái là đủ. Hai mô hình phân tích đó sẽ được trình bày ở chương này.

5.1. CÁC YẾU TỐ THỂ HIỆN SỰ TƯƠNG TÁC

5.1.1. Các sự kiện và hành động của hệ thống

Trong quá trình tương tác với hệ thống, các tác nhân gây ra các *sự kiện* làm cho hệ thống hoạt động và yêu cầu hệ thống phải thực hiện một số thao tác để đáp ứng các yêu cầu của những tác nhân đó. Các sự kiện phát sinh bởi các tác nhân có liên quan chặt chẽ với những hành động mà hệ thống cần thực hiện. Điều này suy ra là

chúng ta phải xác định được các hoạt động của hệ thống thông qua các sự kiện mà các tác nhân gây ra.

Sự kiện là một hành động kích hoạt hệ thống để nó hoạt động, hoặc tác động lên hệ thống để nó hoạt động tiếp theo một cách nào đó. Nói cách khác, sự kiện là cái gì đó xảy ra và kết quả là nó có thể gây ra một số hoạt động sau đó của hệ thống.

Ví dụ: Sau khi nhập vào hết các mặt hàng mà khách đã chọn mua, *người bán hàng* nhấn phím *"Kết thúc" (EndSale),* thì hệ thống chuyển sang thực hiện chức năng *thanh toán* với khách mua hàng. Việc người bán hàng nhấn phím *"Kết thúc"* chính là sự kiện làm cho hệ thống chuyển sang trạng thái khác.

Các sự kiện có thể là *độc lập* hoặc có *liên hệ* với nhau. Ví dụ: *Nhập thông tin về các mặt hàng* và *Thanh toán* là hai sự kiện phụ thuộc, sự kiện *Thanh toán* phải xảy ra sau sự kiện *nhập thông tin*, còn sự kiện *Trả tiền mặt* và *trả bằng séc* là độc lập với nhau.

Những sự kiện độc lập có thể là những sự kiện đồng thời. Bởi vì những sự kiện này không phụ thuộc vào nhau nên có thể xảy ra trong cùng một thời điểm. Ví dụ: *Hiển thị số tiền dư trả lại cho khách* và *Cập nhật các mặt hàng* trong hệ thống bán hàng là hai sự kiện độc lập với nhau và có thể xảy ra đồng thời.

Các sự kiện cũng có thể chia thành hai loại: các sự kiện bên trong và các sự kiện bên ngoài.

- *Sự kiện bên trong* là sự kiện xảy ra ngay bên trong hệ thống, ở trong một đối tượng và được kích hoạt bởi đối tượng khác.
- *Sự kiện bên ngoài* là sự kiện được tạo ra ở bên ngoài phạm vi của hệ thống. *Sự kiện vào của hệ thống* là những sự kiện bên ngoài tác động vào hệ thống và do các tác nhân tạo ra.

Hoạt động của hệ thống là những thao tác mà hệ thống phải thực hiện để trả lời, đáp ứng cho những sự kiện vào. Một số hoạt động của hệ thống có thể tạo ra những *sự kiện ra* cho các tác nhân để thông báo những sự kiện tiếp theo của hệ thống có thể xảy ra, hoặc nhắc các tác nhân phải hành động như thế nào để có những thông tin mong muốn. Như vậy, *các sự kiện vào sẽ kích hoạt hệ thống hoạt động và hệ thống hoạt động là để trả lời cho các sự kiện vào mà các tác nhân tạo ra.*

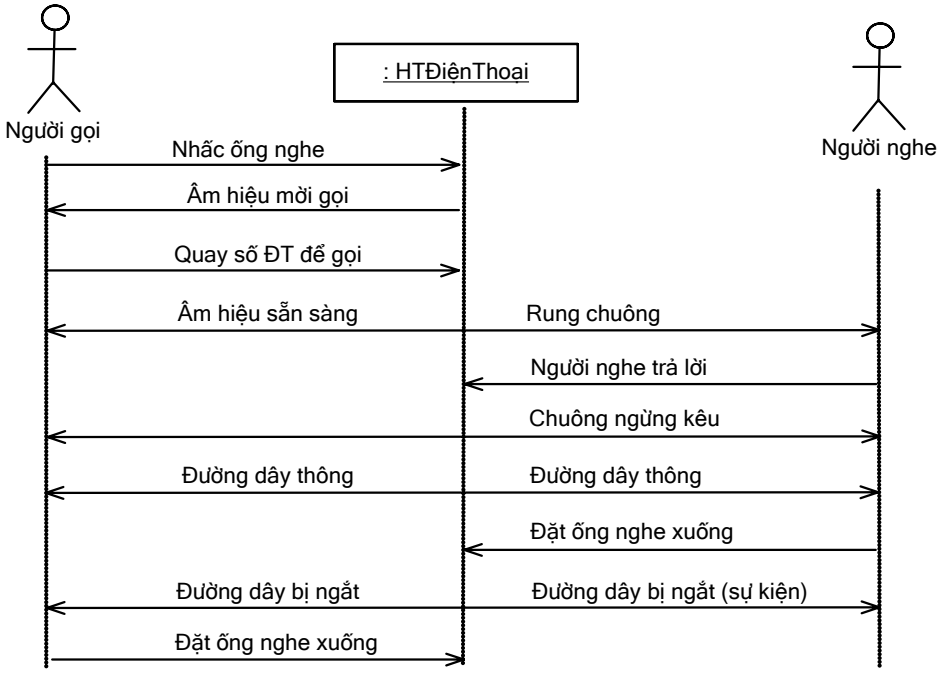
Các sự kiện và hoạt động của hệ thống thường được sử dụng để mô tả "kịch bản" cho ca sử dụng.

Ví dụ: Trong kịch bản của ca sử dụng "*Gọi điện thoại*" có hai tác nhân là *người gọi* và *người nghe.* Dãy các sự kiện của ca sử dụng này được mô tả bằng bảng 5.1.

Bảng 5.1. Mô tả hình thức kịch bản ca sử dụng "gọi điện thoại"

Các sự kiện vào	Các sự kiện ra
1. Người gọi nhấc ống nghe	2. Tiếng bíp bíp báo hiệu máy điện thoại sẵn sàng cho cuộc gọi
3. Người gọi quay số (ví dụ 5652288)	4. Tín hiệu điện thoại được nối với người nghe
	5. Điện thoại của người được gọi rung chuông (nếu không bận đàm thoại)
6. Người nghe nhấc ống nghe và trả lời	7. Chuông ngừng kêu
	8. Đường dây điện thoại được kết nối để hai người đàm thoại với nhau
9. Người nghe đặt ống nghe xuống	10. Đường dây bị ngắt
11. Người gọi đặt ống nghe xuống	

Các sự kiện vào là do *người gọi* tạo ra, các sự kiện ra lại tác động đến *người nghe*. Hệ thống điện thoại sẽ hoạt động để trả lời cho các sự kiện vào đồng thời phát sinh ra các sự kiện ra. Dãy các sự kiện và hoạt động của *Hệ thống điện thoại* được mô tả một cách trực quan hơn bằng sơ đồ:



Hình 5.1. Sơ đồ dãy các sự kiện khi thực hiện ca sử dụng "Gọi điện thoại"

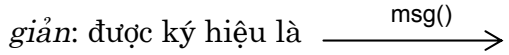
5.1.2. Trao đổi thông điệp giữa các đối tượng

Trong các sơ đồ tương tác, các đối tượng trao đổi với nhau bằng các thông điệp. Các đối tượng thường được gửi, nhận theo:

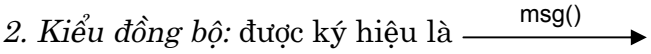
- Các giao thức trao đổi tin (*Communication Protocol*),
- Các lời gọi hàm: một đối tượng gọi một hàm của đối tượng khác để xử lý các yêu cầu.

Có thể thực hiện việc trao đổi thông tin trên mạng, hoặc trong một máy tính và phần lớn thực hiện trong thời gian thực (*Real-Time*).

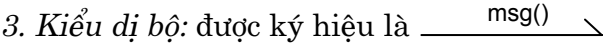
Có ba kiểu thông điệp trao đổi chính:

1. *Kiểu đơn giản*: được ký hiệu là 

Biểu diễn cho dòng điều khiển để chuyển thông điệp *msg()* từ đối tượng này sang đối tượng khác mà không cần biết chi tiết về sự trao đổi thông tin.

2. Kiểu đồng bộ: được ký hiệu là 

Biểu diễn cho dòng điều khiển được đồng bộ hóa, nghĩa là khi có nhiều thông điệp gửi đến (nhận được) thì thông điệp trước (có mức độ ưu tiên cao) phải được xử lý xong và sau khi đã kết thúc công việc thì thông điệp tiếp theo mới được xử lý.

3. Kiểu dị bộ: được ký hiệu là 

Biểu diễn cho dòng điều khiển thông điệp không cần đồng bộ, nghĩa là khi có nhiều thông điệp gửi đến (hay nhận được) thì các thông điệp đó được xử lý mà không cần biết những thông điệp khác đã kết thúc hay chưa và thứ tự thực hiện là không quan trọng.

5.2. SƠ ĐỒ TRÌNH TỰ

Theo yêu cầu của giai đoạn phân tích, chúng ta chỉ cần định nghĩa hệ thống như một *hộp đen*, trong đó hành vi của hệ thống thể hiện được *những gì* (*What?*) nó cần thực hiện và không cần thể hiện những cái đó thực hiện như thế nào (*How?*). Vì vậy, nhiệm vụ chính của chúng ta trong giai đoạn này là xác định và mô tả được các hoạt động của hệ thống theo yêu cầu của các tác nhân. Nghĩa là phải tìm được các sự kiện, các thao tác (sự tương tác) của các đối tượng trong từng ca sử dụng. Sơ đồ trình tự giúp chúng ta thực hiện được những nhiệm vụ đó.

Sơ đồ trình tự (sequence diagram) là sơ đồ tương tác theo trình tự thời gian của các giao tiếp bằng thông điệp giữa các đối tượng đang hoạt động trong hệ thống. Mỗi ca sử dụng có nhiều luồng dữ liệu. Mỗi sơ đồ trình tự biểu diễn một luồng dữ liệu.

5.2.1. Các thành phần của sơ đồ trình tự

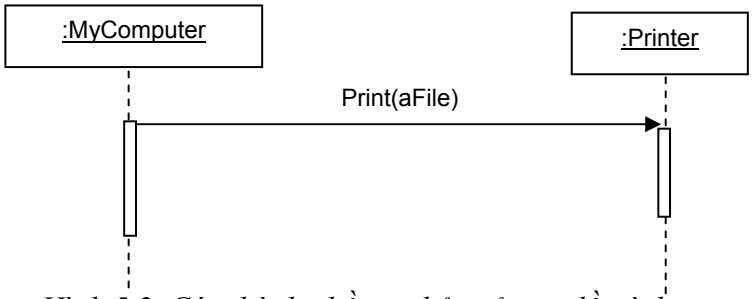
Sơ đồ trình tự bao gồm các phần tử biểu diễn đối tượng, thông điệp và thời gian. Sơ đồ trình tự được thể hiện theo hai trục:

- Trục dọc trên xuống chỉ thời gian xảy ra các sự kiện, hay sự truyền thông điệp, được biểu diễn bằng các đường gạch - gạch thẳng đứng bắt đầu từ đỉnh đến đáy của sơ đồ.
- Trục ngang từ trái qua phải là dãy các đối tượng tham gia vào việc trao đổi các thông điệp với nhau theo chiều ngang, có thể có cả các tác nhân.

Đối tượng được biểu diễn bằng hình chữ nhật trong đó có tên đối tượng cụ thể và/hoặc tên lớp cùng được gạch dưới (hoặc tên lớp được gạch dưới biểu diễn cho một đối tượng bất kỳ của lớp đó).

Sơ đồ trình tự được đọc từ trên xuống dưới, từ trái sang phải. Thứ tự các đối tượng trong sơ đồ phải được sắp xếp sao cho đơn giản nhất có thể để dễ quan sát. Thời gian thực hiện một thông điệp của một đối tượng, hay còn gọi là hoạt động của đối tượng được biểu diễn bằng hình chữ nhật hẹp dọc theo trục thẳng đứng của đối tượng đó.

Ví dụ:

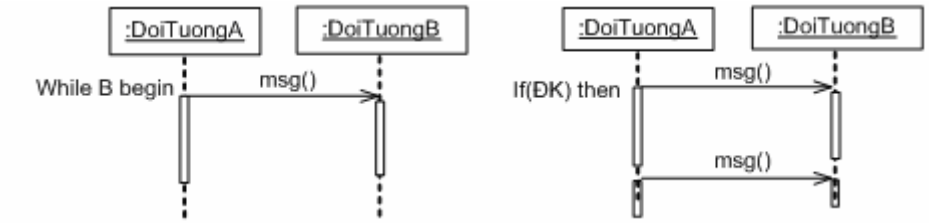


Hình 5.2. Các thành phần cơ bản của sơ đồ trình tự

Mỗi thông điệp đều có tên gọi thể hiện được ý nghĩa của thông tin cần gửi và các tham số về dữ liệu liên quan. Thông thường đó là các *lời gọi hàm*. Khi định nghĩa các lớp sau này thì mỗi thông điệp nhận được sẽ trở thành một phương thức. Một đối tượng có thể gửi thông điệp tới chính nó. Những thông điệp này gọi là phản thân, nó chỉ ra rằng đối tượng gọi chính các thao tác của mình để thực hiện.

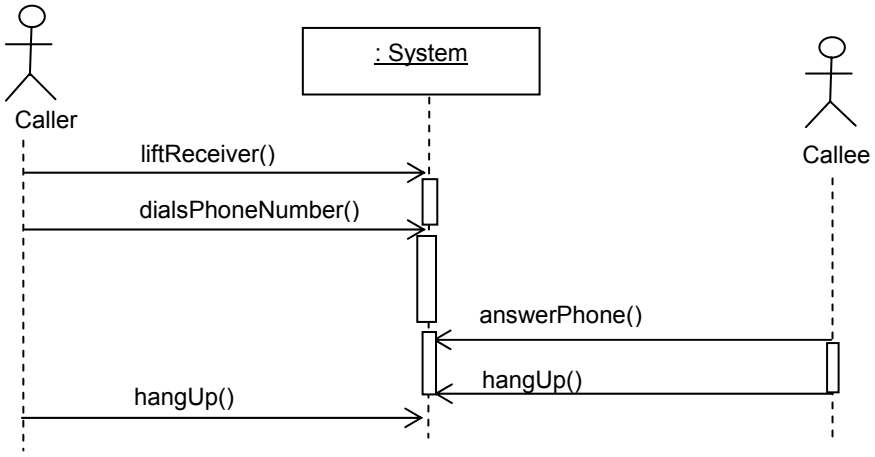
Chú ý rằng tác nhân ngoài kích hoạt các đối tượng trong sơ đồ trình tự hoạt động, nhưng nó không phải là phần tử của sơ đồ loại này.

Mặt khác, trong khi *Notes* được sử dụng để chú thích cho các đối tượng trong sơ đồ trình tự thì *Scripts* được sử dụng để chú thích cho các thông điệp. Chúng được đặt bên trái của một hoạt động của đối tượng, ngang với mức thông điệp cần chú thích. *Scripts* được sử dụng để mô tả các điều kiện logic hoặc điều kiện lặp, ví dụ lệnh IF hay WHILE,... trong quá trình trao đổi thông điệp. *Scripts* không hỗ trợ để tạo ra mã chương trình, nhưng nó giúp cho người phát triển hệ thống biết được những điều kiện cần phải tuân theo.



Hình 5.3. Scripts trong sơ đồ trình tự

Ca sử dụng "*Gọi điện thoại",* trong đó có hai tác nhân là *Caller* (người gọi) và *Callee* (người nghe). Hệ thống nhận được các sự kiện vào được ký hiệu là các lời gọi hàm: *liftReceiver()* (nhấc ống nghe), *dialsPhoneNumber()* (quay số), *answerPhone()* (trả lời điện thoại) và *hangUp()* (đặt ống nghe xuống, nhấc tay lên). Sơ đồ trình tự mô tả cho các hoạt động trên được xác định như hình 5.4:



Hình 5.4. Sơ đồ trình tự mô tả hoạt động "Gọi điện thoại"

5.2.2. Xây dựng sơ đồ trình tự cho một luồng dữ liệu trong mỗi ca sử dụng

1. Các bước xây dựng sơ đồ trình tự

- Xác định các tác nhân, các đối tượng tham gia vào ca sử dụng và vẽ chúng theo hàng ngang trên cùng theo đúng các ký hiệu,

- Xác định những thông điệp (lời gọi hàm) mà tác nhân cần trao đổi với một đối tượng nào đó, hoặc giữa các đối tượng tương tác với nhau theo trình tự thời gian và vẽ lần lượt các hoạt động đó từ trên xuống theo thứ tự thực hiện trong thực tế. Cần xác định chính xác các loại thông điệp trao đổi giữa các đối tượng là đơn giản, đồng bộ hay dị bộ.

2. Kỹ thuật xây dựng sơ đồ trình tự liên quan đến CSDL

Cũng giống như sơ đồ cộng tác, sơ đồ trình tự được xây dựng theo các bước sau:

+ Bước thứ nhất chỉ tập trung vào phục vụ khách hàng. Không ánh xạ ngay thông điệp thành thao tác và không ánh xạ ngay đối tượng thành lớp. Sơ đồ tạo ra trong bước này dành cho nhà phân tích, khách hàng và những ai chỉ quan tâm đến tác nghiệp, để biết luồng logic chạy trong hệ thống như thế nào.

+ Bước thứ hai tập trung vào phục vụ những người tham gia dự án ngoài khách hàng như người phát triển hệ thống, người kiểm tra chất lượng,... bằng cách bổ sung chi tiết hơn các đối tượng mới, đặc biệt là *đối tượng điều khiển* để điều khiển trình tự xuyên qua kịch bản. Đối tượng điều khiển không xử lý tác nghiệp nào, chúng chỉ gửi thông điệp đến các đối tượng khác. Nó có trách nhiệm điều phối công việc của các đối tượng khác nên được coi là đối tượng quản trị. Lợi thế của việc sử dụng đối tượng này là khả năng tách logic tác nghiệp khỏi logic trình tự. Nếu trình tự thay đổi thì chỉ có đối tượng điều

khiển thay đổi. Ngoài ra có thể bổ sung đối tượng quản lý an toàn, quản lý lỗi hay kết nối CSDL. Rất nhiều đối tượng chỉ xây dựng một lần và có thể sử dụng lại trong các ứng dụng khác nhau.

Có hai chức năng thường được sử dụng trong CSDL là lưu trữ và truy xuất. Nếu đối tượng biết về CSDL thì nó tự lưu trữ vào CSDL và truy xuất được từ CSDL. Khi đó ta nói rằng logic ứng dụng và logic CSDL không tách rời nhau. Ngược lại khi chúng tách rời nhau, chúng ta phải tạo ra một đối tượng mới có trách nhiệm với CSDL. Đối tượng mới này được gọi là đối tượng *quản trị giao dịch (transaction manager).* Nó biết cách truy vấn một đối tượng dữ liệu cụ thể từ CSDL cũng như lưu trữ đối tượng đó vào CSDL.

+ Sau khi bổ sung đầy đủ các đối tượng, bước tiếp theo là ánh xạ từng đối tượng vào lớp có sẵn hay lớp mới, ánh xạ từng thông điệp trong sơ đồ thành thao tác. Cuối cùng là đặc tả đối tượng và đặc tả thông điệp như gán đồng bộ, gán tần suất thông điệp và cách thức lưu trữ đối tượng...

Rose cho khả năng thiết lập thuộc tính lưu trữ cho mỗi đối tượng trong sơ đồ (đã đề cập trong chương 4).

5.2.3. Ví dụ về xây dựng các sơ đồ trình tự cho hệ thống bán hàng

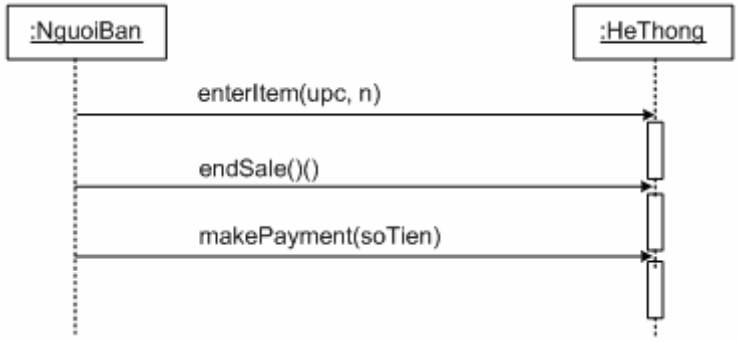
Đối với một số sơ đồ trình tự mô tả các hành vi của hệ thống, hành vi của hệ thống được thể hiện trong các sơ đồ trình tự thông qua sự tương tác của các đối tượng với nhau. Sự tương tác đó được thể hiện bằng các lời gọi hàm và sẽ được khai báo công khai (*public*) trong các lớp.

1. Mô tả hoạt động của hệ thống bán hàng

Người ta mô tả bằng lời hoạt động giữa người bán hàng và hệ thống: sau khi chọn đủ hàng, người mua sẽ đưa giỏ (hoặc xe) hàng

đến quầy để trả tiền. Người bán hàng phải nhập vào các thông tin và số lượng của mỗi mặt hàng. Những thông tin về mặt hàng thường được xác định thông qua mã sản phẩm *upc (*thuộc lớp *UPC).* Vậy đối tượng :NguoiBan phải gửi đến cho :HeThong/:HeBanHang (một hệ thống bán hàng) thông điệp *enterItems()(upc, n),* để nhập vào mã sản phẩm *upc* và *n* đơn vị. Khi đã nhập xong tất cả các mặt hàng của khách muốn mua thì phải báo cho hệ thống biết là đã nhập xong bằng cách nhấn nút *Kết thúc*, nghĩa là gửi cho hệ thống thông điệp *endSale().* Hệ thống sẽ thông báo cho khách mua hàng biết tổng số tiền phải trả. Nếu khách hàng trả bằng tiền mặt thì :NguoiBan gửi tiếp đến cho :HeThong thông điệp *makePayment(soTien).*

Các hoạt động trên được mô tả bằng sơ đồ trình tự như hình 5.5:

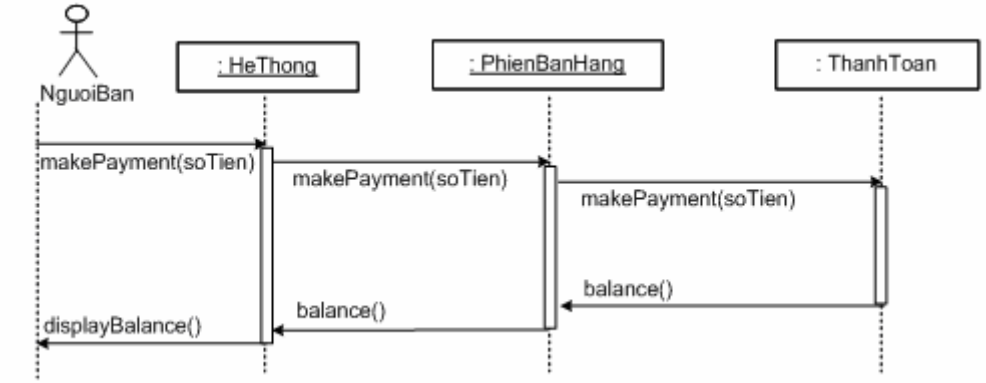


Hình 5.5. Sơ đồ trình tự mô tả sự tương tác giữa người bán và hệ thống

2. Sơ đồ trình tự mô tả ca sử dụng "Thanh toán tiền mặt"

Khi đối tượng :HeThong nhận được yêu cầu thanh toán của khách hàng thông qua *người bán*, nghĩa là nó nhận được thông điệp *makePayment(soTien)* thì nó phải gửi cho một đối tượng *một phiên bán hàng*: PhienBanHang một yêu cầu tương tự. Đối tượng *một phiên bán hàng* lại yêu cầu đối tượng tạo ra một đối tượng *một lần thanh toán*: ThanhToan để thanh toán theo *soTien* mà khách đưa. Đối tượng *một lần thanh toán* gửi lại thông điệp *balance()* cho hệ

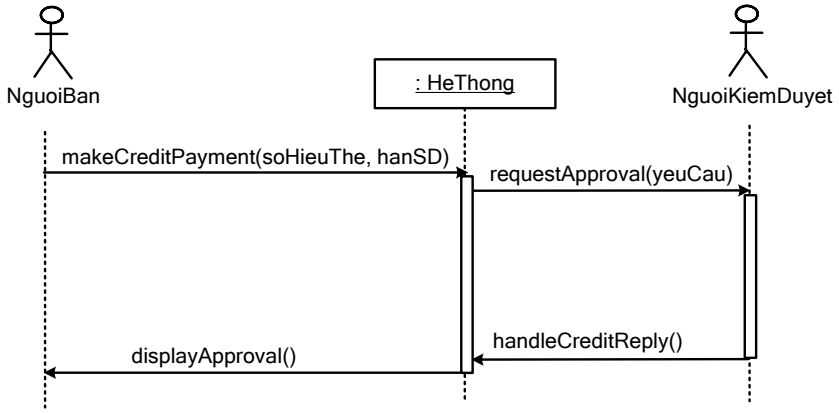
thống *:HeThong biết số tiền dư (soTien* khách đưa trừ đi số tiền mua hàng) phải trả lại cho khách và để hiển thị *(displayBalance())* lên màn hình, để *người bán* thông báo lại cho khách hàng khi khách trả tiền mặt. Sơ đồ trình tự mô tả dãy các hành động trao đổi giữa các đối tượng nêu trên được thiết lập như hình 5.6:



Hình 5.6. Sơ đồ trình tự mô tả ca sử dụng "Thanh toán (tiền mặt)"

3. Sơ đồ trình tự mô tả ca sử dụng "Thanh toán bằng thẻ tín dụng"

Khi thanh toán tiền hàng, khách hàng có thể trả bằng thẻ tín dụng. Thẻ tín dụng được xác định thông qua *soHieuThe* (*số hiệu thẻ*) và *hanSD* (*hạn sử dụng*). Điều đó được thể hiện bằng việc gửi một thông điệp *makeCreditPayment(soHieuThe, hanSD)* cho hệ thống. Thông thường hệ thống phải gửi một yêu cầu kiểm duyệt thẻ *requestApproval(yeuCau))* tới bộ phận kiểm duyệt thẻ tín dụng là tác nhân ngoài của hệ thống. Hệ thống sẽ dựa vào kết quả trả lời của bộ phận kiểm duyệt để thanh toán với khách hàng bằng cách trừ đi số tiền mua hàng trong thẻ tín dụng nếu nó hợp pháp, nghĩa là thực hiện *handleCreditReply()* (xử lý thẻ đã kiểm duyệt), dựa vào kết quả của *bộ phận kiểm duyệt*. Sơ đồ trình tự mô tả sự tương tác giữa *người bán hàng, hệ thống và bộ phận kiểm duyệt thẻ* được thiết lập như hình 5.7:



Hình 5.7. Sơ đồ trình tự mô tả sự trao đổi giữa người bán, hệ thống và bộ phận kiểm duyệt

Tương tự như trên, chúng ta có thể tạo ra những sơ đồ trình tự cho các ca sử dụng khác của hệ thống.

