Giáo trình

PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG HƯỚNG ĐỐI TƯỢNG SỬ DỤNG UML

Biên soạn

ThS. Phạm Nguyễn Cương

TS. Hồ Tường Vinh

Giới thiệu

Hệ thống phần mềm càng ngày càng trở nên phức tạp. Các ứng dụng hôm nay có những yêu cầu và kiến trúc đòi hỏi phức tạp hơn rất nhiều so với quá khứ. Các kỹ thuật, công cụ, và phương pháp luận phát triển hệ thống phần mềm đang thay đổi một cách nhanh chóng. Các phương pháp phát triển phần mềm chúng ta sẽ sử dụng trong tương lai có lẽ sẽ khác so với các phương pháp hiện hành đang sử dụng. Tuy nhiên, một điều hiển nhiên là phát triển hướng đối tượng và các khái niệm cơ bản của nó đang được sử dụng rộng rãi. Nhiều trường học đã nhận ra được điều này và đã tạo ra những khoá học phát triển hệ thống hướng đối tượng như một phần chính yếu của hệ thống thông tin tin học hoá và các chương trình khoa học máy tính. Giáo trình này dự kiến sẽ cung cấp một kiến thức nền tảng về phát triển các hệ thống hướng đối tượng cho các đối tượng sinh viên những năm cuối. Mục tiêu của giáo trình là cung cấp một mô tả rõ ràng về các khái niệm nền tảng phát triển hệ thống hướng đối tượng. Trong đó, nhấn mạnh đến tính đơn giản của tiếp cận giúp sinh viên có kiến thức về UML có thể dễ dàng nắm bắt để phát triển một hệ thống hướng đối tượng.

Mục tiêu

Sau khi học xong môn học này sinh viên có thể:

- Hiểu các nguyên lý nền tảng của kỹ thuật hướng đối tượng và các khái niệm về sự trừu tượng, tính bao bọc, tính thừa kế, và tính đa hình.
- Hiểu về một số quy trình phát triển hệ thống, nội dung các giai đoạn cơ bản của một qui trình phát triển, và một số phương pháp phân tích thiết kế hướng đối tượng.
- Tiếp cận toàn bộ qui trình phát triển hệ thống sử dụng các kỹ thuật hướng đối tượng
- Sử dụng UML như là một công cụ mô hình hoá trong quá trình phát triển hệ thống
- Phát triển hệ thống từ các mô hình use case được xem như là một mô hình phân tích nhằm biểu diễn đầy đủ yêu cầu hệ thống.
- Áp dụng một qui trình lặp, tập trung vào kiến trúc để phát triển một mô hình thiết kế đủ chi tiết, đủ mạnh đáp ứng với các nhu cầu:
    - Phù hợp với các yêu cầu hệ thống đã được thống nhất qua mô hình use case trong giai đoạn phân tích.
    - Tái sử dụng.
    - Dễ dàng để cài đặt hệ thống trong một ngôn ngữ và môi trường cụ thể.

PHẦN 1: TỔNG QUAN

Chương 1

GIỚI THIÊU VỀ PHƯƠNG PHÁP VÀ PHƯƠNG PHÁP LUẬN PHÁT TRIỂN HỆ THỐNG HƯỚNG ĐỐI TƯỢNG

Giới thiệu về phương pháp phát triển hướng chức năng

Đây là phương pháp cận truyền thống của ngành công nghiệp phần mềm trong đó quan điểm về phần mềm như là một tập hợp các chương trình (hoặc chức năng) và dữ liệu giả lập. Vậy chương trình là gì? Theo Niklaus Wirth, người tạo ra ngôn ngữ lập trình Pascal thì: "Chương trình = thuật giải + cấu trúc dữ liệu". Điều này có nghĩa rằng có hai khía cạnh khác nhau của hệ thống được tiếp cận, hoặc tập trung vào các chức năng của hệ thống hoặc tập trung vào dữ liệu. Chúng ta chia hướng tiếp cận này thành hai thời kỳ: thời kỳ vào những năm thập niên 70, tiếp cận phân tích và thiết kế hệ thống theo phương pháp gọi là Descartes. Ý tưởng chính trong cách tiếp cận này là một quá trình lặp phân rã hệ thống thành các chức năng và ứng dụng phương pháp lập trình cấu trúc đơn thể chương trình, việc phân rã kết thúc khi một chức năng được phân rã có thể lập trình được. Trong thời kỳ này, người ta chưa quan tâm đến các thành phần không được tin học hoá mà chỉ xoay quanh đến các vấn đề trong hệ thống để lập trình, tập trung vào chức năng và ít tập trung vào dữ liệu (vì thời kỳ nay đang chuẩn hoá và phát triển về cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu)

Thời kỳ vào những thập niên 80, tiếp cận phân tích thiết kế theo phương pháp gọi là *hệ thống*. Quan điểm chính của phương pháp này là tiếp cận hệ thống theo 2 thành phần, thành phần xử lý (thành phần động) và thành phần dữ liệu (thành phần tĩnh) của hệ thống. Cách tiếp cận của các phương pháp trong giai đoạn này tuân theo hai tính chất : tính toàn thể : tiếp cận hệ thống qua việc mô tả các hệ thống con và sự tương tác giữa chúng ; tính đúng đắn : tìm kiếm sự phân rã, kết hợp các hệ thống con sao cho hành vi của nó tiêu biểu nhất của hệ thống trong môi trường tác động lên hệ thống con đó. Cách tiếp cận hệ thống theo hai thành phần chính là tiền đề cho cách tiếp cận hướng đối tượng trong các giai đoạn sau. Tuy nhiên, việc tiếp cận chủ yếu là hướng xoay quanh dữ liệu để thu thập và tổ chức dữ liệu nhằm khai thác mặt đáp ứng nhu cầu thông tin. Hướng tiếp cận gây khó khăn trong những hệ thống lớn và thường xuyên thay đổi cũng như là trong việc thiết kế nhằm tái sử dụng một thành phần đã có.

Giới thiệu về phương pháp phát triển hướng đối tượng

Vào thập niên 90, phương pháp tiếp cận phân tích thiết kế đối tượng là sự tổng hợp của phương pháp Descartes và phương pháp hệ thống. Trong khi các mô hình được đưa ra trong những thập niên trước thường đưa ra dữ liệu và xử lý theo hai hướng độc lập nhau. Khái niệm đối tượng là sự tổng hợp giữa khái niệm xử lý và khái niệm dữ liệu chung trong một cách tiếp cận, và một hệ thống là một tập hợp các đối tượng liên kết nội. Có nghĩa rằng việc xây dựng hệ thống chính là việc xác định các đối tượng đó bằng cách cố gắng ánh xạ các đối tượng của thế giới thực thành đối tượng hệ thống, thiết kế và xây dựng nó, và hệ thống hình thành chính là qua sự kết hợp của các đối tượng này. Phương pháp hướng đối tượng được xem là phương pháp phân tích thiết kế thế hệ thứ ba, các phương pháp tiêu biểu là OOD, HOOD, BON, OSA, … và sau này là OOSA, OOA, OMT, CRC, OOM, OOAD, OOSE, RUP/UML

Đặc trưng cơ bản

- *Tính bao bọc (encapsulation):* quan niệm mối quan hệ giữa đối tượng nhận và đối tượng cung cấp thông qua khái niệm hộp đen. Nghĩa là đối tượng nhận chỉ truy xuất đối tượng cung cấp qua giao diện được định nghĩa bởi đối tượng cung cấp, đối tượng nhận không được truy cập đến các đặc trưng được xem là "nội bộ" của đối tượng cung cấp.

- *Tính phân loại (classification):* gom nhóm các đối tượng có cùng cấu trúc và hành vi vào một lớp (class).
- *Tính kết hợp (aggregation):* kết hợp các đối tượng và các đối tượng cấu thành nó để mô tả cấu trúc cục bộ của đối tượng (ví dụ: toà nhà <-> phòng, xe <-> sườn xe, bánh xe,…) , hoặc sự liên kết phụ thuộc lẫn nhau giữa các đối tượng.
- *Tính thừa kế (heritage):* phân loại tổng quát hoá và chuyên biệt hoá các đối tượng, và cho phép chia sẽ các đặc trưng của một đối tượng.

Phân loại

Phương pháp lập trình hướng đối tượng được chia thành 2 hướng như sau:

- Hướng lập trình: từ lập trình đơn thể chuyển sang lập trình hướng đối tượng với lý thuyết cơ bản dựa trên việc trừu tượng hóa kiểu dữ liệu.
- Hướng hệ quản trị CSDL: phát triển thành CSDL hướng đối tượng

Có 2 cách tiếp cận riêng biệt:

- Phương pháp kỹ thuật: hướng công nghệ phần mềm như OOD, HOOD, BON, BOOCH, MECANO, OODA,…
- Phương pháp toàn cục: hướng về HTTT như OOA, OOSA, OOAD, OMT, OOM,…

Ưu điểm

- Cấu trúc hoá được các cấu trúc phức tạp và sử dụng được cấu trúc đệ qui: các phương pháp đối tượng đều sử dụng các mô hình bao gồm nhiều khái niệm để biểu diễn nhiều ngữ nghĩa khác nhau của hệ thống. Ví dụ: trong mô hình lớp của OMT có khái niệm mối kết hợp thành phần cho phép mô tả một đối tượng là một thành phần của đối tượng khác, trong khi nếu dùng mô hình ER truyền thống không có khái niệm này do đó không thể biểu diễn được quan hệ thành phần.
- Xác định được đối tượng của hệ thống qua định danh đối tượng[1]
- Tính thừa kế được đưa ra tạo tiền đề cho việc tái sử dụng

Mô hình

Mô hình (model) là một dạng thức trừu tượng về một hệ thống, được hình thành để hiểu hệ thống trước khi xây dựng hoặc thay đổi hệ thống đó. Theo Efraim Turban, mô hình là một dạng trình bày đơn giản hoá của thế giới thực. Bởi vì, hệ thống thực tế thì rất phức tạp và rộng lớn và khi tiếp cận hệ thống, có những chi tiết những mức độ phức tạp không cần thiết phải được mô tả và giải quyết. Mô hình cung cấp một phương tiện (các khái niệm) để quan niệm hoá vấn đề và giúp chúng ta có thể trao đổi các ý tưởng trong một hình thức cụ thể trực quan, không mơ hồ.

Các đặc điểm của một mô hình:

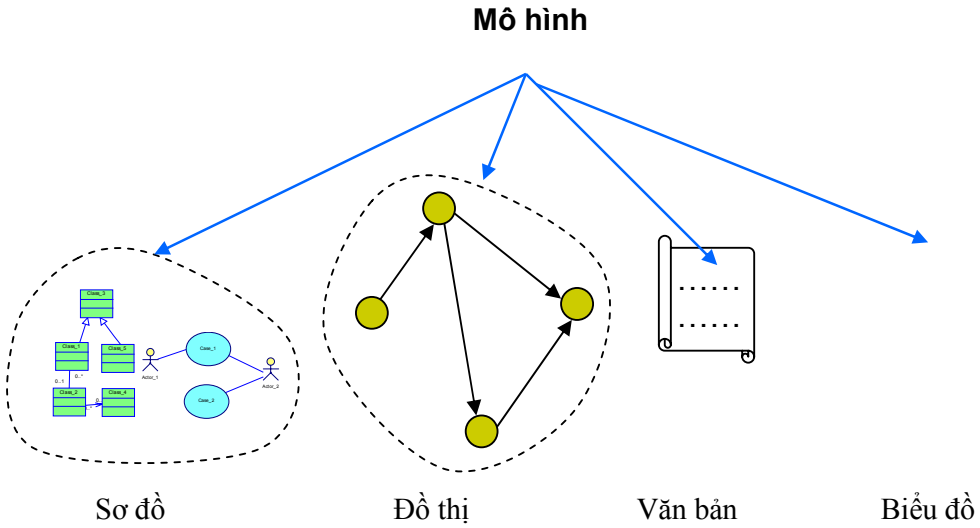
- Diễn đạt một mức độ trừu tượng hóa (ví dụ: mức quan niệm, mức tổ chức, mức vật lý,…)
- Tuân theo một quan điểm (quan điểm của người mô hình hoá)
- Có một hình thức biểu diễn (văn bản, đồ họa: sơ đồ, biểu đồ, đồ thị,…)

Hầu hết các kỹ thuật mô hình hóa sử dụng trong phân tích thiết kế là các ngôn ngữ đồ họa (đa số là sơ đồ - diagram), các ngôn ngữ này bao gồm một tập hợp các ký hiệu. Các ký hiệu này

[1] OID: Object Iden tifier

được dùng đi kèm theo các qui tắc của phương pháp luận giúp cho việc trao đổi các quan hệ thông tin phức tạp được rõ ràng hơn việc mô tả bằng văn bản.

Ví dụ :



Mô hình tĩnh và mô hình động

Mô hình tĩnh (static model): được xem như là hình ảnh về thông số hệ thống tại một thời điểm xác định. Các mô hình tĩnh được dùng để trình bày cấu trúc hoặc những khía cạnh tĩnh của hệ thống.

Mô hình động (dynamic model): được xem như là một tập hợp các hành vi, thủ tục kết hợp với nhau để mô tả hành vi của hệ thống. Các mô hình động được dùng để biểu diễn sự tương tác của các đối tượng để thực hiện công việc hệ thống.

Mục đích của mô hình hoá

Đứng trước sự gia tăng mức độ phức tạp của một hệ thống, việc trực quan hoá và mô hình hóa ngày càng trở nên chính yếu trong cách tiếp cận về một hệ thống, đặc biệt là cách tiếp cận hướng đối tượng. Việc sử dụng các ký hiệu để trình bày hoặc mô hình hóa bài toán có các mục đích sau:

- Làm sáng tỏ vấn đề: chúng ta có thể đưa ra được các lỗi hoặc các thiếu xót của hệ thống từ việc tiếp cận trực quan đồ họa hơn là các dạng trình bày khác như văn bản, đoạn mã,… Hơn nữa, việc mô hình hoá giúp chúng ta dễ dàng hiểu được hệ thống.
- Mô phỏng được hình ảnh tương tự của hệ thống: hình thức trình bày của mô hình có thể đưa ra được một hình ảnh giả lập như hoạt động thực sự của hệ thống thực tế, điều này giúp cho người tiếp cận cảm thấy thuận tiện khi làm việc với mô hình (là hình ảnh thu nhỏ của hệ thống thực tế)
- Gia tăng khả năng duy trì hệ thống: các ký hiệu trực quan có thể cải tiến khả năng duy trì hệ thống. Thay đổi các vị trí được xác định trực quan và việc xác nhận trực quan trên mô hình các thay đổi đó sẽ giảm đi các lỗi. Do đó, chúng ta có thể tạo ra các thay đổi nhanh hơn và các lỗi được kiểm soát hoặc xảy ra ít hơn.
- Làm đơn giản hóa vấn đề: mô hình hoá có thể biểu diễn hệ thống ở nhiều mức, từ mức tổng quát đến mức chi tiết, mức càng tổng quát thì ký hiệu sử dụng càng ít (do đó càng đơn giản hoá việc hiểu) và hệ thống được biểu diễn càng tổng quát.

Phương pháp luận phát triển hệ thống

Phương pháp luận phát triển hệ thống bao gồm hai thành phần :

- Qui trình (process) : bao gồm các giai đoạn (phase) và tiến trình trong đó định nghĩa thứ tự các giai đoạn và các luật hình thành nên một quá trình phát triển hệ thống từ các công việc khởi tạo đến các công việc kết thúc của một dự án hệ thống.
- Các khái niệm (notation), phương pháp : các mô hình (bao gồm các phương pháp mô hình hoá của mô hình) cho phép mô hình hoá các kết quả của quá trình phát triển hệ thống.

Các giai đoạn cơ bản trong một qui trình :

Để tự động hóa hoạt động xử lý, hệ thống phải trải qua một quá trình gồm nhiều bước được gọi là quá trình phát triển hệ thống. Cũng giống như nhiều tiến trình khác, phát triển hệ thống tự động cũng theo chu trình được gọi là vòng đời (Life cycle). Khái niệm vòng đời là một khái niệm rộng nó bắt đầu từ sự khởi đầu xây dựng cho đến kết thúc việc khai thác hệ thống. Nếu chúng ta chỉ chú trọng đến giai đoạn xây dựng và triển khai thì gọi là phát triển hệ thống. Vòng đời phát triển hệ thống - SDLC (Systems Development Life Cycle) là một phương pháp luận chung để phát triển nhiều loại hình hệ thống khác nhau. Tuy nhiên, các giai đoạn trong quá trình này cũng thay đổi khác nhau khoảng từ 3 cho đến 20 tùy theo qui mô và loại hình hệ thống chúng ta đang tiếp cận.

Phần sau đây sẽ giới thiệu các giai đoạn cơ bản làm nền tảng chung cho hầu hết quá trình phát triển hệ thống:

Giai đoạn khởi tạo

Hoạt động chính của giai đoạn này là khảo sát tổng quan hệ thống, vạch ra các vấn đề tồn tại trong hệ thống và các cơ hội của hệ thống, cũng như trình bày lý do tại sao hệ thống nên hoặc không nên được đầu tư phát triển tự động hóa. Một công việc quan trọng tại thời điểm này là xác định phạm vi của hệ thống đề xuất, trưởng dự án và nhóm phân tích viên ban đầu cũng lập một kế hoạch các hoạt động của nhóm trong các giai đoạn tiếp theo của dự án phát triển hệ thống. Kế hoạch này xác định thời gian và nguồn lực cần thiết. Đánh giá khả thi của dự án và nhất là phải xác định được chi phí cần phải đầu tư và lợi ít mang lại từ hệ thống. Kết quả của giai đoạn này là xác định được dự án hoặc được chấp nhận để phát triển, hoặc bị từ chối, hoặc phải định hướng lại.

Giai đoạn phân tích

Giai đoạn phân tích bao gồm các bước sau:

- Thu thập yêu cầu hệ thống: các phân tích viên làm việc với người sử dụng đề xác định tất cả những gì mà người dùng mong muốn từ hệ thống đề xuất.
- Nguyên cứu các yêu cầu và cấu trúc hoá (mô hình hoá) để dễ dàng nhận biết và loại bỏ những yếu tố dư thừa.
- Phát sinh các phương án thiết kế chọn lựa phù hợp với yêu cầu và so sánh các phương án này để xác định giải pháp nào là đáp ứng tốt nhất các yêu cầu trong một mức độ cho phép về chi phí, nhân lực, và kỹ thuật của tổ chức. Kết quả của giai đoạn này là bản mô tả về phương án được chọn.

Trong phân tích hướng đối tượng giai đoạn này quan tâm đến mức độ trừu tượng hoá đầu tiên bằng cách xác định các lớp và các đối tượng đóng vai trò quan trọng nhằm diễn đạt các yêu cầu cũng như mục tiêu hệ thống. Để hiểu rõ các yêu cầu hệ thống chúng ta cần xác định ai là người dùng và là tác nhân hệ thống. Trong phương pháp phát triển hướng đối tượng cũng như phương pháp truyền thống, các mô tả kịch bản hoạt động được sử dụng để trợ giúp các phân tích viên hiểu được yêu cầu. Tuy nhiên, các kích bản này có thể được mô tả không đầy đủ hoặc không theo một hình thức. Do đó, khái niệm use case được dùng trong giai đoạn này

nhằm biểu diễn chức năng hệ thống và sự tương tác người dùng hệ thống. Các kích bản hoạt động lúc này sử dụng các mô hình động (dynamic diagram) nhằm mô tả nội dung của use case để làm rõ sự tương tác giữa các đối tượng, vai trò cũng như sự cộng tác của các đối tượng trong hoạt động của use case hệ thống. Trong giai đoạn phân tích, chỉ có các lớp tồn tại trong phạm vi hệ thống (ở thế giới thực) mới được mô hình hoá và như vậy thì kết quả mô hình hoá trong giai đoạn này sẽ phản ánh phạm vi của hệ thống. Các lớp về kỹ thuật, giao diện định nghĩa phần mềm cũng không quan tâm ở giai đoạn này.

Giai đoạn thiết kế

Trong giai đoạn này kết quả của giai đoạn phân tích sẽ được chi tiết hoá để trở thành một giải pháp kỹ thuật để thực hiện. Các đối tượng và các lớp mới được xác định để bổ sung vào việc cài đặt yêu cầu và tạo ra một hạ tầng cơ sở kỹ thuật về kiến trúc. Ví dụ: các lớp mới này có thể là lớp giao diện (màn hình nhập liệu, màn hình hỏi đáp, màn hình duyệt,…). 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.

Về mức độ thiết kế thì có thể chia kết quả của giai đoạn này thành hai mức:

Thiết kế luận lý

Đặc tả hệ thống ở mức độ trừu tượng hóa dựa trên kết quả của giải pháp được chọn lựa từ giai đoạn phân tích. Các khái niệm và mô hình được dùng trong giai đoạn này độc lập với phần cứng, phần mềm sẽ sử dụng và sự chọn lựa cài đặt. Theo quan điểm lý thuyết, ở bước này hệ thống có thể cài đặt trên bất kỳ trên nền tảng phần cứng và hệ điều hành nào, điều này cho thấy giai đoạn này chỉ tập trung để biểu diễn khía cạnh hành vi và tính năng của đối tượng hệ thống.

Thiết kế vật lý

Chuyển đổi kết quả thiết kế luận lý sang các đặc tả trên phần cứng, phần mềm và kỹ thuật đã chọn để cài đặt hệ thống. Cụ thể là đặc tả trên hệ máy tính , hệ quản trị cơ sở dữ liệu, ngôn ngữ lập trình đã chọn,…. Kết quả của bước này là các đặc tả hệ thống vật lý sẵn sàng chuyển cho các lập trình viên hoặc những người xây dựng hệ thống khác để lập trình xây dựng hệ thống.

Giai đoạn xây dựng

Trong giai đoạn xây dựng (giai đoạn lập trình), các lớp của giai đoạn thiết kế sẽ được biến thành những dòng mã lệnh (code) cụ thể trong một ngôn ngữ lập trình hướng đối tượng (không nên dùng một ngôn ngữ lập trình hướng chức năng!). Phụ thuộc vào khả năng của ngôn ngữ được sử dụng, đây có thể là một công việc khó khăn hoặc 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 mã lệnh. 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 viết mã lệnh 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 chuyển thành các mã lệnh.

Giai đoạn thử nghiệm

Một hệ thống phần mềm thường được thử nghiệm qua nhiều giai đoạn và với nhiều nhóm thử nghiệm khác nhau. Các nhóm sử dụng nhiều loại biểu đồ UML khác nhau làm nền tảng cho công việc của mình: Thử nghiệm đơn vị sử dụng biểu đồ lớp (class diagram) và đặc tả lớp, thử nghiệm tích hợp thường sử dụng biểu đồ thành phần (component diagram) và biểu đồ cộng tác (collaboration diagram), và giai đoạn thử nghiệm hệ thống sử dụng biểu đồ Use case

(use case diagram) để đảm bảo hệ thống có phương thức hoạt động đúng như đã được định nghĩa từ ban đầu trong các biểu đồ này.