5.2.4. Ghi nhận các hoạt động của các lớp đối tượng

Khi xây dựng các lớp, chúng ta tập trung chủ yếu vào việc phát hiện các thuộc tính của chúng. Khi xây dựng được các sơ đồ trạng thái và sơ đồ trình tự, chúng ta đã hiểu hơn về sự tương tác giữa các đối tượng trong hệ thống. Trên cơ sở đó, chúng ta có thể dần dần xác định được hành vi ứng xử của các đối tượng, nghĩa là xác định các *thao tác* của các lớp.

Nghiên cứu các sơ đồ trình tự chúng ta nhận thấy: *Khi có một thông điệp mgs() gửi đến cho một đối tượng thì lớp của đối tượng đó phải có thao tác là msg().*

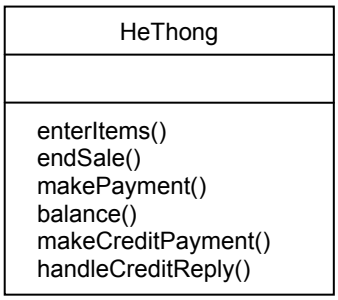
Dựa vào nguyên lý này và căn cứ vào các sơ đồ trình tự ở trên, lớp HeThong sẽ có các thao tác sau (hình 5.8):

Nhận xét:

- Mỗi lần xây dựng thêm một sơ đồ trình tự mới có liên quan đến một lớp nào đó, thì lớp đó có thể lại được bổ sung thêm một số thao tác mới.

- *Để đơn giản,* trong giai đoạn xây dựng sơ đồ, chúng ta có thể bỏ qua các đối số của các thao tác của các lớp.

- Các thao tác cũng có phạm vi truy cập giống như các thuộc tính trong các lớp đối tượng. Do vậy, người thiết kế có thể khai báo chúng là công khai (*public*), được bảo vệ (*protected*), sở hữu riêng (*private*) hay *mặc định* như đã nêu ở phần định nghĩa các thuộc tính.



Hình 5.8. Các thao tác của hệ thống được ghi nhận vào lớp HeThong

5.2.5. Các hợp đồng/các đặc tả về hoạt động của hệ thống

Qua các phân tích trên, sơ đồ trình tự mới chỉ cho chúng ta biết tên của những nhiệm vụ mà mỗi đối tượng cần phải thực hiện khi nhận được một thông điệp, nhưng chưa mô tả cách thực hiện những công việc đó như thế nào. Để hiểu rõ hơn về những hành vi của các đối tượng và để hỗ trợ cho thiết kế và cài đặt các lớp sau này, chúng ta cần xây dựng các *hợp đồng (Contract)* hay các đặc tả cho *những hoạt động (thao tác, hàm)* đã xác định được. Hợp đồng cho các hoạt động của hệ thống mô tả về sự thay đổi trạng thái mà khi bắt đầu thì thỏa mãn *tiền điều kiện (pre-conditions)* và sau đó, khi kết thúc sẽ thỏa mãn *hậu điều kiện (post-conditions).* Những hợp đồng này có thể lưu ở những tệp khác nhau và được gán vào sơ đồ tương ứng.

Một cách hình thức, hợp đồng cho hoạt động Op() có thể viết theo bộ ba của Hoare:

{Pre-conditions} Op() {Post-conditions}

thể hiện rằng, nếu hoạt động *Op()* bắt đầu từ trạng thái thỏa mãn *Pre-conditions* thì sau khi kết thúc thực hiện, hệ thống sẽ ở trạng thái thỏa mãn *Post-conditions.*

- *Pre-conditions:* là những điều kiện mà trạng thái của hệ thống được giả thiết là thỏa mãn trước khi thực hiện một thao tác, một hàm.
- *Post-conditions:* là những điều kiện mà trạng thái của hệ thống phải thỏa mãn sau khi thực hiện xong một thao tác, một hàm.

Pre-condition và *Post-condition* là những mệnh đề *boolean* (mệnh đề điều kiện logic).

Ngoài *Pre-conditions* và *Post-coditions*, chúng có thể bổ sung thêm *mô tả tóm tắt về trách nhiệm, kiểu, lớp nào, tham chiếu tới những chức năng nào, chú thích, những ngoại lệ và kết quả,...*

1. Các bước xây dựng các hợp đồng/đặc tả

- Từ các sơ đồ trình tự, xác định các thao tác, hàm, hoạt động của các lớp trong hệ thống,

- Với mỗi thao tác trên hãy xây dựng một hợp đồng/đặc tả,

- Mô tả tóm tắt những trách nhiệm chính mà hệ thống phải thực hiện khi thực thi thao tác này.

- Trong *Post-conditions* phải nêu được các trạng thái của các đối tượng sau khi kết thúc thao tác.

- Để mô tả *Post-conditions*, hãy căn cứ vào:

+ Việc tạo lập, hủy bỏ các đối tượng

+ Những thay đổi của các thuộc tính

+ Thiết lập hay hủy bỏ các mối liên kết.

2. Đặc tả các thao tác của lớp HệThống

a. Hợp đồng/đặc tả thao tác nhập các thông tin về các mặt hàng

Bảng 5.2. Hợp đồng/đặc tả thao tác enterItems()

Tên gọi thao tác:	enterItems(upc: UPC, n: Int) (UPC- mã sản phẩm)
Trách nhiệm:	Nhập lần lượt *các thông tin* về những mặt hàng mà khách đã chọn mua và đưa chúng vào *phiên bán hàng*. Hiển thị các thông tin mô tả và giá bán của từng mặt hàng.
Kiểu/Lớp:	System (Hệ thống)
Tham chiếu tới:	R1.1, R1.3, R1.9 và ca sử dụng Bán hàng
Chú ý:	Sử dụng phương pháp truy nhập nhanh vào CSDL
Ngoại lệ:	Nếu upc không hợp lệ thì có lỗi
Kết quả(Output):	
Pre-conditions:	UPC được biết trước
Post-conditions:	+ Nếu mặt hàng nhập vào là đầu tiên thì phải tạo ra một đối tượng phienBanHang của lớp PhienBanHang và kết hợp nó vào hệ thống, + Tạo ra một đối tượng dongBanHang của lớp DongBanHang cho mặt hàng vừa nhập vào và gán dongBanHang.soLuong = n (đối số của hàm), + dongBanHang được liên kết với moTaMatHang dựa vào mã upc.

b. Hợp đồng/đặc tả thao tác kết thúc nhập hàng

Bảng 5.3. Hợp đồng thao tác kết thúc nhập hàng

Tên gọi:	endSale()()
Trách nhiệm:	Ghi nhận những mặt hàng đã được nhập. Hiển thị tổng số tiền bán hàng.
Kiểu/Lớp:	System (Hệ thống)
Tham chiếu tới:	R1.2, và ca sử dụng Bán hàng
Chú ý:	
Ngoại lệ:	Nếu phiên bán hàng không thực hiện được thì có lỗi
Kết quả (Output):	
Pre-conditions:	UPC được biết trước
Post-conditions:	PhienBanHang.ketThuc nhận giá trị true.

c. Hợp đồng/đặc tả thao tác thực hiện thu tiền mặt

Bảng 5.4. Hợp đồng thao tác thực hiện thu tiền mặt

Tên gọi:	makePayment(soTien: DVT), trong đó DVT là lớp các loại tiền
Trách nhiệm:	Ghi nhận các thông tin liên quan đến việc thanh toán, tính số tiền dư cần trả lại cho khách.
Kiểu/Lớp:	System (Hệ thống)
Tham chiếu tới:	R2.1, và ca sử dụng Bán hàng
Chú ý:	
Ngoại lệ:	Nếu phiên bán hàng không kết thúc thì có lỗi. Nếu soTien nhỏ hơn tongSoTien thì cũng có lỗi
Kết quả (Output):	
Pre-conditions:	
Post-conditions:	+ Một đối tượng thanhToan được tạo lập, + số tiền dư là soTien - ThanhToan.tongSoTien + thanhToan được liên kết với phienBanHang đã được tạo lập ở hợp đồng trước để thực hiện việc cập nhật những mặt hàng đã bán, tổng số tiền,...

Lưu ý: Đối với những ca sử dụng có nhiều đối tượng tham gia thì sơ đồ trình tự là khá phức tạp, do vậy nó không thích hợp. Muốn hiểu rõ hoạt động của các đối tượng thì tốt nhất là *nên phân tách những ca sử dụng phức hợp thành các ca sử dụng tương đối đơn giản, dễ hiểu và mô tả được bằng sơ đồ trình tự một cách đơn giản hơn.*

5.3. SƠ ĐỒ TRẠNG THÁI

Sơ đồ trạng thái (State Diagram, State Machine Diagram, State Chart Diagram) mô tả các thông tin về các trạng thái khác nhau của đối tượng, thể hiện các đối tượng chuyển từ trạng thái này sang trạng thái khác như thế nào, hoạt động của đối tượng trong mỗi trạng thái ra sao. Sơ đồ trạng thái thể hiện chu kỳ hoạt động của đối tượng, các hệ thống con và của cả hệ thống, từ khi chúng được tạo ra cho đến khi kết thúc.

Sơ đồ trạng thái mô tả:

- Các trạng thái mà các đối tượng có thể có,

- Các sự kiện: các thông điệp nhận được, các lỗi có thể xuất hiện, điều kiện nào đó có thể trở thành đúng (true), khoảng thời gian đã qua,... tác động lên trạng thái để làm biến đổi.

Sơ đồ trạng thái là giải pháp tốt để mô hình hóa hành vi động của các lớp đối tượng. Trong một dự án, không nhất thiết phải tạo ra các sơ đồ trạng thái cho tất cả các lớp. Tuy nhiên, đối với những lớp có nhiều hành vi động, có nhiều trạng thái hoạt động khác nhau thì sơ đồ trạng thái là hữu ích, giúp chúng ta hiểu rõ hệ thống hơn.

5.3.1. Trạng thái và sự biến đổi trạng thái

Mọi đối tượng trong hệ thống đều có chu kỳ sống và mỗi thời điểm đều có một trạng thái nào đó. Ví dụ, *người bán hàng* trong hệ thống bán hàng *đang bán hàng, phiên bán hàng* đã được *thanh toán,...*

Trạng thái là một trong các điều kiện có thể để đối tượng tồn tại, là kết quả của một hoạt động trước đó của đối tượng. Trạng thái của đối tượng thường được mô tả trong hình chữ nhật góc tròn và được xác định bởi:

- *Tên gọi trạng thái,* thường bắt đầu bằng động từ,
- *Biến trạng thái* mô tả các giá trị hiện thời của trạng thái,
- *Hoạt động* là hành vi mà đối tượng sẽ thực hiện khi nó ở vào trạng thái đó.

Hoạt động của trạng thái được mô tả hình thức như sau:

event_name argument_list '/' action_exp

Trong đó:

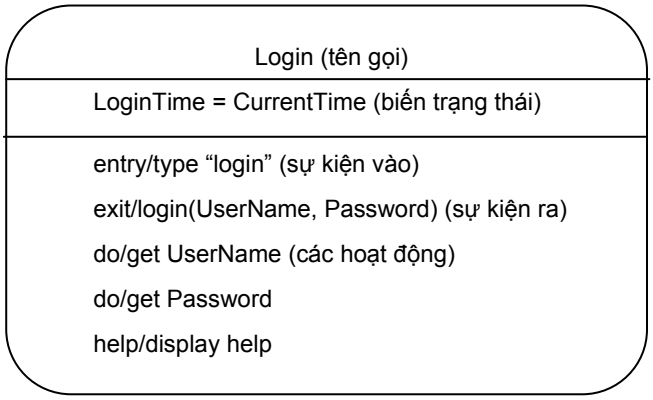
- event_name: Tên của sự kiện, có thể là một trong các sự kiện chuẩn: exit (thoát ra), entry (vào), do (thực hiện).

- argument_list: danh sách các sự kiện,

- action_exp: những hoạt động cần thực hiện bao gồm các lời gọi hàm, thao tác trên các biến trạng thái,...

Ví dụ:

trạng thái Login được mô tả trong UML:



Hình 5.9. Trạng thái Login

Khi hệ thống ở trạng thái *Login* thì biến *LoginTime (thời gian khi khởi nhập)* được gán là *CurrentTime (thời gian hiện thời)* của máy tính. Sự kiện vào của trạng thái này là gõ từ "*login*" và để thoát ra khỏi trạng thái này thì phải thực hiện lời gọi hàm *login (UserName, Password).* Các hoạt động của đối tượng ở trạng thái này là: Nhận vào *UserName (tên người sử dụng), Password (mật khẩu),* và hiển thị sự trợ giúp *display help.*

Lưu ý:

- Khi không cần mô tả chi tiết thì có thể chỉ cần tên gọi để xác định trạng thái trong các sơ đồ.

- Có hai trạng thái đặc biệt là:

+ *trạng thái bắt đầu* được ký hiệu: ●⟶

+ *trạng thái kết thúc*, được ký hiệu: ⟶◉

Sơ đồ trạng thái thường có trạng thái bắt đầu, còn trạng thái kết thúc thì có thể có hoặc không tùy vào chu kỳ hoạt động của các đối tượng.

Trong sơ đồ, *đường mũi tên chỉ ra sự biến đổi từ một trạng thái sang trạng thái khác* khi có các sự kiện xảy ra làm thay đổi các trạng thái. Trạng thái của đối tượng sẽ *bị thay đổi* khi có cái gì đó xảy ra, nghĩa là khi có một hay nhiều sự kiện xuất hiện. *Sự biến đổi trạng thái hay sự chuyển trạng thái* thể hiện mối quan hệ giữa các trạng thái với nhau.

Sự chuyển trạng thái được thể hiện trong sơ đồ bằng mũi tên có nhãn là sự kiện, thao tác (hàm có đối số), hoặc *điều kiện thường trực (guard)*. Sự chuyển trạng thái có thể là đệ quy, nghĩa là trong một điều kiện nhất định, một đối tượng có thể quay lại trạng thái cũ của nó.

5.3.2. Xác định các trạng thái và các sự kiện

Để xác định được các trạng thái và các sự kiện chúng ta cần trả lời cho các câu hỏi sau:

- Một đối tượng có thể ở những trạng thái nào? Liệt kê tất cả các trạng thái có thể có trong hệ thống của mỗi đối tượng.

- Những sự kiện nào có thể xuất hiện? Bởi vì sự kiện có thể làm biến đổi trạng thái, do vậy, từ các sự kiện có thể xác định được các trạng thái của đối tượng.

- Những trạng thái mới nào sẽ xuất hiện? Từ một trạng thái, đối tượng có thể chuyển sang trạng thái mới khi một số sự kiện xác định xuất hiện.

- Ở mỗi trạng thái, hoạt động của đối tượng là gì?

- Sự tương tác giữa các đối tượng là gì? Sự tương tác giữa các đối tượng thường gắn chặt với các trạng thái của đối tượng.

- Những sự kiện, hay chuyển đổi trạng thái nào là không thể xảy ra? Một số sự kiện, hay trạng thái không thể chuyển đổi sang trạng

thái khác được, ví dụ khi khách mua hàng *trả bằng thẻ tín dụng không hợp pháp* thì *phiên bán đó không thực hiện được.*

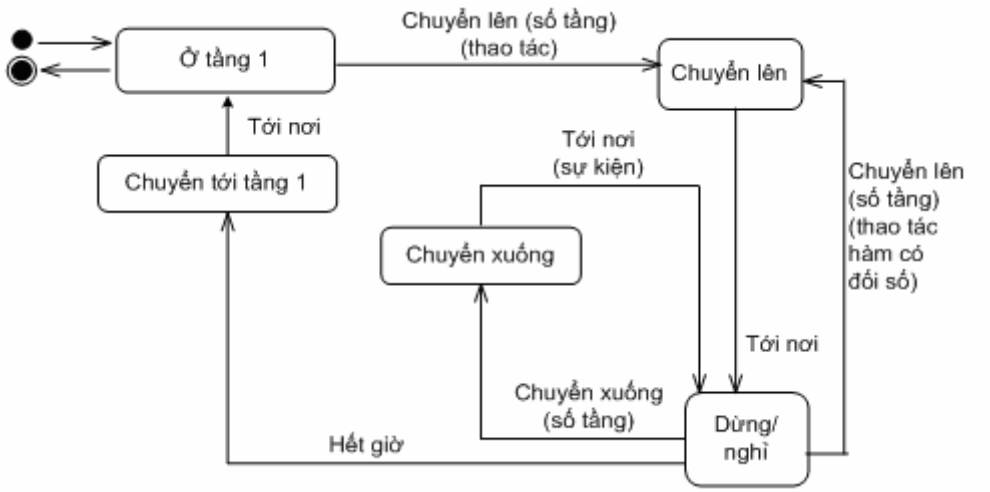
- Cái gì làm cho đối tượng được tạo ra? Đối tượng thường được tạo ra bởi một, hay một số sự kiện.

- Cái gì làm cho đối tượng bị hủy bỏ? Đối tượng thường được loại bỏ khi không còn cần thiết nữa?

5.3.3. Xây dựng sơ đồ trạng thái

Sơ đồ trạng thái được sử dụng để chỉ ra cách các đối tượng phản ứng lại đối với các sự kiện và cách *biến đổi các trạng thái theo các sự kiện đó.*

Ví dụ 1: Mô tả hoạt động của hệ thống thang máy:

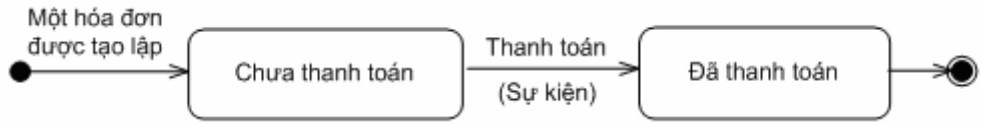


Hình 5.10. Sơ đồ trạng thái của lớp ThangMay

Thường thang máy bắt đầu hoạt động từ tầng một (OnFirstFloor). Khi đang ở OnFirstFloor và có người ở tầng trên (floorNum) nhấn nút yêu cầu thang máy (goUp(floorNum)) thì nó chuyển sang trạng thái *chuyển lên (MovingUp*). Khi chuyển *đến tầng yêu cầu (arrived*) thì nó chuyển sang trạng thái *dừng/nghỉ (Idle)* để mở cửa cho người vào/ra khỏi thang máy. Đang ở trạng thái *nghỉ*, nếu có ai ở tầng trên yêu cầu

thì nó lại *chuyển lên*, nếu có người ở tầng dưới yêu cầu thì thang máy *chuyển xuống (MovingDown)*, còn khi *hết giờ (time-out)* nó sang trạng thái *chuyển về tầng một (MovingtoFirstFloor)* rồi về tầng một.

Ví dụ 2: Sơ đồ trạng thái cho lớp HoaDon:

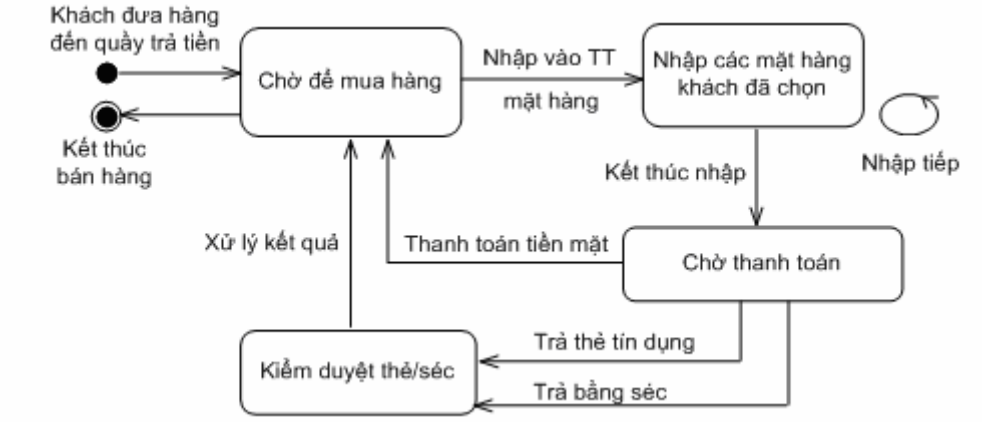


Hình 5.11. Sơ đồ các trạng thái của lớp HoaDon

Khi *một hóa đơn (đối tượng của lớp HoaDon)* được tạo lập thì nó ở trạng thái *chưa thanh toán,* sau đó khi có sự kiện *khách hàng thanh toán,* nghĩa là khách trả tiền cho các mặt hành đã chọn mua thì nó chuyển sang trạng thái *đã thanh toán.*

Như đã đề cập ở trên, các ca sử dụng là rất quan trọng, nó thể hiện những nhiệm vụ mà hệ thống phải thực hiện. Vì vậy, thường chúng cần có các sơ đồ trạng thái kèm theo để mô tả cho các lớp trong những ca sử dụng quan trọng nhất của hệ thống.

Ví dụ 3. Sơ đồ trạng thái của lớp hệ bán hàng:



Hình 5.12. Sơ đồ trạng thái của lớp HệBánHàng

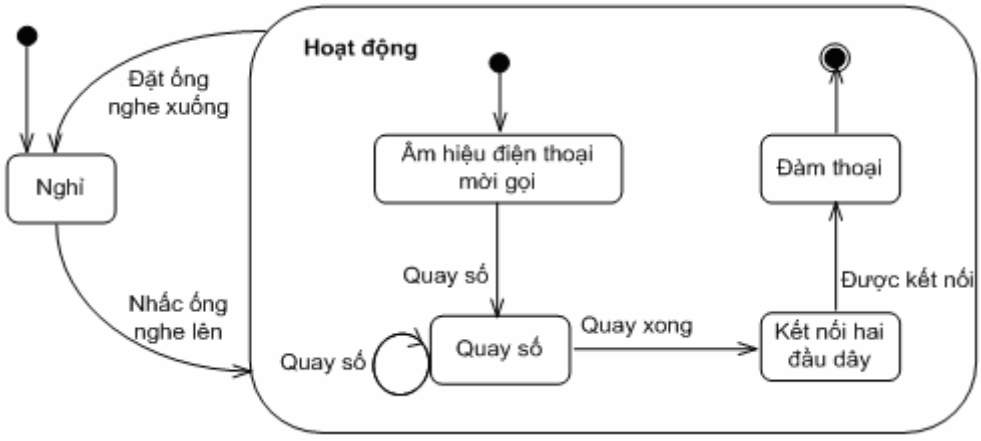
Trạng thái của một đối tượng cũng có khi là trạng thái phức hợp, nghĩa là nó có thể chứa các trạng thái con được lồng bên trong. Một

số trạng thái, ví dụ trạng thái *Kiểm duyệt thẻ* trong sơ đồ trên có thể tiếp tục làm mịn hơn ở pha sau.

Ví dụ 4. Sơ đồ trạng thái của lớp Telephone:

Trong ca sử dụng "*Gọi điện thoại*", Telephone đã được mô tả bằng sơ đồ vết các sự kiện ở trên. Telephone có hai trạng thái chính: *nghỉ* (*Idle)* và *hoạt động* (*Active*). Trạng thái *hoạt động* lại có thể phân tách tiếp thành *âm hiệu điện thoại mời gọi* (*PlayingDialTone*), *quay số* (*Dialing*), *kết nối hai đầu dây* (*Connecting*) và *đàm thoại* (*Talking*).

Sơ đồ trạng thái cho các hoạt động trên được đặc tả như hình 5.13.



Hình 5.13. Sơ đồ trạng thái của Telephone

Máy điện thoại ở trạng thái *nghỉ*, khi người gọi *nhấc ống nghe lên* (*off hook*) thì nó chuyển sang trạng thái *hoạt động* sẵn sàng phục vụ đàm thoại giữa hai điểm trong mạng điện thoại. *Trạng thái hoạt động* lại được mô tả dưới dạng một sơ đồ trạng thái con. Bắt đầu là trạng thái *Có âm hiệu điện thoại mời gọi*, khi người gọi *quay số* (*digit*) nó chuyển sang trạng thái *Quay số (Dialing).* Khi *quay xong* (*completed*), nó chuyển tiếp sang trạng thái *Kết nối* và khi đường dây *được kết nối (connected)* thì hai người có thể *đàm thoại* được với nhau. Trạng thái *đàm thoại* lại có thể mô tả chi tiết hơn bằng một sơ

đồ trạng thái con nếu cần thiết. Còn sự kiện *đặt ống nghe xuống* (*on hook)* làm chuyển trạng thái từ *hoạt động* sang *nghỉ*.

Lưu ý:

- Sơ đồ trạng thái chỉ cần xây dựng cho những đối tượng có nhiều hoạt động quan trọng trong hệ thống,

- Dựa vào các ca sử dụng để xây dựng sơ đồ trạng thái,

- Dựa vào các thuộc tính liên quan để định nghĩa các trạng thái.

Tóm lại, sơ đồ trạng thái là cần thiết vì nó giúp người phân tích, thiết kế và người lập trình hiểu, nắm bắt được các hành vi ứng xử của các đối tượng tham gia vào các ca sử dụng. Họ không chỉ cài đặt đối tượng mà còn cần phải làm cho chúng thực hiện những công việc mà hệ thống yêu cầu. Tuy nhiên sơ đồ trạng thái không được sử dụng để sinh mã tự động trong khâu lập trình sau này.

Sơ đồ trạng thái trong phân tích hướng đối tượng cũng tương tự như sơ đồ khối trong phân tích có cấu trúc, nó mô tả các bước cần thực hiện (thuật toán) của hệ thống.

6

CÁC MÔ HÌNH THIẾT KẾ TƯƠNG TÁC

6.1. SƠ ĐỒ HOẠT ĐỘNG

Sơ đồ hoạt động (*Activity Diagram*) trong UML gần giống với lưu đồ (*Flow Chart*) mà chúng ta đã quen sử dụng trong phân tích thiết kế có cấu trúc. Nó chỉ ra các bước thực hiện, các hoạt động, các nút quyết định và điều kiện rẽ nhánh để điều khiển luồng thực hiện của hệ thống. Sơ đồ hoạt động mô tả các hoạt động và các kết quả của những hoạt động đó và:

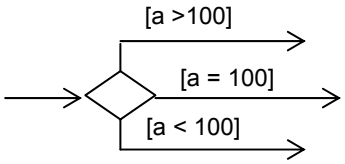
- Nhấn mạnh hơn về công việc thực hiện khi cài đặt một thao tác của từng đối tượng,
- Tương tự như sơ đồ trạng thái, nhưng khác chủ yếu ở chỗ nó tập trung mô tả về các hoạt động (công việc và những thao tác cần thực thi) cùng những kết quả thu được từ việc thay đổi trạng thái của các đối tượng.
- Trạng thái trong sơ đồ hoạt động là các trạng thái hoạt động, nó sẽ được chuyển sang trạng thái sau, nếu hoạt động ở trạng thái trước được hoàn thành.

6.1.1. Trạng thái và sự chuyển trạng thái

Trạng thái và sự chuyển đổi trạng thái được ký hiệu và cách sử dụng hoàn toàn giống như trong sơ đồ trạng thái.

6.1.2. Nút quyết định và rẽ nhánh

Một đối tượng khi hoạt động thì từ một trạng thái có thể rẽ nhánh sang những trạng thái khác nhau tùy thuộc vào những điều kiện, những sự kiện xảy ra để quyết định. Điều kiện rẽ nhánh thường là các biểu thức *Boolean*. Trong UML, nút quyết định rẽ nhánh được biểu diễn bằng hình thoi có các đường rẽ nhánh với những điều kiện đi kèm để lựa chọn như hình 6.1.

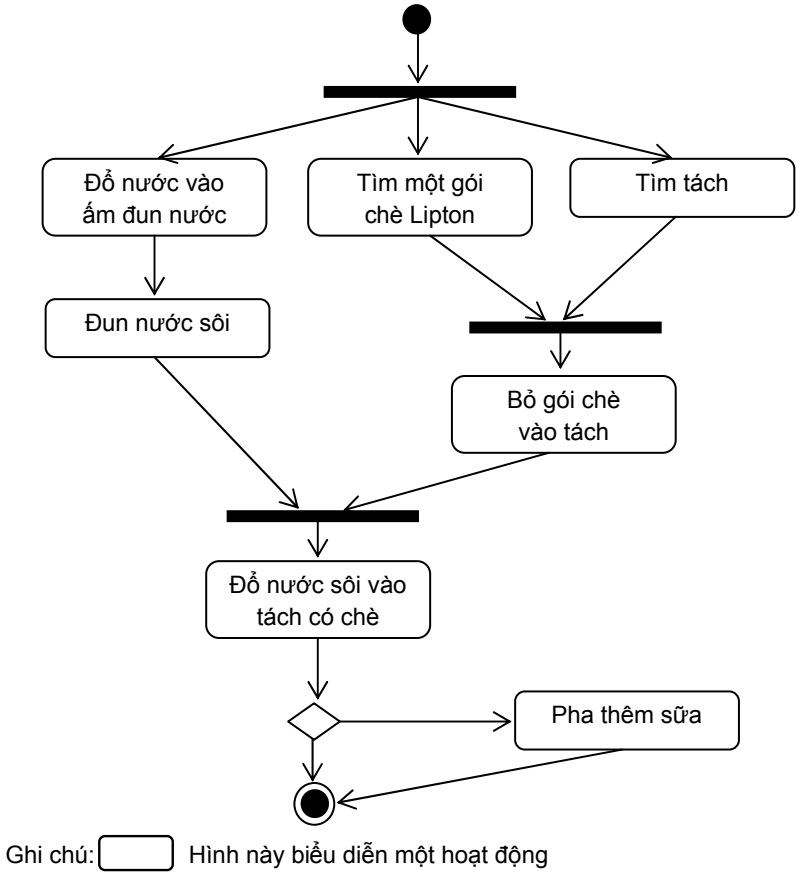


Hình 6.1. Nút rẽ nhánh trong sơ đồ hoạt động

6.1.3. Thanh tương tranh hay thanh đồng bộ

Trong hoạt động của hệ thống, có thể có nhiều luồng hoạt động được bắt đầu thực hiện hay kết thúc đồng thời. Trong UML, *thanh đồng bộ* được vẽ bằng đoạn thẳng đậm được sử dụng để kết hợp nhiều luồng hoạt động đồng thời và để chia nhánh cho những luồng có khả năng thực hiện song song.

Ví dụ: Vẽ sơ đồ hoạt động mô tả các hoạt động "*Đun nước và pha một tách chè Lipton*". Chúng ta thấy một số hoạt động có thể thực hiện song hành như "*Đun nước*", "*Tìm một gói chè Lipton*", "*Tìm tách*",... Sơ đồ hoạt động cho các hoạt động trên có thể mô tả như hình 6.2.



Hình 6.2. Sơ đồ hoạt động "Đun nước và pha chè"

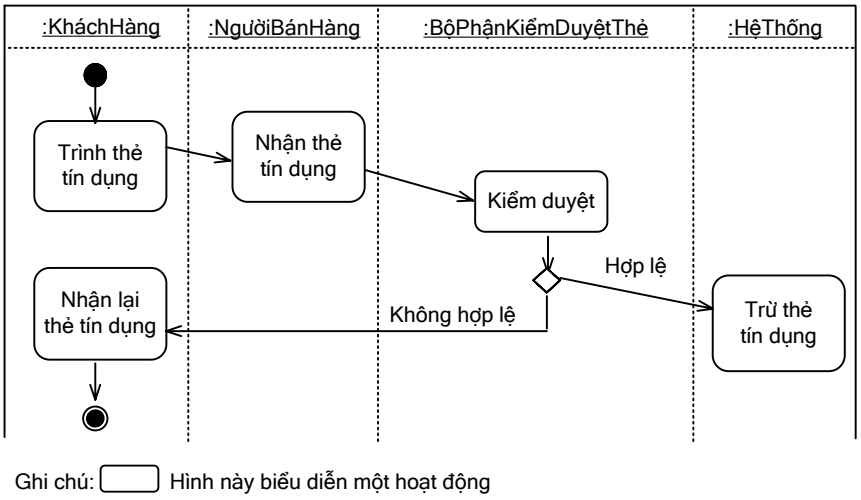
6.1.4. Tuyến công việc

Tuyến công việc (đường bơi) được sử dụng để phân hoạch các hoạt động (trạng thái) theo các nhóm đối tượng hay theo tuyến hoạt động của từng đối tượng. Giống như trong một cuộc *thi bơi*, trong bể bơi mỗi vận động viên bơi lội chỉ được bơi theo một tuyến đã được xác định. Trong hệ thống phần mềm cũng vậy, mỗi đối tượng hoạt động theo tuyến đã được xác định, nhưng có khác là giữa các tuyến này có sự chuyển đổi thông tin với nhau.

Ví dụ: Xét các hoạt động xảy ra khi khách mua hàng chọn phương thức thanh toán bằng *thẻ tín dụng* (*Credit).* Người bán hàng

nhận thẻ từ khách hàng, chuyển thẻ cho bộ phận kiểm duyệt. Nếu là thẻ hợp lệ thì trừ vào thẻ số tiền mua hàng của khách phải trả và giao lại thẻ cho khách. Các hoạt động trên được mô tả trong sơ đồ hoạt động như ở hình 6.3.

Sơ đồ hoạt động sử dụng để thể hiện những hoạt động sẽ thực hiện của mỗi đối tượng và chỉ ra cách thực hiện các hoạt động nghiệp vụ thông qua các dòng công việc, theo tổ chức của các đối tượng.



Hình 6.3. Các tuyến công việc trong sơ đồ hoạt động

6.2. SƠ ĐỒ CỘNG TÁC

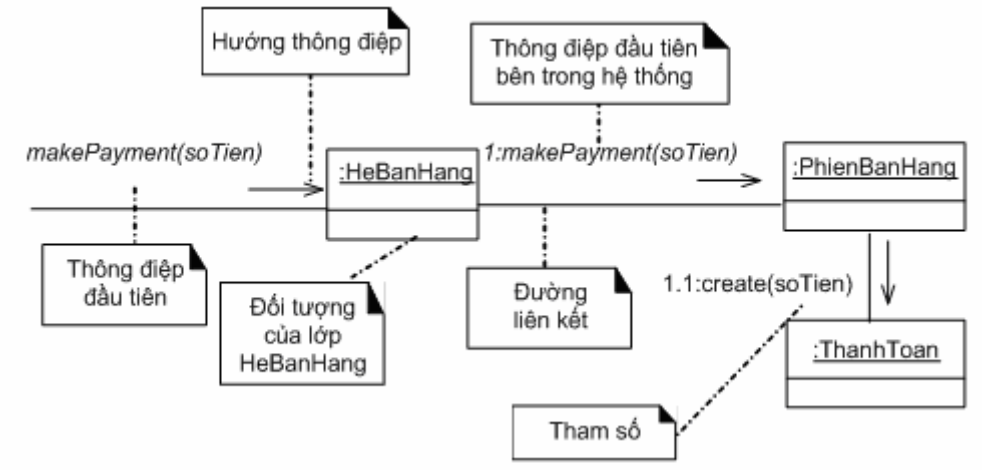
Sơ đồ cộng tác (collaboration diagram) gần giống như sơ đồ trình tự, mô tả sự tương tác của các đối tượng với nhau, nhưng khác với sơ đồ trình tự là ở đây tập trung vào ngữ cảnh và không gian thực hiện công việc.

Sơ đồ trình tự có trật tự theo thời gian, còn sơ đồ cộng tác tập trung nhiều vào quan hệ giữa các đối tượng, tập trung vào các tổ chức cấu trúc của các đối tượng gửi hay nhận thông điệp. Sơ đồ trình tự tập trung vào điều khiển, còn sơ đồ cộng tác lại tập trung vào *luồng dữ liệu (data flows).* Sơ đồ trình tự và cộng tác chỉ phù hợp với

việc mô tả từng biến thể của thủ tục, không phù hợp với việc xác định đầy đủ các hành vi trên một sơ đồ. Cả hai sơ đồ trình tự và cộng tác đều liên quan đến các đối tượng cài đặt chức năng thực hiện trong ca sử dụng. Chúng được xây dựng cho đối tượng, lớp, hay cả hai.

Sơ đồ cộng tác chính là một đồ thị chỉ ra một số các đối tượng và những sự liên kết giữa chúng, trong đó các đỉnh là các đối tượng còn cạnh thể hiện sự trao đổi thông điệp giữa các đối tượng.

Ví dụ: Xét vấn đề thanh toán trong hệ thống bán hàng. Giả sử khách hàng trả tiền mua hàng bằng tiền mặt. Người bán phải ghi lại số tiền mà khách đưa và qua đó hệ thống bán hàng nhận được thông điệp *makePayment(soTien). soTien* là số tiền khách đưa, thường là lớn hơn số tiền hàng và hệ thống phải tính số dư để trả lại cho khách. Để thực hiện được yêu cầu thanh toán thì lớp HệBánHàng lại gửi một thông điệp tương tự cho lớp *PhiênBánHàng* và kết quả là tạo ra một đối tượng của lớp ThanhToan theo thông điệp *create(soTien)* để thực hiện thanh toán với khách hàng. Hình 5.6 mô tả sơ đồ trình tự trao đổi các thông điệp trên lần lượt theo thời gian. Sau đây chúng ta sử dụng sơ đồ cộng tác để thể hiện cùng mối tương tác đó nhưng theo ngữ cảnh thực hiện công việc.



Hình 6.4. Một phần Sơ đồ cộng tác để thực hiện thanh toán tiền mặt

Sơ đồ trên được đọc như sau:

- Thông điệp đầu *makePayment(soTien)* (chính là lời gọi hàm) được gửi tới một đối tượng của hệ thống :HeBanHang.
- Đối tượng :HeBanHang gửi tiếp thông điệp tới một đối tượng của PhienBanHang là :PhienBanHang. Hàm này được đánh số là 1.
- :PhienBanHang gọi toán tử tạo lập của lớp ThanhToan để tạo ra một đối tượng :ThanhToan theo thông điệp *created(soTien)* để làm nhiệm vụ thu tiền. Thông điệp này được đánh số là 1.1.

Qua các phân tích trên, chúng ta có thể khẳng định: *sơ đồ cộng tác của một hoạt động thể hiện thuật toán để thực thi hoạt động đó.*

Từ sơ đồ trình tự và sơ đồ cộng tác, người thiết kế và người phát triển có thể xác định các lớp sẽ xây dựng, quan hệ giữa các lớp, thao tác và trách nhiệm của mỗi lớp. Hai loại sơ đồ này trở thành nền tảng cho các công việc còn lại khi thiết kế. Sơ đồ trình tự có ích cho việc quan sát luồng logic kịch bản, còn sơ đồ cộng tác có ích cho việc quan sát giao tiếp giữa các đối tượng.

6.2.1. Các cấu phần trong sơ đồ cộng tác

1. Danh sách các thành phần trong sơ đồ

Tổng quát, sơ đồ cộng tác chứa các thành phần sau:

- Đối tượng

- Liên kết: thể hiện của kết hợp

- Thông điệp: giúp lớp hay đối tượng có thể yêu cầu lớp hay đối tượng khác thực hiện chức năng cụ thể. Ví dụ: form có thể yêu cầu đối tượng report tự in.

- Chú thích (notes) và các ràng buộc như các sơ đồ khác.

2. Các ký hiệu, cách biểu diễn các thông điệp và một số quy ước trong sơ đồ cộng tác

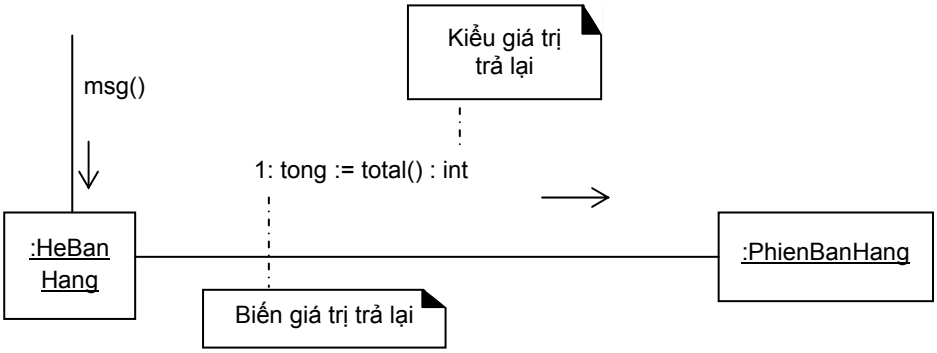
a. Thể hiện giá trị trả lại (return value)

Một số thông điệp được gửi đến cho một đối tượng và yêu cầu có giá trị trả lại cho đối tượng gửi. Giá trị này có thể chỉ ra thông qua phép gán cho một tên biến và tên của thông điệp đó. Trong lập trình, thực chất đây là lời gọi hàm có kiểu giá trị trả lại (trong C, đó là những hàm có kiểu trả lại khác *void*).

Cú pháp chung của thông điệp này có dạng:

ReturnVariableName:=
message(parameter:ParameterType): ReturnType

Ví dụ:

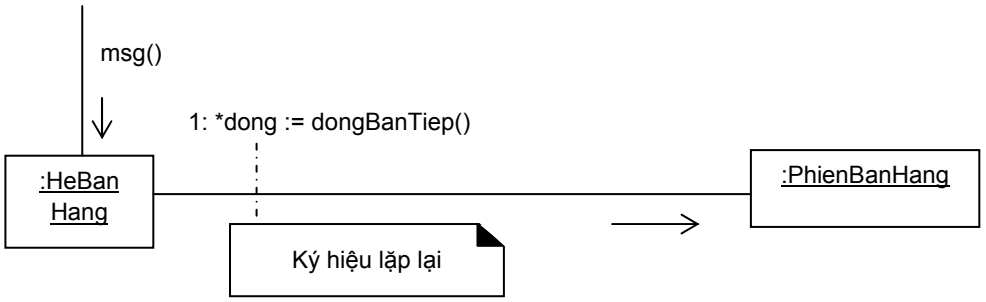


Hình 6.5. Thông điệp có giá trị trả lại

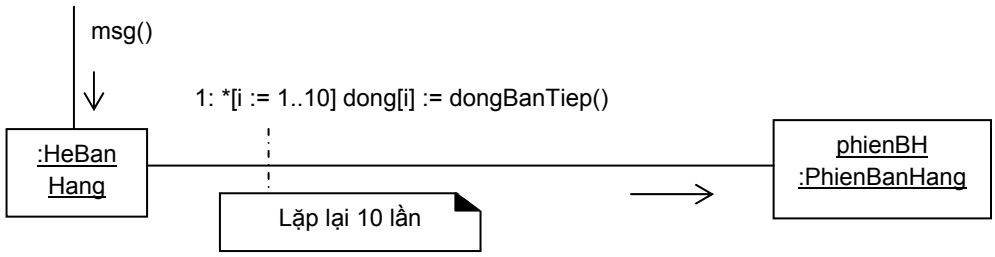
b. Thể hiện những thông điệp lặp

Một đối tượng có thể gửi một thông điệp lặp lại một số lần cho đối tượng khác. Thông điệp được gửi lặp lại nhiều lần có thể biểu diễn bằng dấu '*' trước thông điệp.

Ví dụ:



Hình 6.6. Thông điệp lặp lại với số lần không xác định



Hình 6.7. Thông điệp lặp lại với số lần xác định

Có một số cách khác nhau để biểu diễn cho chu trình lặp, ví dụ thay vì viết:

*[i := 1..10] ta có thể viết *[i < 11].

Như đã khẳng định từ trước, khi một đối tượng nhận được một thông điệp, ví dụ đối tượng :HeBanHang nhận được *msg()* như hình 6.7, thì lớp HeBanHang phải có hàm thành phần *msg()*. Mặt khác, sơ đồ cộng tác thể hiện thuật toán mô tả hoạt động của hệ thống. Vậy dựa vào sơ đồ cộng tác ở hình 6.4, người lập trình có thể viết hàm *msg()* (trong C) cho lớp HeBanHang như sau:

```
void msg(){
  for(int i = 1; i < 11; i++)
```

```
            dong[i] = phienBH.dongBanTiep();
        }
```

Hàm này gọi tới hàm *dongBanTiep()* của lớp PhienBanHang thông qua đối tượng *phienBH* của nó để đọc liên tiếp 10 *dòng bán hàng.*

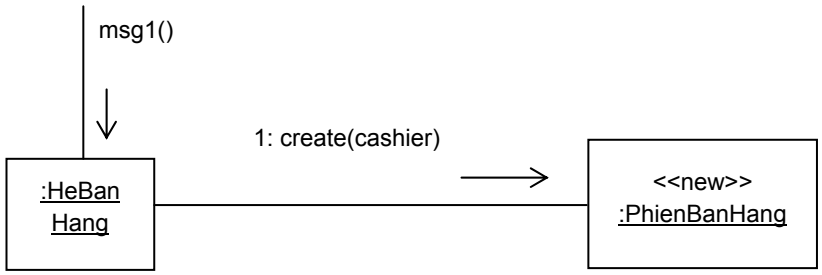
Lưu ý:

Một đối tượng có thể gửi thông điệp cho chính nó, nghĩa là thông điệp có thể quay vòng tròn.

3. Tạo lập đối tượng mới

UML sử dụng thông điệp *create()* để tạo lập một đối tượng mới (một thể hiện nào đó của một lớp). Trong sơ đồ có thể sử dụng thêm ký tự <<new>> ở đối tượng nhận thông điệp.

Ví dụ:

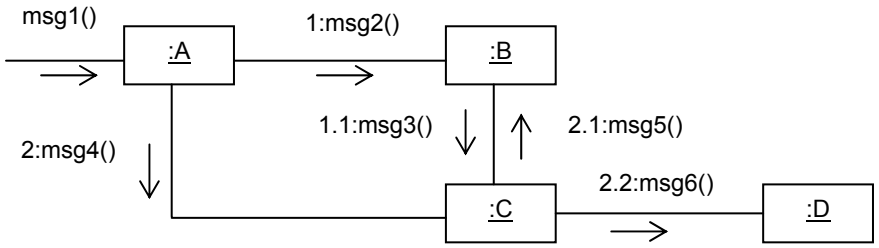


Hình 6.8. Tạo lập đối tượng mới

4. Quy tắc đánh số các thông điệp

- Thông điệp đầu không được đánh số.
- Những thông điệp được gửi tới cho các đối tượng trực tiếp được đánh số tăng dần 1:--, 2:--,...
- Các thông điệp gửi tiếp cho các đối tượng tiếp theo được đánh số theo quy tắc "dấu chấm" ('.').

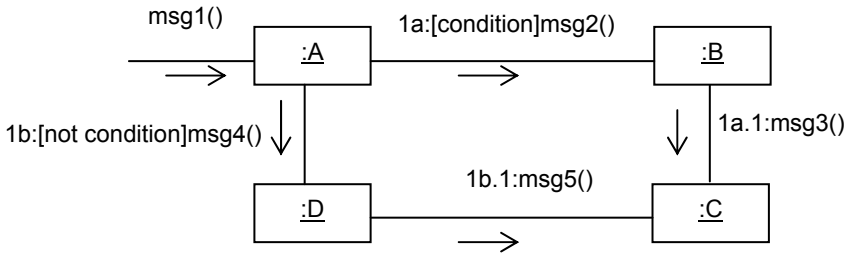
Ví dụ:



Hình 6.9. Đánh số các thông điệp

5. Các điều kiện gửi thông điệp

Đôi khi, một thông điệp có thể *bị kiểm soát (guarded)* được gọi là *thông điệp có điều kiện* vì nó được gửi đến cho một đối tượng khác chỉ khi thỏa mãn một điều kiện nào đó. Điều kiện *condition* được để trong cặp ngoặc '[' và ']'. Ví dụ:



Hình 6.10. Các thông điệp được gửi đi có điều kiện

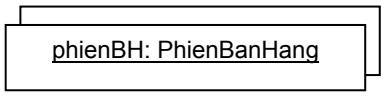
Sơ đồ trên thể hiện:

+ Thông điệp số *1a* được gửi cho :B nếu điều kiện *condition* thỏa mãn,

+ Ngược lại, nếu *condition sai* (*non condition*) thì thông điệp *1b* sẽ được gửi đến cho :D.

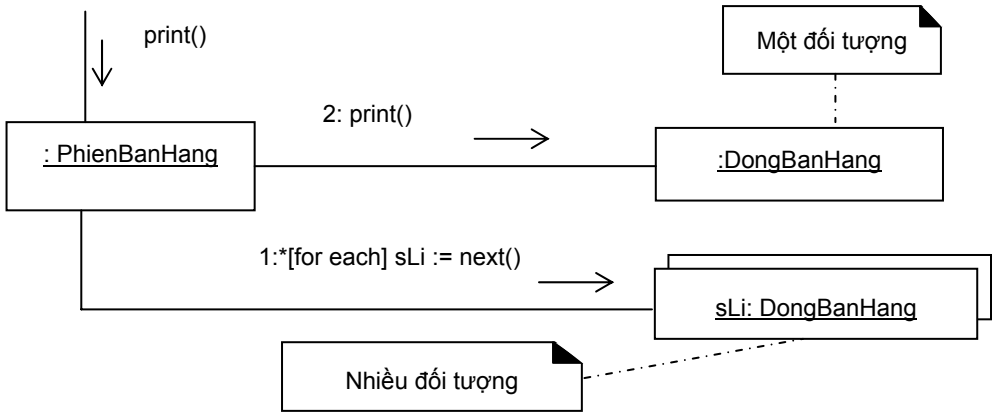
6. Các thông điệp gửi cho một tập (nhiều) các đối tượng

UML ký hiệu tập các thể hiện (đối tượng) như là một kho các đối tượng dạng:



Hình 6.11. Ký hiệu tập các đối tượng phienBH của lớp PhienBanHang.

Ví dụ:



Hình 6.12. Gửi thông điệp cho nhiều đối tượng

6.2.2. Thiết kế các sơ đồ cộng tác và các lớp đối tượng

Nhiệm vụ chính của pha thiết kế là xây dựng các sơ đồ cộng tác mô tả chính xác các hoạt động của hệ thống và từ đó thiết kế chi tiết các lớp. Một trong những khó khăn chính của công việc trên là *gán trách nhiệm cho các đối tượng* như thế nào. Nguyên lý tổng quát để gán giá trị cho các đối tượng được gọi là mẫu (*pattern*) gán trách nhiệm.

Việc tạo lập các sơ đồ cộng tác phụ thuộc vào các kết quả trong pha phân tích:

- *Mô hình khái niệm*: từ những mô hình này, người thiết kế có thể chọn để định nghĩa các lớp ứng với từng khái niệm trong hệ thống. Các đối tượng của những lớp này tương tác với nhau và được thể hiện trong các *sơ đồ tương tác* như sơ đồ trình tự, sơ đồ trạng thái.

- *Các hợp đồng/đặc tả hoạt động của hệ thống:* từ những đặc tả này, người thiết kế xác định được các trách nhiệm và các điều kiện (*Pre-condition, Post-condition*) trong các sơ đồ tương tác.
- *Các ca sử dụng cốt yếu và thực tế:* từ những trường hợp này, người thiết kế có thể lượm lặt được những thông tin về những nhiệm vụ mà các sơ đồ tương tác phải thỏa mãn.

1. Ca sử dụng thực tế

Những ca sử dụng được xác định trong pha phân tích các yêu cầu thường là ít liên quan đến kỹ thuật cài đặt và cũng chưa có liên hệ nhiều với giao diện sử dụng.