Giai đoạn cài đặt và bảo trì

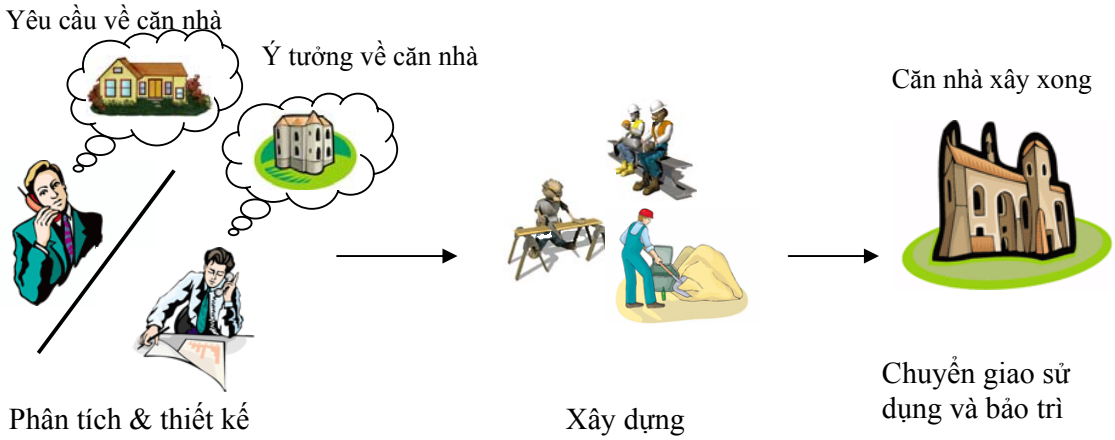
Điều chỉnh hệ thống phù hợp với nhu cầu sử dụng, các thay đổi phát sinh bao gồm:

- Chức năng sử dụng chưa phù hợp tốt nhất với người sử dụng hoặc khó sử dụng
- Các điều kiện và yêu cầu của người dùng hệ thống thay đổi, đòi hỏi phải chỉnh sửa sao cho hệ thống vẫn hữu dụng
- Các lỗi hệ thống phát sinh do quá trình kiểm tra còn xót lại
- Nâng cấp phiên bản mới của hệ thống

Bảo trì hệ thống không nên xem như là một giai đoạn tách rời mà nên xem như là một sự lặp lại chu trình của những giai đoạn trước đòi hỏi phải được nghiên cứu đánh giá và cài đặt. Tuy nhiên, nếu một hệ thống không còn hoạt động như mong muốn do có sự thay đổi quá lớn về hoạt động, hoặc nhu cầu mới đặt ra vượt quá sự giải quyết của hệ thống hiện tại, hoặc chi phí để bảo trì là quá lớn. Lúc này yêu cầu về hệ thống mới được xác lập để thay thế hệ thống hiện tại và một qui trình lại bắt đầu.

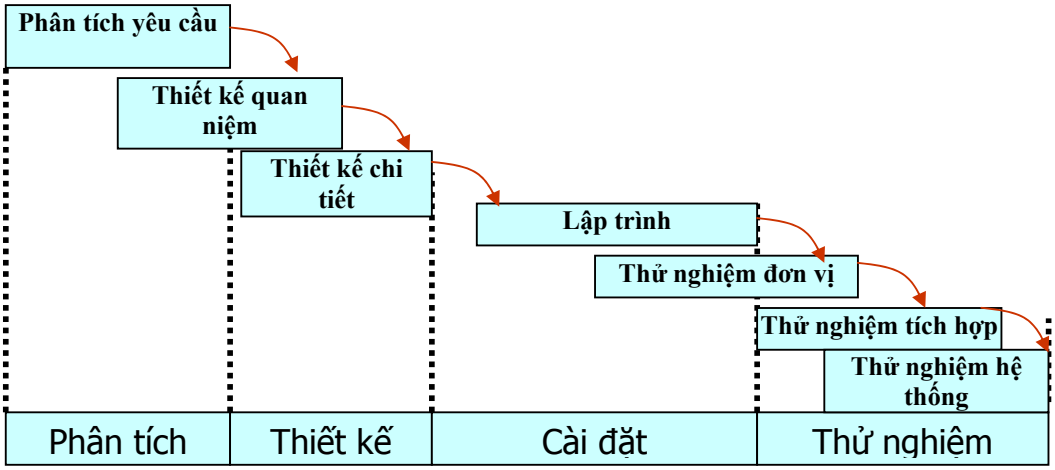
Ví dụ về qui trình phát triển

Chúng ta có thể hình dung rằng chúng ta muốn xây dựng một căn nhà. Công việc đầu tiên là chúng ta chắc chắn là chúng ta dự tính xem chúng ta sẽ bỏ số tiền bao nhiêu để xây dựng căn nhà này, dựa trên số tiến này chúng ta tìm kiếm và phác hoạ (có thể chỉ trong ý tưởng) căn nhà này phải như thế nào? Loại căn nhà theo kiểu gì, có mấy phòng, chiều rộng và chiều dài bao nhiêu, rồi nào đến nền nhà, màu sắc, tiện nghi ?,… Rồi sau đó, chúng ta sẽ chọn một đơn vị xây dựng (trong số nhiều đơn vị mà thoả yêu cầu nhất). Tất cả các yêu trên sẽ phải trao đổi với đơn vị xây dựng này nhằm thống nhất về giá cả cũng như các điều khoản về yêu cầu xây dựng. Giai đoạn này được xem như là giai đoạn phân tích. Tiếp đó, đơn vị xây dựng sẽ thực hiện công việc thiết kế chi tiết của căn nhà, và từng đơn vị trong căn nhà (phòng, tường, trần, mái, phòng khách, phòng ăn, phòng ngủ,…). Giai đoạn này được xem là giai đoạn thiết kế. Sau đó các bản thiết kế chi tiết của căn nhà sẽ được bộ phận thi công dựa vào đó để tiến hành việc xây dựng. Giai đoạn này được xem là giai đoạn xây dựng. Căn nhà sau khi hoàn tất sẽ được chuyển giao để sử dụng, tất nhiên trong quá trình sử dụng nếu có các hư hỏng thì đơn vị xây dựng sẽ phải tiến hành bảo trì và sửa chữa.



Mộ số qui trình phát triển

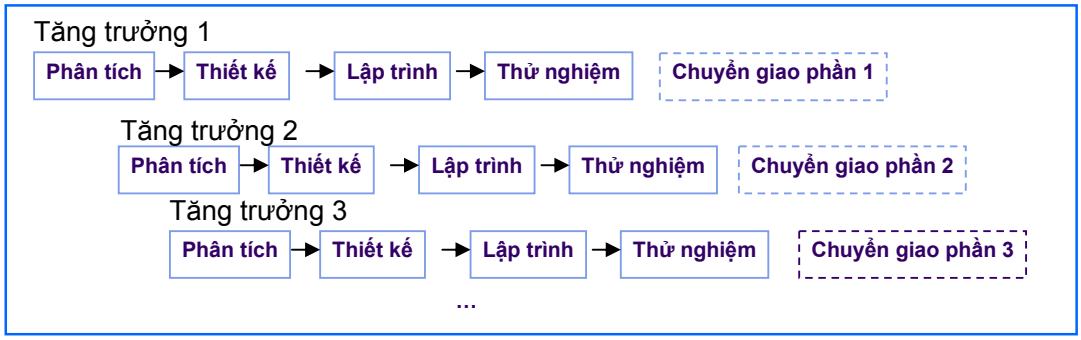
Qui trình thác nước (waterfall – Boehm 1970)



Đây là một qui trình đầu tiên được đề xuất và đã đưa ra được các giai đoạn căn bản nhất và đầy đủ cho một quá trình phát triển hệ thống, các giai đoạn bao gồm : phân tích, thiết kế, cài đặt và thử nghiệm hệ thống. Từ khi được đề xuất qui trình này nhanh chóng được phổ cập sử dụng rộng rãi trong giới công nghiệp và cho đến bây giờ đã có nhiều cải tiến hoàn thiện.
Nhược điểm :

- Qui trình là các giai đoạn tuần tự nối tiếp nhau, có nghĩa là giai đoạn phân tích phải được hoàn thành rồi đến giai đoạn thiết kế,… không cho phép sự quay lui và do đó, khi áp dụng qui trình này sẽ khó khăn khi ở giai đoạn trước có sự thay đổi (do sai xót, do nhu cầu người dùng thay đổi hoặc do có sự tiến hoá hệ thống,…).

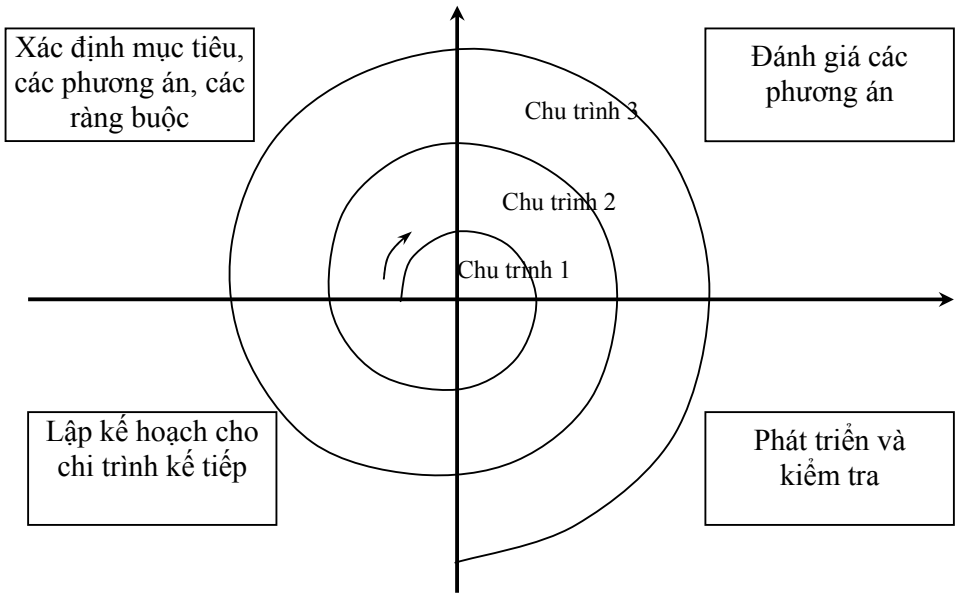
Qui trình tăng trưởng (D.R. Graham 1988)



- Quan điểm chính của qui trình này là phát triển từng phần (phân hệ con) của hệ thống dùng qui trình thác nước.
- Lặp : phân chia hệ thống thành những phần có thể phát triển một cách độc lập. Mỗi thành phần trong quá trình phát triển sẽ được áp dụng qui trình thác và được xem như một tăng trưởng của hệ thống. Khi thành phần cuối cùng hoàn tất thì quá trình phát triển toàn bộ hệ thống kết thúc.

Nhược điểm : qui trình này không thể áp dụng cho những hệ thống có sự phân chia không rõ ràng hoặc không thể phân chia thành những thành phần tác biệt.

Qui trình xoắn ốc (Boehm 88)

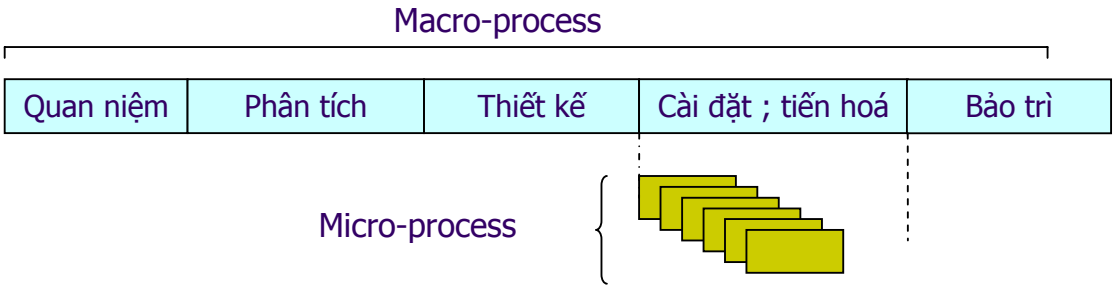


Là một quá trình gồm nhiều vòng lặp dựa trên bốn giai đoạn :

- Giai đoạn 1 :
  - Đối với vòng lặp đầu tiên : phân tích yêu cầu
  - Từ vòng lặp thứ hai trở đi : thiết lập mục tiêu cho vòng lặp, xác định các phương án để đạt mục tiêu đó ; các ràng buộc xuất phát từ các kết quả của các vòng lặp trước.
- Giai đoạn 2 :
  - Đánh giá các phương án dựa trên các sản phẩm đạt được và tiến trình thực thi phương án.
  - Xác định và giải quyết các rũi ro.
- Giai đoạn 3 :
  - Phát triển và kiểm tra sản phẩm.
- Giai đoạn 4 :
  - Lập kế hoạch cho vòng lặp tiếp theo.

Qui trình xoắn ốc cũng có thể áp dụng qui trình khác, ví dụ giai đoạn 3 có thể được thực hiện áp dụng qui trình thác nước.

Qui trình Booch (1996)



Gồm hai tiến trình :

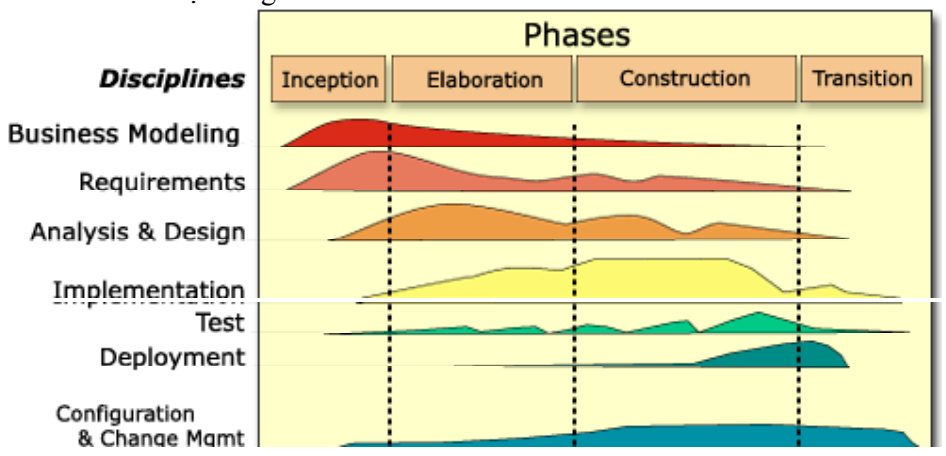
- Macro process : đóng vai trò như là bộ khung của micro process và bao phủ toàn bộ phạm vi dự án. Công việc chính của macro process là liên quan đến quản lý kỹ thuật của hệ thống trong việc chú trọng đến yêu cầu của người dùng và thời gian hoàn thành sản phẩm mà ít quan tâm đến chi tiết thiết kế hệ thống. Macro porcess gồm :

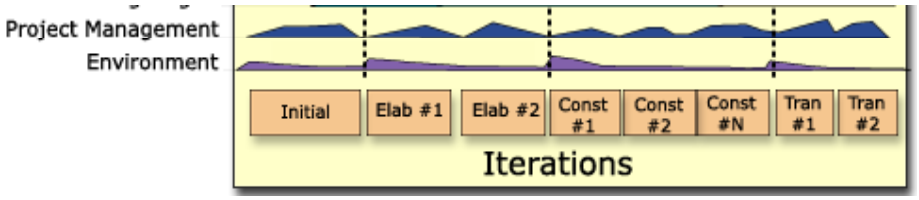
  - Quan niệm hoá (conceptualization) : xác định yêu cầu căn bản, mục tiêu của hệ thống
  - Phân tích và phát triển mô hình : sử dụng sơ đồ để mô hình hoá đối tượng hệ thống ; xác định vai trò và trách nhiệm của các đối tượng ; mô hình hoá hành vi của hệ thống thông qua các kịch bản mô tả hành vi.
  - Thiết kế : thiết kế kiến trúc của hệ thống, các mối quan hệ giữa các lớp, các lớp sẽ được cài đặt, các vị trí định vị xử lý.
  - Cài đặt, tiến hoá : tinh chế hệ thống thông qua nhiều vòng lặp. Lập trình cài đặt phần mềm.
  - Bảo trì : điều chỉnh lỗi phát sinh, cập nhật các yêu cầu mới
- Micro process : mô tả các hoạt động chi tiết của mỗi giai đoạn thông qua việc phân chia thành các hoạt động chi tiết theo từng nhóm phát triển hoặc theo từng đơn vị thời gian (giờ, ngày, tuần,…).

RUP/UML (Rational Unified Process)

Qui trình bao gồm bốn giai đoạn chính và đan xen nhiều dòng hoạt động (activity flow) như là : mô hình hoá nghiệp vụ, phân tích yêu cầu, phân tích và thiết kế, cài đặt, thử nghiệm triển khai, …Mỗi giai đoạn được hình thành từ những bước lặp (iteration).

- Khởi tạo (inception) :
  - Thiết lập phạm vi dự án, các điều kiện ràng buộc phạm vi, các kiến trúc đề xuất của hệ thống,
  - Xác định chi phí và thời gian của dự án,
  - Xác định độ rũi ro và môi trường hệ thống,
  - Xác định các thay đổi bổ sung, các tác động của các thay đổi này, các rũi ro nếu có,…
- Tinh chế (elaboration) :
  - Tinh chế kiến trúc hệ thống, yêu cầu hệ thống và đảm bảo kế hoạch sự ổn định của kế hoạch,
  - Đánh giá độ rũi ro, các thành phần sử dụng,
  - Xây dựng nền kiến trúc nền tảng hệ thống,…
- Xây dựng (construction) :
  - Quản lý tài nguyên, kiểm soát và thực hiện tối ưu hoá,
  - Hoàn thành việc phát triển các thành phần của sản phẩm, thử nghiệm sản phẩm,
  - Đánh giá sản phẩm cài đặt từ các tiêu chuẩn đã được thoả thuận,…
- Chuyển giao (transition) :
  - Thực hiện cài đặt hệ thống,
  - Thử nghiệm sản phẩm đã triển khai,
  - Thu thập các phản hồi từ phía người dùng,
  - Bảo trì hệ thống





Phương pháp (method)

- Phương pháp là một quá trình tập trung vào một hoặc một vài giai đoạn của toàn bộ qui trình phát triển. Ví dụ :
  - Phương pháp phân tích yêu cầu : mô tả cách thức và qui trình nhằm thu thập các yêu cầu của hệ thống,
  - Phương pháp phân tích thiết kế : tập trung vào giai đoạn phân tích và thiết kế
  - Phương pháp thử nghiệm : qui trình và cách thức cũng như các hoạt động thử nghiệm hệ thống
  - …
- Một phương pháp bao gồm một tập các ký hiệu đồ hoạ và văn bản, các luật sử dụng để mô tả các yếu tố hệ thống
- Một phương pháp thường được áp dụng trong một qui trình của phương pháp luận nhằm hướng dẫn cách thức thực hiện chi tiết của giai đoạn trong qui trình phát triển.

Một số phương pháp

Phần sau đây sẽ tổng kết nội dung của một số phương pháp phát triển hệ thống hướng đối tượng:

OOD (Object Oriented Design - G.Booch 1991)

- Không đưa vào giai đoạn phân tích trong các phiên bản đầu tiên. Các bước phân tích hệ thống chuẩn bị cho giai đoạn thiết kế gồm :
  - Xác định vấn đề thế giới thực
  - Phát triển một chiến lược không hình thức hiện thức hoá từng phần đối với các vấn đề đã xác định
  - Hình thức hoá chiến lược này
- Việc hình thức hoá chiến lược bao gồm một thứ tự các công việc sau :
  - Xác định lớp và đối tượng ở mức trừu tượng hoá
  - Xác định ngữ nghĩa cho các lớp và đối tượng
  - Xác định mối quan hệ giữa các lớp và các đối tượng
  - Cài đặt các lớp và đối tượng
- Đưa vào khái niệm gói (package) và dùng như một thành phần tổ chức của mô hình.
- Cài đặt các lớp và đối tượng thông qua việc đào sâu các chi tiết của lớp và đối tượng và cách thức cài đặt chúng trong một ngôn ngữ lập trình; cách thức tái sử dụng các thành phần và xây dựng các mô đun từ các lớp và đối tượng.
- Trong giai đoạn thiết kế, phương pháp này nhánh mạnh sự phân biệt giữa tầng luận lý (trong thuật ngữ lớp và đối tượng) và tầng vật lý (trong thuật ngữ môđun và xử lý) và phân chia mô hình thành các mô hình động và mô hình tĩnh.
  - Sơ đồ lớp (mô hình tĩnh)

- Sơ đồ đối tượng(mô hình tĩnh)
- Sơ đồ trạng thái (mô hình động)
- Sơ đồ thời gian (mô hình động)
- Sơ đồ mô- đun
- Sơ đồ xử lý

HOOD (Hierarchical Object Oriented Design)

- Khía cạnh tĩnh được biểu diễn qua sơ đồ đối tượng; văn bản hình thức cho phép hoàn thiện sơ đồ này thông qua việc chỉ dẫn các ràng buộc đồng bộ.
- Cấu trúc phân cấp được mô tả thông qua cấu trúc phân rã đối tượng
- Các giai đoạn cơ bản của giai đoạn thiết kế như sau :
  - Xác định vấn đề : xác định ngữ cảnh của đối tượng với mục đích tổ chức và cấu trúc hoá dữ liệu từ các yêu cầu của giai đoạn phân tích.
    - Diễn đạt vấn đề
    - Phân tích và cấu trúc hoá dữ liệu yêu cầu : thu thập và phân tích tất cả thông tin liên quan đến vấn đề, bao gồm môi trường mà hệ thống được thiết kế.
  - Phát triển chiến lược giải pháp: phác hoạ giải pháp vấn đề thông qua việc xác định các đối tượng ở mức trừu tượng hoá cao.
  - Hình thức hoá và mô hình hoá chiến lược : xác định các đối tượng và các toán tử. Phát sinh một giải pháp thiết kế dùng sơ đồ cho phép trực quan hoá các khái niệm, bao gồm năm bước :
    - Xác định đối tượng
    - Xác định các toán tử
    - Nhóm các đối tượng và các toán tử
    - Mô tả đồ hoạ
    - Điều chỉnh các quyết định thiết kế
  - Hình thức hoá giải pháp : giải pháp được hình thức hoá thông qua
    - Mô tả hình thức giao diện đối tượng
    - Mô tả hình thức đối tượng và các cấu trúc điều khiển toán tử

OMT (Object modeling Technique)

Cung cấp ba tập khái niệm diễn đạt ba cách nhìn về hệ thống. Sử dụng một phương pháp để dẫn dắt tới ba mô hình tương ứng với ba cách nhìn hệ thống. Các mô hình đó là :

- Mô hình đối tượng mô tả cấu trúc tĩnh của các đối tượng bên trong hệ thống và các quan hệ của chúng. Các khái niệm chính là :
  - Lớp
  - Thuộc tính
  - Toán tử
  - Thừa kế
  - Mối kết hợp (association)
  - Mối kết hợp thành phần (aggregation)
- Mô hình động hệ thống mô tả các khía cạnh của hệ thống có thể thay đổi theo thời gian. Mô hình này được sử dụng để xác định và cài đặt các khía cạnh điều khiển của một hệ thống. Các khái niệm đó là :

  - Trạng thái
  - Trạng thái con/ cha
  - Sự kiện
  - Hành động
  - Hoạt động
- Mô hình chức năng mô tả việc chuyển đổi giá trị dữ liệu bên trong hệ thống. Các khái niệm đó là :
  - Xử lý
  - Kho dữ liệu
  - Dòng dữ liệu
  - Dòng điều khiển
  - Tác nhân (nguồn, đích)
- Phương pháp được phân chia thành bốn giai đoạn :
  - Phân tích : xây dựng một mô hình thế giới thực dựa vào việc mô tả vấn đề và yêu cầu hệ thống. Kết quả của giai đoạn này là :
    - Bản mô tả vấn đề
    - Mô hình đối tượng = sơ đồ lớp đối tượng + tự điển dữ liệu
    - Mô hình động = sơ đồ trạng thái + sơ đồ dòng sự kiện toàn cục
    - Mô hình chức năng = Sơ đồ dòng dữ liệu + các ràng buộc
  - Thiết kế hệ thống : phân chia hệ thống thành các hệ thống con dựa trên việc kết hợp kiến thức về lãnh vực vấn đề và kiến trúc đề xuất cho hệ thống. Kết quả của giai đoạn thiết kế là :
    - Sưu liệu thiết kế hệ thống : kiến trúc hệ thống cơ sở và các quyết định chiến lược ở mức cao.
  - Thiết kế đối tượng : xây dựng một mô hình thiết kế dựa trên mô hình phân tích được làm giàu với các chi tiết cài đặt, bao gồm các lớp nền tảng các đối tượng cài đặt máy tính. Kết quả của giai đoạn này :
    - Mô hình đối tượng chi tiết
    - Mô hình động chi tiết
    - Mô hình chức năng chi tiết
  - Cài đặt : chuyển đổi các kết quả thiết kế vào một ngôn ngữ và phần cứng cụ thể. Đặc biệt nhấn mạnh trên các đặc điểm có thể truy vết, khả năng uyển chuyển và dễ mở rộng.