Một *ca sử dụng được gọi là cốt yếu* nếu nó mô tả quá trình hoạt động chủ yếu và các động cơ thúc đẩy những hoạt động đó. Ngược lại, *ca sử dụng được gọi là thực tế* nếu nó mô tả một quá trình hoạt động thông qua những thiết kế theo thực tế và được ủy thác cho công nghệ vào/ra đã được xác định trước.

Như vậy, ca sử dụng thực tế là một thiết kế cụ thể về một ca sử dụng, trong đó đã xác định kỹ thuật vào/ra và hỗ trợ cho cài đặt.

Ví dụ: ***Mô tả ca sử dụng thực tế cho ca sử dụng Thanh toán tiền mặt.***

Ca sử dụng: Thanh toán tiền mặt (Buy Items with Cash)

Tác nhân: Người mua (khách hàng), người bán hàng

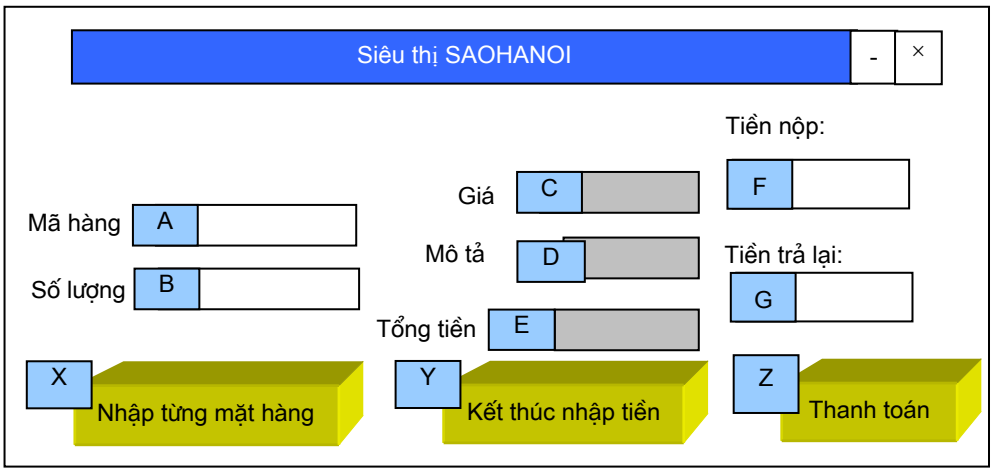
Mục đích: Ghi nhận các thông tin về phiên bán hàng và thu tiền hàng

Mô tả tóm tắt: Sau khi chọn đủ hàng, người mua đưa giỏ hàng (xe hàng) đến quầy trả tiền. Người bán ghi nhận thông tin về các mặt hàng và thu tiền bán hàng bằng tiền mặt. Thanh toán xong, khách hàng có thể đưa hàng ra khỏi cửa hàng.

Kiểu: Thực tế

Tham chiếu: R1.1, R1.2, R1.3, R1.7, R1.9, R2.1.

Trên cơ sở khảo sát bài toán thực tế, người thiết kế có thể xây dựng màn hình thực hiện nhiệm vụ trên như hình 6.13.



Hình 6.13. Màn hình giao diện của ca sử dụng thực tế "Bán hàng"

Kịch bản mô tả ca sử dụng thực tế trên được viết cụ thể như sau:

Bảng 6.1. Kịch bản mô tả ca sử dụng thực tế

Hoạt động của tác nhân	Hoạt động của hệ thống
1. Ca sử dụng bắt đầu khi khách đưa hàng đến quầy trả tiền.	
2. Với mỗi mặt hàng, người bán nhập vào cửa sổ **A** mã hàng. Nếu số lượng mua nhiều hơn 1 thì nhập số đó vào cửa sổ **B**. Ấn **X** sau mỗi lần nhập xong một mặt hàng.	3. Bổ sung các thông tin của từng mặt hàng vào phiên bán hàng. Mô tả của từng mặt hàng vừa nhập vào được hiển thị ở ô **D** và giá bán ở ô **C.**
4. Khi nhập xong các mặt hàng thì ấn **Y.**	5. Tính toán và hiển thị tổng số tiền phải trả ở ô **E.**
6. Khách hàng đưa một số tiền để trả tiền mua hàng, người nhập số đó vào ô **F** và ấn **Z**, số đó không nhỏ hơn *tổng tiền*.	7. Hệ thống xác định số tiền trả lại cho khách và hiển thị ở ô **G.**
. . .	

2. Mẫu gán trách nhiệm

Nhiệm vụ chính của người thiết kế là định nghĩa được các lớp để các đối tượng của chúng thực hiện được những nhiệm vụ mà hệ thống yêu cầu. Muốn vậy, chúng ta phải thiết kế được chi tiết các sơ đồ cộng tác mô tả chính xác từng hoạt động của hệ thống và gán nhiệm vụ cho các lớp đối tượng.

a. Trách nhiệm

Trách nhiệm (*Responsibility*) được mô tả trong hợp đồng, là nghĩa vụ của từng đối tượng. Hoạt động (hành vi) của các đối tượng liên quan chặt chẽ tới các trách nhiệm của các đối tượng đó.

Nói chung có hai loại trách nhiệm:

- *Các trách nhiệm thực hiện:* đó là những hoạt động mà đối tượng thực hiện bao gồm:

+ Tự động thực hiện cái gì đó,

+ Khởi động một hoạt động hoặc kích hoạt một thao tác của những đối tượng khác,

+ Điều khiển hoặc phối hợp các hoạt động giữa các đối tượng khác nhau,

- *Những trách nhiệm để biết:* những hiểu biết về các đối tượng, bao gồm:

+ Hiểu biết về dữ liệu riêng được bao gói và bị che giấu,

+ Hiểu biết về những đối tượng liên quan,

+ Hiểu biết về những sự vật có thể xuất phát hoặc những tính toán nào đó.

Tất cả các thông tin trong hệ thống hướng đối tượng đều được lưu trữ theo các đối tượng và nó chỉ được xử lý khi các đối tượng đó

được ra lệnh thực hiện những nhiệm vụ tương ứng. Để sử dụng được các đối tượng, ngoài các thuộc tính, chúng ta phải biết cách giao tiếp với chúng, nghĩa là phải biết chúng có những hàm thành phần nào. Điều này có thể tìm thấy trong các sơ đồ cộng tác.

b. Các bước thiết kế sơ đồ cộng tác

Việc tạo ra sơ đồ cộng tác có thể thực hiện như sau:

Bước 1: Xác định các trách nhiệm từ các ca sử dụng, mô hình khái niệm (sơ đồ lớp) và các hợp đồng hoạt động của hệ thống,

Bước 2: Gán trách nhiệm cho các đối tượng, quyết định xem đối tượng nào phải thực thi những trách nhiệm trên,

Bước 3: Lặp lại hai bước trên cho đến khi gán hết các trách nhiệm trong sơ đồ cộng tác.

Lưu ý: Các trách nhiệm của đối tượng sẽ được cài đặt trong lớp của những đối tượng đó bằng các hàm thành phần (phương thức).

Trong UML, các trách nhiệm sẽ được gán cho đối tượng khi tạo ra sơ đồ cộng tác và sơ đồ cộng tác thể hiện cả hai khía cạnh, gán trách nhiệm và sự cộng tác giữa các đối tượng trong sơ đồ đó.

c. Các mẫu gán trách nhiệm cơ bản

Những người thiết kế hướng đối tượng có kinh nghiệm thường sử dụng những nguyên lý chung và những phương pháp giải phù hợp với một ngôn ngữ lập trình, được gọi là *mẫu (pattern)* hướng dẫn để tạo ra phần mềm. Một cách đơn giản hơn, *mẫu* là cách gọi một cặp (*bài toán/lời giải*) có thể áp dụng trong những ngữ cảnh mới, với những lời khuyên làm thế nào để áp dụng được vào tình hình mới.

Có năm mẫu gán trách nhiệm cơ bản (GRASP: General Responsibility Assignment Software Patterrn): Expert (Chuyên gia),

Creator (Tạo lập mới), Low Coupling (Móc nối yếu), Hight Cohensin (Cố kết chặt), và Controller (Điều khiển).

Mẫu gán trách nhiệm theo phương pháp chuyên gia (Expert)

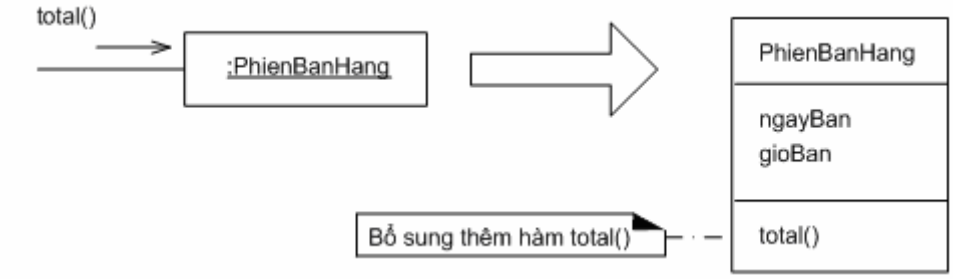
Ví dụ trong hệ thống bán hàng, một số lớp cần phải biết số tiền của mỗi phiên bán hàng và để gán được các trách nhiệm thì phải trả lời:

Ai có trách nhiệm để biết về số tiền của mỗi phiên bán hàng?

Theo ý kiến *chuyên gia,* để trả lời câu hỏi trên thì phải tìm những lớp chứa những thông tin trên. Đó là: Tất cả các *dòng bán hàng* của mỗi *phiên bán hàng,* trong đó cần tính tổng (*total*) của các mục *thành tiền* (*subtotal*) của từng *dòng bán hàng.*

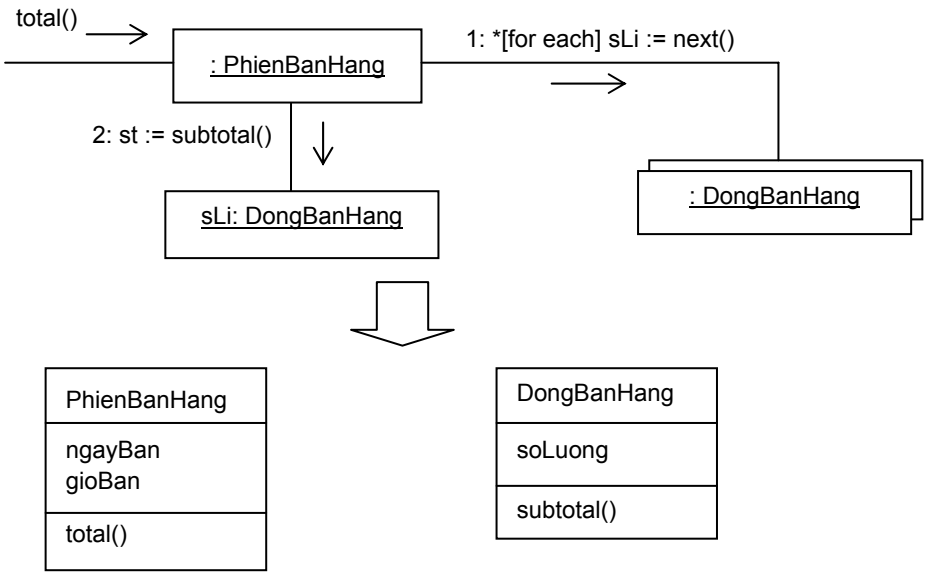
Chỉ có *phienBanHang* (đối tượng phiên bán hàng hiện thời) biết về thông tin này, vì vậy, theo chuyên gia thì gán trách nhiệm cho lớp PhienBanHang.

Chúng ta bắt đầu vẽ sơ đồ cộng tác liên quan đến việc gán trách nhiệm tính tiền bán hàng (*total()*) cho lớp PhienBanHang.



Hình 6.14. Phần đầu của sơ đồ cộng tác

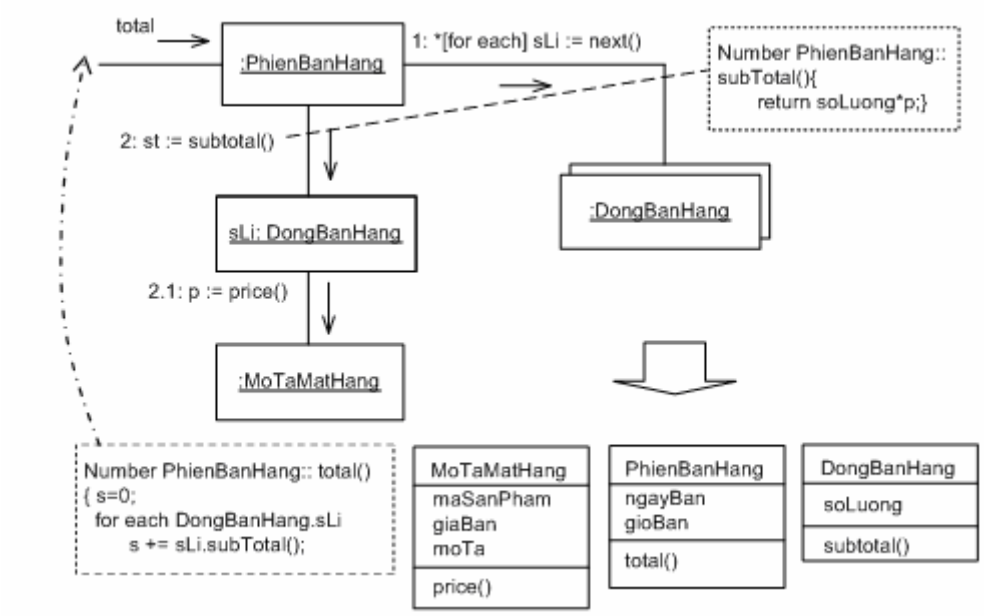
Sau khi gán *total()* cho lớp PhienBanHang thì lại phải biết thêm là ai cung cấp những mục con (số tiền bán từng mặt hàng). Thông tin này được xác định thông qua hàm *subtotal().* Vì vậy, :PhienBanHang phải gửi tiếp thông điệp *subtotal()* cho từng :DongBanHang như hình 6.15.



Hình 6.15. Trao đổi giữa các đối tượng để tính được total()

Để biết và tính được *subtotal()* thì :DongBanHang phải biết thông tin về giá bán được lấy từ đâu. *subtotal()* được xác định từ :DongBanHang và là tích của *DongBanHang.soluong* MoTaMatHang.giaBan,* do vậy sơ đồ cộng tác của hoạt động tính *total()* sẽ được xây dựng như hình 6.16.

Lưu ý: mẫu gán trách nhiệm theo ý kiến chuyên gia được áp dụng nhiều hơn những mẫu khác, nó là *nguyên lý hướng dẫn chung trong thiết kế hướng đối tượng*. Mẫu này thể hiện được những cảm giác trực quan chung về các đối tượng, chúng phải thực hiện những gì để có được những thông tin cần có. Sử dụng loại mẫu này cũng đảm bảo *duy trì được tính chất bao gói*, che giấu thông tin vì các đối tượng chỉ sử dụng những thông tin riêng mà chúng có để thực hiện những nhiệm vụ được giao.



Hình 6.16. Sơ đồ cộng tác để thực hiện việc tính total()

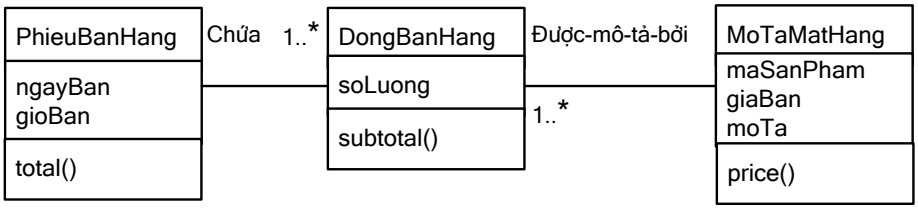
- Mẫu gán trách nhiệm theo phương pháp tạo lập (Creator)

Tạo lập các đối tượng khi cần thiết là một trong những hoạt động quan trọng của hệ thống hướng đối tượng. Do đó cần phải có nguyên lý chung để gán trách nhiệm tạo lập đối tượng trong hệ thống.

Phương pháp: Gán trách nhiệm cho lớp B để tạo ra đối tượng của lớp A (B là phần tử tạo lập các đối tượng của A) nếu có một trong các điều kiện sau:

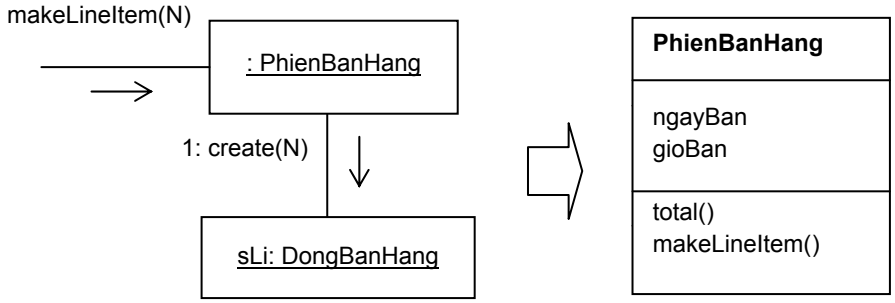
- ♦ B gồm các đối tượng của A,
- ♦ B chứa các đối tượng của A,
- ♦ B ghi chép các thể hiện của A,
- ♦ B sử dụng trực tiếp các đối tượng của A,
- ♦ B có những dữ liệu ban đầu sẽ được truyền cho các đối tượng của A khi chúng được tạo ra.

Ví dụ: hãy xét một phần của mô hình khái niệm như hình dưới đây (trích từ hình 4.14 sau khi được bổ sung thêm các hàm được xác định ở bước trước).



Hình 6.17. Một phần của sơ đồ lớp

Một :PhienBanHang chứa (thực chất là bao gồm) nhiều đối tượng :DongBanHang, do vậy theo mẫu này thì có thể gán trách nhiệm PhienBanHang để tạo lập các đối tượng của DongBanHang. Trách nhiệm này được gán khi có yêu cầu cần tạo ra một dòng bán hàng với *N* đơn vị, nghĩa là khi :PhienBanHang nhận được thông điệp *makeLineItem(N)* thì nó sẽ gửi cho :DongBanHang thông điệp *create(N)* như hình 6.18.



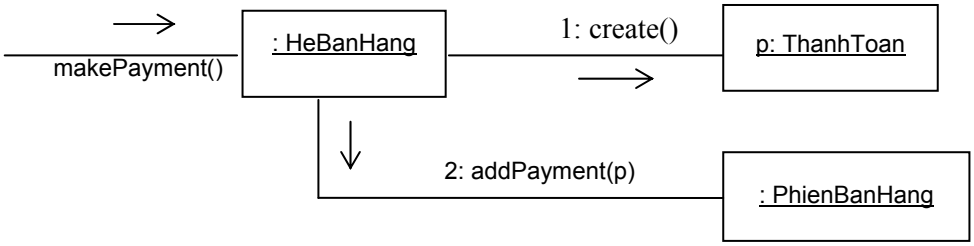
Hình 6.18. Tạo ra DongBanHang

- Mẫu gán trách nhiệm theo phương pháp móc nối yếu (Low Coupling)

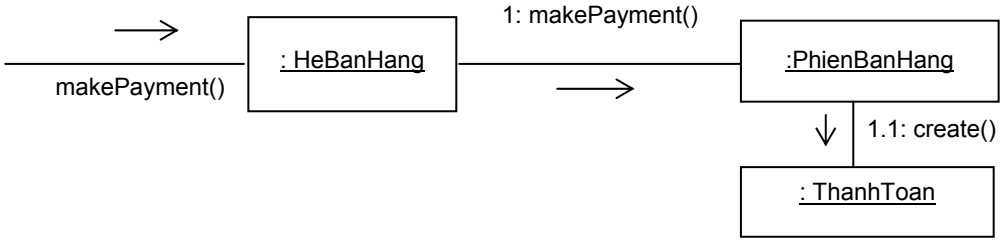
Độ móc nối là một độ đo về sự kết nối của một lớp để biết về/phụ thuộc vào các lớp khác như thế nào. Một lớp có độ *móc nối yếu thì không phụ thuộc nhiều vào nhiều lớp khác*. Ngược lại, một lớp có độ móc nối mạnh thì phụ thuộc vào nhiều lớp khác.

Do đó, khi gán trách nhiệm cho một lớp, hãy cố gắng sao cho độ móc nối giữa các lớp ở mức yếu.

Ví dụ: Xét mối quan hệ giữa các lớp ThanhToan, HeBanHang và PhienBanHang trong hệ thống bán hàng. Giả sử cần phải tạo ra đối tượng p:ThanhToan tương ứng với :PhienBanHang và :HeBanHang nhận được yêu cầu thanh toán *makePayment()*. Bởi vì HeBanHang phải " ghi nhận" :ThanhToan nên theo mẫu tạo lập mới ở trên thì lớp HeBanHang là đại biểu để tạo lập. Điều này dẫn đến thiết kế sơ đồ cộng tác như hình 6.19.



Hình 6.19. Đối tượng của HeBanHang tạo ra đối tượng p:ThanhToan



Hình 6.20. Đối tượng của PhienBanHang tạo ra :ThanhToan

Với cách gán trách nhiệm như hình 6.19 thì HeBanHang phải biết về lớp ThanhToan. Nhưng thực tế điều này không đòi hỏi, vì chỉ cần PhienBanHang cần biết về ThanhToan là đủ. Từ đó chúng ta có thiết kế tốt hơn, thể hiện được mức độ móc nối lỏng hơn như hình 6.20. Sau đó lại áp dụng các quy tắc khác để gán trách nhiệm cho các đối tượng.

Trong các ngôn ngữ lập trình hướng đối tượng như C++, Java, ClassA với ClassB được móc nối với nhau khi:

- ClassA có những thuộc tính mà đối tượng của ClassB cần tham khảo
- ClassA có những hàm mà đối tượng của ClassB cần sử dụng, hay tham chiếu
- ClassA là lớp con của ClassB.

- Mẫu gán trách nhiệm theo phương pháp Cố kết cao (High Cohension)

Cố kết là độ đo về mức độ liên kết trong lớp, tập trung vào các trách nhiệm được gán cho mỗi lớp. Một lớp có *tính cố kết cao* nếu các nhiệm vụ có liên quan chức năng với nhau. *Lớp cố kết là lớp có tối thiểu số các hàm đơn giản và chúng phải có liên hệ chức năng với nhau.*

Vấn đề quan trọng trong thiết kế hướng đối tượng là phải gán được trách nhiệm cho các lớp sao cho một mặt phù hợp với thực tế của bài toán, mặt khác đảm bảo có mối liên hệ chặt chẽ về chức năng nhưng không để có những lớp phải làm việc quá tải.

Ví dụ: Khi xét mối quan hệ giữa các lớp ThanhToan, HeBanHang và PhienBanHang trong hệ thống bán hàng ta có thể đưa ra một thiết kế sơ đồ cộng tác. Vì HeBanHang là lớp chính, nên việc để cho lớp HeBanHang đảm nhận vai trò tạo lập *create()* :ThanhToan và thực hiện thêm nhiều công việc khác có thể dẫn đến quá tải. Mặt khác, nhiệm vụ *create()* không thực sự liên hệ với các nhiệm vụ còn lại, nên có thể gán trách nhiệm này cho lớp PhienBanHang như hình 6.20 thì hợp lý hơn.

Những lớp không cố kết sẽ khó hiểu, khó sử dụng lại và rất khó duy trì hoạt động của hệ thống cho có hiệu quả.

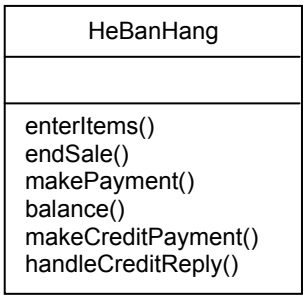
- Mẫu gán trách nhiệm theo phương pháp điều khiển (Controller)

Những sự kiện vào (input) sẽ tác động và làm cho hệ thống được kích hoạt. Việc gán trách nhiệm để xử lý các sự kiện vào của hệ thống cho các lớp được thực hiện như sau:

- Gán trách nhiệm cho những lớp đại diện cho cả hệ thống (điều khiển bề mặt),
- Gán trách nhiệm cho những lớp đại diện cho toàn bộ nghiệp vụ hoặc cho cả tổ chức (điều khiển bề mặt),
- Gán trách nhiệm cho những lớp đại diện cho cái gì đó trong thế giới thực mà nó là tích cực và có thể bị lôi kéo theo trong công việc (điều khiển vai trò),
- Gán trách nhiệm cho những lớp đại diện cho bộ phận nhân tạo để xử lý các sự kiện vào ở mỗi ca sử dụng thường được đặt tên "*<UseCaseName> Handler*" (điều khiển ca sử dụng).

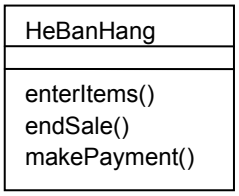
Ví dụ: Trong hệ thống bán hàng, chúng ta đã xác định một số thao tác chính như: *enterItems()* (nhập dữ liệu của mặt hàng), *endSale()* (kết thúc việc nhập dữ liệu của một phiên bán hàng) và *makePayment()* (thu tiền, thanh toán),...

Trong pha phân tích, những thao tác này của hệ thống đã được ghi nhận vào lớp HeBanHang.



Hình 6.21. Các thao tác của hệ thống được ghi nhận vào lớp có tên là HeBanHang

Tuy nhiên, điều này không có nghĩa là lớp có tên HeBanHang phải thực hiện tất cả những nhiệm vụ đó. Trong giai đoạn thiết kế, những thao tác của hệ thống có thể được gán cho lớp điều khiển. Do vậy, ở pha thiết kế có thể gán trách nhiệm thực hiện các thao tác của hệ thống cho một hay một số lớp điều khiển như gán cho lớp HeBanHang đại diện cho cả hệ thống.



Hình 6.22. Gán các thao tác của hệ thống cho lớp điều khiển

Một số chú ý về việc gán trách nhiệm cho đối tượng nhận thông điệp:

- Vì thực hiện bổ sung thông điệp vào sơ đồ có nghĩa là gán trách nhiệm cho đối tượng nhận thông điệp nên phải đảm bảo rằng phải gán trách nhiệm phù hợp cho đối tượng tương ứng.

- Trong phần lớn ứng dụng, đối tượng màn hình và đối tượng mẫu báo cáo thường không thực hiện tác nghiệp nào. Chúng chỉ được sử dụng để người dùng nhập liệu và quan sát thông tin . Do vậy nếu logic tác nghiệp thay đổi sẽ không ảnh hưởng gì đến giao diện hệ thống và ngược lại, thay đổi giao diện sẽ không ảnh hưởng đến tác nghiệp.

Ví dụ: Nếu cần in báo cáo về cân đối tài khoản của mọi tài khoản, đối tượng *tàiKhoảnÔngA* không có trách nhiệm làm việc này. Hãy gán trách nhiệm cho đối tượng khác, nó có khả năng tìm kiếm trong mọi tài khoản để hình thành báo cáo.

6.2.3. Ví dụ thiết kế hệ thống bán hàng

Trong phần này chúng ta hãy áp dụng GRASP để gán trách nhiệm cho các lớp và tạo ra các sơ đồ cộng tác. Chúng ta thực hiện mẫu đối với ca sử dụng "*Thanh toán tiền mặt*" và "Khởi động" (*Start Up*), sau đó thực hiện tương tự đối với những ca sử dụng khác.

1. Định hướng thiết kế sơ đồ cộng tác

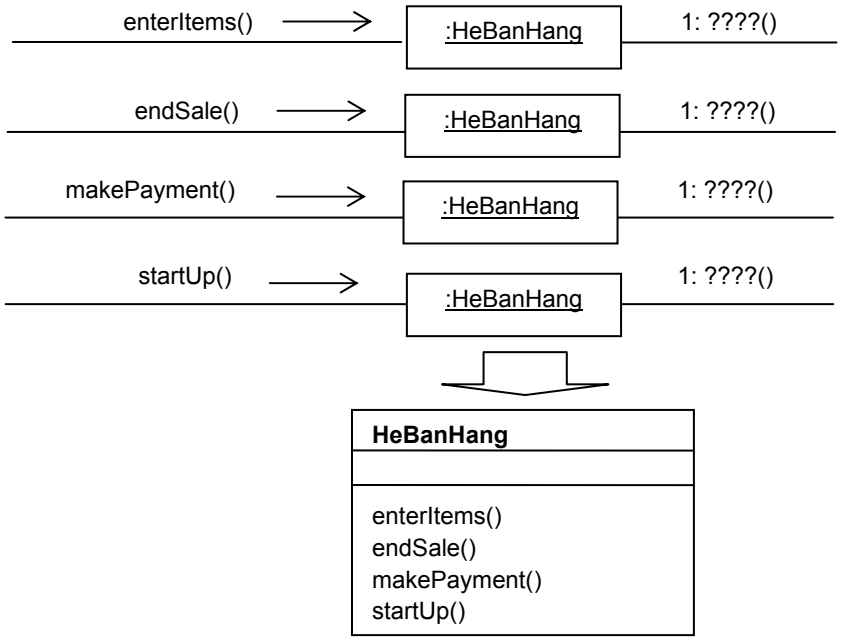
+ Với mỗi thao tác (hoạt động chính) đã được mô tả trong các *hợp đồng* của hệ thống nên tạo ra một sơ đồ riêng.

+ Nếu sơ đồ này phức tạp thì có thể chia nó thành những sơ đồ đơn giản hơn.

+ Sử dụng các hợp đồng trách nhiệm, trong đó dựa vào những điều kiện cần thỏa mãn sau khi hoàn thành (*post-condition*) và các mô tả ca sử dụng để xác định trách nhiệm của các đối tượng trong hệ thống theo mẫu gán trách nhiệm như đã nêu trên.

Để thực hiện ca sử dụng "*Thanh toán tiền mặt* và "Khởi động", hệ thống phải thực hiện: *enterItems()* (nhập các mặt hàng), *endSale()* (kết thúc một phiên bán hàng), *makePayment()* (thanh toán) và *startUp()* (khởi động). Theo hướng dẫn trên, chúng ta thiết kế các sơ đồ cộng tác cho những thao tác đó.

Chúng ta bắt đầu từ bốn sơ đồ như hình 6.23:



Hình 6.23. Các thao tác của hệ thống

2. Sơ đồ cộng tác cho enterItems()

Trước hết chúng ta xem lại *hợp đồng/mô tả* thực hiện *enterItems()* để biết hệ thống phải làm những gì. Đó là:

- Hiển thị thông tin mô tả và giá bán của mặt hàng được nhập vào

Nói chung, các đối tượng của hệ thống không có trách nhiệm hiển thị các thông tin. Những công việc này có thể được các đối tượng giao diện đảm nhiệm bằng cách truy nhập vào CSDL về các mặt hàng và gửi các thông điệp chứa những thông tin tương ứng cho các đối tượng đang hoạt động.

- Tạo lập phiên bán hàng mới

Như trong hợp đồng đã nêu rõ: khi nhập vào mặt hàng đầu tiên của mỗi phiên bán hàng thì phải tạo lập *phienBanHang* mới. Trách nhiệm này được gán cho HệBánHàng và nó có nhiệm vụ là phải ghi lại các phiên bán hàng như thế.

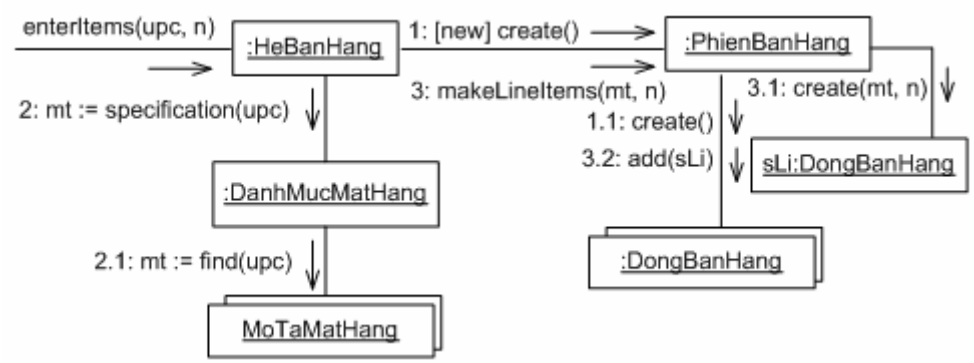
Hơn nữa, mỗi khi *phienBanHang* được tạo mới thì nó là tập rỗng và sau đó được bổ sung thêm các *dongBanHang* mỗi khi nhập xong một mặt hàng.

Áp dụng các mẫu gán trách nhiệm nêu trên, gán việc tạo lập *dongBanHang* mới cho PhienBanHang là hợp lý.

- Xác định các thông tin mô tả và giá bán

Sau khi các *dongBanHang* được tạo lập thì phải được đưa bổ sung vào *phienBanHang* đang hoạt động, do đó nó cần gửi thông điệp *add()* (bổ sung vào) cho những *dongBanHang* đang được nhập vào. Những *dongBanHang* được đưa vào *phienBanHang* phải được xác định từ DanhMucMatHang và sau đó là MoTaMatHang. Vậy để có được những thông tin trên thì đối tượng :HeBanHang phải gửi thông điệp *specification(upc),* xác định mô tả mặt hàng có mã là *upc* cho :DanhMucMatHang, đối tượng này lại gửi tiếp đi thông điệp *find(upc),* để tìm trong tập các mô tả mặt hàng có mã *upc.*

Dựa vào phân tích trên, sơ đồ cộng tác sẽ được xây dựng như hình 6.24:



Hình 6.24. Sơ đồ cộng tác cho enterItems()

3. Tầm nhìn của các lớp

Các đối tượng trong hệ thống tương tác với nhau bằng cách trao đổi các thông điệp, cụ thể hơn như trong các sơ đồ tương tác là trao đổi thông qua các lời gọi hàm. Trong sơ đồ ở hình 6.24, khi hệ thống cần xác định những thông tin về mặt hàng có mã *upc* cho trước, như giá bán chẳng hạn thì nó gửi tới cho :DanhMucMatHang lời gọi hàm *specification(upc)*, đối tượng này lại gửi cho :MoTaMatHang lời gọi hàm *find(upc).*

Một đối tượng muốn có được những thông tin từ những đối tượng khác thì đối tượng đó phải có khả năng nhìn thấy được những gì mà nó cần. Một cách hình thức hơn, *để đối tượng :A có thể gửi một thông điệp cho đối tượng :B thì lớp A phải được liên kết với lớp B.*

Trong thiết kế hướng đối tượng, sự liên kết có liên quan chặt chẽ với khái niệm *khả năng nhìn thấy được của các đối tượng.*

- Nếu :A có thể nhìn thấy :B thì phải có một liên kết giữa hai đối tượng đó, nghĩa là giữa hai lớp tương ứng có quan hệ kết hợp.
- Nếu giữa hai đối tượng :A và :B hiện thời có liên kết với nhau thì một trong hai đối tượng đó có một đối tượng nhìn thấy đối tượng kia.

Về sự trao đổi thông điệp giữa các đối tượng có thể phát biểu chính xác như sau: *Để đối tượng :A gửi được thông điệp cho đối tượng :B thì đối tượng :A phải nhìn thấy được đối tượng :B.*

Có bốn cách để đối tượng A nhìn thấy được đối tượng :B.

Cách 1: Tầm nhìn thuộc tính: :B là thuộc tính của :A.

Cách 2: Tầm nhìn tham số: :B là tham số của một hàm nào đó của :A.

Cách 3: Tầm nhìn khai báo cục bộ: :B được khai báo là đối tượng cục bộ trong định nghĩa hàm của :B.

Cách 4: Tầm nhìn toàn cục: :B là toàn cục.

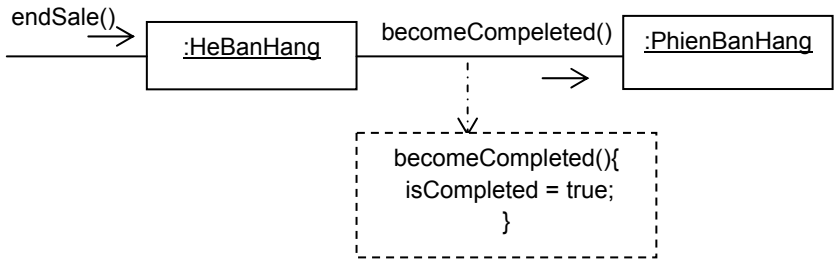
Dựa vào phạm vi quan sát của các đối tượng trong các sơ đồ để khai báo các đặc tính private, protected, public cho các thuộc tính và hàm thành phần trong các lớp đối tượng.

4. Sơ đồ cộng tác cho endSale

Có thể chọn :HeBanHang để điều khiển thao tác này của hệ thống. Hợp đồng của thao tác này yêu cầu:

> *PhienBanHang.isCompleted* được gán trị *true* khi kết thúc nhập dữ liệu.

Như vậy, :HeBanHang có thể gửi thông điệp becomeCompleted() cho PhienBanHang để đặt thuộc tính isCompleled nhận giá trị true.

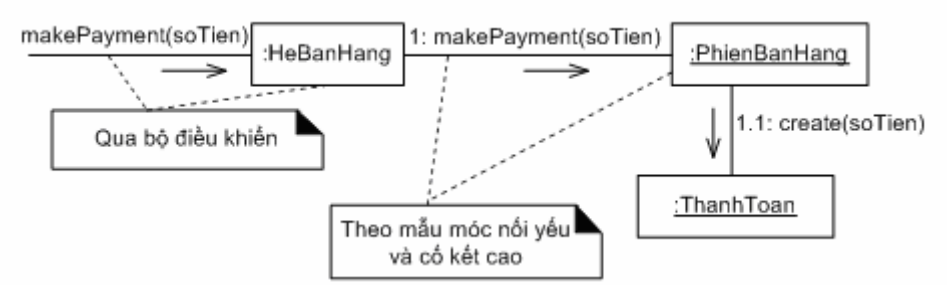


Hình 6.25. Sơ đồ cộng tác thể hiện kết thúc nhập dữ liệu endSale()

5. Sơ đồ cộng tác cho makePayment

Lớp HeBanHang được xem như là bộ điều khiển. Khi nghiên cứu về mẫu gán trách nhiệm để đảm bảo mức độ móc nối giữa các lớp yếu, nhưng độ cố kết lại cao, chúng ta đã gán trách nhiệm tạo lập *đối tượng ThanhToan* cho lớp PhienBanHang, chứ không gán cho lớp HeBanHang.

Sơ đồ cộng tác mô tả thao tác *makePayment()* được vẽ lại như hình 6.26:



Hình 6.26. Sơ đồ cộng tác cho makePayment()

Sơ đồ này đáp ứng những điều kiện sau:

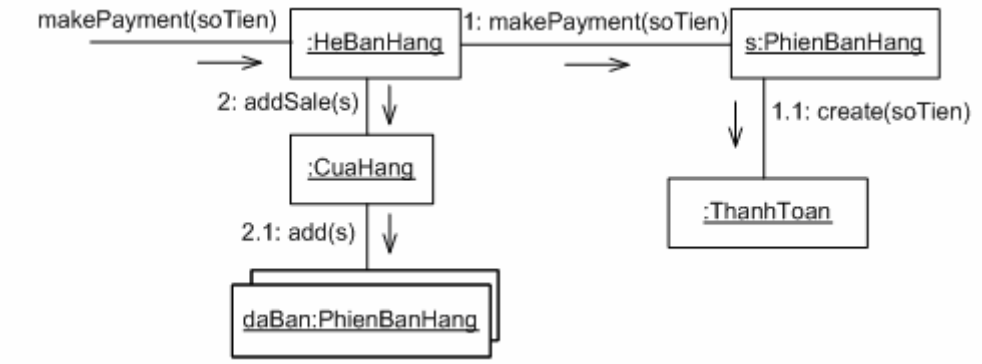
+ Một đối tượng của lớp ThanhToan đảm nhận việc thanh toán,

+ ThanhToan được kết hợp với PhienBanHang

+ *ThanhToan.soluong = soTien.*

6. Ghi nhận những phiên bán hàng đã kết thúc

Mỗi khi kết thúc một phiên bán hàng, theo yêu cầu trong hoạt động kinh doanh của Công ty, hệ thống phải ghi lại ký sự của những phiên bán hàng đó. Theo kinh nghiệm của chuyên gia, ta có thể gán trách nhiệm này, trách nhiệm *addSale()* cho CuaHang sau khi đã gán trách nhiệm *makePayment*() cho PhienBanHang. Sơ đồ ở hình 6.26 được bổ sung thành:



Hình 6.27. Sơ đồ cộng tác cho makePayment() và ghi nhận thông tin đã bán

7. Sơ đồ cộng tác cho ca sử dụng StartUp

Phần lớn các hệ thống (nhưng không phải tất cả) đều có ca sử dụng "*Khởi động*" và một số thao tác liên quan đến việc khởi tạo các giá trị cho một ứng dụng khi bắt đầu thực thi. Mặc dù thao tác này thường là thao tác đầu tiên hệ thống phải thực hiện, nhưng chúng ta để lại thiết kế sau để đảm bảo mọi thông tin liên quan đến các hoạt động sau này của hệ thống đều đã được phát hiện.

Sơ đồ cộng tác của *StartUp* chỉ ra những gì có thể xảy ra khi đối tượng của miền ứng dụng được tạo ra. Nghĩa là trong hệ thống bán hàng phải gán trách nhiệm *create*() cho những lớp chính, đó là HeBanHang hoặc CuaHang.

Chúng ta chọn lớp CuaHang bởi vì HeBanHang đã được chọn làm bộ điều khiển cho các hoạt động của cả hệ thống.

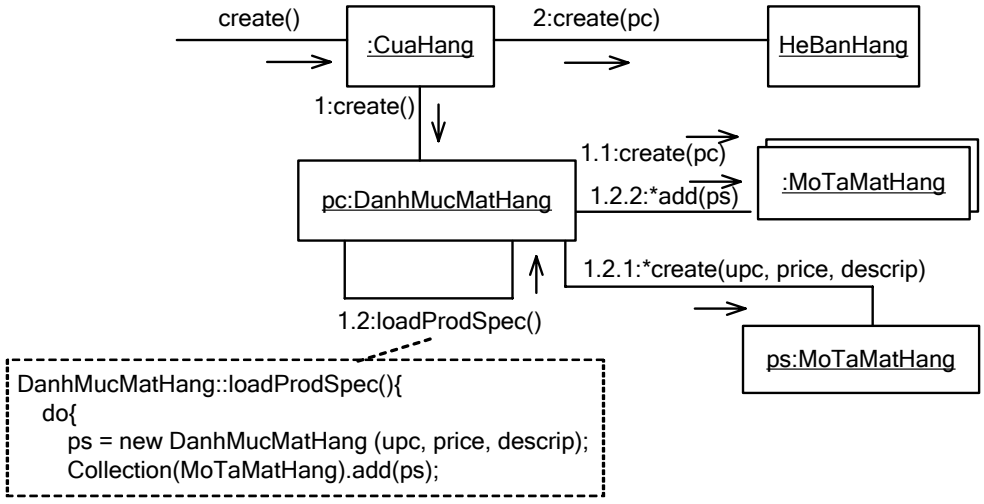
Căn cứ vào hợp đồng của *StartUp* và phân tích trên, ta có thể thiết kế sơ đồ cộng tác cho *StartUp* như sau:

- Bắt đầu gửi thông điệp *create()* cho CuaHang,
- Để tạo ra đối tượng thuộc lớp HeBanHang và cho phép những đối tượng đó gửi được các thông điệp cho DanhMucMatHang để yêu cầu các thông tin về các mặt hàng (xem sơ đồ cộng tác

của *enterItems*) thì trước tiên nó cần phải tạo ra các đối tượng DanhMucMatHang. Nghĩa là nó gửi thông điệp *create()* trước cho DanhMucMatHang, sau đó mới gửi cho HeBanHang thông điệp tương tự.

- Khi một đối tượng DanhMucMatHang đã được tạo lập thì nó sẽ yêu cầu tạo lập MoTaMatHang và sau đó bổ sung vào danh sách các mô tả mặt hàng, đồng thời bản thân nó tự nạp những thông tin mô tả những mặt hàng tiếp theo.

Sơ đồ này được vẽ như hình 6.28.



Hình 6.28. Sơ đồ cộng tác cho StartUp
*(dấu * biểu diễn thông điệp được giữ lặp nhiều lần)*

8. Sự khác biệt giữa kết quả phân tích và thiết kế

Trong sơ đồ cộng tác cho startUp mới chỉ thể hiện việc tạo ra *một đối tượng* đại diện cho một điểm bán hàng đầu cuối. Trong khi ở mô hình khái niệm ở pha phân tích, nó được xây dựng để mô hình cho những cửa hàng thực tế có thể có nhiều điểm bán hàng đầu cuối, nghĩa là khi hệ thống hoạt động thì sẽ có nhiều đối tượng của lớp HeBanHang được tạo ra.

Từ đó có thể thấy có một số điểm khác biệt giữa kết quả phân tích và thiết kế:

- *Đối tượng* :CuaHang trong các sơ đồ không phải là cửa hàng thực, nó là đối tượng trừu tượng,
- *Đối tượng* :HeBanHang trong các sơ đồ không phải là điểm bán hàng đầu cuối cụ thể, nó là đối tượng trừu tượng,
- Sơ đồ cộng tác thể hiện mối quan hệ tương tác giữa các đối tượng :CuaHang và :HeBanHang.
- Tổng quát hóa từ một đối tượng của lớp HeBanHang sang nhiều đối tượng đòi hỏi CuaHang phải tạo ra nhiều thể hiện. Và những sự tương tác giữa các đối tượng của lớp HeBanHang với nhiều đối tượng khác cần phải được đồng bộ hóa để đảm bảo sự toàn vẹn trong việc chia sẻ các thông tin trong hệ thống.

7

MÔ HÌNH KIẾN TRÚC LOGIC

7.1. KIẾN TRÚC HỆ THỐNG

7.1.1. Định nghĩa

Kiến trúc hệ thống là cấu trúc tổ chức của hệ thống. Kiến trúc gồm nhiều bộ phận có thể ở nhiều mức khác nhau, tương tác với nhau thông qua các giao diện, các mối quan hệ kết nối và các ràng buộc để kết hợp chúng thành một thể thống nhất.

7.1.2. Phân loại kiến trúc hệ thống

Kiến trúc hệ thống được chia thành hai loại: *logic* và *vật lý.*

- *Kiến trúc logic:* chỉ ra các lớp và đối tượng, các quan hệ và sự cộng tác để hình thành chức năng của hệ thống. Kiến trúc logic được mô tả bởi các sơ đồ ca sử dụng, sơ đồ lớp và các sơ đồ tương tác.

- *Kiến trúc vật lý:* đề cập đến việc mô tả chi tiết hệ thống về phương diện phần cứng và phần mềm của hệ thống. Đồng thời nó cũng mô tả cấu trúc vật lý và sự phụ thuộc của các mô-đun cộng tác trong cài đặt những khái niệm đã được định nghĩa trong kiến trúc logic.

7.2. PHÂN TÍCH KIẾN TRÚC

Phân tích kiến trúc bao gồm việc chính là *xác định các gói, các lớp thực thể hiển nhiên*. Ngoài ra người ta còn xác định các yêu cầu chuyên biệt chung nhất.

7.2.1. Xác định các gói

Việc xác định các gói bao gồm:

- Xác định các gói phân tích
- Xác định các gói dịch vụ
- Xác định mối quan hệ phụ thuộc giữa các gói phân tích

1. Xác định các gói phân tích

Từ yêu cầu chức năng nghiệp vụ, xác định các gói phân tích. Đó là phương tiện tổ chức hệ thống thành các "cụm" nhỏ hơn cho dễ quản lý. Một gói phân tích có thể chứa các *lớp phân tích, các thực thi ca sử dụng* và các gói phân tích khác (*lớp phân tích, các thực thi ca sử dụng* sẽ được trình bày chi tiết ở chương sau).

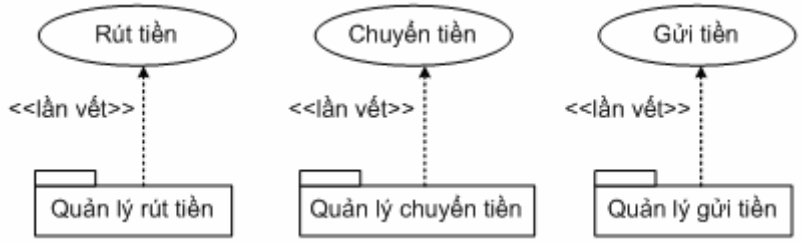
Hai bước xác định các gói phân tích:

Bước 1: Bố trí một số ca sử dụng vào các gói riêng (có thể đã thực hiện ở bước xác định nhu cầu) theo một số tiêu chí gợi ý sau:

- Các ca sử dụng cần có để hỗ trợ một quá trình nghiệp vụ cụ thể
- Các ca sử dụng cần có để hỗ trợ một tác nhân cụ thể của hệ thống
- Các ca sử dụng có quan hệ với nhau bằng các quan hệ tổng quát hóa, mở rộng và quan hệ bao gồm.

Bước 2: Tiến hành thực thi chức năng tương ứng bên trong gói đó

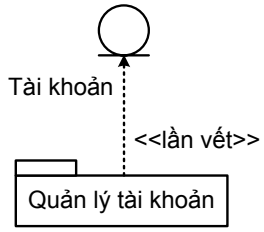
Ví dụ:



Hình 7.1. Xác định các gói phân tích riêng biệt từ các ca sử dụng trong cùng một gói (cùng gói với chúng có liên quan đến cùng một quá trình nghiệp vụ thanh toán tín dụng)

Chú ý: Trong nhiều trường hợp, ta có thể tìm thấy các phần chung trong các gói phân tích, chẳng hạn như một lớp phân tích nào đó. Để xử lý vấn đề này, ta đặt phần chung đó vào một gói riêng nằm ngoài các gói chứa nó, sau đó để các gói khác có liên quan phụ thuộc vào gói mới chứa lớp chung này. Các lớp có phần chia sẻ chung như vậy thường là các lớp thực thể. Chúng có thể tìm thấy bằng cách lần vết tới các lớp thực thể lĩnh vực hoặc nghiệp vụ. Do vậy ta nên nghiên cứu các lớp thực thể lĩnh vực hay lớp thực thể nghiệp vụ để tìm ra phần chung và tạo nên gói phân tích tổng quát.

Ví dụ: Trong hệ thống giao dịch tín dụng cả 3 gói rút tiền, chuyển tiền và gửi tiền đều có một lớp chung là lớp thực thể tài khoản. Lớp này là đại diện của lớp thực thể miền quan trọng là tài khoản và chúng được chia sẻ trong các gói phân tích nêu trên. Do vậy ta cần tạo ra một gói riêng cho nó là gói quản lý tài khoản:



Hình 7.2. Gói quản lý tài khoản
là gói chung được xác định từ một lớp miền

2. Xác định các gói dịch vụ

Xác định các gói dịch vụ là trường hợp riêng của việc xác định gói. Công việc này thích hợp sau khi các yêu cầu chức năng đã được hiểu rõ và đã xác định được phần lớn các lớp phân tích.

Gói dịch vụ dùng để mô tả các gói phân tích được sử dụng ở một mức thấp hơn trong sơ đồ phân cấp cấu trúc các gói của hệ thống. Một gói dịch vụ có thể có các tính chất sau:

- Chứa một tập hợp các lớp có liên quan với nhau về mặt chức năng

- Không thể chia nhỏ hơn

- Có thể tham gia vào một hay nhiều thực thi ca sử dụng

- Phụ thuộc rất ít vào các gói dịch vụ khác

- Các chức năng mà nó cung cấp có thể được quản lý như một đơn vị riêng biệt

Ví dụ: gói chung quản lý tài khoản chính là gói dịch vụ.