OOA (Object Oriented Analysis– Coad 90, 91)

OOA sử dụng các nguyên lý cấu trúc hoá và kết hợp chúng với quan điểm hướng đối tượng tập trung vào giai đoạn phân tích. Phương pháp bao gồm năm bước :

- Tìm lớp và đối tượng : xác định cách thức tìm lớp và đối tượng. Tiếp cận đầu tiên bắt đầu với lãnh vực ứng dụng và xác định các lớp, các đối tượng hình thành nền tảng cho ứng dụng.
- Xác định cấu trúc : được thực hiện qua hai cách :
  - Xác định cấu trúc tổng quát hoá – chuyên biệt hoá và xác định sự phân cấp giữa các lớp đã tìm được

  - Cấu trúc tổng thể - thành phần (whole – part) được dùng để mô hình hoá cách thức một đối tượng là một phần của đối tượng khác, và cách thức các đối tượng kết hợp thành các loại lớn hơn.

- Xác định các chủ đề : phân chia các mô hình lớp, đối tượng thành các đơn vị lớn hơn gọi là chủ đề.
- Xác định thuộc tính : xác định các thông tin và các mối liên kết cho mỗi thể hiện. Điều này bao gồm luôn việc xác định các thuộc tính cần thiết để đặc trưng hoá mỗi đối tượng. Các thuộc tính được tìm thấy sẽ được đưa vào đúng mức trong cấu trúc phân cấp.
- Xác định các dịch vụ : định nghĩa các toán tử cho lớp bằng cách xác định các trạng thái và các dịch vụ nhằm truy cập và thay đổi trạng thái đó.

Kết quả của giai đoạn phân tích là một mô hình gồm năm lớp:

- Lớp chủ đề
- Lớp các lớp và đối tượng
- Lớp cấu trúc (sự thừa kế, mối quan hệ,…)
- Lớp thuộc tính
- Lớp dịch vụ

Một mô hình thiết kế hướng đối tượng bao gồm các thành phần sau:

- Thành phần lãnh vực vấn đề (Problem Domain Component) : kết quả của phân tích hướng đối tượng đưa trực tiếp vào thành phần này.
- Thành phần tương tác (Humain Interaction Component) : bao gồm các hoạt động như là : phân loại người dùng, mô tả kích bản nhiệm vụ, thiết kế cấu trúc lệnh, thiết kế tương tác chi tiết, lập bản mẫu giao diện tương tác người – máy, định nghĩa các lớp của thành phần tương tác.
- Thành phần quản lý nhiệm vụ (Task Management Component) : bao gồm việc xác định các nhiệm vụ (xử lý), các dịch vụ được cung cấp, mức độ ưu tiên, các sự kiện kích hoạt, và cách thức các xử lý trao đổi (với các xử lý khác và với bên ngoài hệ thống).
- Thành phần quản lý dữ liệu (Data Management Component) : phụ thuộc rất nhiều vào công nghệ lưu trữ sẵn có và dữ liệu yêu cầu.

Chương 2

CÁC KHÁI NIỆM CƠ BẢN VỀ HƯỚNG ĐỐI TƯỢNG

Đối tượng (object)

Đối tượng là thành phần trọng tâm của cách tiếp cận hướng đối tượng. Một đối tượng là một đại diện của bất kỳ sự vật nào cần được mô hình trong hệ thống và đóng một vai trò xác định trong lãnh vực ứng dụng.

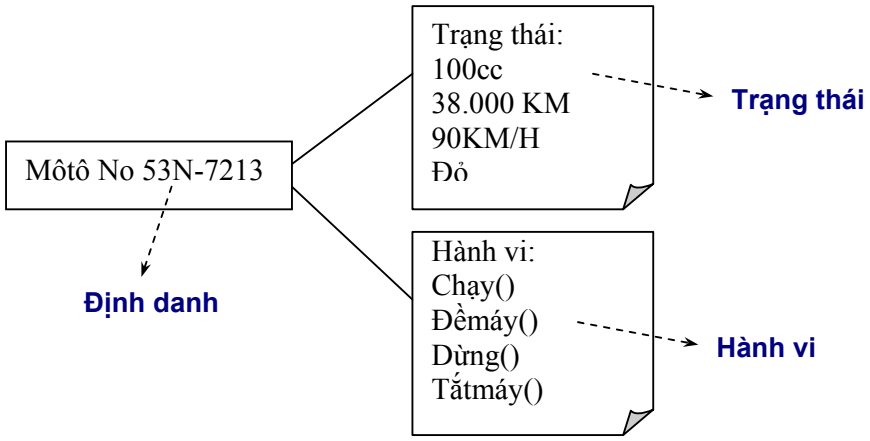
- Là một biểu diễn từ thế giới thực sang thể hiện của tin học (ví dụ : một chiếc xe ô tô trong thế giới thực được biểu diễn trong tin học dùng một khái niệm đối tượng xe ô tô).
- Là một sự trừu tượng hoá, một khái niệm có ý nghĩa trong lãnh vực ứng dụng.
- Diễn đạt một thực thể vật lý, hoặc một thực thể quan niệm, hoặc một thực thể phần mềm.
- Đối tượng có thể là một thực thể hữu hình trực quan (ví dụ : một con người, một vị trí, một sự vật,…) hoặc một khái niệm, một sự kiện (ví dụ : phòng ban, bộ phận, kết hôn, đăng ký, …).

Một thực thể phải thoả ba nguyên lý :

- Phân biệt (distinction): đơn vị duy nhất (định danh)
- Thường xuyên (permanence) : quá trình sống (trạng thái)
- Hoạt động (activity) : vai trò, hành vi

Đối tượng = định danh + trạng thái + hành vi

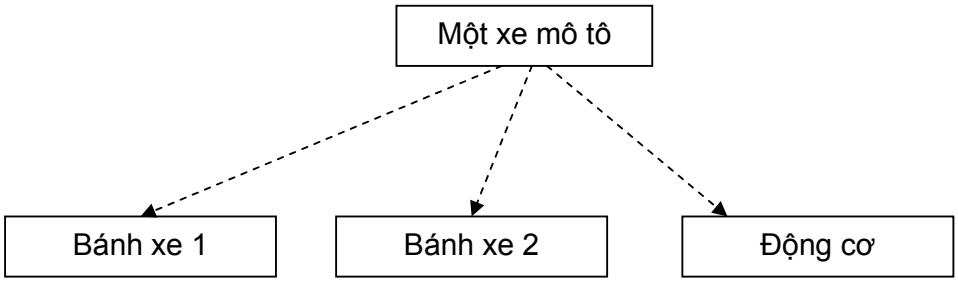
Ví dụ : một đối tượng xe mô tô



Liên kết giữa các đối tượng

- Mối kết hợp (association) – liên kết ngữ nghĩa :

- Phân cấp (hierarchy) – liên kết cấu trúc :

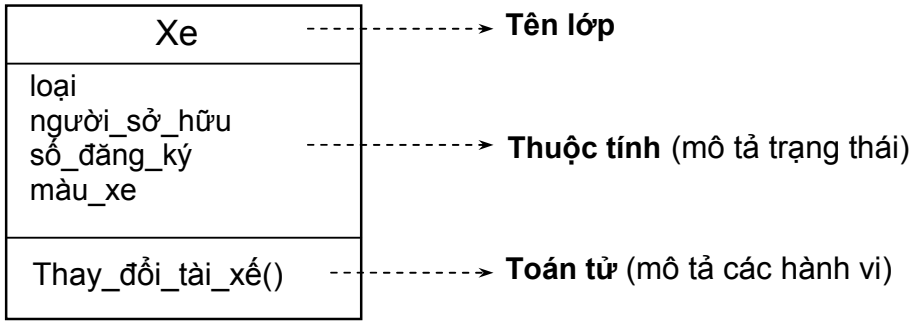


Đối tượng persistent/ transient

- Đối tượng transient : là đối tượng có quá trình sống tối đa tương ứng với quá trình chạy ứng dụng (đối tượng không được lưu trữ trạng thái)
- Đối tượng persistent : đối tượng có trạng thái được lưu trữ trong máy tính và có thể được thực thi bởi một ứng dụng khác ứng dụng tạo ra nó (quá trình sống của nó kéo dài và có thể từ ứng dụng này qua ứng dụng khác do trạng thái của nó được lưu trữ trong máy tính). Thông thường, trạng thái của đối tượng này sẽ được lưu trữ vào cơ sở dữ liệu trong quá trình sử dụng, và việc lưu trữ này sẽ duy trì được tình trạng của đối tượng và cung cấp tình trạng này cho những lần thực thi khác của ứng dụng hoặc cung cấp trạng thái của đối tượng cho những ứng dụng khác.

Lớp (class)

- Một lớp là một mô tả của một tập hợp/ một loại các dối tượng có :
  - Cùng cấu trúc (định danh, đặc trưng)
  - Cùng hành vi (trạng thái, vai trò)
- Trình bày của lớp : là một hình chữ nhật bao gồm ba phần (không bắt buộc)



- Trong giai đoạn cài đặt, định danh của lớp được cài đặt từ một khoá. Khoá này cho phép phân biệt rõ các đối tượng của lợp một cách duy nhất. Khái niệm khoá có thể cho phép truy cập bởi người dùng một cách tường minh hoặc ngầm định. Một khoá tường minh có thể được khai báo chung với trạng thái của lớp trong khi đó khái niệm định danh là một khái niệm độc lập. và có các ý nghĩa sau :
  - Xác định tính duy nhất của đối tượng
  - Có ý nghĩa sử dụng đối với người dùng

Ví dụ : trong lớp Xe có thể khai báo số_đăng_ký là một khoá.

Thể hiện của lớp (instance)

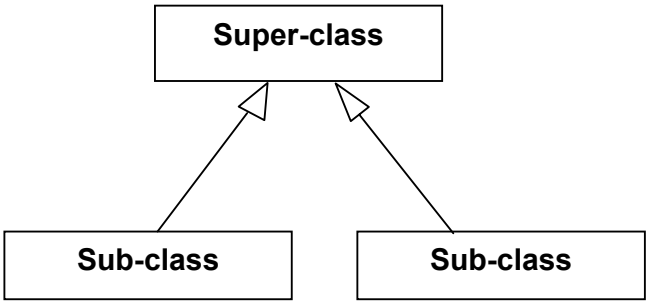
Thể hiện của lớp là một đối tượng cụ thể được tạo ra trên mô hình lớp :

- Các toán tử của lớp mô tả các hành vi chung của các thể hiện
- Tất cả các thể hiện của một lớp có chung các thuộc tính

Phân cấp (hierarchy)

Là cơ chế hỗ trợ việc tổng quát hoá theo cách thức sau :

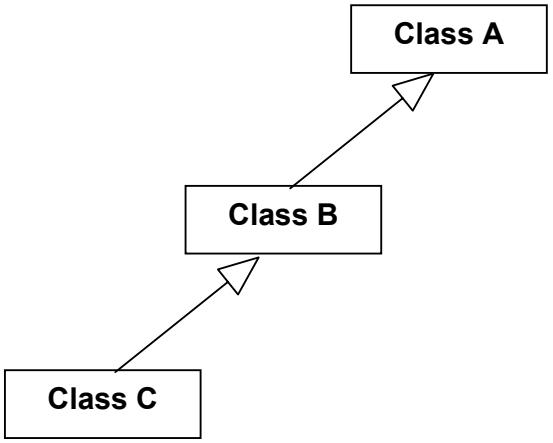
- Tổng quát hoá các đặc tính chung (định nghĩa các supper-class)
- Định nghĩa các đặc tính chuyên biệt nhất của các trường hợp cụ thể (định nghĩa các sub-class)



- Tổng quát hoá (generalisation) : xây dựng một lớp tổng quát từ các lớp khác cụ thể để đạt được một mức độ trừu tượng hoá có thể.
- Chuyên biệt hoá (specialisation) :
    - Sự phân cấp của các lớp có phép mô tả các lớp chuyên biệt có thể từ các lớp trừu tượng
    - Sự chuyên biệt hoá cũng có thể được tạo ra để :
        - Làm giàu thông tin : thêm mới thuộc tính hoặc toán tử vào lớp chuyên biệt so với các lớp trừu tượng
        - Có thể thay thế hoặc định nghĩa lại các thuộc tính, toán tử trong các lớp chuyên biệt từ thuộc tính, toán tử của các lớp trừu tượng

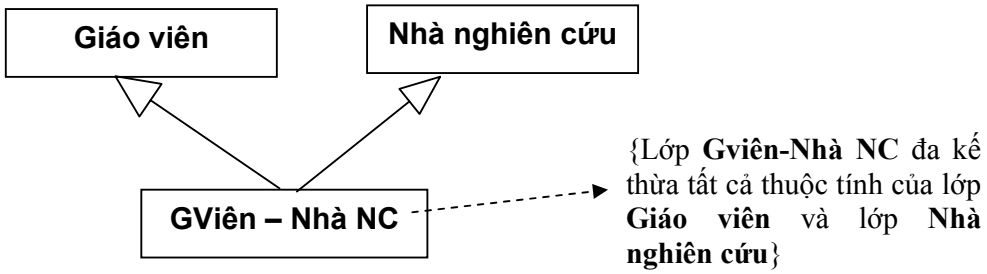
Trong quá trình phân tích hoặc thiết kế hệ thống hướng đối tượng, việc chuyên biệt hoá và tổng quát hoá cho phép định nghĩa các mối quan hệ tập con và làm sáng tỏ tính thừa kế. (Jacobson 1992). Nếu một lớp B thừa kế từ một lớp A, thì có nghĩa rằng tất cả các toán tử và các thuộc tính của lớp A trở thành toán tử và thuộc tính của lớp B.

- Quan hệ kế thừa là quan hệ:
    - Có tính bắc cầu



{Nếu lớp B là một chuyên biệt hoá của lớp A và lớp C là một chuyên biệt hoá của lớp B thì lớp C cũng là một chuyên biệt hoá của lớp A}

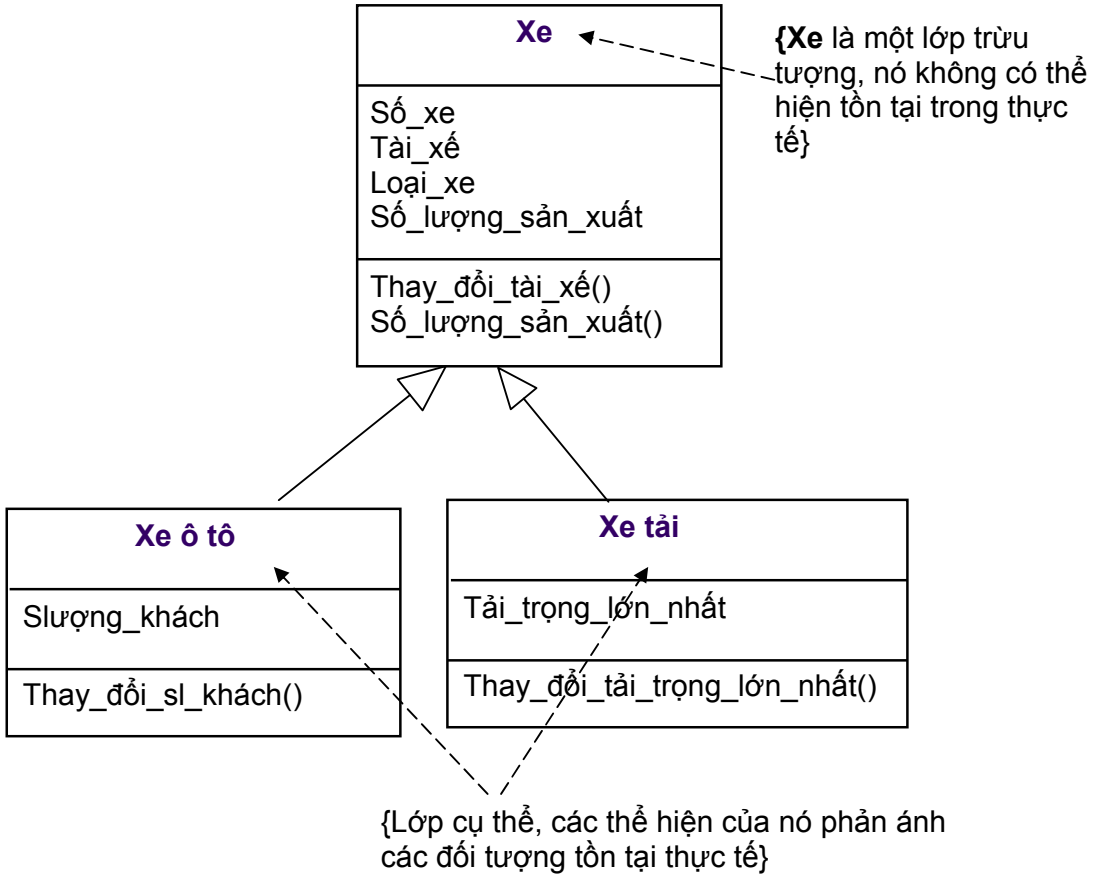
    - Có thể đa kế thừa



- Và được cài đặt với mục tiêu tái sử dụng

- Khái niệm lớp trừu tượng (abstract) – lớp cụ thể (concrete) :

Khi xây dựng một cấu trúc lớp phân cấp, chúng đã hình thành các lớp tổng quát và được gọi là lớp trừu tượng. Trong đó, tất cả các thể hiện đối tượng của một lớp trừu tượng đều xuất phát từ một trong những lớp cụ thể của nó. Một lớp trừu tượng không chứa đựng trực tiếp các đối tượng, các thể hiện của nó chỉ là sự xác định trừu tượng hơn của các thể hiện đối tượng trong các lớp cụ thể. Ngược lại, một lớp cụ thể thực sự chứa đựng các đối tượng và thể hiện. Trong ví dụ dưới đây, các lớp Xe tải và Xe ô tô là các lớp cụ thể bởi vì nó sẽ có các thể hiện xe ô tô và xe tải.

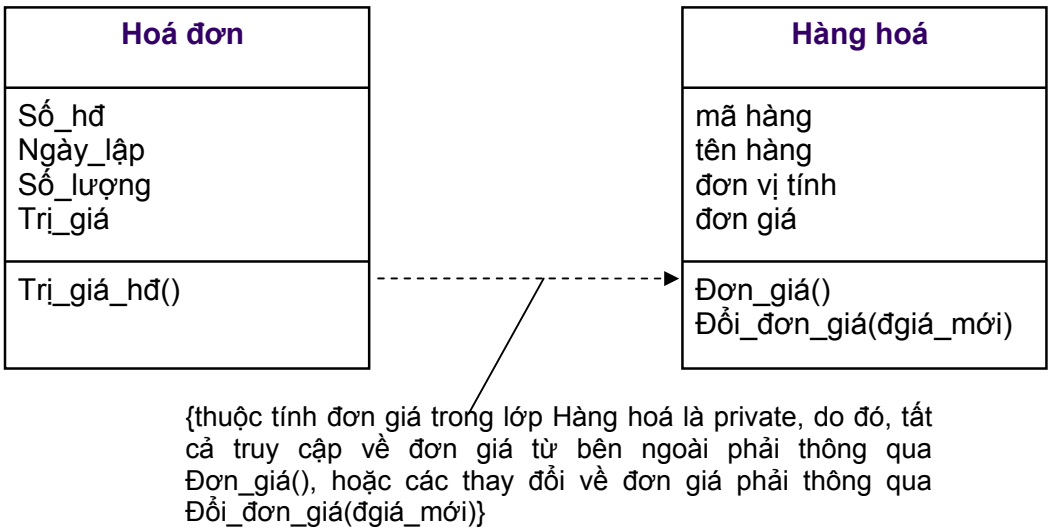


Tính bao bọc(encapsulation)

Che dấu thông tin là nguyên lý để che dấu những dữ liệu và thủ tục bên trong của một đối tượng và cung cấp một giao diện tới mỗi đối tượng như là một cách để tiết lộ ít nhất có thể được về nội dung bên trong của đối tượng.
Các cơ thể bao bọc đối tượng tổng quan bao gồm : public, private, và protected

- public : thuộc tính và hành vi của đối tượng có thể được truy cập từ mọi nơi
- private : thuộc tính và hành vi của đối tượng chỉ được bên trong lớp
- protected : thuộc tính và hành vi của đối tượng chỉ được truy cập từ các lớp con

Tính bao bọc là một mục tiêu trong thiết kế hướng đối tượng. Thay vì cho phép một đối tượng truy cập trực tiếp đến dữ liệu của một đối tượng khác, thì đối tượng này sẽ yêu cầu dữ liệu đó thông qua việc gọi thi hành một hành vi đã được thiết kế cho việc cung cấp dữ liệu và một thông điệp sẽ được gởi tới đối tượng đích thông tin được yêu cầu. Điều này không chỉ đảm bảo rằng các lệnh đang hoạt động trong dữ liệu đúng mà còn không cho phép các đối tượng có thể thao tác trực tiếp lên dữ liệu của đối tượng khác.



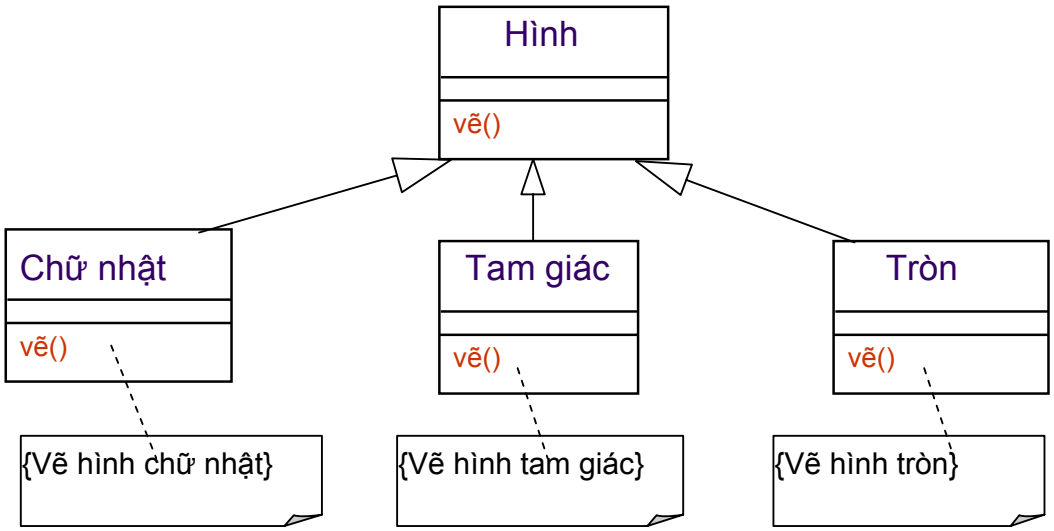
Một yếu tố quan trọng của tính bao bọc là việc thiết kế khác nhau của các đối tượng có thể sử dụng một phương thức (protocol) chung hoặc giao diện chung cho người dùng đối tượng. Điều này lý giải rằng nhiều đối tượng sẽ trả lời tới cùng thông điệp nhưng mỗi đối tượng sẽ thi hành thông điệp sử dụng các toán tử đã được biến đổi thích ứng tới lớp của nó. Bằng cách này, một chương trình có thể gởi một thông điệp tổng quát và để lại việc cài đặt cho đối tượng nhận. Điều này làm giảm sự phụ thuộc lẫn nhau và gia tăng số lượng trao đổi và tái sử dụng của đối tượng.