Các bước xác định các gói dịch vụ:

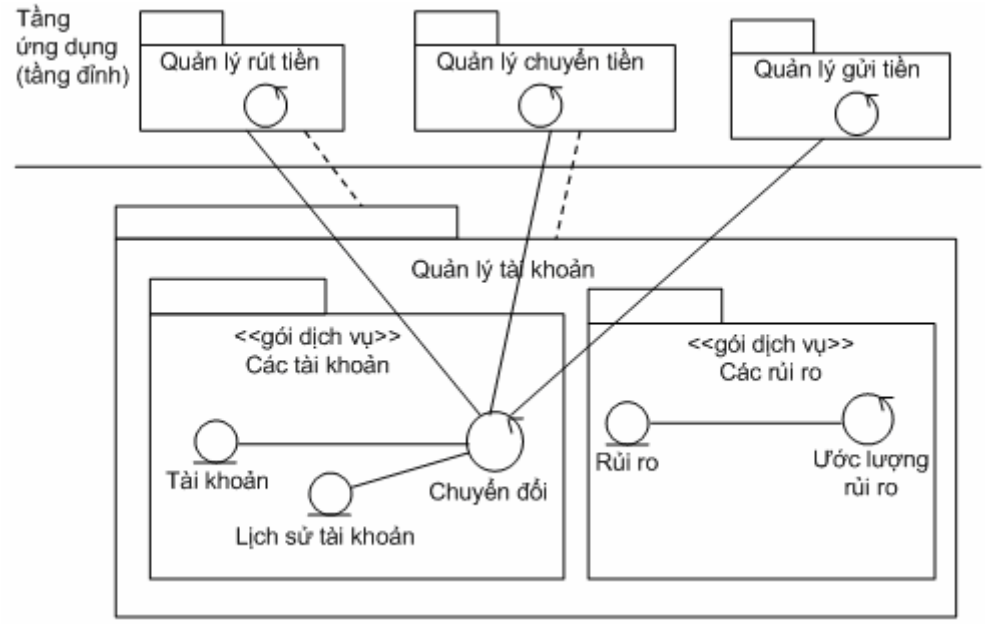
- Xác định một gói dịch vụ cho mỗi dịch vụ được chọn.

- Xác định một gói dịch vụ cho mỗi dịch vụ mà có thể trở thành tùy chọn cho nhiều ca sử dụng. Ta sẽ xác định một gói dịch vụ cho mỗi dịch vụ được cung cấp bởi các lớp có liên quan về mặt chức năng.

3. Xác định mối quan hệ phụ thuộc giữa các gói phân tích

Mục tiêu đặt ra là tìm các gói phân tích tương đối độc lập với các gói khác, tức là chúng được ghép nối lỏng lẻo với nhau nhưng có tính kết dính cao bên trong. Với mục tiêu này, ta cố gắng giảm số lượng các mối quan hệ giữa các lớp thuộc các gói khác nhau. Cách tốt nhất để thực hiện công việc này là tổ chức mô hình phân tích thành các tầng và xếp các gói ứng dụng cụ thể ở tầng đỉnh, các gói chung hơn ở tầng thấp hơn (xem hình 7.3).

Các gói trên tầng ứng dụng đều chứa một lớp điều khiển (điều khiển rút tiền, điều khiển chuyển tiền và điều khiển gửi tiền). Các lớp điều khiển này đều liên kết (đường nối liền) với lớp điều khiển chuyển đổi trong gói quản lý tài khoản. Do đó đòi hỏi phải có mối quan hệ phụ thuộc (mũi tên có đường đứt đoạn) tương ứng giữa 3 gói trên tầng ứng dụng đó với gói quản lý tài khoản.



Hình 7.3. Xác định các gói dịch vụ
gói các lớp có liên quan về chức năng

4. Phân tích một gói

Mục đích của việc phân tích một gói:

- Đảm bảo rằng gói phân tích càng độc lập đối với các gói khác nếu có thể

- Đảm bảo rằng gói phân tích hoàn thành mục đích của nó là thực thi những lớp miền hoặc các ca sử dụng nào đó

- Mô tả các mối quan hệ phụ thuộc sao cho có thể ước tính được hiệu ứng của các thay đổi này

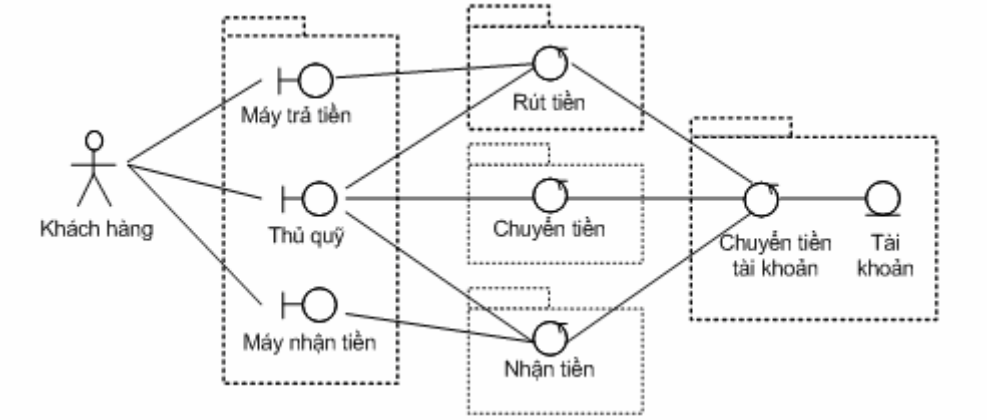
Một số nguyên tắc chung phân tích thành gói:

- Xác định và duy trì các mối quan hệ phụ thuộc giữa hai gói có chứa các lớp liên kết với nhau.

- Mỗi gói chứa các lớp đúng, nghĩa là hãy cố gắng làm cho gói trở thành kết dính bằng cách chỉ đưa các đối tượng có liên quan về mặt chức năng vào trong gói.

- Hạn chế tối đa các mối quan hệ phụ thuộc tới các gói khác bằng cách bố trí lại các lớp chứa trong một gói sang gói khác nếu nó quá phụ thuộc vào các gói khác.

Ví dụ:



Hình 7.4. Mô hình lớp phân tích của hệ thống giao dịch tín dụng với các gói

(đường đứt nét là hình ảnh các gói. Các gói có đường viền đậm là những gói có tầm quan trọng về mặt kiến trúc vì chúng liên quan đến ca sử dụng rút tiền là ca sử dụng quan trọng nhất)

7.2.2. Xác định các lớp thực thể hiển nhiên

Việc xác định các lớp thực thể quan trọng nhất dựa trên các lớp miền hoặc các thực thể nghiệp vụ đã được xác định trong quá trình nắm bắt các yêu cầu. Mỗi lớp thực thể này có thể đưa vào một gói riêng. Ở bước này không nên xác định quá nhiều lớp và đi quá sâu vào chi tiết. Một phác thảo ban đầu chỉ gồm các lớp có ý nghĩa về mặt kiến trúc là đủ.

7.2.3. Xác định các yêu cầu chuyên biệt chung nhất

Yêu cầu chuyên biệt là yêu cầu nảy sinh trong quá trình phân tích và việc nắm bắt nó là quan trọng. Các yêu cầu này có thể là:

- Tính lâu bền (cần lưu trữ)

- Phân tán đối tượng trong suốt (các đối tượng được phân tán nhưng người sử dụng không cần biết nó được cư trú ở đâu)

- Sự phân bố và tính tương tranh

- Các đặc trưng về an toàn,

- Phát hiện lỗi, khắc phục lỗi, dung sai về lỗi

- Quản lý giao dịch...

7.3. THIẾT KẾ KIẾN TRÚC

Thiết kế kiến trúc bao gồm việc thiết kế các cấu hình mạng cho hệ thống, thiết kế các hệ thống con và các giao diện của chúng, xác định các lớp quan trọng về mặt kiến trúc.

7.3.1. Thiết kế cấu hình mạng cho hệ thống

Các cấu hình mạng chung thường dùng một dạng mẫu ba hay hai tầng (three/tier pattern) trong đó các ứng dụng khách hàng được phân vào một tầng, chức năng CSDL vào một tầng, và logic nghiệp vụ/ứng dụng vào một tầng. Dạng đơn giản của kiến trúc *client/server* là một trường hợp đặc biệt của dạng mẫu 2 tầng, trong đó logic nghiệp vụ/ứng dụng được bố trí vào cùng một tầng với tầng *client* hoặc tầng *CSDL.*

Kiến trúc phổ biến chung hiện nay là kiến trúc ba tầng: tầng trình diễn/tầng giao diện, tầng tác nghiệp/tầng logic ứng dụng và tầng lưu trữ:

1. *Tầng trình diễn/tầng giao diện*: biểu diễn và giới thiệu các thành phần của hệ thống thông qua các giao diện đồ họa, các Window, các hộp thoại,... Các thực thể trong hệ thống được thể hiện sao cho vừa thân thiện với người sử dụng, vừa phù hợp với bài toán ứng dụng.

2. *Tầng logic ứng dụng/tầng tác nghiệp*: mô tả các đối tượng thực thi các nhiệm vụ và các quy luật điều hành các tiến trình của hệ thống. Hệ thống phần mềm có hai loại đối tượng cơ bản:

- *Những đối tượng nghiệp vụ:* là những đối tượng đại diện cho các khái niệm của miền ứng dụng.

- *Những đối tượng dịch vụ* (*Service Object*): những đối tượng không nằm trong phạm vi bài toán nhưng cung cấp các dịch vụ cho hệ thống như: làm môi giới, tương tác với CSDL, trao đổi thông tin, lập báo cáo, đảm bảo an toàn, an ninh dữ liệu,...

Tầng hai được thể hiện ở các sơ đồ: lớp, trạng thái, cộng tác, hoạt động và sơ đồ triển khai.

3. *Tầng lưu trữ* (*Storage*): thể hiện cơ chế lưu trữ đảm bảo nhất quán và bền vững dữ liệu. Hiện nay có hai mô hình CSDL chính đang được sử dụng phổ biến là: *mô hình dữ liệu quan hệ và mô hình dữ liệu hướng đối tượng.* Để lưu trữ dữ liệu cho hệ thống, ta phải lựa chọn hệ quản trị CSDL và những phương pháp biến đổi dữ liệu, biến đổi các câu truy vấn cho phù hợp với công nghệ hiện đại và với phạm vi ứng dụng. Kiến trúc sư đối tượng phải lựa chọn công nghệ CSDL thích hợp để phát triển hệ thống. Khi phân tích, thiết kế hướng đối tượng và nếu lựa chọn mô hình dữ liệu quan hệ thì người phát triển hệ thống phải quan tâm đến những vấn đề về tích hợp dữ liệu để đảm bảo cho phép người sử dụng truy cập được tới tất cả các dữ liệu từ nhiều mô hình khác nhau một cách trong suốt.

Quá trình phát triển phần mềm cũng giống như quá trình học và nhận thức của con người, đó là quá trình lặp, tích lũy để phát triển, hoàn thiện liên tục. Vì vậy, kiến trúc của hệ thống cũng phải được xây dựng sao cho phù hợp với sự phát triển và khả năng mở rộng của hệ thống.

Trong triển khai cài đặt, kiến trúc ba tầng thường được tổ chức theo nhiều phương án khác nhau. Có thể thực hiện:

- ♦ Cả 3 tầng trên cùng máy (hệ thống nhỏ).
- ♦ Tầng trình diễn và tầng logic ứng dụng trên máy khách (Client Computer), tầng lưu trữ trên Sever (hệ thống loại vừa).

♦ Tầng trình diễn trên máy người sử dụng, tầng logic ứng dụng trên máy trạm phục vụ ứng dụng (Application Server), còn tầng lưu trữ tổ chức ở các máy trạm phục vụ dữ liệu (Data Server) (những hệ thống lớn).

Hiện nay có những ngôn ngữ lập trình hướng đối tượng như Java và các công nghệ CSDL hướng đối tượng đã sẵn sàng hỗ trợ cài đặt phân tán theo hai phương án cuối.

Những đặc trưng của các cấu hình mạng bao gồm:

- Những nút nào liên kết với nhau, các khả năng về công suất xử lý và kích cỡ bộ nhớ của chúng là bao nhiêu?

- Kết nối giữa các nút thuộc loại nào, các giao thức truyền thông giữa chúng là gì?

- Các đặc trưng của các kết nối và các giao thức truyền thông (băng thông, tính sẵn sàng, chất lượng dịch vụ của nó)?

- Nhu cầu về khả năng xử lý dư thừa, về chế độ hỏng hóc, về sự di trú tiến trình xử lý, về việc duy trì các bản sao lưu dữ liệu dự phòng,…?

7.3.2. Thiết kế các hệ thống con và các giao diện của chúng

Mục tiêu thiết kế một hệ thống con bao gồm:

- Đảm bảo cho một hệ thống con là độc lập đối với các hệ thống con khác đến mức tối đa có thể được thông qua các giao diện của chúng.

- Đảm bảo các hệ thống con cung cấp các giao diện đúng.

- Đảm bảo hệ thống con thực thi đúng các thao tác đã được xác định bởi các giao diện mà nó cung cấp.

Việc phân hệ thống thành các hệ thống con là một cách thức để tổ chức mô hình thiết kế thành các cụm có thể quản lý được. Hơn nữa, việc này giúp chúng ta có thể lập luận khi thiết kế và đánh giá được các cơ hội tái sử dụng.

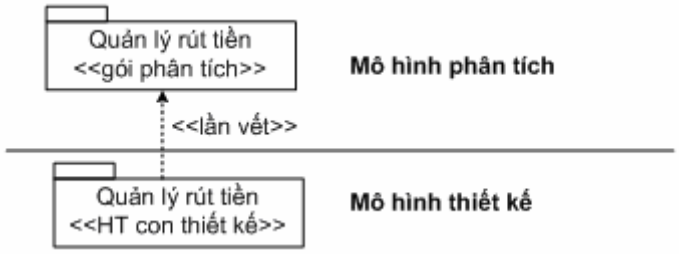
1. Xác định các hệ thống con

Chúng ta cần xác định các hệ thống con trong các tầng ứng dụng cụ thể và tầng ứng dụng tổng quát (2 tầng mức đỉnh)

Để xác định các hệ thống con, người ta thường sử dụng các gói phân tích đã tìm thấy trong quá trình phân tích. Tất nhiên sau này cần tinh chế lại.

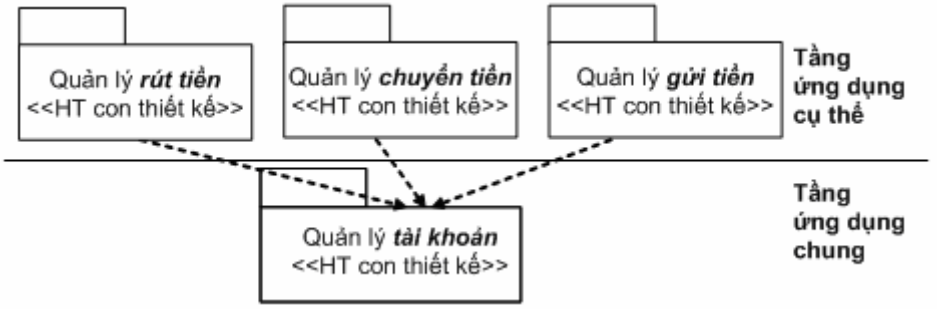
Hệ thống con thiết kế có thể bao gồm các lớp thiết kế, các thực thi ca sử dụng, các giao diện và các hệ thống con khác. Hệ thống con có thể cung cấp các giao diện thể hiện các chức năng xuất ra từ nó dưới dạng các tác vụ mà nó cung cấp.

Ví dụ:



Hình 7.6. Xác định các hệ thống con dựa trên các gói phân tích đã có

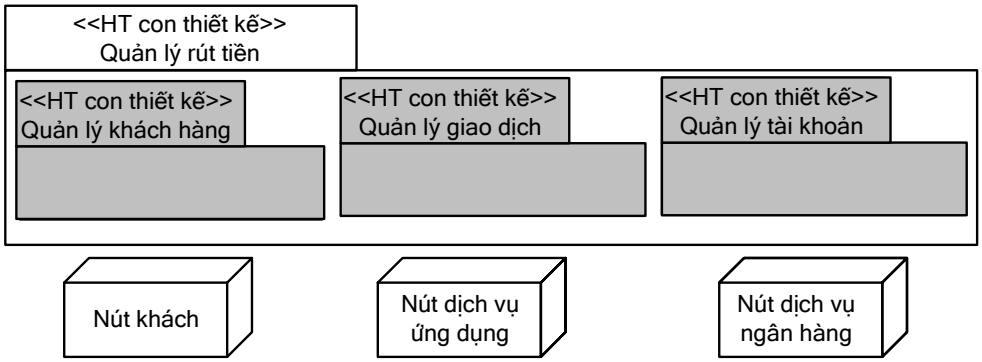
Chúng ta cần tinh chế các hệ thống con để xử lý các chức năng dùng chung. Ví dụ:



Hình 7.7. Hệ thống con dịch vụ quản lý tài khoản cung cấp một dịch vụ tổng quát có thể được sử dụng trong nhiều ca sử dụng khác nhau

Các lớp thiết kế thực hiện ca sử dụng cần được phân phối cho nhiều nút khác nhau khi bố trí để phục vụ các tác nhân. Vì thế các lớp của nó cần được phân vào nhiều thành phần nhỏ hơn để có thể bố trí chúng một cách thích hợp vào các nút tương ứng.

Ví dụ:



Hình 7.8. Phân bố các hệ thống con trên các nút của mạng dựa vào việc phân các lớp thiết kế thực hiện ca sử dụng rút tiền vào 3 thành phần nhỏ hơn

Một hệ thống con hoàn thành mục tiêu của nó nếu nó cung cấp một thực thi đúng các thao tác xác định bởi giao diện mà nó cung cấp. Một số vấn đề liên quan ở đây là:

- Mỗi giao diện do hệ thống con cung cấp phải được các lớp thiết kế hoặc các hệ thống con khác bên trong hệ thống con đó cung cấp phương tiện thực thi.

- Để làm sáng tỏ cách thức mà các thiết kế bên trong của một hệ thống con thực thi bất kỳ một giao diện nào hoặc các ca sử dụng của nó, ta có thể tạo ra các cộng tác dưới dạng các yếu tố được chứa trong hệ thống con đó.

2. Xác định các mối quan hệ giữa các hệ thống con

Sử dụng các mối quan hệ phụ thuộc giữa các gói phân tích làm cơ sở xác định các mối quan hệ của các hệ thống con thiết kế. Các mối quan hệ phụ thuộc nên được biểu diễn tường minh giữa các tầng của kiến trúc.

Các mối quan hệ phụ thuộc phải được xác định và duy trì từ hệ thống con này đến hệ thống con khác có chứa các phần tử được liên kết với nó. Tuy nhiên, nếu các hệ thống con khác đó cung cấp các giao diện thì các mối quan hệ phụ thuộc cần được khai báo hướng về các giao diện đó. Một phụ thuộc tốt là phụ thuộc vào một giao diện mà không phụ thuộc vào hệ thống con. Ta nên tối thiểu hóa các phụ thuộc vào các hệ thống con và/hoặc các giao diện bằng việc bố trí lại các lớp được chứa mà không quá phụ thuộc vào các hệ thống con khác.

3. Xác định các giao diện của các hệ thống con

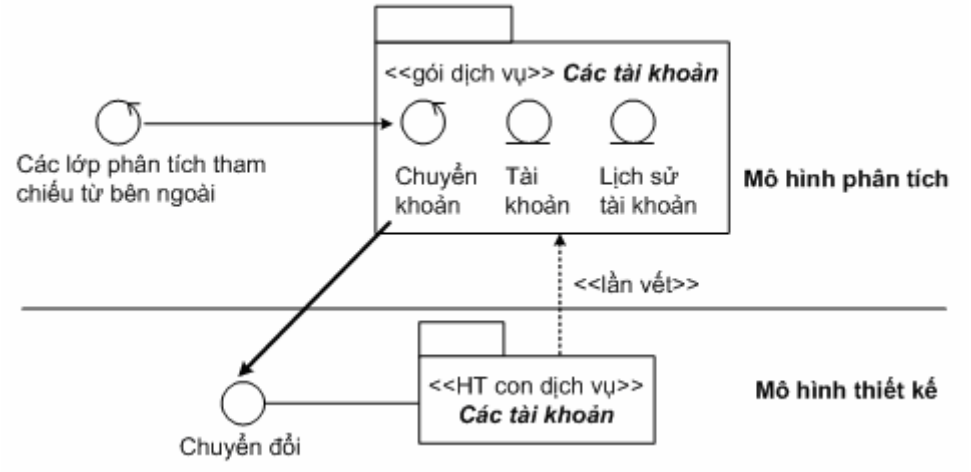
Các giao diện mà một hệ thống con cung cấp xác định các thao tác mà từ "bên ngoài" hệ thống con đó có thể truy nhập đến nó. Các giao diện này do các lớp hoặc các hệ thống con khác bên trong hệ thống con đó cung cấp.

Giao diện sử dụng các đặc tả các tác vụ mà các lớp thiết kế và các hệ thống con cung cấp. Phần lớn các giao diện giữa các hệ thống con đều mang ý nghĩa về mặt kiến trúc, xác định phạm vi và cách thức mà các hệ thống con được phép tương tác với nhau.

Các thao tác được xác định qua các giao diện được cung cấp bởi một hệ thống con cần phải hỗ trợ mọi vai trò mà hệ thống con này đóng góp trong thực thi các ca sử dụng khác nhau. Ngay cả khi các giao diện đã được phác thảo, các giao diện này có thể phải tinh chế khi phát triển mô hình thiết kế. Một hệ thống con và các giao diện của nó có thể được sử dụng bên trong nhiều thực thi ca sử dụng, do đó sẽ cung cấp nhiều yêu cầu mới nữa trên các giao diện.

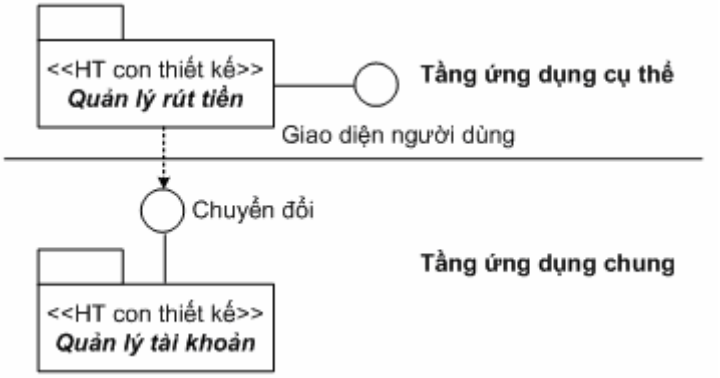
Ví dụ: Tìm các giao diện của hệ con dựa trên mô hình phân tích:

Trong mô hình thiết kế chúng ta có thể xác định một giao diện ban đầu của hệ thống con dịch vụ gọi là ***các tài khoản*** do nó cung cấp:



Hình 7.9. Giao diện của một hệ thống con được xác định từ gói thiết kế tương ứng

Tương tự ta xác định được các giao diện của các hệ thống con thiết kế khác thuộc hai tầng ứng dụng và các tham chiếu của các hệ thống con khác đến các giao diện này.



Hình 7.10. Giao diện của các hệ thống con thuộc các tầng ứng dụng

7.3.3. Xác định các lớp thiết kế quan trọng về mặt kiến trúc

1. Xác định các lớp thiết kế từ các lớp phân tích

Trong quá trình phân tích, người ta xác định được các lớp phân tích quan trọng về mặt kiến trúc, từ đó phác thảo được một số lớp

thiết kế. Các mối quan hệ giữa các lớp phân tích quan trọng này được dùng để xác định các mối quan hệ giữa các lớp thiết kế tương ứng.

2. Xác định các lớp hoạt động

Các lớp hoạt động do hệ thống yêu cầu có thể được xác định bằng cách xem xét các yêu cầu đồng thời đối với hệ thống, chẳng hạn như:

- Các yêu cầu về hiệu năng, lưu lượng thông qua, về tính sẵn sàng của hệ thống mà các tác nhân yêu cầu cần có khi tương tác với hệ thống. Chẳng hạn, nếu có một tác nhân nào đó có các yêu cầu cao về thời gian phúc đáp, thì yêu cầu đó được quản lý bằng một đối tượng hoạt động dành riêng để nhận đầu vào từ tác nhân và cung cấp đầu ra cho tác nhân đó.
- Sự phân bố hệ thống trên các nút. Các đối tượng hoạt động cần hỗ trợ bằng sự phân phối trên nhiều nút khác nhau
- Các yêu cầu khác như là các yêu cầu về sự khởi động và kết thúc hệ thống, tính sống động, việc tránh bế tắc, tránh thiếu tài nguyên, tái cấu hình các nút và năng lực kết nối.

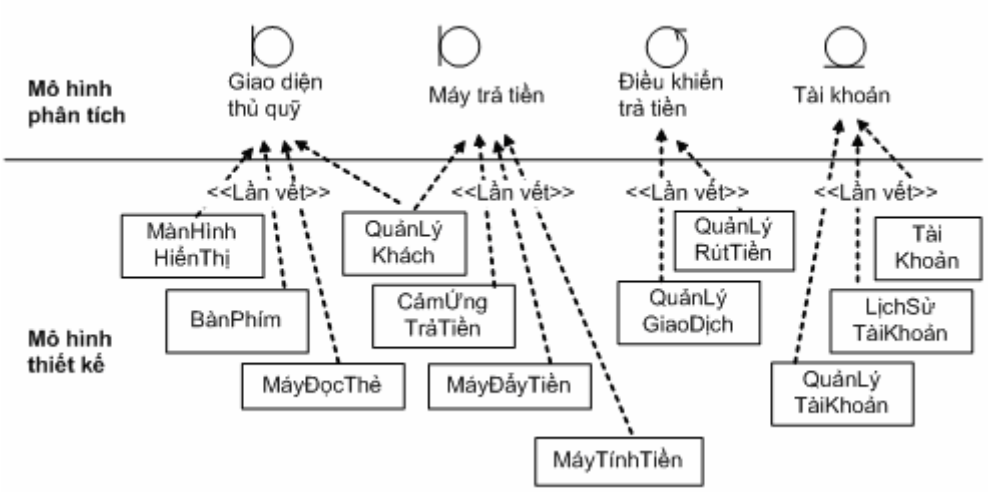
Để phác thảo các lớp hoạt động ban đầu có thể sử dụng các kết quả phân tích và mô hình triển khai làm đầu vào rồi sau đó bố trí các thiết kế tương ứng của các lớp phân tích (hoặc các bộ phận của chúng) cho các nút thông qua các lớp hoạt động.