Ví dụ : các động cơ xe ô tô có thể khác nhau về cách cài đặt và vận hành cụ thể, giao diện giữa tài xế và xe ô tô là thông qua một phương thức chung : ví dụ, đạp cần gas để tăng lực và nhã cần gas để giảm lực của xe. Tất cả tài xế đều biết phương thức này và tất cả tài xế đều sử dụng phương thức này trong tất cả xe ô tô mà không qua tâm đến động cơ của xe ô tô được thực hiện như thế nào (tất nhiên, các động cơ khác nhau thì có cách vận hành khác nhau).

Tính đa hình (polymorphism)

Trong hệ thống hướng đối tượng, thuật ngữ đa hình dùng để mô tả các đối tượng có nhiều dạng thức. Đa hình có nghĩa rằng cùng một toán tử có thể xử lý một cách khác nhau trong các lớp khác nhau có chung một (vài) lớp cha (superclass).

- Cùng toán tử có thể thi hành khác nhau trong các lớp khác nhau.
- Các phương thức khác nhau cùng cài đặt cho toán tử này trong các lớp khác nhau phải có cùng ký hiệu (tên, tham số và giá trị trả về)
- Cài đặt của toán tử được xác định bởi lớp đối tượng mà được sử dụng trực tiếp



Cấu trúc phân cấp trên cho thấy, lớp Hình là lớp tổng quát chung cho các lớp : Chữ nhật, Tam giác, Tròn. Vì cả ba lớp này đều có thể vẽ, do đó, có thể xác định một phương thức vẽ() chung trong lớp Hình. Tuy nhiên, các đối tượng trong các lớp chuyên biệt có thể được thực hiện trong một cách thức có thể khác so với phương thức vẽ() chung. Do đó, mỗi lớp chuyên biệt có thể cài đặt lại phượng thức vẽ() chồng lên phương thức vẽ() của lớp tổng quát.

Nếu hệ thống xử lý một danh sách các đối tượng của lớp Hình, thì hệ thống sẽ dò tìm và thực hiện phương thức vẽ() phù hợp cho mỗi đối tượng. Nếu đối tượng đó là của lớp con Chữ nhật thì nội dung cài đặt của phương thức vẽ() trong lớp Chữ nhật sẽ được thực thi. Tương tự cho đối tượng của lớp con Tam giác và Tròn.

Đa hình ở đây là cho phép có nhiều hình thức cài đặt của cùng một hành vi (phương thức). Hệ thống sẽ tự động thực hiện phương thức thích hợp cho mỗi đối tượng.

Câu hỏi và bài tập

Chương 3

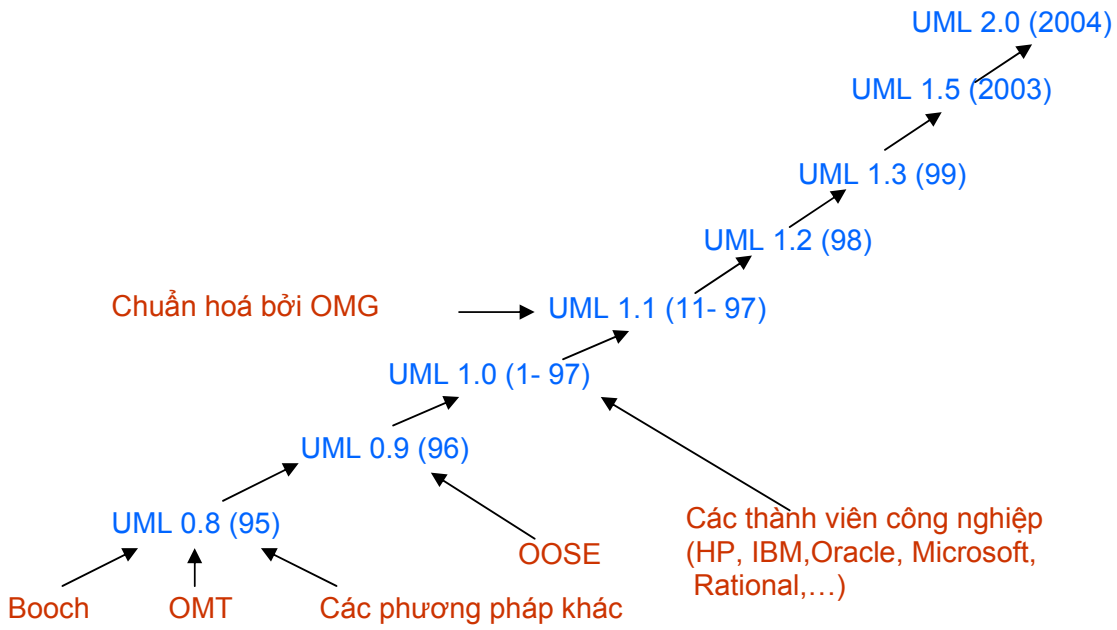
UML (UNIFIED MODELING LANGUAGE)

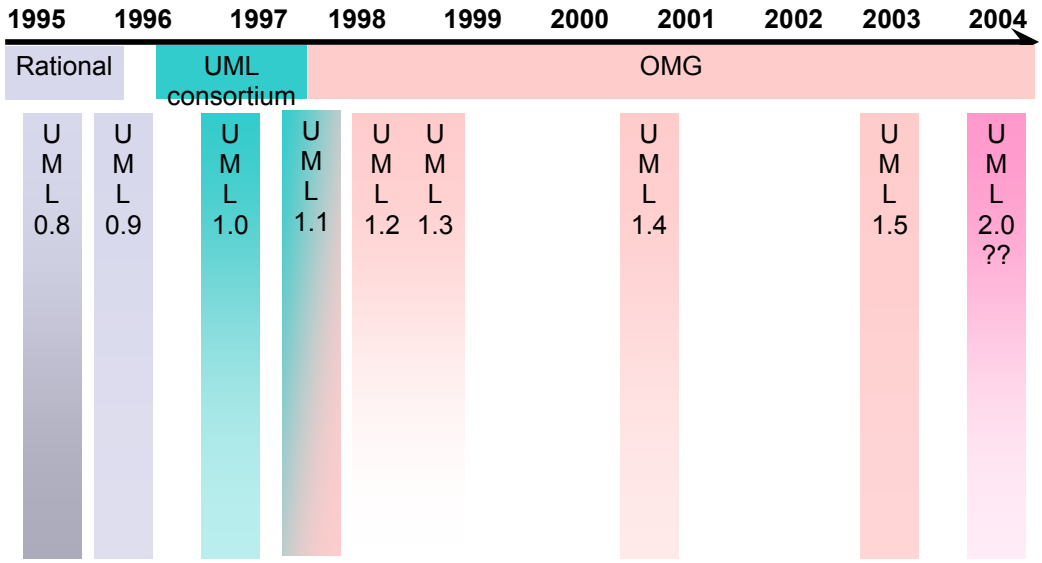
Lịch sử của UML

- Số lượng các phương pháp luận hướng đối tượng gia tăng từ dưới 10 đến 50 trong khoảng những năm 1989 đến 1994, và do đó nảy sinh vấn đề là làm cho người phát triển khó tìm thấy một phương pháp luận duy nhất thoả mãn đầy đủ nhu cầu của họ.
- Vào tháng mười năm 1994, Rumbaugh đã liên kết với công ty Booch (Rational Sofware Corporation) để kết hợp phương pháp Booch và phương pháp OMT. Và cho ra một bản phác thảo về phương pháp có tên là Unified Process vào tháng mười năm 1995.
- Cũng trong năm 1995, Jacobson đã nỗ lực tích hợp phương pháp này với OOSE. Và những tài liệu đầu tiên về UML đã được trình làng vào trong năm 1996.
- Phiên bản 1.0 của UML đã được công bố vào tháng giêng 1997, bao gồm các công việc của các thành viên của UML consortium :

DEC	MCI Systemhouse
HP	Microsoft
i-Logix	Oracle
Intellicorp	Rational Software
IBM	TI
ICON	Computing Unisys

- Bản thảo về UML phiên bản 1.5 đã được tạo vào tháng ba năm 2003.
- Phiên bản UML 2.0 sẽ được tạo vào 2004.





UML ?

- UML được tạo ra nhằm chuẩn hoá ngôn ngữ mô hình hoá, UML không phải là một chuẩn về tiến trình và do đó, UML phải được sử dụng kết hợp với một tiến trình phương pháp luận.
- UML là một ngôn ngữ dùng để đặc tả, trực quan hoá, và tư liệu hoá phần mềm hướng đối tượng. Nó không mô tả một tiến trình hay một phương pháp mà trong đó chúng ta dùng nó để mô hình hoá. Ví dụ : Công ty Rational Software đề xuất một quy trình RUP (Rational Unified Process) được xem như là một phương pháp luận phát triển hệ thống và có ngôn ngữ mô hình hoá là UML.
- UML phủ tất cả các mức mô hình hoá khác nhau trong qui trình phát triển bao gồm chín loại sơ đồ, trong đó, năm sơ đồ dùng biểu diễn khía cạnh tĩnh và bốn sơ đồ biểu diễn khía cạnh động của hệ thống.

Các đặc trưng của một tiến trình sử dụng UML

Thuở ban đầu, qui trình tuần tự được xem là phương pháp hợp lý nhất để phát triển hệ thống. Tuy nhiên, trải qua vài thập niên, đã cho thấy các dự án sử dụng qui trình tuần tự thường ít thành công, bởi những nguyên do sau đây:

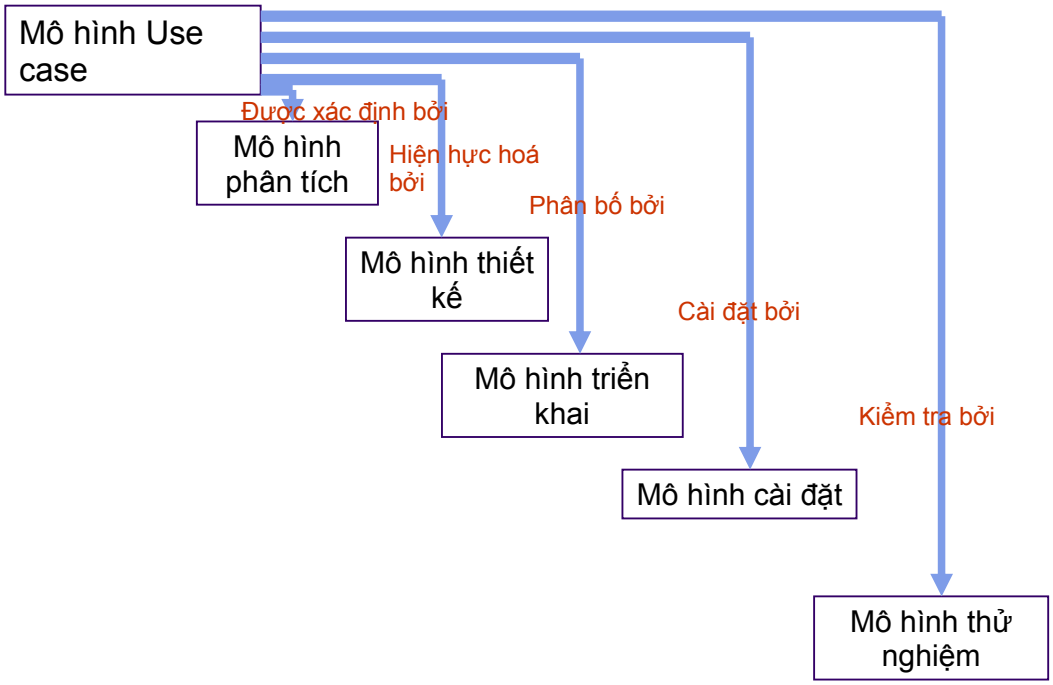
- Sự giả định ban đầu có sai sót
- Thất bại trong việc kết hợp các nhân tố con người
- Các hệ thống ngày càng lớn và thường hay thay đổi
- Chúng ta vẫn còn đang trong giai đoạn thăm dò của công nghệ phần mềm, và không có nhiều kinh nghiệm. Đây là lý do chính.

Một phương pháp luận sử dụng UML phải kết hợp với một qui trình lặp và điều này sẽ làm giảm đi các hạn chế của qui trình tuần tự. Tính chất lặp gồm các đặc trưng cơ bản sau :

- Tính lặp (iterative)
  - Thay vì nỗ lực xác định tất cả các chi tiết của mô hình trong một thời điểm, chúng ta chỉ xác định các chi tiết đã « đáp ứng » cho thời điểm đó để thực hiện, và
  - Lặp lại một (hoặc nhiều) vòng lặp khác bổ sung thêm các chi tiết
- Gia tăng (incremental)

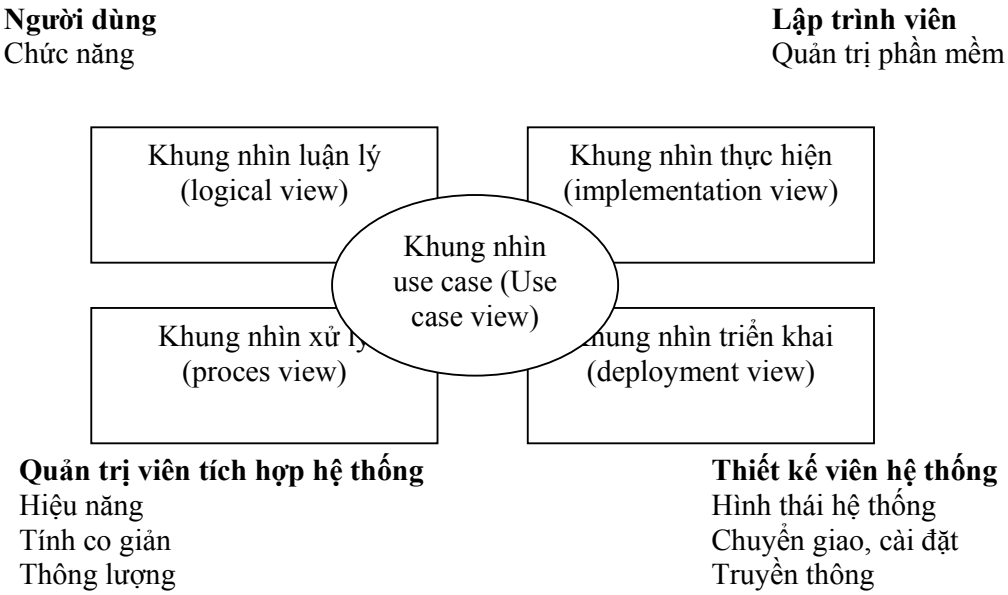
- Hệ thống tiến hoá thông qua một tập các sự gia tăng
- Mỗi sự gia tăng sẽ bù đắp thêm vào hệ thống các tính năng khác

- Tập trung vào người dùng (user – concentrated)
  - Phân tích viên xác định các tính năng của hệ thống thông qua các use case
  - Người dùng xác nhận các use case này
  - Thiết kế viên và người phát triển hiện thực hoá các use case
  - Người thử nghiệm kiểm tra hệ thống về việc thoả mãn các use case được đặt ra.



- Hướng kiến trúc (well-defined structure)
  - Hệ thống được phân chia thành các hệ thống con
  - Mức luận lý và vật lý phải được xác lập một cách tách biệt trong hệ thống
- Các khung nhìn về hệ thống :
  - Khung nhìn luận lý (logical view): mô tả các yêu cầu chức năng của hệ thống, tức những gì hệ thống nên làm cho người dùng cuối. Đó là sự trừu tượng của mô hình thiết kế và xác định các gói thiết kế chính, các subsystem và lớp chính. Trong UML khung nhìn này có thể được trình bày dùng sơ đồ lớp, sơ đồ đối tượng, sơ đồ mô tả các gói, hệ thống con.
  - Khung nhìn thực hiện (implementation view): mô tả tổ chức của các đơn thể (module) phần mềm tĩnh (như mã nguồn, tập tin dữ liệu, thành phần, tập tin thực thi, và các thành phần kèm theo khác) trong môi trường phát triển. Trong UML khung nhìn này có thể dùng sơ đồ thành phần để trình bày.
  - Khung nhìn xử lý (process view): mô tả các khía cạnh xảy ra đồng thời của hệ thống thời gian thực (run-time) (tasks, threads, processes cũng như sự tương tác giữa chúng). Khung nhìn này tập trung vào sự đồng hành, song song, khởi động và đóng hệ thống, khả năng chịu đựng hư hỏng, và sự phân tán các đối tượng.

- Khung nhìn triển khai (deployment): cho thấy các tập tin thực thi và các thành phần khác nhau được triển khai trên các hệ thống như thế nào. Nó giải quyết các vấn đề như triển khai, cài đặt, và tốc độ. Trong UML, khung nhìn này có thể sử dụng sơ đồ triển khai để mô tả.
- Khung nhìn use-case: đóng một vai trò đặc biệt đối với kiến trúc. Nó chứa một vài kịch bản hay use case chủ yếu. Ban đầu, chúng được dùng để khám phá và thiết kế kiến trúc trong các giai đoạn bắt đầu và giai đoạn đặc tả, nhưng sau đó chúng sẽ được dùng để xác nhận các khung nhìn khác nhau. Trong UML, khung nhìn này có thể sử dụng sơ đồ use case để minh hoạ.

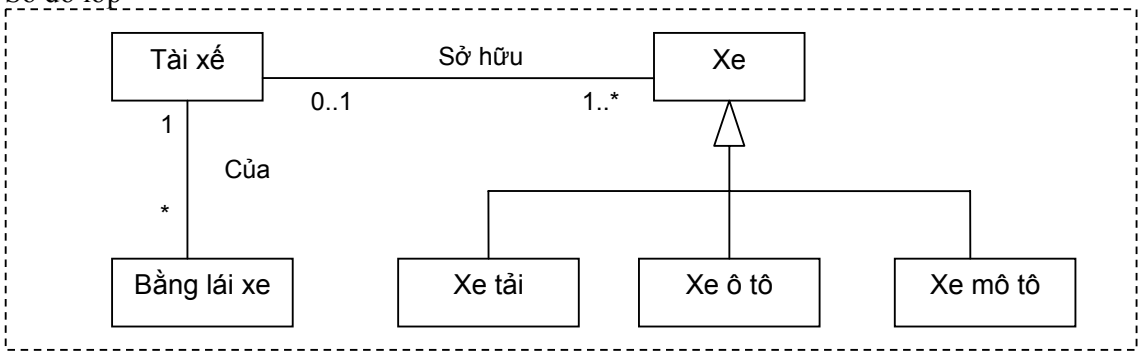


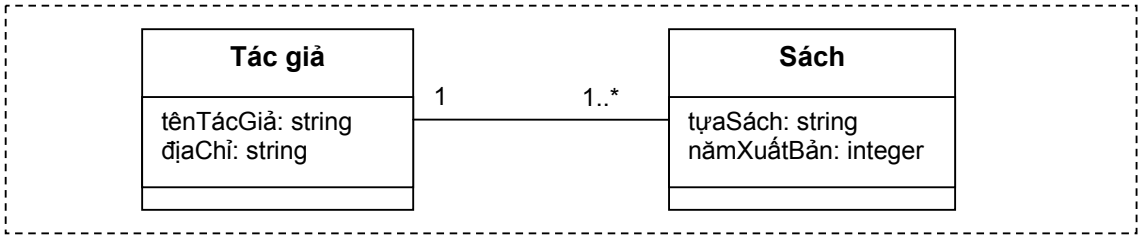
Cần phân biệt khung nhìn kiến trúc với mô hình: mô hình là sự trình bày hoàn chỉnh về hệ thống, trong khi khung nhìn kiến trúc chỉ tập trung vào những gì có ý nghĩa về mặt kiến trúc, tức là những gì có tác động rộng lớn đến cấu trúc của hệ thống và lên tốc độ, sự hoàn thiện, tính tiến hóa của nó.

Các sơ đồ trong UML

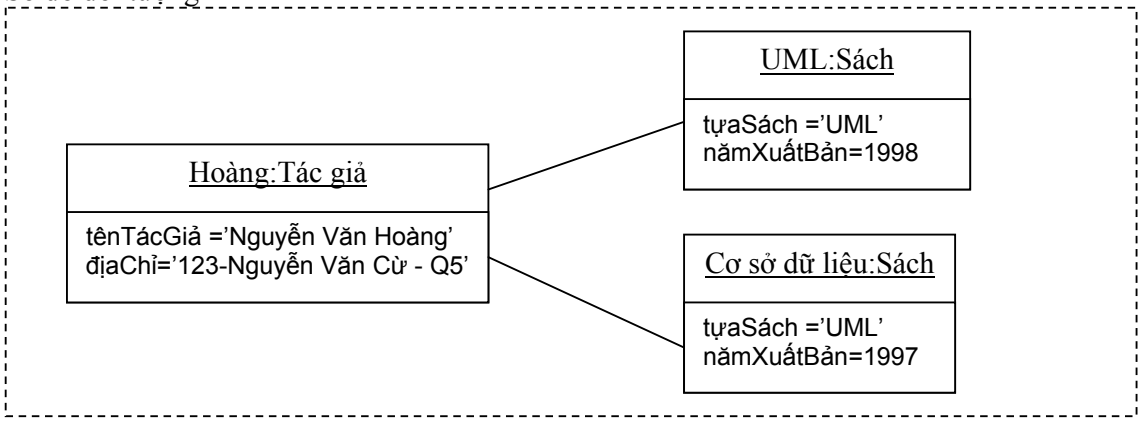
- Các sơ đồ mô tả khía cạnh tĩnh
  - Sơ đồ đối tượng (object diagram)
  - Sơ đồ lớp (class diagram)
  - Sơ đồ use case (use case diagram)
  - Sơ đồ thành phần (component diagram)
  - Sơ đồ triển khai (deployment diragram)
- Các sơ đồ mô tả khía cạnh động
  - Các sơ đồ tương tác (interaction diagram)
    - Sơ đồ tuần tự (sequence diagram)
    - Sơ đồ hợp tác (collaboration diagram)
  - Sơ đồ hoạt động (activity diagram)
  - Sơ đồ chuyển dịch trạng thái (state transition diagram)

Sơ đồ lớp và đối tượng: được sử dụng để mô hình hoá cấu trúc tĩnh của hệ thống trong quá trình phát triển. Mỗi sơ đồ chứa đựng các lớp và các mối quan hệ giữa chúng (quan hệ kế thừa (heritage), quan hệ kết hợp (association), quan hệ tập hợp (aggregation), quan hệ thành phần (composition)). Chúng ta cũng có thể mô tả các hoạt động của lớp (operation).
Sơ đồ đối tượng là một thể hiện của sơ đồ lớp. Nó mô tả trạng thái chi tiết của hệ thống tại một thời điểm cụ thể và là bức tranh của hệ thống tại một thời điểm, do đó, biểu đồ đối tượng được dùng để minh hoạ một trường hợp thực tế của sơ đồ lớp. Sơ đồ đối tượng có cùng ký hiệu với biểu đồ lớp. Sơ đồ đối tượng được dùng để minh hoạ một trường hợp phức tạp của bức tranh thực tế về hệ thống trong các thể hiện cụ thể.
Ví dụ :
Sơ đồ lớp



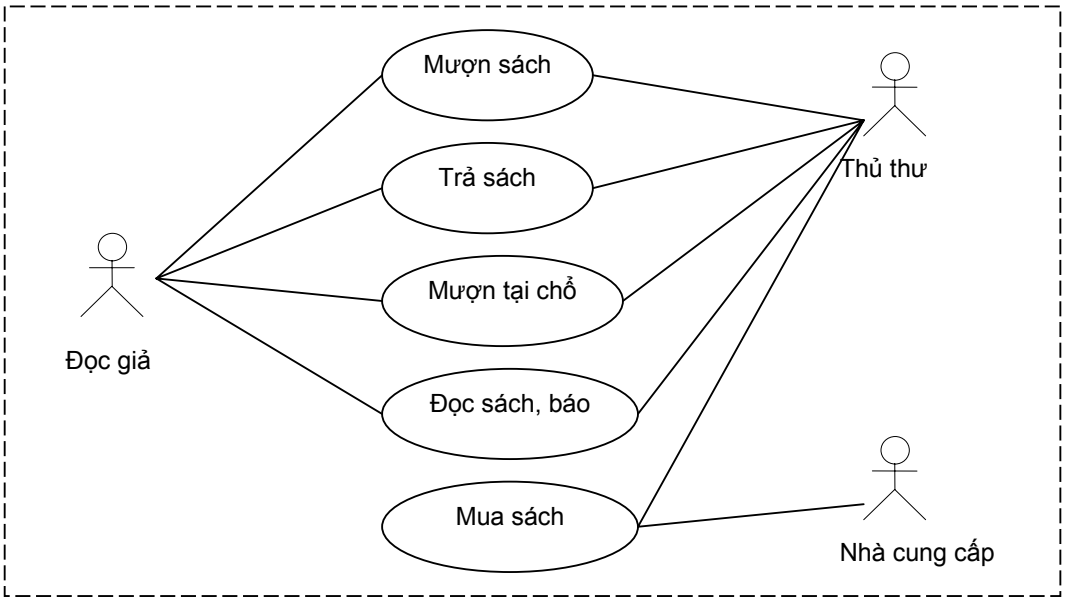


Sơ đồ đối tượng

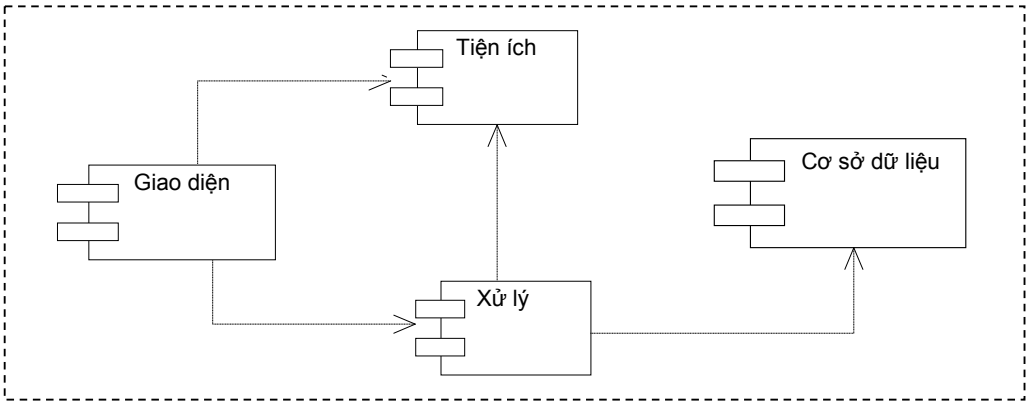


Sơ đồ use case : xuất phát từ các mô hình use case của phương pháp OOSE (Jacobson). Nó mô tả giao diện với một hệ thống từ quan điểm và cách nhìn của người sử dụng. Một sơ đồ use case mô tả các tình huống tiêu biểu của việc sử dụng một hệ thống. Nó biểu thị các trường hợp sử dụng (trong việc mô hình hoá các tính năng hệ thống) và các tác nhân (trong việc mô hình hoá các vai trò tham gia bởi các cá nhân tương tác với hệ thống), và mối quan hệ giữa các use case và các tác nhân.

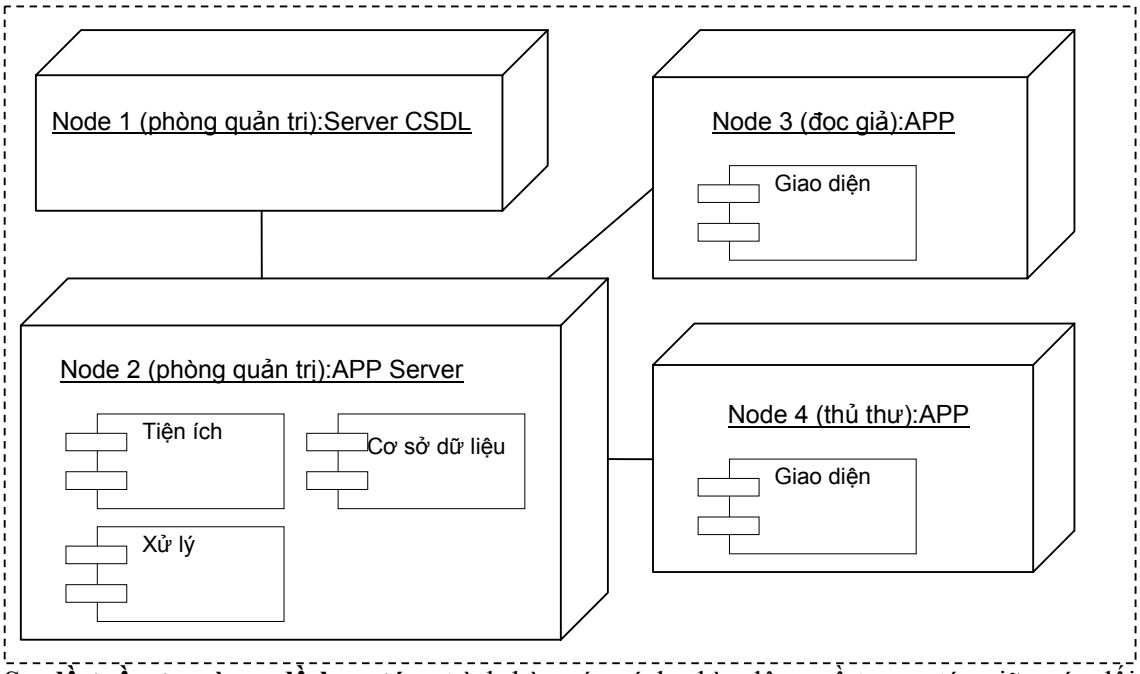
Ví dụ : sơ đồ use case một hệ thống quản lý một thư viện



Sơ đồ thành phần : được sử dụng để biểu thị các nhìn tĩnh trong việc cài đặt một hệ thống. Mỗi sơ đồ bao gồm các thành phần (component) và các mối quan hệ phụ thuộc giữa chúng trong môi trường cài đặt. Một thành phần đại diện cho một yếu tố cài đặt vật lý của môi trường (mã nguồn, mã thực thi, tập tin, cơ sở dữ liệu, một thư viện hàm,…).
Ví dụ : sơ đồ thành phần của một hệ thống phần mềm quản lý thư viện

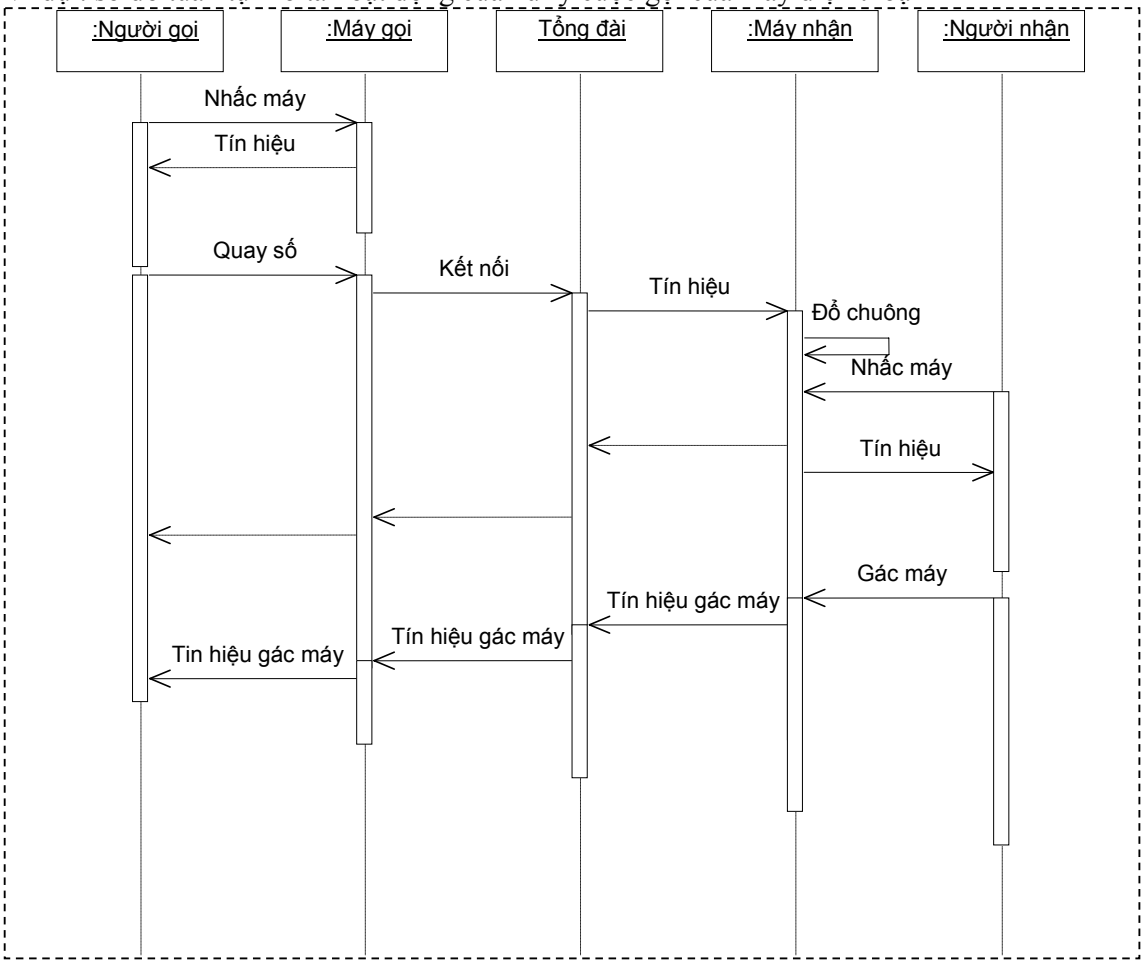


Sơ đồ triển khai : mô tả cách bố trí vật lý các thiết bị và sự phấn phối các thành phần trú ngụ tại các thiết bị này. Một sơ đồ triển khai bao gồm các nút (node) đại diện cho các tài nguyên thiết bị và các thành phần được cài đặt trong thiết bị, các liên kết trong sơ đồ dùng để mô tả sự trao đổi giữa các nút. Sơ đồ triển khai biểu thị một sự tương ứng giữa cấu trúc phần mềm của một hệ thống và kiến trúc về bố trí thiết bị của nó.
Ví dụ : sơ đồ triển khai của hệ thống quản lý thư viện

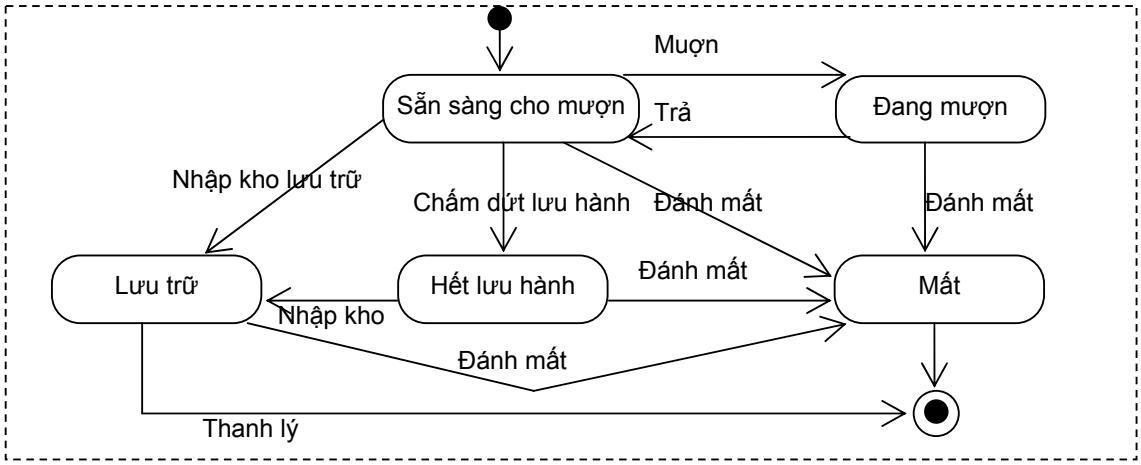


Sơ đồ tuần tự và sơ đồ hợp tác : trình bày các cách nhìn động về tương tác giữa các đối tượng của hệ thống trong quá trình phát triển. Sơ đồ hợp tác mô tả sự hợp tác giữa một nhóm các đối tượng trong hoạt động để đạt một mục tiêu cụ thể. Sơ đồ tuần tự thêm vào chiều thời gian nhằm thể hiện trực quan thứ tự trao đổi của các thông điệp (message).

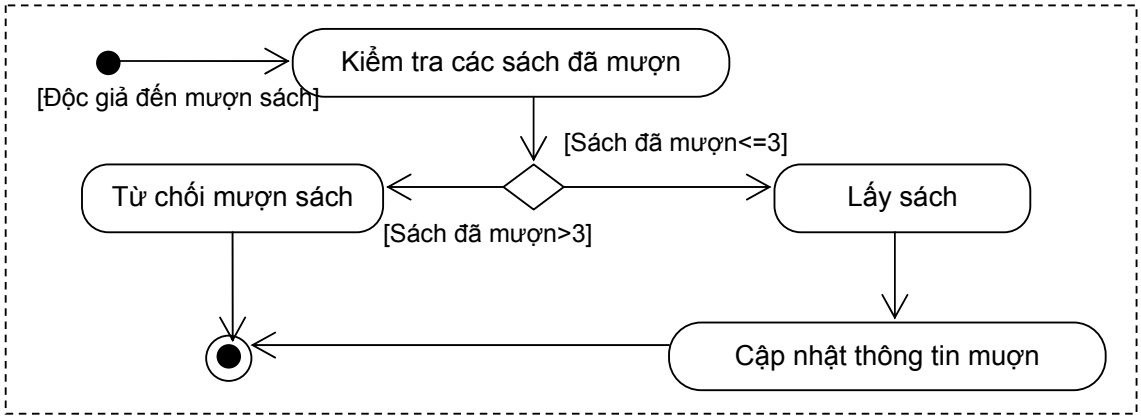
Ví dụ : sơ đồ tuần tự mô tả hoạt động của xử lý cuộc gọi của máy điện thoại



Sơ đồ chuyển đổi trạng thái : hình thành từ phương pháp OMT và Booch. Mỗi sơ đồ được dùng có liên quan đến một lớp để biểu thị các trạng thái khác nhau của đối tượng của lớp và các biến cố kích hoạt sự chuyển dịch giữa các trạng thái.
Ví dụ : sơ đồ trạng thái của sách trong thư viện



Sơ đồ hoạt động : dùng để mô hình hoá các dòng hoạt động liên kết tới các lớp như là trong trường hợp của một nhóm các lớp hợp tác cùng thực hiện trong một loại tiến trình. Mỗi lớp sẽ đảm nhiệm các hoạt động và các chuyển dịch như được mô tả trong sơ đồ chuyển dịch trạng thái. Tuy nhiên, một sơ đồ hoạt động có thể liên quan đến nhiều lớp hơn là một lớp. Mặt khác nó mô tả tiến trình tuần tự các hoạt động, sự đồng bộ hoá các dòng điều khiển song song, các điều kiện và quyết định, điểm bắt đầu và điểm kết thúc tiến trình.
Ví dụ : sơ đồ hoạt động đơn giản của hoạt động mượn sách thư viện



Các hệ thống sử dụng UML trong việc mô hình hoá

Việc sử dụng UML trong quá trình mô hình hoá có thể áp dụng trong các loại hệ thống sau :

Hệ thống nghiệp vụ (business system) : mô tả tài nguyên (nhân lực, tài lực, tài sản,…), mục tiêu, luồng công việc, các qui tắc, ràng buộc trong hoạt động sản xuất kinh doanh của doanh nghiệp. Ví dụ : các công ty sản xuất, cửa hàng kinh doanh, y tế, giáo dục, …

Hệ thống thông tin (informaton system) : thu thập và lưu trữ, biến đổi dữ liệu nhằm cung cấp thông tin đáp ứng nhu cầu người nhận trong các tổ chức hoạt động nghiệp vụ. Hệ thống thông tin cũng được chia thành nhiều loại tuỳ thuộc vào quy mô và độ phức tạp : hệ thống thông tin tác nghiệp : là hệ thống chuyên xử lý việc thu thập và truy tìm thông tin trong môi hoạt động nghiệp vụ. Hệ thống thông tin quản lý : xử lý tổng hợp dữ liệu thông qua các thống kê báo cáo nhằm đáp ứng thông tin cho các nhà quản lý theo dõi tình hình hoạt động. Hệ thống thông tin chuyên gia, hệ hỗ trợ ra quyết định : xử lý và tri thức hoá các dữ liệu hiện

tại nhằm đáp ứng các nhu cầu nâng cao về mặt thông tin như là hỗ trợ giải đáp tự động, hỗ trợ ra quyết định, dự báo tình hình tương lai,…

Phần mềm hệ thống (System software) : xây dựng các công cụ phần mềm cơ sở cho các phần mềm khác sử dụng như là hệ điều hành, hệ quản trị cơ sở dữ liệu, công cụ phát triển,….

Hệ thống nhúng (embeded system) : là một loại hệ thống phần mềm được xây dựng gắn trên một loại thiết bị như : điện thoại di động, thiết bị điều khiển,… Các hệ thống nhúng này thường được lập trình dùng ngôn ngữ cấp thấp hoặc chuyên dụng và có bộ nhớ lưu trữ cũng như màn hình.

Hệ thống kỹ thuật (Technical system) : xử lý và điều khiển các thiết bị kỹ thuật như hệ thống viễn thông, hệ thống quân sự, hay các quá trình xử lý kỹ thuật công nghiệp (dây chuyền sản xuất, vận hành máy móc,…). Đây là loại thiết bị phải xử lý các giao tiếp đặc biệt , không có phần mềm chuẩn và thường là các hệ thống thời gian thực (real time).

PHẦN 2

PHÂN TÍCH HỆ THỐNG

Mục tiêu

Giúp cho người học nắm vững các nội dung về:

- Tiến trình phân tích hướng đối tượng
- Tiến trình, nội dung và các phương pháp khảo sát yêu cầu
- Phân tích chức năng hệ thống bằng mô hình hoá use case
- Hiểu về hệ thống nghiệp vụ và mô hình hoá hệ thống nghiệp vụ
- Phân loại và xác định đối tượng hệ thống bằng việc xây dựng sơ đồ lớp

Giới thiệu

Bước đầu tiên để tìm ra một giải pháp thích hợp cho vấn đề hệ thống là hiểu vấn đề và lãnh vực của hệ thống đó. Mục tiêu chính của phân tích là nắm bắt một hình ảnh đầy đủ, không mơ hồ, và nhất quán về yêu cầu hệ thống và những gì hệ thống sẽ phải làm để đáp ứng đòi hỏi và nhu cầu người dùng. Điều này được thực hiện bằng cách xây dựng các mô hình của hệ thống với mục tiêu tập trung vào khía cạnh biểu diễn hệ thống về mặt nội dung (nghĩa là các mô hình tập trung vào làm rõ hệ thống có những gì) hơn là cách thức mà hệ thống thực hiện nội dung đó. Do đó, kết quả của giai đoạn phân tích là làm rõ các yếu tố hệ thống từ quan điểm và cách nhìn của người sử dụng mà không quan tâm đến cách thức chi tiết mà máy tính thực hiện nội dung đó.

Phân tích là một tiến trình chuyển đổi một định nghĩa vấn đề từ một tập mờ các sự kiện, các dự kiến mang tính tưởng tượng thành một diễn đạt chặt chẽ các yêu cầu hệ thống. Thực sự, phân tích là một hoạt động mang tính sáng tạo bao gồm việc hiểu vấn đề, các ràng buộc liên quan đến vấn đề đó và các phương pháp để khống chế hoặc giải quyết những ràng buộc. Đây là một tiến trình lặp cho đến khi các vấn đề phải được rõ ràng.

Phần này gồm ba chương bao gồm: xác định yêu cầu hệ thống; mô hình hoá use case; mô hình hoá nghiệp vụ; và xây dựng sơ đồ lớp.

Chương 4

XÁC ĐỊNH YÊU CẦU HỆ THỐNG

Mục tiêu

Qua chương này, chúng ta có thể hiểu:

- Mục tiêu của việc khảo sát hệ thống
- Nội dung của việc khảo sát và các đối tượng để tiếp cận khảo sát
- Môt số phương pháp khảo sát: phỏng vấn, bảng câu hỏi, phỏng vấn nhóm, phân tích tài liệu, thiết kế kết hợp người dùng, bản mẫu (prototype)
- Yêu cầu và việc phân loại các yêu cầu hệ thống

Mục đích khảo sát yêu cầu

Khảo sát hệ thống là nhằm thu thập tốt nhất thông tin phản ánh về hệ thống hiện tại, để từ đó làm cơ sở cho việc phân tích và xây dựng hệ thống mới giải quyết tồn tại bất cập của hệ thống. Vậy khảo sát yêu cầu bao gồm những mục tiêu sau:

- Tiếp cận với nghiệp vụ chuyên môn, môi trường của hệ thống
- Tìm hiểu vai trò, chức năng, nhiệm vụ và cách thức hoạt động của hệ thống
- Nêu ra được các điểm hạn chế, bất cập của hệ thống cần phải thay đổi
- Đưa ra được những vấn đề của hệ thống cần phải được nghiên cứu thay đổi.