Một khả năng khác để phác thảo các lớp hoạt động là sử dụng các hệ thống con được xác định trước đó và phân toàn bộ một hệ thống con cho một nút riêng bằng cách xác định một lớp hoạt động bên trong hệ thống con.

Ví dụ: Tìm các lớp thiết kế từ các lớp phân tích khi tinh chế nó để phù hợp với yêu cầu của môi trường triển khai:

Phác thảo các lớp hoạt động: trước đây, chúng ta đã xác định các lớp phân tích như lớp *quản lý khách, quản lý giao dịch (rút tiền), tài*

khoản. Bây giờ, chúng ta nhận thấy khách hàng quan tâm tới chức năng được cung cấp bởi lớp phân tích *quản lý khách*. Còn việc xử lý yêu cầu là nhiệm vụ của lớp quản lý rút tiền và lớp quản lý tài khoản đã được tách riêng đặt trên các máy dịch vụ. Do vậy ta xác định được các lớp hoạt động như các lớp có hình viền đậm trên hình 7.11.



Hình 7.11. Các lớp thiết kế nhận được từ các lớp phân tích khi làm mịn

8

MÔ HÌNH KIẾN TRÚC VẬT LÝ

Kiến trúc vật lý đề cập đến việc mô tả chi tiết hệ thống về phương diện phần cứng và phần mềm của hệ thống. Đồng thời nó cũng mô tả cấu trúc vật lý và sự phụ thuộc của các mô-đun cộng tác trong cài đặt những khái niệm đã được định nghĩa trong kiến trúc logic.

Kiến trúc vật lý của hệ thống liên quan nhiều đến cài đặt, do vậy, nó được mô hình hóa trong các *sơ đồ thành phần* (*Component Diagram*) và *sơ đồ triển khai* (*Deployment Diagram*). Sơ đồ thành phần chứa các thành phần bao gồm các đơn vị mã chương trình và cấu trúc các tệp (mã nguồn và nhị phân). Sơ đồ triển khai chỉ ra kiến trúc hệ thống khi thực thi, bao gồm các thiết bị vật lý và những phần mềm đặt trên đó.

8.1. SƠ ĐỒ THÀNH PHẦN

Sơ đồ thành phần (*Component Diagram*) là sơ đồ mô tả các thành phần và sự phụ thuộc của chúng trong hệ thống. Nói rõ hơn, sơ đồ thành phần được xem như là tập các biểu tượng thành phần biểu diễn cho các thành phần vật lý trong một hệ thống. Ý tưởng cơ bản của sơ đồ thành phần là tạo ra cho những người thiết kế và phát triển hệ thống một bức tranh chung về các thành phần của hệ thống.

8.1.1. Các thành phần trong sơ đồ

Các thành phần của hệ thống có thể là:

- *Thành phần mã nguồn*, có ý nghĩa vào thời điểm dịch chương trình. Thông thường nó là tập các chương trình cài đặt các lớp. Ví dụ, trong C++, mỗi tệp *.cpp* và *.h* là một thành phần. Trước khi phát sinh mã chương trình, phải thực hiện ánh xạ từng tệp vào thành phần tương ứng, thông thường mỗi lớp được ánh xạ vào hai tệp (*.cpp,* và *.h*).

- *Thành phần mã nhị phân* là mã trình nhị phân được dịch từ mã chương trình nguồn. Nó có thể là tệp mã đích (*.obj*), tệp thư viện tĩnh (*.lib*) hay tệp thư viện động (*.dll*). Thành phần nhị phân được sử dụng để liên kết, hoặc để thực thi chương trình (đối với thư viện động).

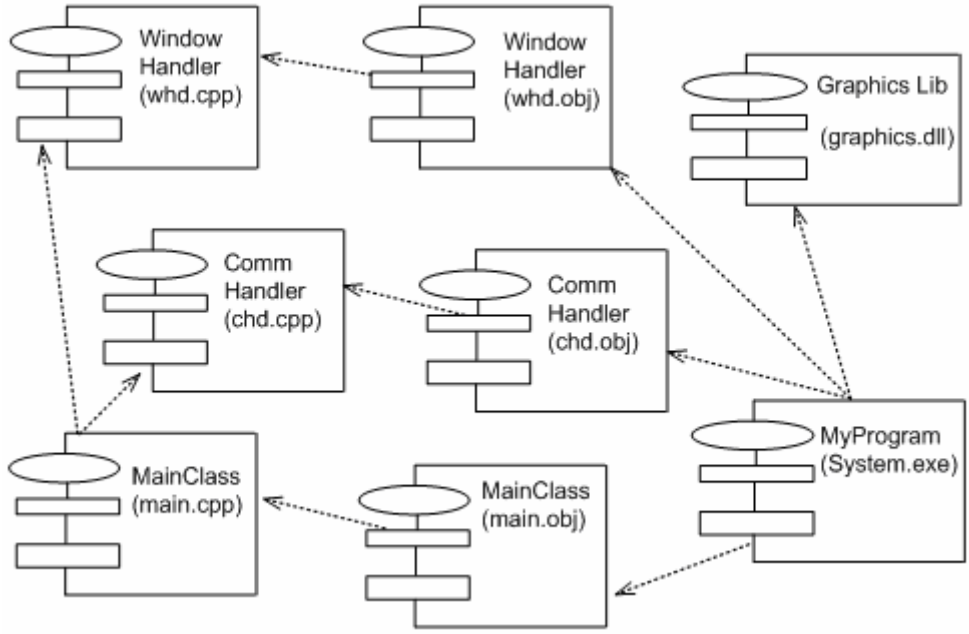
- *Thành phần thực thi* là tệp chương trình có thể thực thi được (các tệp *.exe*). Nó là kết quả của chương trình liên kết các thành phần nhị phân.

Với sơ đồ thành phần, người phát triển hệ thống thực hiện dịch hay triển khai hệ thống sẽ biết thư viện mã trình nào tồn tại và những tệp có thể thực thi (*.exe*) khi dịch và liên kết thành công.

Giữa các thành phần chỉ có một loại quan hệ phụ thuộc được biểu diễn bằng đường mũi tên đứt nét. Kết nối phụ thuộc cho biết thành phần phụ thuộc phải dịch sau thành phần kia.

Lưu ý, nên tránh phụ thuộc vòng trong sơ đồ thành phần.

Ví dụ: sơ đồ thành phần mô tả sự phụ thuộc giữa các thành phần của hệ thống (hình 8.1).



Hình 8.1. Sự phụ thuộc của các thành phần trong sơ đồ thành phần

Trong UML có một số biểu tượng biểu diễn cho các thành phần:

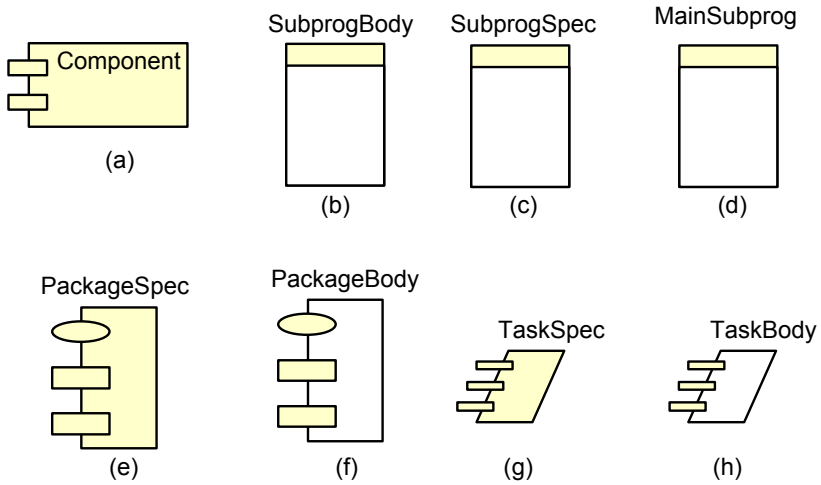
- *Thành phần:* biểu tượng thành phần (hình 8.2.a) được sử dụng để biểu diễn mô-đun chương trình có các giao diện. Trong đặc tả có xác định kiểu *Stereotype* (AciveX, Applet, DLL, exe,...).

- *Đặc tả và thân chương trình con:* biểu tượng thành phần cho đặc tả chương trình con (hình 8.2.b), và cài đặt của chương trình con (hình 8.2.c). Chương trình con không chứa các định nghĩa lớp.

- *Chương trình chính:* biểu tượng thành phần (tệp) chứa điểm vào của chương trình chính (hình 8.2.d), ví dụ trong C/C++ đó là tệp chứa hàm *main()*.

- *Đặc tả và thân của gói:* đặc tả gói (hình 8.2.e) là tệp *header* chứa thông tin về các hàm thành phần của lớp, ví dụ đặc tả gói trong C/C++ là tệp *.h* định nghĩa các hàm *prototype*. Biểu tượng cho thân gói (hình 8.2.f) gồm mã các lệnh của các hàm thành phần của lớp chứa trong gói. Trong C/C++, thành phần này là tệp *.cpp*.

- *Đặc tả và nội dung nhiệm vụ:* các biểu tượng (hình 8.2.g, 8.2.h) biểu diễn cho phần đặc tả và nội dung của những nhiệm vụ độc lập:



Hình 8.2. Các thành phần của hệ thống

Tương tự như các phần tử khác trong UML, các thành phần có thể bổ sung một số đặc tả chi tiết:

+ Stereotype: điều khiển biểu tượng nào sẽ được sử dụng để biểu diễn thành phần. Nó có thể là một trong các lựa chọn: <none>, đặc tả chương trình con, chương trình chính, đặc tả gói, nội dung của gói, đặc tả nhiệm vụ, nội dung công việc, ActiveX, Applet, ứng dụng,...

+ *Ngôn ngữ:* Rose cho phép lựa chọn ngôn ngữ lập trình cho từng thành phần, như C++, Java, Visual Basic, Oracle 8,...

+ *Khai báo:* phụ thuộc được gộp vào mã chương trình cho mỗi thành phần. Lệnh #include của C++ được xem như là lệnh khai báo.

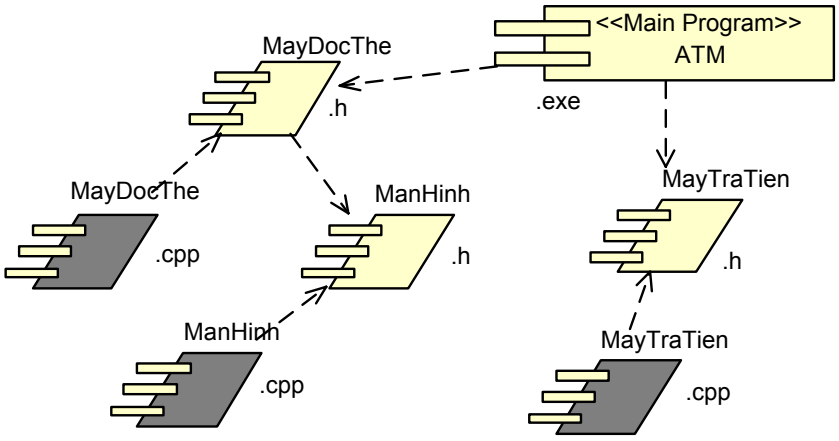
+ *Lớp:* trước khi phát sinh mã chương trình thì lớp phải được ánh xạ vào thành phần. Điều này báo cho Rose biết mã chương trình của lớp sẽ được ghi vào tệp nào. Có thể ánh xạ một hay nhiều lớp vào một thành phần.

8.1.2. Sơ đồ thành phần trong ATM

Sơ đồ thành phần cho ta cái nhìn vật lý về mô hình hệ thống. Có ba loại chính trong sơ đồ, đó là thành phần khả thi, thành phần mã nguồn và các thư viện. Trong Rose, mỗi lớp có thể được ánh xạ vào một thành phần mã nguồn. Hình 8.3 mô tả sơ đồ thành phần của ATM trên máy trạm. Nếu chúng ta chọn ngôn ngữ lập trình C++ để cài đặt thì mỗi lớp sẽ có hai tệp tương ứng là *.cpp* và *.h* riêng biệt.

Ví dụ:

Lớp ManHinh ánh xạ thành hai thành phần và sinh mã tương ứng thành hai tệp *.h* (header) và tệp *.cpp* (thân của lớp). Khi tất cả các lớp dịch xong thì ta mới có thành phần thực thi *ATMClient.exe*.

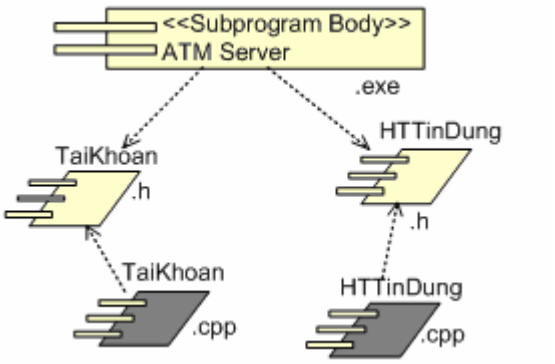


Hình 8.3. Sơ đồ thành phần của ATM trên máy trạm

Hệ thống ATM có hai tiến trình xử lý chính:

- *Tiến trình thứ nhất* là ATM trên máy trạm gồm các thành phần Máy trả tiền, Máy đọc thẻ và Màn hình.

- *Tiến trình thứ hai* là máy chủ ATM có Tài khoản và HT tín dụng. Sơ đồ thành phần của máy chủ ATM được mô tả như hình 8.4.



Hình 8.4. Sơ đồ thành phần của Máy chủ ATM

Có thể có nhiều sơ đồ thành phần cho một hệ thống, số lượng này phụ thuộc vào các hệ thống con của nó. Mỗi hệ thống con là gói thành phần, do vậy, hệ thống ATM có hai gói: gói *Máy trạm* và gói *Máy chủ*.

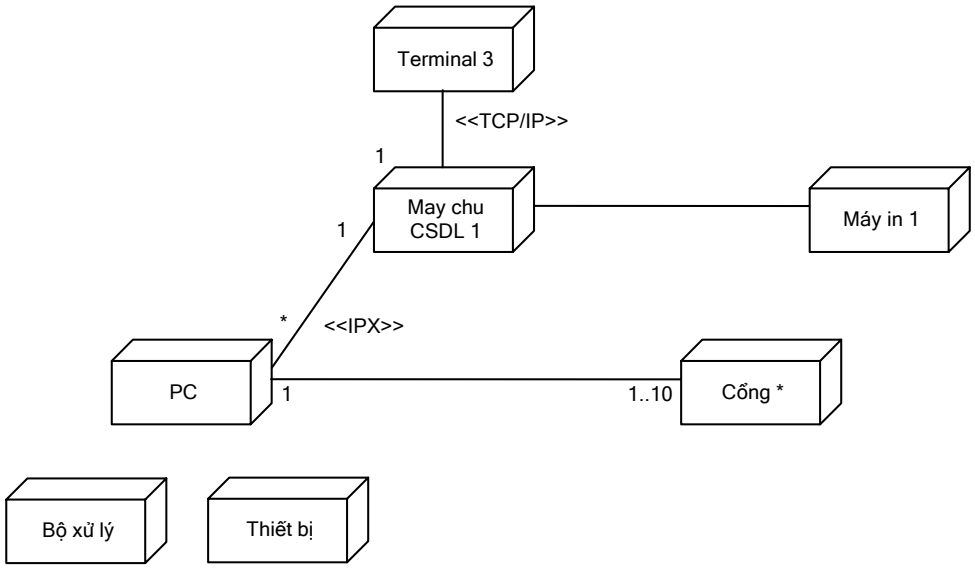
8.2. SƠ ĐỒ TRIỂN KHAI

Sơ đồ triển khai (Deployment Diagram) chỉ ra cấu hình các phần tử xử lý lúc chương trình chạy, các nút trên mạng và các tiến trình phần mềm thực hiện trên những phần tử đó. Nó chỉ ra mối quan hệ giữa các phần cứng và phần mềm của hệ thống.

Các tiến trình (Process) phần mềm thực hiện trên các phần tử xử lý được gọi tắt là tiến trình. Đó là luồng thực hiện của một chương trình trong một bộ xử lý. Một chương trình thực thi được xem như là một tiến trình. Các tiến trình thường được gán các mức ưu tiên và bộ xử lý sẽ thực hiện những tiến trình có mức ưu tiên cao hơn.

Sơ đồ triển khai chỉ ra toàn bộ các nút trên mạng, các mối kết nối giữa chúng và các tiến trình chạy trên chúng. Mỗi nút là một đối tượng vật lý (các thiết bị) có tài nguyên tính toán. Chúng có thể là máy tính, máy in, máy đọc ảnh, thiết bị truyền tin,... Các nút được kết nối với nhau thông qua các giao thức (*protocol*) như các giao thức "TCP/IP".

Ví dụ:



Hình 8.5. Sơ đồ triển khai hệ thống đóng/mở cửa trong 1 tòa nhà

Hệ thống bao gồm máy chủ nối với các máy tính PC điều khiển đóng/mở cửa ra vào. Số lượng PC chưa xác định. Mỗi PC có thể điều khiển 10 cửa. Có 3 máy tính đầu cuối. Máy chủ và PC kết nối với nhau thông qua giao thức IPX,..

8.2.1. Các phần tử (nút) của sơ đồ triển khai

1. Bộ xử lý (Processor)

Bộ xử lý (Processor) của máy tính, máy chủ, trạm làm việc,... Các bộ xử lý được đặc tả chi tiết bằng cách bổ sung thêm các thông tin:

- *Stereotype:* để phân nhóm các bộ xử lý.

- *Đặc tính:* mô tả các tính chất vật lý của mỗi bộ xử lý như: tốc độ tính toán, dung lượng bộ nhớ,...

- *Lịch biểu (Schelduling):* mô tả loại lịch biểu thời gian xử lý, bao gồm:

+ *Preemptive:* cho phép những tiến trình có mức ưu tiên cao hơn có thể chiếm quyền xử lý đối với những tiến trình có mức ưu tiên thấp hơn

+ *Non Preemptive:* không có ưu tiên, một tiến trình chỉ dừng khi nó tự kết thúc

+ *Cyclic:* chỉ ra chu kỳ điều khiển giữa các tiến trình

+ *Executive:* các lịch biểu được điều khiển bằng thuật toán, bằng chương trình.

+ *Manual:* tiến trình được điều khiển bằng người sử dụng.

2. Thiết bị

Thiết bị là máy móc hay bộ phận phần cứng nhưng không phải là bộ xử lý trung tâm, như màn hình, máy in, máy vẽ,... Thiết bị cũng có thể đặc tả một số thông tin chi tiết như: *Stereotype và một số tính chất vật lý.*

3. Kết nối

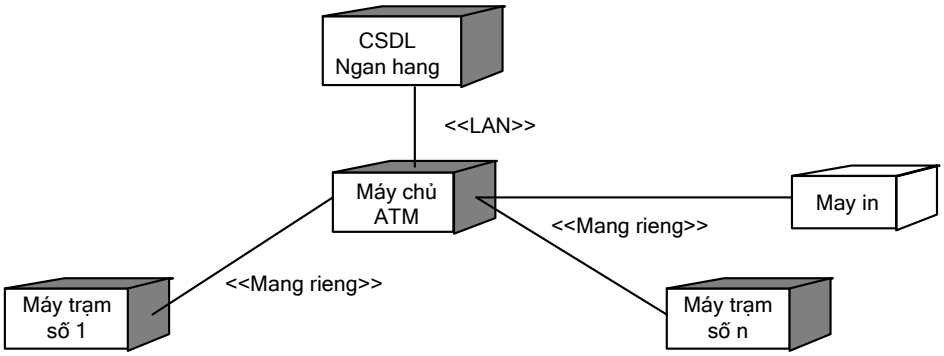
Kết nối là liên kết vật lý giữa 2 bộ xử lý, 2 thiết bị hay giữa thiết bị và bộ xử lý. Thông thường, kết nối biểu diễn kết nối mạng vật lý giữa các nút trong mạng. Chúng có thể là liên kết *internet* giữa các nút. Có thể gán stereotype cho kết nối, đồng thời nó cũng được gán đặc tính nói về chi tiết của kết nối vật lý.

4. Tiến trình

Tiến trình là luồng thực hiện đơn chạy trong bộ xử lý. Một tệp khả thi được coi là một tiến trình. Tiến trình có thể được hiển thị hoặc không được hiển thị trong sơ đồ triển khai. Nếu được hiển thị thì nó được liệt kê ngay dưới bộ xử lý nơi nó chạy. Các tiến trình được gán mức ưu tiên. Nếu bộ xử lý mà trên đó các tiến trình chạy sử dụng *scheduding* theo mức ưu tiên, thì mức ưu tiên của tiến trình sẽ được xác định khi nó chạy.

8.2.2. Sơ đồ triển khai của hệ thống ATM

Sơ đồ triển khai chỉ ra cách bố trí vật lý các thành phần của hệ thống trên mạng. Hệ thống ATM có nhiều tiến trình con chạy trên các thiết bị (máy trạm) được gọi là các nút và chúng kết nối với máy chủ. Máy chủ ATM được kết nối với CSDL Ngân hàng qua mạng LAN. Như vậy, hệ thống ATM được xây dựng theo kiến trúc ba tầng: tầng CSDL, tầng máy chủ (ATM Server) và tầng các máy trạm (ATMClient) được mô tả trong sơ đồ triển khai như hình 8.6.



Hình 8.6. Sơ đồ triển khai của hệ thống ATM

9

MÔ HÌNH PHÂN TÍCH VÀ THIẾT KẾ MỘT CA SỬ DỤNG

Trong chương 3, những khái niệm chung nhất về mô hình ca sử dụng đã được trình bày. Đến chương này, chúng ta đi sâu nghiên cứu về các mô hình phân tích và thiết kế một ca sử dụng.

9.1. PHÂN TÍCH MỘT CA SỬ DỤNG

Việc phân tích một ca sử dụng bao gồm:

- Xác định các lớp phân tích mà các đối tượng của nó cần thiết phải thực hiện luồng các công việc tương ứng với các sự kiện của ca sử dụng.

- Phân phối hành vi của ca sử dụng cho các đối tượng phân tích tương tác với nhau.

- Nắm bắt các yêu cầu chuyên biệt để thực thi ca sử dụng.

9.1.1. Xác định các lớp phân tích

1. Mục đích của việc xác định các lớp phân tích

Lớp phân tích thể hiện một sự trừu tượng của một hoặc nhiều lớp và/hoặc hệ thống con. Như đã trình bày trong chương 4, có 3 kiểu lớp phân tích cơ bản (3 khuôn mẫu-steoreotype): *lớp biên, lớp điều khiển và lớp thực thể.*

Lớp biên được sử dụng để mô hình hóa sự tương tác giữa hệ thống và các tác nhân của nó. Tương tác này bao gồm có việc nhận và hiển thị thông tin từ các yêu cầu (đến và đi) của người dùng và các hệ thống ngoài. Mỗi lớp biên phải liên quan đến ít nhất 1 tác nhân.

Các lớp biên thường thể hiện như các trừu tượng hóa của các cửa sổ (window), các biểu mẫu (form), các bảng hiển thị (pane), các giao diện truyền thông (communication interface), giao diện máy in (printer interface), các bộ cảm biến (sensor), các đầu cuối (terminal),… nhưng ở một mức độ tương đối cao và mang tính khái niệm chứ không đi quá sâu vào chi tiết.

Lớp điều khiển thể hiện sự phối hợp, sắp xếp trình tự, các giao dịch, sự điều khiển của các đối tượng và thường được sử dụng để gói lại các điều khiển liên quan đến một ca sử dụng cụ thể. Các khía cạnh động của hệ thống được mô hình hóa qua các lớp điều khiển.

Lớp thực thể được dùng để mô hình hóa các thông tin tồn tại lâu dài và có thể được lưu trữ. Nó thường thể hiện các cấu trúc dữ liệu logic và góp phần làm rõ về các thông tin mà hệ thống phải thao tác trên chúng.

2. Khái niệm về thực thi ca sử dụng

Thực thi ca sử dụng là một sự cộng tác trong mô hình phân tích. Nó mô tả cách thức làm thế nào để một ca sử dụng cụ thể được thực hiện và thể hiện ra dưới dạng các lớp phân tích và các đối tượng phân tích tương tác với nhau.

3. Xác định các lớp phân tích

Để xác định các lớp phân tích, ta làm mịn ca sử dụng dựa trên luồng các sự kiện mô tả ca sử dụng bằng văn bản. Cụ thể các bước như sau:

- Xác định các lớp thực thể bằng việc nghiên cứu mô tả ca sử dụng và mô hình lĩnh vực chi tiết, sau đó xét xem thông tin nào cần được lưu trữ và cần thao tác trong thực thi ca sử dụng.

- Xác định một lớp biên trung tâm (quan trọng, có ý nghĩa về mặt cấu trúc hệ thống) đối với tác nhân là người và chọn lớp này làm cửa sổ cơ bản trong giao diện NSD mà tác nhân tương tác với nó. Nếu tác nhân đã tương tác với một lớp biên nào đó, thì có thể dùng lại lớp này.

- Xác định một lớp biên cho mỗi lớp thực thể (đã được xác định ở bước trước).

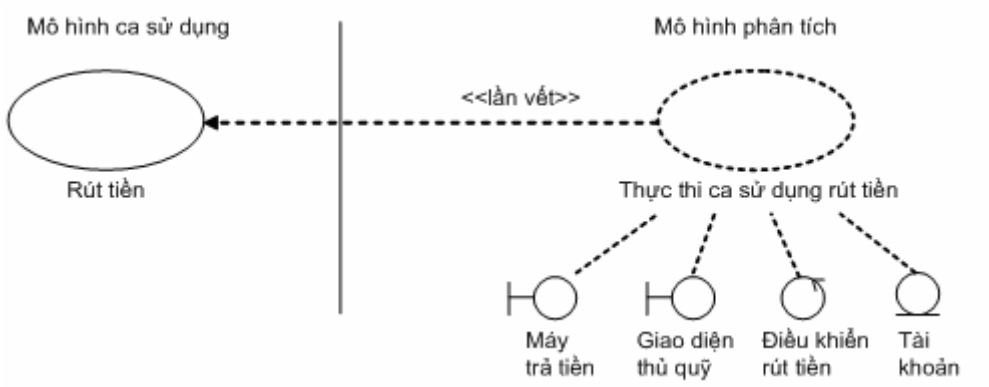
- Xác định một lớp biên trung tâm cho mỗi tác nhân là hệ thống ngoài, chọn nó làm đại diện cho giao diện truyền thông.