Nội dung khảo sát

Nội dung khảo sát phải tìm hiểu được các nội dung của hệ thống như sau:

- Các mục tiêu hoạt động của đơn vị, các công việc và cách thức hoạt động để đạt những mục tiêu đó.
- Những thông tin cần để thực hiện công việc từng loại công việc
- Các nguồn dữ liệu (định nghĩa, cấu trúc, dung lượng, kích thước,…) bên trong và bên ngoài đơn vị. Có thể bao gồm:
    - Các hồ sơ, sổ sách, tập tin
    - Biểu mẫu, báo cáo, qui tắc, quy định, công thức.
    - Các qui tắc, qui định ràng buộc lên dữ liệu
    - Các sự kiện tác động lên dữ liệu
- Tìm hiểu khi nào, như thế nào, và bởi ai các dữ liệu đó được tạo ra, di chuyển, biến đổi và được lưu trữ. Ứng với mỗi xử lý thực hiện tìm hiểu:
    - Phương pháp: cách thức thực hiện
    - Tần suất: số lần thực hiện trong một đơn vị thời gian
    - Khối lượng: độ lớn thông tin thực hiện
    - Độ phức tạp: xử lý là một là một quá trình phức tạp liên quan đến nhiều loại dữ liệu hay chỉ là một tính toán đơn giản với một vài loại dữ liệu.
    - Độ chính xác: độ chính xác của kết quả thực hiện
- Thứ tự và các phụ thuộc khác giữa các hoạt động truy xuất dữ liệu khác nhau

- Các chính sách, hướng dẫn mô tả hoạt động quản lý, thị trường và môi trường hệ thống
- Các phương tiện, tài nguyên có thể sử dụng
- Trình độ chuyên môn sử dụng vi tính của các đối tượng xử lý thông tin hệ thống
- Môi trường hệ thống (kinh tế, xã hội, cơ quan chủ quản)
- Các đánh giá, phàn nàn về hệ thống hiện tại; các đề xuất giải quyết.

Đối tượng khảo sát

Có nhiều nguồn có thể cung cấp thông tin để đáp ứng nội dung khảo sát yêu cầu. Mỗi nguồn có một hình thức khác nhau do đó phải có một cách tiếp cận khảo sát khác nhau. Các đối tượng khảo sát đó là:

Người dùng

- *Các cán bộ lãnh đạo, cán bộ quản lý:* các đối tượng này sẽ giúp cho phân tích viên nắm bắt được tổng quan cấu trúc hệ thống, mục tiêu chung mà hệ thống mới mong muốn mang lại. Các thông tin mà đối tượng này mang lại thường là chiều rộng, mang tính tổng thể, chiến lược không mô tả chi tiết cách thức phải thực hiện.
- *Người sử dụng, nhân viên nghiệp vụ*: các đối tượng này sẽ cung cấp thông tin chi tiết cách thức mà họ đang thực hiện công việc gồm các bước cụ thể, các giấy tờ biểu mẫu liên quan. Các thông tin mà đối tượng mang lại thường là chiều sâu, chi tiết, cục bộ bỏ qua ý tưởng chiến lược mang tính tổng thể.
- *Nhân viên kỹ thuật*: các đối tượng này sẽ cung cấp thông tin về tình trạng công nghệ, trang thiết bị, phần mềm hiện hành đang sử dụng và khả năng, trình độ về kỹ thuật của họ. Các đối tượng này thường trợ giúp rất lớn trong việc huấn luyện, triển khai và bảo trì hệ thống mới.

Tài liệu

- *Tài liệu về sổ sách, biểu mẫu, tập tin*: nguồn cung cấp các thông tin về dữ liệu, luồng dữ liệu, giao dịch và xử lý giao dịch. Đặc biệt là các biểu mẫu đây chính là kết quả đầu ra của hệ thống.
- *Tài liệu về qui trình, thủ tục*: nguồn cung cấp thông tin về qui trình xử lý, vai trò xử lý của các nhân viên, chi tiết mô tả công việc của nhân viên, các qui định thủ tục.
- *Các thông báo*: các mẫu thông báo của hệ thống đối với môi trường ngoài, giữa các bộ phận trong hệ thống (ví dụ: thông báo họp mặt khách hàng, thông bao mời thầu, thông báo từ chối đơn hàng, … hoặc các thông báo nội bộ như là thông báo bổ nhiệm, thông báo nâng lương,…)

Chương trình máy tính

- Các chương trình phần mềm hệ thống đang sử dụng, các chương trình này giúp xác định được cấu trúc dữ liệu hệ thống, thói quen của người sử dụng, chức năng mà hệ thống mới chưa đáp ứng được, số liệu thử nghiệm hệ thống.

Phương pháp xác định yêu cầu

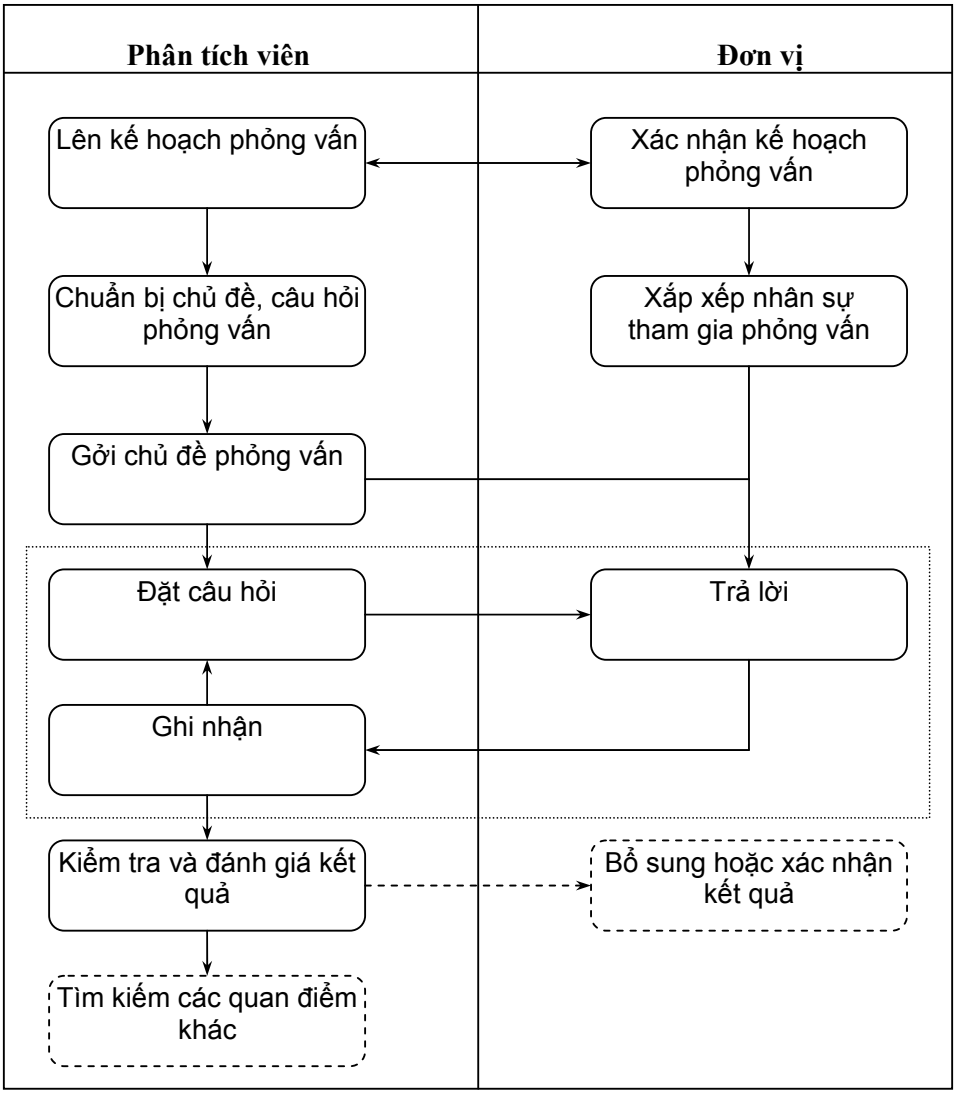
Các phương pháp truyền thống xác định yêu cầu

Phỏng vấn

Phỏng vấn là một hình thức khảo sát thu thập thông tin trực tiếp từ các đối tượng sẽ sử dụng hệ thống. Vì mỗi người dùng sẽ có những hiểu biết nhất định về một phần công việc của mình trong hệ thống hiện tại và mong muốn hệ thống mới về những gì sẽ phục vụ và trợ giúp

cho công việc của họ. Ví dụ: một kế toán viên chi tiết thì biết được chi tiết các loại chứng từ, cách sắp xếp và xử lý chứng từ,… còn kế toán viên tổng hợp thì chỉ quan tâm đến những số liệu nào và cách thức để tổng hợp số liệu đó để tạo ra các báo cáo thống kê, tổng hợp,… Do đó, việc phỏng vấn phải được thực hiện trên nhiều người dùng khác nhau (ba loại người dùng) nhằm thu thập nhiều nhất yêu cầu hệ thống.

Phỏng vấn là một cách thức đối thoại trực tiếp trong đó, phân tích viên sẽ ra câu hỏi và đối tượng phỏng vấn sẽ trả lời câu hỏi. Qui trình các bước thực hiện như sau:



Hình 1. Sơ đồ mô phỏng quá trình phỏng vấn

Đầu tiên phân tích viên chuẩn bị một kế hoạch phỏng vấn tổng quát, kế hoạch này sẽ liệt kê tất cả các lãnh vực của hệ thống cần khảo sát và thời gian dự kiến cho từng lãnh vực. Mẫu kế hoạch như sau:

Kế hoạch phỏng vấn tổng quan

Hệ thống:……..

Người lập:……. *Ngày lập:../../….*

STT	*Chủ đề*	*Yêu cầu*	*Ngày bắt đầu*	*Ngày kết thúc*

Kế hoạch phỏng vấn này sẽ được gởi đến đơn vị để được xác nhận về thời gian và bố trí nhân viên nào sẽ tham gia trả lời phỏng vấn. Một cuộc phỏng vấn sẽ hiệu quả hơn khi người phỏng vấn chuẩn bị các câu hỏi và thiết lập cho mình một hướng dẫn phỏng vấn và đối tượng trả lời biết trước được các câu hỏi để chuẩn bị thì chắc chắn thông tin trả lời sẽ xác định hơn và thời gian phỏng vấn sẽ được rút ngắn. Sau khi kết thúc phỏng vấn, phân tích viên phải dành thời gian để tổng hợp lại các kết qủa ghi nhận được, loại bỏ các thông tin trùng lắp, tìm ra vấn đề nào vẫn chưa rõ ràng cần phải hỏi lại, nếu cần thiết gởi bản kết quả phỏng vấn đến người được phỏng vấn nhờ xác nhận lại. Sau đó, phân tích viên nên tham khảo thêm các quan điểm khác về vấn đề đã phỏng vấn để có một quan điểm tổng quan hơn trong việc đánh giá kết quả ghi nhận được.

Bảng kế hoạch hướng dẫn buổi phỏng vấn *Hệ thống:……*	
Người phỏng vấn:……..	*Phân tích viên:……..*
Vị trí/ phương tiện Văn phòng, phòng họp, điện thoại,…	
Mục tiêu: Dữ liệu gì? Lãnh vực nào?	
Chi tiết buổi phỏng vấn Giới thiệu Tổng quan của hệ thống Chủ đề 1 Các câu hỏi Chủ đề 2 Các câu hỏi ... Tóm tắt các điểm chính Câu hỏi của người trả lời phỏng vấn Kết thúc	*Thời gian ước lượng (phút)* Tổng:
Quan sát tổng quan	
Phát sinh ngoài dự kiến	

Bảng câu hỏi mẫu dành cho phân tích viên để chuẩn bị câu hỏi và ghi nhận kết quả phỏng vấn (kết quả trả lời và kết quả quan sát về thái độ cử chỉ biểu hiện bên ngoài)

Người được phỏng vấn:……	*Ngày:../../….*
Câu hỏi	*Ghi nhận*
Câu hỏi 1:	*Trả lời:*

	Kết quả quan sát:

Ví dụ:

Người được phỏng vấn: Trần Thị X…	*Ngày: 05/08/2003*
Câu hỏi	*Ghi nhận*
Câu hỏi 1: Tất cả đơn hàng của khách hàng phải được thanh toán trước rồi mới giao hàng?	*Trả lời:* Phải thanh toán trước hoặc ngay khi giao. *Kết quả quan sát:* Thái độ không chắc chắn
Câu hỏi 2: Anh Chị muốn hệ thống mới sẽ giúp cho Anh Chị điều gì?	*Trả lời* Dữ liệu chỉ nhập một lần và các báo cáo tự động tính toán *Kết quả quan sát* Không tin tưởng lắm, hình như đã triển khai thất bại một lần

Loại câu hỏi phỏng vấn:

Câu hỏi mở: là câu hỏi giúp cho việc trả lời được tự do trong phạm vi hệ thống. Kết quả trả lời không tuân theo một vài tình huống cố định. Mục đích của câu hỏi mở là khuyến khích người trả lời đưa ra được tất cả ý kiến có thể trong khuôn khổ câu hỏi. Do đó, câu hỏi mở dùng để thăm dò, gợi mở vấn đề và người trả lời phải có một kiến thức tương đối.

Ví dụ: "Anh (Chị) đang xử lý thông tin gì?" hoặc "Anh (Chị) có khó khăn gì khi quản lý dữ liệu của mình?"

Câu hỏi đóng: là câu hỏi mà sự trả lời là việc chọn lựa một hoặc nhiều trong những tình huống xác định trước. Do đó, câu hỏi đóng được dùng xác định một tình huống cụ thể.

Ví dụ: "Điều nào dưới đây là tốt nhất đối với HTTT Anh (Chị) đang sử dụng?"

- □ Dễ dàng truy cập đến tất cả dữ liệu cần
- □ Thời gian trả lời tốt nhất của hệ thống
- □ Khả năng chạy đồng thời với các ứng dụng khác

Câu hỏi đóng thường được thiết kế theo một trong những dạng sau:

- Đúng – sai
- Nhiều chọn lựa (có thể một trả lời hoặc trả lời tất cả chọn lựa)
- Tỉ lệ trả lời: từ xấu đến tốt, từ rất đồng ý đến hoàn toàn không đồng ý. Mỗi điểm trên tỉ lệ nên có một nghĩa rõ ràng và nhất quán và thường có một điểm trung lập ở giữa
- Xếp hạng các chọn lựa theo thứ tự mức độ quan trọng

Xắp xếp câu hỏi: thứ tự câu hỏi phải hợp lý, phù hợp với mục tiêu khảo sát và khả năng của người trả lời. Các thứ tự có thể là:

- Thu hẹp dần: ban đầu là những câu hỏi rộng, khái quát và càng về sau thì thu hẹp đến một mục tiêu.
- Mở rộng dần: ban đầu đề cập đến một điểm nào đó rồi mở rộng dần phạm vi đề cập

Câu hỏi mở	Câu hỏi đóng
Ưu điểm - Không ràng buộc kết quả trả lời - Có thể phát sinh ý tưởng mới	- Thời gian trả lời ngắn - Nội dung trả lời tập trung, chi tiết
Khuyết điểm - Thời gian dễ kéo dài - Nội dung trả lời có thể vượt phạm vi câu hỏi	- Mất nhiều thời gian chuẩn bị câu hỏi - Không mở rộng được kết quả trả lời

Khảo sát dùng bảng câu hỏi (questionaire)

Phỏng vấn là một phương pháp hiệu quả để trao đổi và thu thập được những thông tin quan trọng từ phía người dùng. Tuy nhiên, thực hiện phỏng vấn cũng rất tốn kém về thời gian và nguồn lực. Phương phướng khảo sát dùng bảng câu hỏi ít tốn kém hơn, thời gian trả lời lại nhanh hơn, kết quả xác định hơn và thu thập được thông tin từ nhiều đối tượng hơn trong cùng một thời gian ngắn. Tuy nhiên, phương pháp này lại thụ động và ít mang lại chiều sâu hơn phương pháp phỏng vấn.

Để thực hiện, các công việc phân tích viên cần phải làm rõ:

- Tập hợp câu hỏi thành từng nhóm
- Phân loại các đối tượng sử dụng thành nhóm và gởi nhóm câu hỏi nào đến nhóm người nào. Tổng quát , việc góm nhóm sẽ được thực hiện bởi một hoặc sự kết hợp của bốn phương pháp sau:
    - Đối tượng tích cực: đối tượng có vị trí thuận lợi, sẵn lòng để được khảo sát, hoặc những đối tượng có nhiều động lực trả lời nhất.
    - Nhóm ngẫu nhiên: chọn ngẫu nhiên một nhóm người dùng trong danh sách để gởi câu hỏi.
    - Theo chủ định: chọn những người thoả các tiêu chuẩn xác định nào đó. Ví dụ: những người đã làm việc với hệ thống 2 năm trở lên, những người thường xuyên sử dụng hệ thống,…
    - Chọn theo loại: người dùng, quản lý, …

Thông thường, người ta kết hợp các phương pháp lại. Trong bất kỳ trường hợp nào khi nhận được trả lời chúng ta nên kiểm tra lại các trường hợp không trả lời để tìm ra nguyên nhân và xem xét các kết quả trả lời là hợp lệ và đủ để được chấp nhận không.

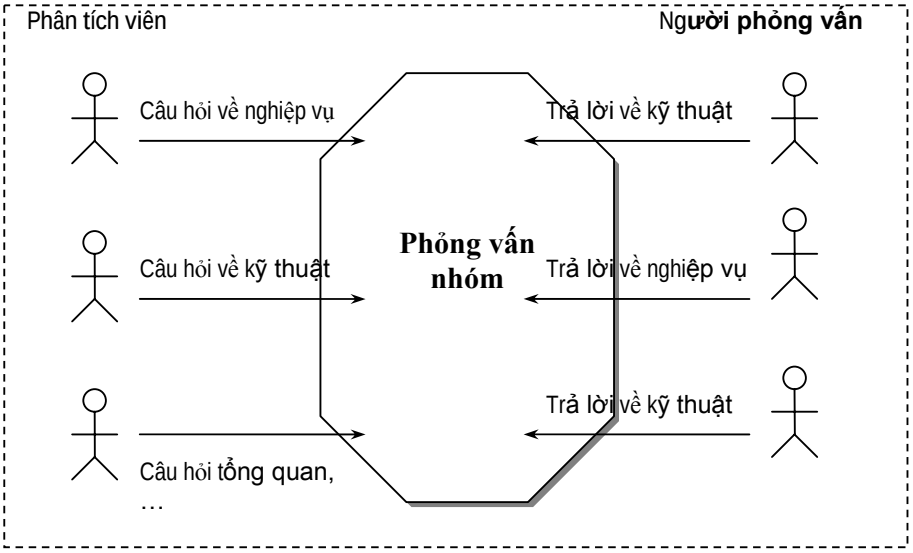
So sánh giữa phỏng vấn và bảng câu hỏi được liệt kê dưới đây

Đặc điểm	Phỏng vấn	Bảng câu hỏi
Sự phong phú thông tin	Cao (qua nhiều kênh: trả lời, cử chỉ,…)	Trung bình tới thấp (chỉ trả lời)
Thời gian	Có thể kéo dài	Thấp, vừa phải
Chi phí	Có thể cao	Vừa phải
Cơ hội nắm bắt và phát hiện	Tốt: việc phát hiện và chọn lọc các câu hỏi có thể được đặt ra bởi hoặc người phỏng vấn	Hạn chế: sau khi thu thập dữ liệu cơ sở

	hoặc người được phỏng vấn	
Tính bảo mật	Mọi người biết lẫn nhau	Không biết người trả lời
Vai trò tham gia	Người được phỏng vấn đóng một vai trò quan trọng và có thể quyết định kết quả	Trả lời thụ động, không chắc chắn quyết định kết quả

Phỏng vấn nhóm

Các yêu cầu được thu thập có thể sử dụng phương pháp phỏng vấn hoặc điều tra dùng bảng câu hỏi. Tuy nhiên, các kết quả phỏng vấn các đối tượng khác nhau có thể dẫn đến sự không nhất quán thông tin về hệ thống hiện hành và yêu cầu về hệ thống mới. Do đó, chúng ta lại phải thực hiện việc kiểm tra, chọn lọc và quyết định chính xác đâu là thông tin đúng và được chấp nhận cuối cùng. Thông thường chúng ta tiếp tục thực hiện các trao đổi và gặp gỡ các nhân vật quan trọng có thể quyết định định và giới hạn được kết quả thông tin. Các cuộc phỏng vấn mới này thường tốn thời gian và có khi lại trả lời lại các câu hỏi mà chúng ta đã được trả lời trước đó bởi những người khác. Do đó, phương pháp phỏng vấn từng cá nhân riêng lẽ vẫn còn những hạn chế nhất định.



Phỏng vấn nhóm là một phương pháp tốt có thể giúp giải quyết được những yêu cầu trái ngược nhau. Các đặc điểm của phỏng vấn nhóm bao gồm:

- Nhiều phân tích viên phụ trách nhiều lãnh vực khác nhau
- Nhiều đối tượng phỏng vấn khác nhau mỗi đối tượng phụ trách một lãnh vực, có thể phân cấp từ quản lý đến nhân viên trực tiếp liên quan.
- Tổ chức một buổi phỏng vấn chung gồm các phân tích viên và các đối tượng phỏng vấn.
- Mỗi phân tích viên có thể đặt câu hỏi và các đối tượng đều có thể trả lời. Phân tích viên có thể ghi nhận lại chỉ những ý kiến liên quan đến lãnh vực của mình.

Lợi điểm:

- Giảm thiểu thời gian phỏng vấn: tất cả các yêu cầu sẽ được thông suốt tại một thời điểm thay vì phải phỏng vấn từng đối một tại những thời điểm khác nhau và thời gian sẽ kéo dài ra

- Cho phép các đối tượng phỏng vấn nghe được ý kiến chủ đạo của lãnh đạo trên những ý kiến bất đồng liên quan đến một vấn đề đặt ra. Đây là cơ hội làm cho các đối tượng thông suốt được ý kiến chủ đạo liên quan đến hệ thống mới.

Nhược điểm:

Nhược điểm chính là rất khó để tổ chức một buổi phỏng vấn nhóm vì khó để tìm được một thời gian và vị trí thích hợp cho tất cả mọi người. Ngày nay với công nghệ truyền thông phát triển cho phép tổ chức một buổi họp với các thành viên ở khoảng cách xa nhau (ví dụ: dùng Video conference).

Quan sát trực tiếp

Quan sát trực tiếp tại nơi làm việc, hiện trường nhằm thu thập chính xác cách thức và qui trình làm việc thực tế của hệ thống.

Ưu điểm:

- Đảm bảo tính trung thực của thông tin. Bởi vì các phương pháp phỏng vấn bị phụ thuộc vào cách thức mà người dùng trả lời, kiến thức và chủ quan của họ,
- Thu thập tốt về thông tin mô tả tổng quan về hệ thống.

Hạn chế

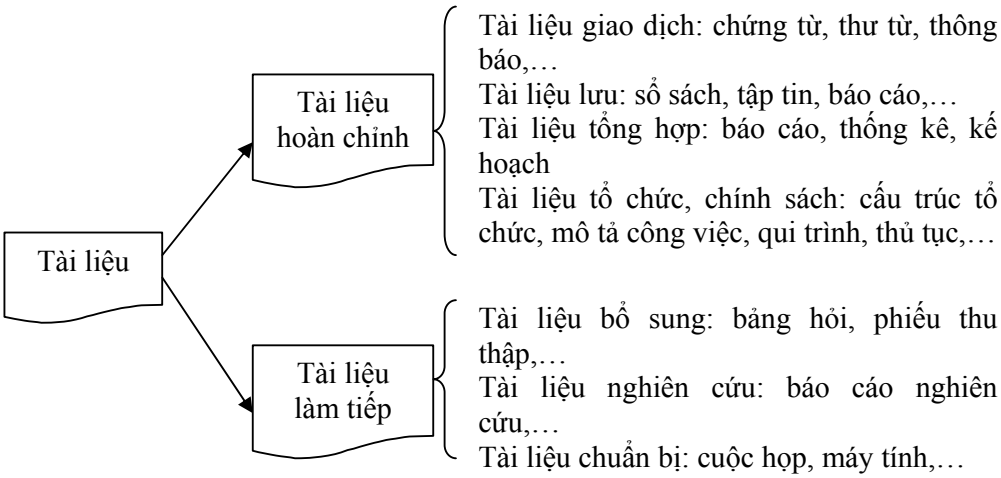
- Thời gian có thể kéo dài.
- Làm cho người dùng khó chịu khi thực hiện công việc, vì có cảm giác như bị theo dõi. Do đó, họ thường thay đổi cách thức làm việc không đúng với hiện trạng.

Thông thường, người ta kết hợp các phương pháp phỏng vấn với phương pháp quan sát để tiến hành khảo sát.

Phân tích tài liệu và thủ tục

Phương pháp quan sát hệ thống hoạt động là phương pháp trực tiếp thì phương pháp nghiên cứu tài liệu và tủ tục là phương pháp quan sát gián tiếp, bởi vì nó không nghiên cứu trực tiếp ở hiện trường hệ thống mà thông qua các văn bản, giấy tờ, tài liệu, tập tin máy tính,… mô tả hệ thống. Phương pháp này giúp xác định chi tiết về hệ thống hiện hành.

Có rất nhiều tài liệu liên mô tả hoạt động hệ thống, các yêu cầu của hệ thống trong tương lai: tài liệu mô tả nhiệm vụ, các kế hoạch kinh doanh, cấu trúc tổ chức, các tra cứu về chính sách, bản mô tả công việc, các thư tín bên trong và bên ngoài, các báo cáo nghiên cứu,… Chúng ta có thể thu thập được nhiều loại thông tin từ các hoạt động chung của đơn vị đến các dữ liệu cơ bản, dữ liệu cấu trúc. Thông thường phương pháp này kết hợp với phương pháp phỏng vấn ở mức thấp.



Phân tích tài liệu sẽ mang lại các thông tin sau:

- Các vấn đề tồn tại trong hệ thống (thiếu thông tin, các bước xử lý dư thừa)
- Các cơ hội để hệ thống đáp ứng nhu cầu mới (ví dụ: việc phân tích tài liệu cho thấy từ dữ liệu lưu trữ mà lâu nay không để ý có thể phân tích thông tin từng loại khách hàng , điều này tạo một cơ hội cho bộ phận bán hàng là có thể đánh giá và phân tích hoạt động bán hàng)
- Phương hướng tổ chức có thể tác động đến các yêu cầu của HTTT (ví dụ: một phương hướng mới của đơn vị là liên kết khách hàng và nhà cung cấp gần gũi hơn nữa với đơn vị mà trước đây chưa tính đến hoặc chưa thực hiện. Phương hướng này làm nảy sinh các nhu cầu mới về HTTT cần có để đáp ứng như là: hệ thống mới cần mở ra các kênh liên lạc thông tin cho khách hàng, xử lý các đánh giá dịch vụ khách hàng,…)
- Lý do tồn tại của hệ thống hiện hành, những chi tiết không được quản lý bởi hệ thống hiện hành và bây giờ thì cần thiết và khả thi trong hệ thống mới.
- Tìm ra tên và vị trí của những cá nhân có liên quan đến hệ thống. Giúp cho việc giao tiếp liên lạc đúng mục tiêu hơn.
- Giá trị của đơn vị, cá nhân có thể trợ giúp để xác định các ưu tiên đối với những khả năng khác nhau đến từ nhiều người dùng khác nhau.
- Các trường hợp xử lý thông tin đặc biệt không thường xuyên không thể được xác định bởi những phương pháp khác.
- Dữ liệu cấu trúc, qui tắc xử lý dữ liệu, các nguyên lý hoạt động được thực hiện bởi HTTT.

Một loại tài liệu hữu dụng khác là các thủ tục mô tả công việc của từng cá nhân hoặc nhóm. Các thủ tục này mô tả cách thức một công việc hoạt động, gồm dữ liệu và thông tin được sử dụng và được tạo ra trong quá trình thực hiện công việc.

Tuy nhiên, việc phân tích tài liệu thủ tục cũng có một số nhược điểm sau:

- Các thủ tục cũng là nguồn thông tin không đúng, trùng lắp
- Thiếu tài liệu
- Tài liệu hết hạn: dẫn đến việc phân tích tài liệu cho một kết quả không đúng với kết quả khi phỏng vấn.

Các phương pháp mới xác định yêu cầu

Thiết kế kết hợp người dùng (JAD - Join Application Design)

Mục tiêu của JAD là một tiến trình xác định yêu cầu trong đó người dùng, nhà quản lý và các nhà phân tích làm việc với nhau trong một vài ngày diễn ra trong các buổi họp tập trung (trong một phòng) để xác định hoặc kiểm tra lại các yêu cầu hệ thống và các thiết kế chi tiết. Do đó, JAD có hình thức như là phương pháp phỏng vấn nhóm. Tuy nhiên, JAD đi theo một cấu trúc vai trò và chương trình đặc biệt hoàn toàn khác với phương pháp phỏng vấn nhóm đó là phân tích viên điều khiển thứ tự câu hỏi được trả lời bởi người dùng.

Mục đích chính của JAD trong giai đoạn phân tích là thu thập yêu cầu hệ thống một cách đồng thời từ nhiều đối tượng khác nhau, kết quả là một tiến trình tập trung, có cấu trúc nhưng có hiệu quả cao. Điểm giống nhau với phỏng vấn nhóm là JAD cũng cho phép các phân tích viên quan sát vá xác định được ở đâu đồng ý và ở đâu có bất đồng trong các người dùng. Các cuộc gặp gỡ diễn ra trong vòng nhiều ngày tạo ra cơ hội để giải quyết bất đồng hoặc ít nhất cũng hiểu được tại sao có bất đồng.

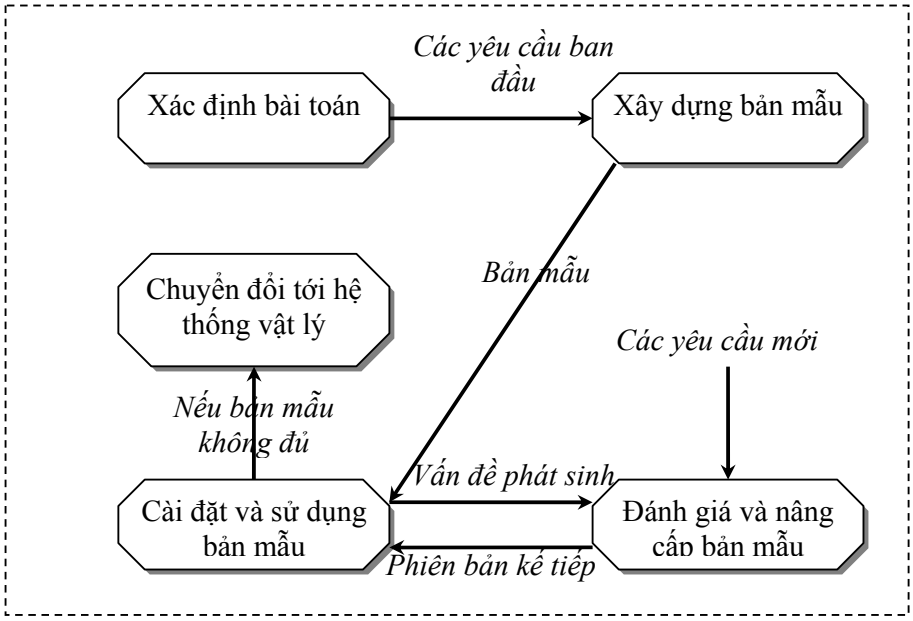
Thành phần của JAD bao gồm:

- Một địa điểm: một địa điểm (phòng họp) có đầy đủ các trang thiết bị hỗ trợ cho cuộc họp, mục đích là làm cho mọi người có tập trung cao trong việc phân tích hệ thống.
- Người tham dự bao gồm:
    - Người chủ trì: điều hành cuộc họp, thiết lập chương trình, giữ thái độ trung lập, tập trung vào việc hướng cuộc họp vào đúng chương trình, giải quyết bất đồng.
    - Người dùng: đại diện người sử dụng hệ thống
    - Phân tích viên hệ thống: các phân tích viên đặt câu hỏi về hệ thống
    - Người ghi chép: ghi chép tất cả các tông tin quá trình diễn ra JAD
    - Nhân viên HTTT: ngoài các phân tích viên, bao gồm thêm các lập trình viên, phân tích viên CSDL.
- Chương trình: chương trình thể hiện nội dung của JAD bao gồm các bước và cuộc họp phải diễn ra đúng chương trình này
- Công cụ: các công cụ trợ giúp phân tích thiết kế (thiết kế bản mẫu, vẽ sơ đồ,…), đánh giá và trợ giúp cho các phân tích để nâng cao hiệu quả và giảm thiểu thời gian của JAD.

Sử dụng bản mẫu (prototype) xác định yêu cầu

Sử dụng bản mẫu như một kỹ thuật xác định yêu cầu, phân tích viên làm việc với người dùng để xác định các yêu cầu cơ bản và ban đầu của hệ thống. Sau đó, phân tích viên dựa trên yêu cầu này để xây dựng một bản mẫu ban đầu. Bản mẫu khi hoàn thành sẽ gởi đến người dùng để người dùng sử dụng thử và kiểm tra. Đặc biệt, việc trực quan hóa các mô tả yêu cầu bằng lời được chuyển đổi thành hệ thống vật lý sẽ nhắc nhở người dùng thay đổi những yêu cầu tồn tại không phù hợp và phát sinh những yêu cầu mới (ví dụ: trong buổi phỏng vấn ban đầu, người dùng muốn xây dựng một form nhập hóa đơn với tất cả thông tin về khách hàng, hoá đơn, dịch vụ, hàng hoá, quá trình thanh toán,… theo cách nghĩ của người dùng là tiện lợi. Tuy nhiên sau khi sử dụng bản mẫu, người dùng sẽ cảm thấy phức tạp, lẫn lộn và sẽ thay đổi yêu cầu với nhiều form khác nhau và sự di chuyển hợp lý giữa các form). Kết quả thử nghiệm của người dùng sẽ phản hồi tới phân tích viên và phân tích viên sẽ dùng thông tin phản hồi này để cải tiến bản mẫu rồi tiếp tục gởi đến người dùng và vòng lặp này cứ tiếp tục như vậy cho đến khi bản mẫu thoả mãn người dùng.

Khi sử dụng phương pháp này, phân tích viên cũng phải sử dụng các phương pháp truyền thống để thu thập thông tin ban đầu.



Hình 2. Sơ đồ xác định yêu cầu dùng phương pháp bản mẫu (The New paradigm for Systems Development – J.D. Naumann & A.M. Jenkins)

Phương pháp bản mẫu sẽ rất hữu dụng để xác định yêu cầu trong các trường hợp sau:

- Yêu cầu chưa rõ ràng và thông suốt, thường là các trường hợp về hệ thống mới hoặc là các trường hợp về hệ hỗ trợ ra quyết định.
- Người dùng và các thành viên khác tham gia vào việc phát triển hệ thống.
- Việc thiết kế phức tạp và đòi hỏi phải có một hình thức cụ thể để đánh giá.
- Có những vấn đề giao tiếp đã tồn tại giữa phân tích viên và người dùng và tất cả đều mong muốn làm sáng tỏ.
- Công cụ (đặc biệt là công cụ phát sinh form và report) và dữ liệu sẵn sàng để xây dựng hệ thống.

Phương pháp này cũng có một số hạn chế:

- Tạo ra một xu hướng làm việc không theo chuẩn tài liệu hình thức về yêu cầu hệ thống, và điều này làm khó khăn hơn để phát triển một hệ thống đầy đủ cần phải có một chuẩn mực tuân theo.
- Các bản mẫu có thể trở thành rất đặc thù phong cách của người dùng ban đầu và khó để thích ứng với những người dùng tiềm năng khác.
- Các bản mẫu thường được xây dựng trên các hệ thống đơn. Do đó, nó bỏ qua các phát sinh về tương tác và chia sẻ dữ liệu với những hệ thống khác.

Ví dụ: mô tả khảo sát hoạt động của hệ thống máy ATM ngân hàng ABC

ATM là một loại máy rút tiền tự động, máy được các ngân hàng lắp đặt để hỗ trợ cho khách hàng có thể rút tiền ở các vị trí thuận tiện mà không phải đến ngân hàng. Hoạt động của máy được mô tả như sau:

Hệ thống được thiết kế để điều khiển một máy giao dịch tự động (ATM – Automated Teller

Machine) có một đầu đọc từ để đọc thẻ ATM, một màn hình giao tiếp (hiển thị và bàn phím), một khe nhỏ để chuyển tiền, một khay đựng tiền, một máy in để in hoá đơn và một công tắc cho phép nhân viên vận hành bật và tắt máy. Máy ATM sẽ giao tiếp với hệ thống ngân hàng thông qua một phương thức thích hợp.

Máy ATM sẽ phục vụ cho một khách hàng tại một thời điểm. Khách hàng của ngân hàng sẽ được lưu trữ thông tin về tên, số thẻ và PIN code (gồm 4 ký số) dùng để nhận dạng khách hàng, khách hàng có thể gởi và rút tiền từ tài khoản của mình tại máy ATM.

Một khách hàng phải có một tài khoản tại ngân hàng. Với tài khoản này, khách hàng có thể thực hiện các giao dịch được cung cấp bởi máy ATM của ngân hàng. Khi khách hàng đến máy ATM để sử dụng, khách hàng sẽ được yêu cầu đưa thẻ vào máy, hoặc nhập vào số thẻ và mã PIN kiểm tra. Sau khi kiểm tra thành công, khách hàng có thể thực hiện một số giao dịch trên máy như sau:

- Rút tiền: khách hàng nhập vào số tiền cần rút. Nếu số tiền dư trong tài khoản tiền gởi nhỏ hơn số tiền rút, hệ thống tự động tạo thêm một giao dịch rút tiền từ tài khoản tiết kiệm. Nếu số dư trong tài khoản vẫn không đủ hệ thống sẽ thông báo cho khách hàng và kết thúc giao dịch.
- Gửi tiền: khách hàng có thể thực hiện việc gởi tiền vào tài khoản tiền gởi hoặc tiết kiệm.
- Xem thông tin tài khoản: khách hàng có thể chọn xem thông tin về tài khoản của mình sau khi đăng nhập vào hệ thống.

Khách hàng cũng có thể huỷ bỏ thực hiện một dịch vụ bằng việc chọn "huỷ bỏ" hoặc "đóng" từ giao diện máy.

Yêu cầu

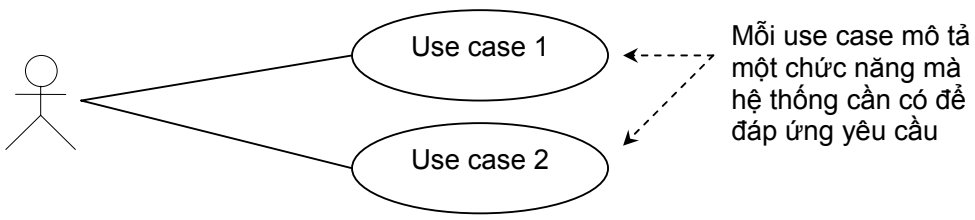
Yêu cầu là: "một điều kiện, hoặc khả năng mà hệ thống phải đáp ứng". Yêu cầu có thể phân thành hai loại lớn là yêu cầu chức năng (functional requirement) và yêu cầu phi chức năng (nonfunctional requirement):

Yêu cầu chức năng

Khi mô tả về một hệ thống, chúng ta nghĩ ngay đến hệ thống sẽ có những gì để thực hiện trên quan diểm người sử dụng. Những việc thực hiện này được xem như là các hành động mà hệ thống phải thi hành và được mô tả như là chức năng, hành vi, và yêu cầu.

Yêu cầu chức năng được dùng để diễn đạt hành vi của một hệ thống bằng việc xác định tất cả điều kiện đầu vào và đầu ra để đạt được một kết quả mong muốn.

Việc biểu diễn yêu cầu chức năng thường thông qua các sơ đồ. Trong UML chúng ta có thể dùng sơ đồ use case, sơ đồ hoạt động, sơ đồ tương tác. Ví dụ, trong một sơ đồ use case. Mỗi use case dùng để biểu diễn một chức năng của hệ thống cần có để cung cấp tới một đối tượng tác nhân.



Yêu cầu phi chức năng

Là các đặc điểm chất lượng của chức năng mà hệ thống cần đáp ứng nhằm thoả mãn nhu cầu người sử dụng. Các đặc điểm chất lượng này được gọi là các yêu cầu phi chức năng. Chúng ta phân loại yêu cầu phi chức năng như sau:

- Sự tiện lợi (usability): là các yêu cầu về yêu tố thẩm mỹ con người, tính dễ học, dễ sử dụng và sự nhất quán của giao diện, tài liệu sử dụng và các tài nguyên huấn luyện.
- Sự tin cậy (realibility): là các yêu cầu về tần suất và giới hạn về hỏng hóc, khả năng phục hồi, khả năng dự đoán và độ chính xác.
- Hiệu năng (performance): là các điều kiện áp đặt lên các yêu cầu chức năng. Ví dụ: tỉ lệ giao tác thực hiện, tốc độ thực hiện, tính sẵn sàng, độ chính xác, thời gian đáp ứng, thời gian phục hồi, dung lượng bộ nhớ sử dụng cho một hoạt động thi hành bởi hệ thống.
- Khả năng chịu đựng (supportability): là các yêu cầu về độ bền, khá năng duy trì, và các yêu cầu khác về chất lượng đòi hỏi hệ thống phải được cập nhật sau thời điểm triển khai.

Phân loại yêu cầu

Với cách tiếp cận truyền thống, các yêu cầu được xem như là các đặc tả văn bản tương ứng với một trong hai loại trên, được diễn đạt qua hình thức:" Hệ thống sẽ …". Tuy nhiên, để quản lý đầy đủ yêu cầu một cách có hiệu quả, các yêu cầu phải được mô tả dựa trên sự hiểu biết của người dùng và các đối tượng có liên quan. Sự hiểu biết này cung cấp cho nhóm phát triển lý do "tại sao?" cũng như "cái gì?" của hệ thống sẽ được phát triển. Vì hệ thống sẽ liên quan đến nhiều loại đối tượng khác nhau do đó, yêu cầu của hệ thống cũng có thể được phân loại theo nhiều cấp khác nhau:

Nhu cầu (need):

Mô tả các yêu cầu ở mức cao thường là các đối tượng có liên quan đến dự án như là: người đầu tư, người hưởng lợi từ dự án, người dùng cuối, cũng như người mua, người thầu, người phát triển, người quản lý,… hoặc những đối tượng khác mà nhu cầu của họ hệ thống phải đáp ứng. Thu thập các nhu cầu này chúng ta phải khảo sát thông qua các phương pháp khảo sát như đã đề cập bên trên. Các nhu cầu được thu thập thông thường mô tả ở mức cao, không rõ ràng, nhọc nhằn và thường bắt đầu như là một "nhu cầu" hoặc "mong muốn".

Ví dụ các nhu cầu có thể là:

"Tôi cần gia tăng khả năng sản xuất",

"Tôi có nhu cầu mở rộng khả năng đáp ứng đơn hàng",

"Tôi có nhu cầu cải tiến hiệu năng hoạt động của hệ thống"

"Tối muốn mở rộng việc khai thác số liệu của khách hàng"

…

Các nhu cầu này được xem như là một tập hợp rất quan trọng giúp chúng ta hiểu về các mong muốn thực sự của các đối tượng liên quan ở mức cao và nó sẽ cung cấp các đầu vào then chốt tới các yêu cầu chi tiết của hệ thống giúp chúng ta xác định các lý do và nội dung hành vi hệ thống.

Đặc điểm hệ thống (feature)

Trong quá trình khảo sát hệ thống, nhu cầu và yêu cầu thường đi đôi với nhau. Trong khi nhu cầu là những cái mong muốn mang lại từ hệ thống trên quan điểm còn không rõ ràng thì yêu cầu ngược lại được mô tả mang tính giải pháp cho những nhu cầu đó.

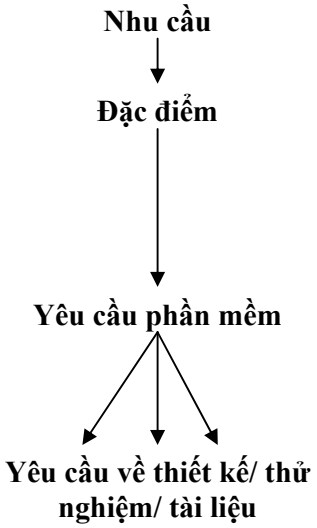
Ví dụ: một nhu cầu "Tôi muốn thông báo đến nhà cung cấp nhanh hơn" thì một yêu cầu là "hệ thống sẽ phát sinh thông báo tự động qua email đến nhà cung cấp"

Các yêu cầu này chính là các biểu thức ở mức cao về hành vi hệ thống – gọi là đặc điểm (feature). Xét về khía cạnh kỹ thuật, đặc điểm được xem như là "một dịch vụ được cung cấp bởi hệ thống để đáp ứng nhu cầu". Như vậy, đặc điểm của hệ thống chính là sự chuyển đổi quan điểm về *cái* gì ("thông báo nhanh hơn") thành *như thế nào* ("email tự động").

Để xác định một đặc điểm chúng ta có thể thêm vào một số thuộc tính khác như là: rũi ro, độ ưu tiên, độ nỗ lực

Yêu cầu phần mềm

Để thích hợp hơn trong quá trình trao đổi với người phát triển về chính xác những gì mà hệ thống sẽ làm. Chúng ta cần một đưa vào thêm một mức đặc tả để chuyển dịch những nhu cầu và đặc điểm thành một đặc tả mà chúng ta có thể thiết kế, cài đặt, thử nghiệm. Các đặc tả này gọi là yêu cầu phần mềm và có thể tiếp cận theo hai loại: yêu cầu chức năng và yêu cầu phi chức năng.



Câu hỏi và bài tập

Câu hỏi

1. Mục đích của khảo sát yêu cầu là gì?
2. Cần tiếp cận những đối tượng nào trong quá trình khảo sát?
3. Cần tu thập các nội dung gì trong quá trình khảo sát?
4. Các phương pháp khảo sát yêu cầu?
5. Ưu và khuyết điểm của phương pháp phỏng vấn?
6. Ưu và khuyết điểm của phượng pháp dùng bảng câu hỏi?
7. Yêu cầu là gì? Như thế nào là yêu cầu chức năng, phi chức năng?
8. Yêu cầu được phân thành bao nhiêu loại? Ý nghĩa của từng loại?