- Xác định một lớp điều khiển chịu trách nhiệm quản lý việc điều khiển và phối hợp hoạt động thực thi ca sử dụng, sau đó làm mịn lớp điều khiển này theo các yêu cầu của thực thi ca sử dụng.

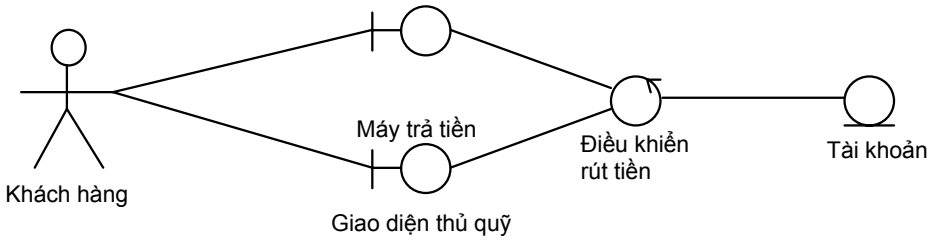
- Nhóm các lớp phân tích tham gia một thực thi ca sử dụng vào một sơ đồ lớp phân tích và sử dụng nó để chỉ ra các mối quan hệ được sử dụng trong việc thực thi ca sử dụng.

4. Ví dụ

Tìm các lớp phân tích tham gia vào thực thi ca sử dụng ***rút tiền*** và sơ đồ lớp phân tích tương ứng:



*Hình 9.1. Các lớp phân tích tham gia vào thực thi ca sử dụng **rút tiền***

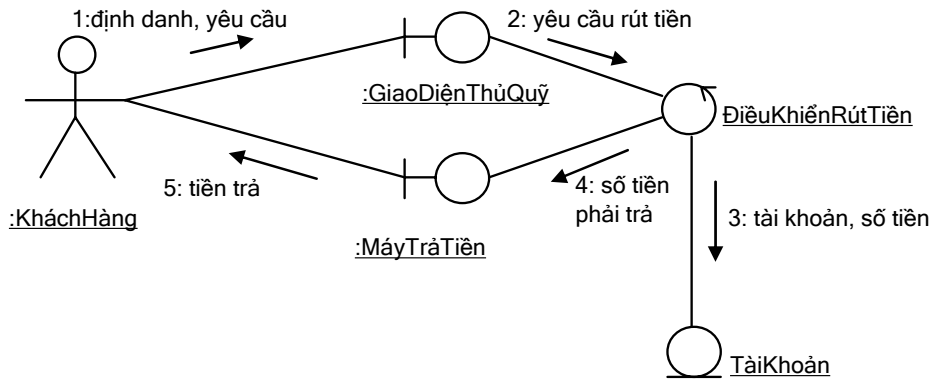


*Hình 9.2. Sơ đồ lớp phân tích mô tả quan hệ liên kết giữa các lớp thực thi ca sử dụng **rút tiền***

9.1.2. Mô tả các tương tác giữa các đối tượng phân tích

Sau khi phác thảo được các lớp phân tích, chúng ta cần mô tả cách thức các đối tượng đó tương tác với nhau như thế nào. Đó chính là hành vi của hệ thống. Hành vi của hệ thống là một bản mô tả những việc hệ thống làm, mà không cần phải giải thích tại sao và làm như thế nào. Mô tả hành vi hệ thống được tiến hành bằng cách sử dụng các sơ đồ cộng tác hay trình tự, chúng chứa các thể hiện của tác nhân tham gia, các đối tượng phân tích và các mối liên kết giữa chúng. Nếu ca sử dụng có các luồng hoặc các luồng công việc khác nhau và tách biệt thì nên tạo một sơ đồ cộng tác riêng cho mỗi luồng.

Ví dụ: Sơ đồ cộng tác tập trung vào việc tìm kiếm các yêu cầu và trách nhiệm của các đối tượng. Trong khi đó sơ đồ trình tự tập trung vào chuỗi tương tác theo thời gian. Trong trường hợp này ta sử dụng chủ yếu là sơ đồ tương tác. Trong sơ đồ tương tác, tương tác giữa các đối tượng được thể hiện bằng đường kết nối giữa các đối tượng và các thông báo ghi bên cạnh kết nối này. Sơ đồ cộng tác xuất phát từ một điểm khởi đầu của luồng sự kiện ca sử dụng, sau đó đi theo luồng từng bước và xác định xem đối tượng phân tích nào và các tương tác của các thể hiện nào là cần thiết để thực thi ca sử dụng đó.



*Hình 9.3. Sơ đồ cộng tác thực thi ca sử dụng **rút tiền***

Lưu ý:

- Một ca sử dụng được gọi bằng một thông báo từ một tác nhân chuyển tới một đối tượng biên.

- Mỗi lớp phân tích được xác định trong bước trước phải có ít nhất một đối tượng phân tích tham gia vào một sơ đồ cộng tác.

- Các kết nối trong sơ đồ là thể hiện của các liên kết giữa các lớp phân tích.

- Cần tập trung vào các quan hệ (liên kết) giữa các đối tượng và các yêu cầu (thể hiện qua các thông báo) đối với các đối tượng cụ thể.

- Sơ đồ cộng tác phải kiểm soát được mọi mối quan hệ cần thiết cho thực thi ca sử dụng.

9.1.3. Mô tả luồng các sự kiện phân tích (flow of events-analysis)

Cùng với sơ đồ, ta cần mô tả bằng văn bản để tăng độ dễ hiểu và dễ dùng. Các văn bản nên viết dưới dạng tương tác giữa các đối tượng với nhau, đặc biệt là đối tượng điều khiển, tương tác với các đối tượng khác để thực hiện ca sử dụng.

Ví dụ: Khách hàng muốn rút tiền sẽ kích hoạt giao diện thủ quỹ (bằng thẻ hay bàn phím) để chỉ ra định danh của mình (1). Giao diện

thủ quỹ tiếp nhận và kiểm tra định danh của khách hàng. Nếu đúng thì yêu cầu khách hàng nhập vào yêu cầu và chuyển yêu cầu cho đối tượng điều khiển rút tiền (2). Đối tượng điều khiển rút tiền yêu cầu đối tượng tài khoản thẩm định lại yêu cầu này, nếu yêu cầu là đúng thì giảm số dư tài khoản và thông báo trả lại (3). Tiếp đó đối tượng điều khiển rút tiền ra lệnh cho máy trả tiền chuyển số tiền khách yêu cầu để trả khách (4). Sau đó khách hàng nhận được số tiền từ máy trả tiền.

Nhận xét:

- Khi so sánh mô tả bằng văn bản như trên với luồng các sự kiện của ca sử dụng được mô tả trong phần nắm bắt yêu cầu, ta thấy mô tả trong phần nắm bắt yêu cầu là mô tả hành vi được quan sát từ bên ngoài của ca sử dụng. Còn ở đây, ta *tập trung vào mô tả các hoạt động bên trong của hệ thống để thực thi ca sử dụng* với sự cộng tác của các đối tượng logic của nó chính là các lớp.

- Sơ đồ cộng tác không mô tả kết quả thực thi của thao tác được gọi. Nó thiếu các chi tiết về sự đáp ứng của hệ thống hay hành vi của hệ thống. Để có thể hiểu được hành vi đầy đủ của hệ thống và tốt hơn ta nên tìm hiểu thêm về trạng thái của hệ thống khi thực hiện các thao tác, đặc biệt đối với ca sử dụng phức tạp. Trong trường hợp này, sơ đồ trạng thái là một công cụ thích hợp được sử dụng bổ sung để mô tả trạng thái các đối tượng tham gia tương tác.

9.1.4. Nắm bắt các yêu cầu chuyên biệt

Ta cần nắm bắt các yêu cầu (phi chức năng) cần cho việc thực thi một ca sử dụng mà đã được xác định trong phân tích nhưng phải được xử lý trong thiết kế và thực thi.

Ví dụ:

Các yêu cầu chuyên biệt cho việc thực thi ca sử dụng rút tiền:

- Khi khách hàng nhập vào thông tin kiểm tra, màn hình yêu cầu phải được hiện ra không chậm hơn 1 giây. Khi khách hàng đã đưa vào yêu cầu rút tiền, nếu không có gì trục trặc, tiền phải được trả sau không quá 10 giây. Trong trường hợp ngược lại phải có thông báo.

- Việc rút tiền phải được thực hiện vào mọi thời gian trong ngày.

9.2. THIẾT KẾ MỘT CA SỬ DỤNG

Mục tiêu của việc thiết kế một ca sử dụng:

- Xác định các lớp thiết kế và/hoặc các hệ thống con mà các thể hiện của chúng là cần thiết để thực hiện luồng các sự kiện của ca sử dụng đó.

- Phân phối hành vi của ca sử dụng cho các đối tượng thiết kế tương tác và/hoặc cho các hệ thống con tham gia.

- Xác định các yêu cầu về các tác vụ của các lớp thiết kế và/hoặc các hệ thống con và các giao diện của chúng.

- Nắm bắt các yêu cầu triển khai cho ca sử dụng đó để thể hiện vào lớp thiết.

9.2.1. Thiết kế một ca sử dụng dưới dạng cộng tác của các lớp

Chúng ta xác định các lớp thiết kế cần thiết để thực thi ca sử dụng thiết kế theo các cách sau:

- Nghiên cứu các lớp phân tích tham gia vào việc thực thi ca sử dụng phân tích. Xác định các lớp thiết kế bằng cách lần vết tới các lớp phân tích tương ứng.

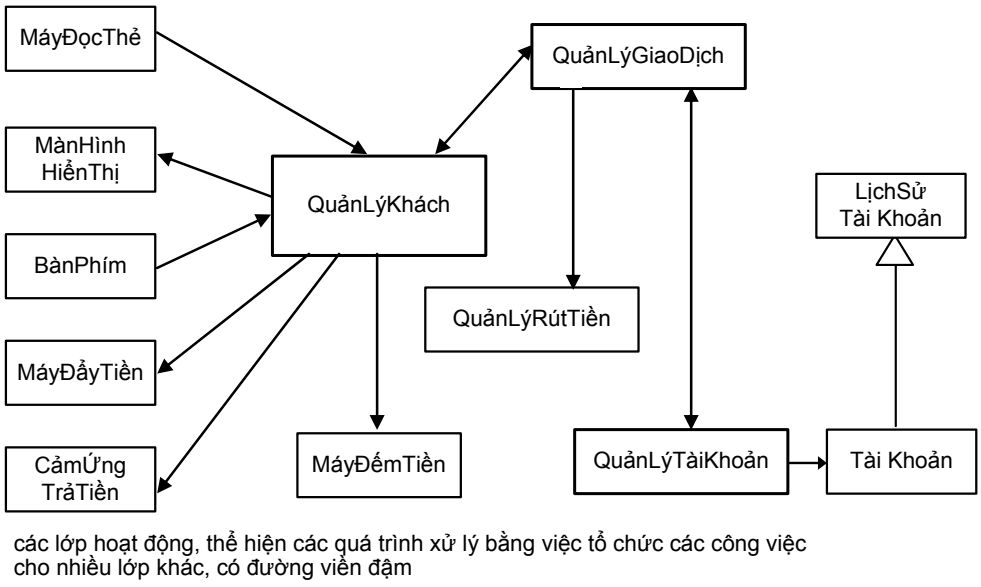
- Nghiên cứu các yêu cầu đặc biệt của việc thực thi ca sử dụng phân tích tương ứng. Xác định các lớp thiết kế cần để thực thi các yêu cầu đặc biệt này.

- Nếu vẫn còn thiếu một lớp thiết kế nào cần để thực hiện một ca sử dụng cụ thể thì lớp được yêu cầu đó phải được xác định thêm.

- Ta nhóm các lớp thiết kế tham gia thực thi ca sử dụng vào một sơ đồ lớp. Sử dụng sơ đồ lớp này để chỉ ra các mối quan hệ đã được dùng trong việc thực thi ca sử dụng.

Như vậy, thực thi ca sử dụng thiết kế là một sự cộng tác trong mô hình thiết kế mô tả cách thức thực thi một ca sử dụng cụ thể và thể hiện dưới dạng sự cộng tác của các lớp thiết kế và các đối tượng của chúng.

Ví dụ [25]: Đầu tiên sử dụng các lớp thiết kế đã tìm được từ các lớp phân tích khi làm mịn (đã làm ở phần trước) và xây dựng sơ đồ lớp thiết kế sao cho ca sử dụng rút tiền được thực hiện với các lớp thiết kế đã nhận được.



Hình 9.4. Sơ đồ lớp tham gia thực hiện ca sử dụng rút tiền

9.2.2. Thiết kế một ca sử dụng dưới dạng cộng tác của các đối tượng trong các lớp

Sau khi phác thảo các lớp thiết kế, chúng ta cần mô tả cách thức mà các đối tượng thiết kế tương tác với nhau. Điều này được tiến

hành bằng cách sử dụng các sơ đồ trình tự chứa các thể hiện của các tác nhân tham gia, các đối tượng thiết kế và sự truyền thông báo giữa chúng. Nếu các ca sử dụng có các luồng hoặc các luồng con khác nhau và tách biệt thì phải tạo ra các sơ đồ trình tự cho mỗi luồng tách biệt đó.

Trước hết nghiên cứu việc thực thi ca sử dụng phân tích tương ứng để đưa ra một phác thảo về chuỗi các thông báo cần thiết giữa các đối tượng thiết kế. Trong một số trường hợp, có thể chuyển trực tiếp một sơ đồ cộng tác thực thi ca sử dụng phân tích thành một phác thảo ban đầu của một sơ đồ trình tự tương ứng.

Sơ đồ trình tự cho một kịch bản của một ca sử dụng mô tả theo thứ tự các sự kiện được phát sinh bởi tác nhân ngoài và các sự kiện bên trong hệ thống. Như vậy để xây dựng sơ đồ trình tự cần xác định các thao tác hệ thống mà một tác nhân yêu cầu. Nghĩa là cần tìm các sự kiện vào dựa trên các ca sử dụng và các kịch bản của chúng.

Có thể tóm tắt các bước xây dựng sơ đồ trình tự như sau:

- Xác định từng tác nhân thao tác trực tiếp với hệ thống.

- Từ tiến trình các sự kiện cơ bản của ca sử dụng, xác định các sự kiện hệ thống được phát sinh bởi mỗi tác nhân.

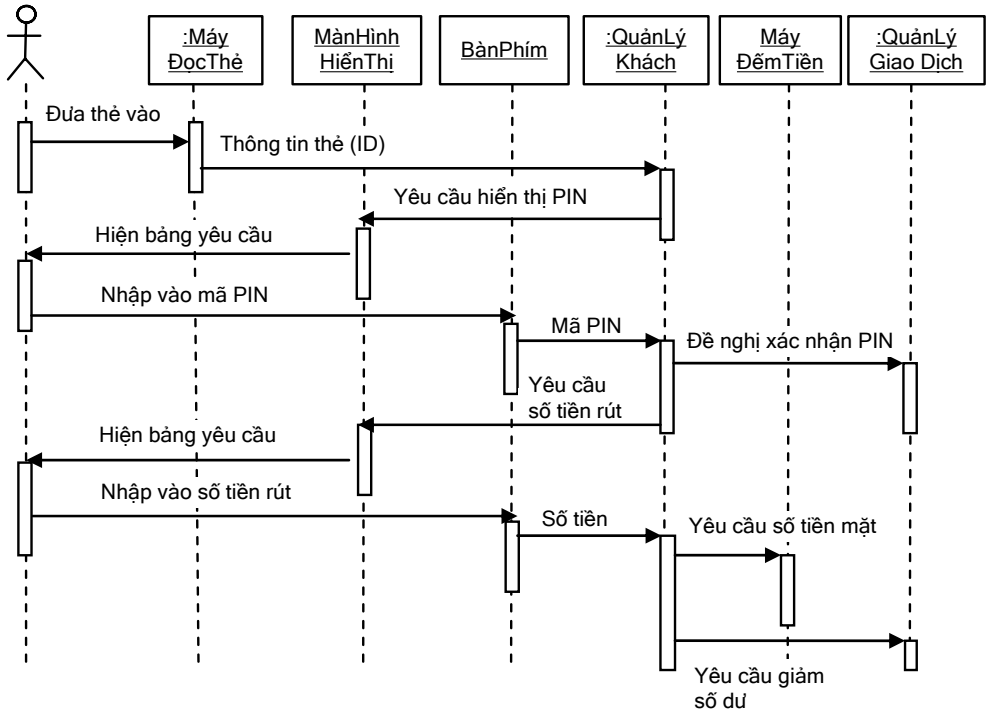
- Nên viết thêm văn bản mô tả ca sử dụng vào để tăng độ dễ hiểu.

Nhận xét:

- Khi chi tiết hóa các sơ đồ tương tác, phần lớn các trường hợp ta sẽ tìm ra con đường mới mà ca sử dụng đó có thể lựa chọn. Những con đường như thế có thể mô tả bằng các nhãn của các sơ đồ hoặc trong chính các sơ đồ tương tác của chúng. Khi đưa thêm thông tin mới vào, người phát triển thường phát hiện ra các ngoại lệ mới mà đã bị bỏ qua trong quá trình nắm bắt hoặc phân tích các yêu cầu.

- Khi thiết kế sơ đồ cộng tác, chúng ta cần cố gắng nhận dạng và phát hiện các mẫu gán trách nhiệm *(GRASP-General Responbilities Assigning Sofware Patterns)* để sử dụng.

Ví dụ:



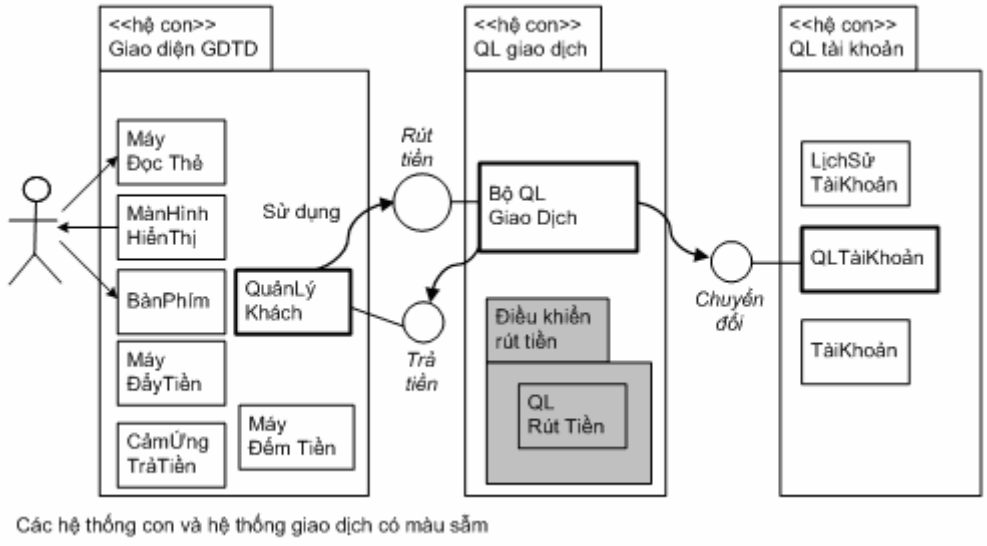
Hình 9.5. Sơ đồ trình tự luồng các sự kiện thiết kế thực hiện một phần ca sử dụng rút tiền

9.2.3. Thiết kế một ca sử dụng dưới dạng các hệ thống con tham gia cùng với các giao diện giữa chúng

Trong cách tiếp cận top-down, cần phải nắm bắt các yêu cầu về các hệ thống con và các giao diện của chúng trước khi thiết kế phần bên trong. Trong những trường hợp như thế, một thực thi ca sử dụng thiết kế có thể được mô tả ở nhiều mức trong hệ phân cấp các hệ thống con.

Việc xác định các hệ thống con đã được trình bày trong chương trước (chương 7). Còn việc tìm ra các hệ thống con cần để thực thi ca sử dụng có thể thực hiện bằng cách <<lần vết>> tới các *lớp phân tích* tham gia vào thực thi ca sử dụng phân tích tương ứng. Xác định các gói phân tích chứa các lớp phân tích đó, nếu có. Sau đó xác định các hệ thống con thiết kế mà chúng lần vết tới các gói phân tích đó. Các hệ thống con cùng tham gia thực thi ca sử dụng được đưa vào một sơ đồ lớp và dùng sơ đồ lớp này để đưa ra các mối quan hệ phụ thuộc giữa các hệ thống con đó và các giao diện đã được dùng khi thực thi ca sử dụng.

Ví dụ [25]:



Hình 9.6. Các hệ thống con, các hệ thống dịch vụ và các giao diện của chúng

Sau khi phác thảo được hệ thống con cần thiết để thực thi ca sử dụng, chúng ta cần mô tả cách thức mà các đối tượng của các lớp trong chúng tương tác trên một cấp độ của hệ thống. Việc này được tiến hành bằng cách sử dụng các sơ đồ trình tự chứa các thể hiện của

tác nhân tham gia, các hệ thống con và những sự truyền thông báo giữa chúng. Một mô tả như vậy trở nên khái quát hơn, đơn giản hơn và cho một khung nhìn kiến trúc thực thi ca sử dụng thiết kế rõ ràng hơn (đặc biệt có lợi trong trường hợp mà các lớp tham gia vào thực thi một ca sử dụng là khá lớn).

9.2.4. Nắm bắt các yêu cầu triển khai

Trong bước này, chúng ta nắm bắt và thể hiện mọi yêu cầu thực thi một ca sử dụng, chẳng hạn các yêu cầu phi chức năng đã được xác định trong thiết kế nhưng sẽ phải được xử lý trong triển khai.

TÀI LIỆU THAM KHẢO

1. Tiếng Anh

[1]. Acura S.T., *Software Process Modeling*, Universided Autonoma de Madrid, Spain, Matalia Jurist, Universided Poli Hemica de Madrid Spain, Springer 2005.

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

[3]. Booch G., Rumbaugh J. and Jacobson I., *The Unified Modeling Language User Guide,* Addision – Wesley, 1999.

[4]. [Larman C., Applying UML and Patterrns: *An Instruction to Object-Oriented Analysis and Design*, Prentice Hall, 1997.

[5]. Liang Y., *From use cases to classes: a way of building object model with UML,* Information and Software Technology, www.elservier.com/locate/infsof, 2003.

[6]. METI, Ministry of Economy, Trade and Industry, *Textbook for Software Design & Development Engineers, Object-Oriented Delevelopment*, Second Edition, Revised and Updated ByJapan Information Processing Development Corporation Japan Information-Technology Engineers Examination Center, 2002.

[7]. Michael B., William P., *Object – Oriented Modeling and Design for Database Applications,* Prentice Hall, New Jersey 1998.

[8]. Oestereich B., *Developing Software with UML, Object-Oriented Analysis and Design in Practice*, Addision – Wesley, 2000.

[9]. OMG, *The OMG Unified Modeling Language Specification*, http:// www.omg.org/uml, 1999.

[10]. Ould, M. *Managing Software Quality and Business Risk*, John Wiley and Sons, 1999.

[11]. Pressman R.S., *Software Engineering, a Practitioner's Approach*. 5th Edition, McGraw Hill, 2001.

[12]. Quatrani T., *Visual Modeling With Rational Rose and UML,* Addison-Wesley, http:// www.rational.com, 2000.

[13]. Royce W., *Software Project Management – A Unified Framework,* Addison-Wesley, 1998.

[14]. Silvia T., Acuna, *Natalia Juristo, Software process modeling,* Springer, Spain, 2005.

[15]. Sommerville I., *Software Engineering*, Addison-Wasley, 6th Edition, 2001, 7th editor Chapter 19, 2004.

[16]. Zhiming L., *Object-Oriented Software Development Using UML,* UNU /IIST, Macau 2001.

2. Tiếng Việt

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

[18]. Đoàn Văn Ban, *Phân tích thiết kế hướng đối tượng bằng UML,* Bài giảng cao học Viện Công nghệ thông tin, 2004.

[19]. Đặng Văn Đức, *Phân tích thiết kế hướng đối tượng bằng UML (Thực hành với Rational Rose),* NXB Khoa học và Kỹ thuật, Hà Nội, 2002.

[20]. Huỳnh Văn Đức và tập thể tác giả, *Giáo trình nhập môn UML,* NXB Lao động Xã hội, 2003.

[21]. Lê Văn Phùng, *Phân tích và thiết kế hệ thống thông tin,* NXB Lao động Xã hội, 2004.

[22]. Lê Văn Phùng, *Kỹ thuật phân tích và thiết kế hệ thống thông tin hướng cấu trúc,* NXB Thông tin và Truyền thông, 2009.

[23]. Lê Văn Phùng, *Kỹ nghệ phần mềm,* NXB Thông tin và Truyền thông, 2010.

[24]. Ngô Trung Việt, *Kỹ nghệ phần mềm - Cách tiếp cận của người thực hành, NXB Giáo dục,* tập 1/1997, tập 2, 3/1999.

[25]. Nguyễn Văn Vỵ, *Phân tích và thiết kế các hệ thống thông tin hiện đại hướng cấu trúc và hướng đối tượng,* NXB Thống kê, 2002.

[26]. Nguyễn Văn Vỵ, *Bài giảng cao học về phân tích thiết kế hướng đối tượng,* Đại học Công nghệ Hà Nội.

[27]. Nguyễn Văn Vỵ, Nguyễn Việt Hà, *Giáo trình kỹ nghệ phần mềm,* NXB Giáo dục Việt Nam, 2009.

MỤC LỤC

Các mô hình cơ bản
TRONG PHÂN TÍCH VÀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Chịu trách nhiệm xuất bản
NGUYỄN THỊ THU HÀ

Biên tập: NGÔ MỸ HẠNH
TRỊNH THU CHÂU
Trình bày sách: BÙI CHÂU LOAN
Sửa bản in: TRỊNH THU CHÂU
Thiết kế bìa: TRẦN HỒNG MINH

In 700 bản, khổ 16 x 24 cm tại Công ty In Hải Nam
Số đăng ký kế hoạch xuất bản: 863-2014/CXB/5-587/TTTT
Số quyết định xuất bản: 245/QĐ-NXB TTTT ngày 21 tháng 08 năm 2014
In xong và nộp lưu chiểu tháng 10 năm 2014.