Chương 5 MÔ HÌNH HOÁ USE CASE

Mục tiêu

Nội dung của chương này cung cấp cho sinh viên:

- Hiểu ý nghĩa của việc sử dụng sơ đồ use case trong biểu diễn yêu cầu hệ thống
- Xác định được các tác nhân và mối quan hệ giữa các tác nhân của một hệ thống phần mềm
- Xác định được các use case biểu diễn chức năng phần mềm hệ thống và mối quan hệ giữa tác nhân cà use case nhằm xây dựng sơ đồ use case mô tả yêu cầu phần mềm hệ thống
- Tinh chế sơ đồ use case nhằm làm gia tăng tính diễn đạt, tính tái sử dụng qua việc sử dụng các liên kết <<extend>>, <<include>>

Giới thiệu

Trong giai đoạn phân tích, kết quả của quá trình khảo sát yêu cầu phản ánh quá trình làm việc của người phát triển với người sử dụng. Các kết quả này phải nhắm đến yếu tố của người dùng. Có nghĩa là người phát triển trước tiên phải diễn đạt bức tranh của hệ thống tương lai theo cách nhìn của người sử dụng. Điều này sẽ giúp cho người dùng có thể thấy được hệ thống sẽ làm thoã mãn các yêu cầu như thế nào và đó chính là chìa khoá đầu vào cho việc phát triển hệ thống trong các giai đoạn về sau. Một công cụ giúp diễn đạt điều này chính là mô hình use case.

Jacobson và cộng sự của ông (1992) là những người tiên phong trong việc sử dụng mô hình use case để phân tích yêu cầu hệ thống. Bởi vì mô hình use case đặt trọng tâm để biểu diễn hệ thống hiện tại làm gì, hệ thống mới sẽ làm gì và môi trường của nó. Nó giúp cho người phát triển có thể hiểu rõ về yêu cầu chức năng hệ thống mà không quan tâm đến chức năng này được cài đặt như thế nào.

Để hiểu yêu cầu của hệ thống, chúng ta phải tìm ra người dùng sẽ sử dụng hệ thống như thế nào. Do đó, từ quan điểm một người dùng, chúng ta phát hiện các tình huống sử dụng khác nhau của người dùng, các tình huống này được thiết lập bởi các use case. Tổng hợp các use case và tác nhân cùng với quan hệ giữa chúng sẽ cho ta một mô hình use case mô tả yêu cầu của hệ thống.

Trong chương 6, quá trình mô hình hoá nghiệp vụ được áp dụng đối với các hệ thống nghiệp vụ và kết quả của nó sẽ cung cấp sơ đồ use case từ việc thống nhất các yêu cầu hệ thống phần mềm để tự động hoá hoạt động của hệ thống nghiệp vụ đó. Tuy nhiên, trong những hệ thống mà không có hoạt động nghiệp vụ (ví dụ: hệ thống nhúng), hoặc các nghiệp vụ của hệ thống không quá phức tạp hoặc không quan tâm để mô hình hoá nghiệp vụ thì việc xây dựng mô hình use case phần mềm sẽ là bước tiếp cận mô hình hoá đầu tiên về hệ thống. Một tiến trình xây dựng sơ đồ use case bao gồm các bước sau:

- Xác định tác nhân hệ thống
    - Ai đang sử dụng hệ thống?
    - Hoặc trong trường hợp phát triển mới thì ai sẽ sử dụng hệ thống?
- Phát triển use case
    - Người dùng (tác nhân) đang làm gì với hệ thống?
    - Hoặc trong trường hợp hệ thống mới thì người dùng sẽ làm gì với hệ thống?
- Xây dựng sơ đồ use case

- Xác định mối quan hệ giữa tác nhân – use case
- Xác định mối quan hệ giữa các use case

- Phân chia sơ đồ use case thành các gói (package)

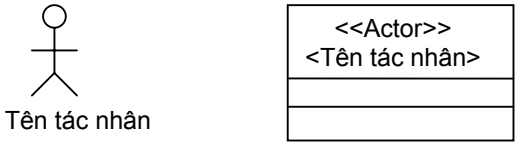
Xác định tác nhân

Tác nhân (actor)

Ý nghĩa: một tác nhân là một đối tượng bên ngoài hệ thống giao tiếp với hệ thống theo một trong những hình thức sau:

- Tương tác, trao đổi thông tin với hệ thống hoặc sử dụng chức năng hệ thống
- Cung cấp đầu vào hoặc nhận các đầu ra từ hệ thống
- Không điều khiển hoạt động của hệ thống

Ký hiệu

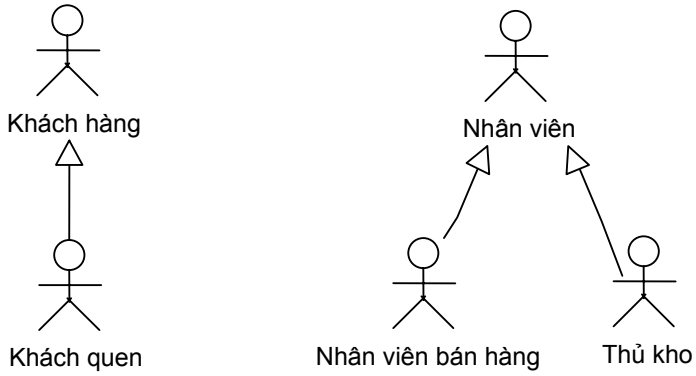


Tên tác nhân: tên tác nhân là một **danh từ**

Quan hệ giữa các tác nhân:

Là quan hệ tổng quát hóa và chuyên biết hoá

Ví dụ:

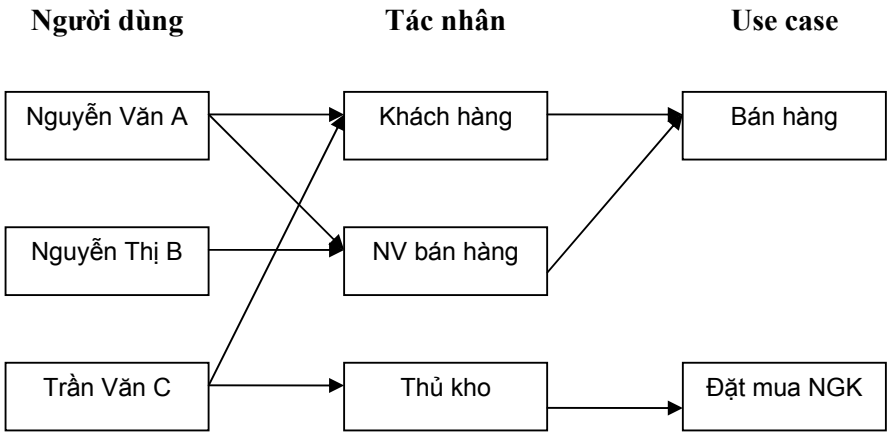


Xác định tác nhân

Xác định tác nhân cũng được xem có tầm quan trọng như xác định class, use case, liên kết,….

Khi xác định người sử dụng phần mềm hệ thống, chúng ta đừng quan trọng vấn đề quan sát người nào đang sử dụng hệ thống mà chúng ta nên xác định xem vai trò chịu trách nhiệm trong việc sử dụng hệ thống. Nghĩa là tác động lên hệ thống theo nghĩa cung cấp thông tin cho hệ thống hoặc nhận kết quả xử lý từ hệ thống.

Tác nhân được hiểu là một vai trò tham gia vào hệ thống không giống như một con người cụ thể hoặc một công việc. Một đối tượng có thể tham gia vào một hoặc nhiều vai trò



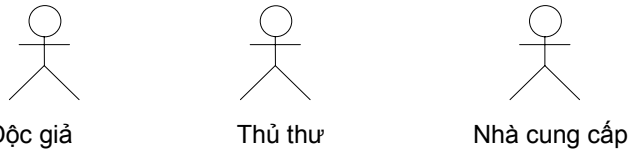
Qua quá trình khảo sát và phân tích tài liệu hệ thống, chúng ta có thể nhận ra các tác nhân thông qua các câu hỏi sau:

- Ai đang sử dụng hệ thống? Hoặc ai được tác động bởi hệ thống? Hoặc nhóm đối tượng nào cần hệ thống trợ giúp để làm công việc? (tác nhân chính)
- Ai tác động tới hệ thống? Những nhóm đối tượng nào hệ thống cần để thực hiện hoạt động của nó (hoạt động gồm chức năng chính và chức năng phụ, như là chức năng quản trị)?
- Những phần cứng hoặc hệ thống bên ngoài nào sử dụng hệ thống?

Ví dụ: trong hoạt động của máy ATM của một ngân hàng, các tác nhân được xác định là:

Trong đó, các tác nhân Khách hàng, Nhân viên ngân hàng là các tác nhân chính (primary actor) của hệ thống ATM. Bởi vì khách hàng là mục tiêu mà hệ thống tượng tác; Nhân viên ngân hàng sử dụng hệ thống để trợ giúp công việc. Trong khi đó, Nhân viên vận hành là tác nhân phụ (secondary actor) bởi vì tác nhân này đảm nhận những chức năng phụ mà hệ thống cần có để thực hiện hoạt động của nó.

Hoặc trong một thư viện của trường đại học, các tác nhân của hệ thống phần mềm quản lý thư viên gồm:



Xác định use case

Use case

Một Use case được xem như một chức năng hệ thống từ quan điểm người dùng, như vậy tập hợp tất cả use case biểu diễn bộ mặt của hệ thống bao gồm các chức năng cần có để cung cấp cho các đối tượng tương tác làm việc với hệ thống. Như vậy, use case dùng để mô tả yêu cầu

của hệ thống mới về mặt chức năng, mỗi chức năng sẽ được biểu diễn như một hoặc nhiều use case.

Ví dụ: hệ thống cửa hàng NGK ta có một vài use case

Bán hàng, quản trị tồn kho,…

Ký hiệu

Xác định use case

Chúng ta bắt đầu từ tập các tác nhân đã xác định trong bước đầu tiên. Ứng với mỗi tác nhân:

- Tìm các nhiệm vụ và chức năng mà tác nhân sẽ thi hành hoặc hệ thống cần tác nhân để thi hành và mô hình hoá nó như là use case. Use case sẽ đại diện một dòng sự kiện dẫn tới một mục tiêu rõ ràng (hoặc trong một vài trường hợp, dẫn tới một vài mục tiêu riêng biệt có thể là các phương án thay thế cho tác nhân hoặc cho hệ thống so với dòng sự kiện chính)
- Đặt tên cho use case: tên use case nên đặt nhằm phản ánh một mô tả tổng quan về chức năng của use case. Tên nên dễn đạt những gì xảy ra khi một thể hiện của use case được thi hành. Một hình thức đặt tên use case phổ biến là : động từ (do) + danh từ (what).
- Mô tả use case một cách ngắn gọn bằng việc áp dụng các thuật ngữ gần gũi với người sử dụng. Điều này sẽ làm cho mô tả use case ít mơ hồ.

Ví dụ: trong hệ thống ATM

Tác nhân Khách hàng sẽ sử dụng hệ thống qua các chức năng:

- Gửi tiền
- Rút tiền
- Truy vấn thông tin về tài khoản

Tác nhân Nhân viên vận hành sẽ sử dụng các chức năng

- Khởi động hệ thống
- Đóng hệ thống

Gửi tiền: khách hàng đăng nhập vào hệ thống và yêu cầu gửi tiền vào tài khoản. Khách hàng sẽ xác định tài khoản và số tiền gửi, hệ thống sẽ tạo một giao tác gửi tiền và lưu vào hệ thống. Các bước như sau:

- Yêu cầu xác định tài khoản
- Hệ thống hỏi số tiền gửi

- Nhập vào số tiền gửi
- Khách hàng đưa tiền vào máy ATM

Rút tiền: khách hàng đăng nhập hệ thống và yêu cầu rút tiền từ tài khoản. Khách hàng xác định tài khoản và lượng tiền rút. Sau khi kiểm tra số dư tài khoản còn đủ, hệ thống sẽ tạo một giao tác rút tiền và lưu vào hệ thống. Các bước như sau:

- Yêu cầu xác định tài khoản
- Yêu cầu xác định số tiền cần rút
- Nhập số tiền rút
- Kiểm tra số dư tài khoản và số tiền hiện có ở máy có đủ không ?
- Chuyển tiền ra ngoài

Truy vấn thông tin tài khoản: khách hàng đăng nhập vào hệ thống và yêu cầu xem thông tin về các giao dịch của tài khoản. Hệ thống hiển thị các thông tin về các giao tác đã tạo lên màn hình cho khách hàng.

Khởi động hệ thống: hệ thống được khởi động khi nhân viên vận hành bật công tắc của máy. Nhân viên vận hành sẽ được yêu cầu nhập vào số tiền hiện hành của máy nằm trong két đựng tiền. Sau đó, hệ thống sẽ thiết lập một kết nối tới ngân hàng và các dịch vụ của máy ATM bắt đầu vận hành.

Đóng hệ thống: hệ thống được đóng lại khi nhân viên vận hành đảm bảo rằng không có khách hàng nào đang sử dụng máy. Khi đó, nhân viên vận hành sẽ lấy các bao tiền gửi ra, bổ sung lượng tiền, giấy,…

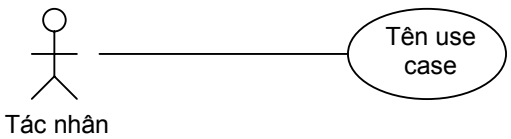
Trong hệ thống quản lý thư viện, các use case được xác định như sau:

Xác định mối quan hệ

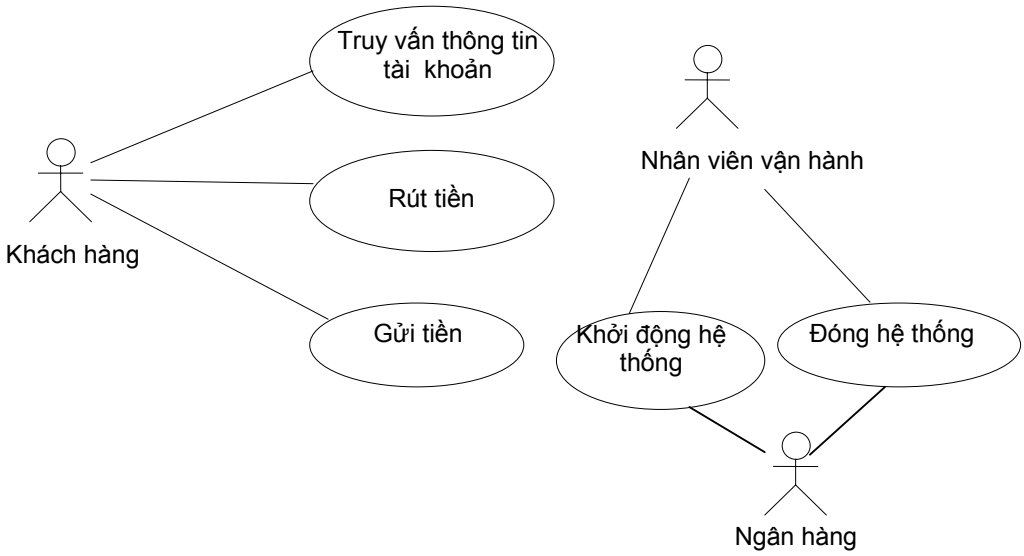
Quan hệ tác nhân – use case

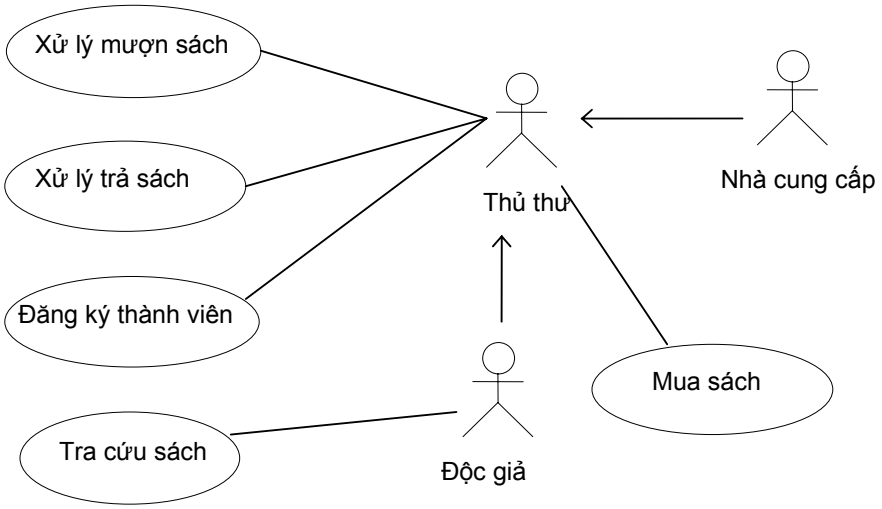
Quan hệ này cho biết tác nhân sẽ tương tác với use case. Một use case luôn luôn được khởi tạo bởi một tác nhân và có thể tương tác với nhiều tác nhân.

Ký hiệu



Ví dụ:





Mối quan hệ giữa các use case

Việc mô tả use case có thể sẽ khó hiểu nếu use case này chứa đựng nhiều dòng phụ hoặc dòng ngoại lệ chỉ xử lý cho những sự kiện trong những điều kiện đặc biệt. Để làm đơn giản mô tả này chúng ta sử dụng thêm các mối kết hợp <<extend>> và <<include>>.

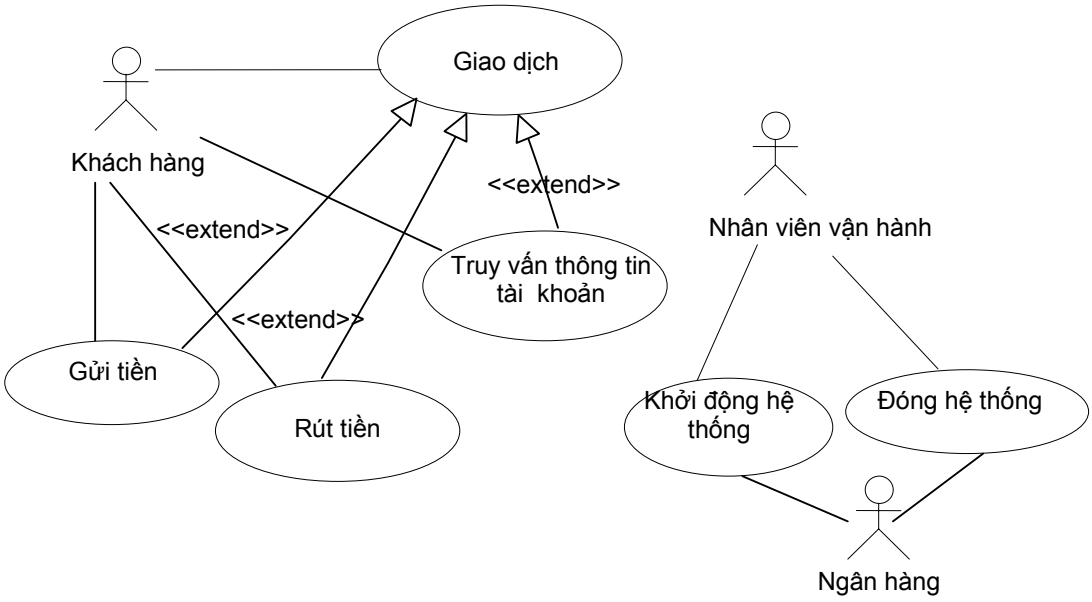
Liên kết mở rộng (<<extend>>): được dùng khi chúng ta có một use case tương tự như use case khác nhưng có nhiều hơn một vài xử lý đặc biệt. Giống như liên kết tổng quát - chuyên biệt, trong đó, use case chuyên biệt là một mở rộng của use case tổng quát bằng việc đưa thêm vào các hoạt động hoặc ngữ nghĩa mới vào use case tổng quát, hoặc bỏ qua hoạt động của use case tổng quát.

Ví dụ: giả sử Đăng nhập là một use case cơ bản. Use case này sẽ đại diện cho tất cả những gì được xem là thực hiện đăng nhập một cách xuyên suốt.Tuy nhiên, nhiều vấn đề có thể tác động đến dòng sự kiện chính. Ví dụ, mã số PIN không hợp lệ, hoặc thẻ không đọc được do bị hư,…. Do đó, chúng ta không phải luôn luôn thi hành các hoạt động thường xuyên của một use case được cho và như vậy, cần thiết tạo ra các use case mới để giải quyết những tình huống mới. Tất nhiên, chúng ta có thể đưa vào use case cơ bản các nội dung xử lý đặc biệt

đó. Tuy nhiên, điều này có thể dẫn đến sự phức tạp với nhiều luận lý riêng biệt và sẽ làm giảm vai trò của dòng chính.

Để giải quyết vấn đề này chúng ta có thể sử dụng quan hệ <<extend>>. Ở đây chúng ta gom các xử lý cơ bản hoặc bình thường vào trong một use case (cơ bản). Các xử lý đặc biệt vào những use case (chuyên biệt) khác. Rồi tạo một liên kết <<extend>> giữa use case cơ bản tới các use case chuyên biệt để khai báo rằng: ngoài xử lý dòng chính (cơ bản), use case cơ bản có mở rộng đến các tình huống xử lý đặc biệt được giải quyết trong các use case chuyên biệt.

Tạo một use case tổng quát có tên là Giao dịch của các use case Rút tiền, Gửi tiền và Truy vấn thông tin tài khoản. Tạo các liên kết <<extend>> từ use case Giao dịch đến các use case này. Như vậy, một rút tiền, hoặc gửi tiền, hoặc truy vấn thông tin tài khoản là một loại giao dịch mà khách hàng có thể sử dụng trên máy ATM. Có nghĩa rằng, các xử lý trong use case Giao dịch sẽ cung cấp một dòng chung và khi khách hàng chọn một loại giao dịch đặc biệt nào đó thì use case này sẽ mở rộng việc giải quyết thông qua các use case chuyên biệt.

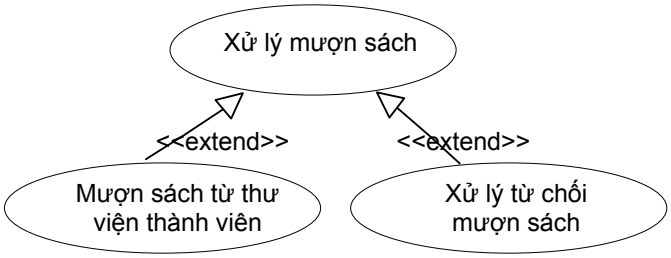


Giao dịch: khách hàng tương tác với hệ thống bắt đầu bằng việc đăng nhập hệ thống. Sau khi đăng nhập, khách hàng có thể thực hiện các giao dịch. Sau đây là các bước:

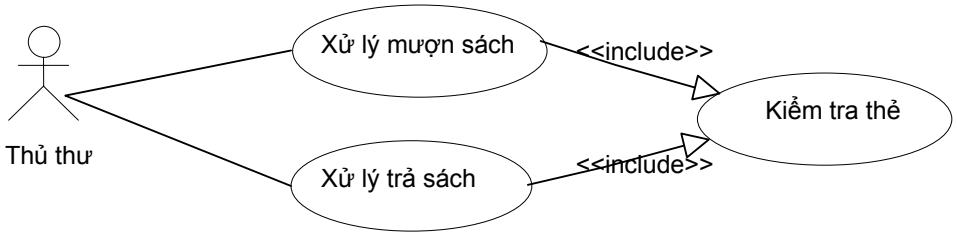
- Đưa thẻ vào máy
- Thực hiện đăng nhập
- Yêu cầu loại giao dịch
- Nhập loại giao dịch
- Thực hiện giao dịch
- Đẩy thẻ ra

- Yêu cầu lấy thẻ
- Lấy thẻ

Trong hệ thống quản lý thư viện, use case Mượn sách ngoài dòng hoạt động chính còn có các dòng phụ. Dòng phụ này sẽ được kích hoạt để giải quyết vấn đề khi một độc giả đến mượn tài liệu nhưng không có trong thư viện và thư viện sẽ mượn tài liệu đó từ những thư viện khác có liên kết. Hoặc do đọc giả không thoả các điều kiện để được mượn (mượn sách quá hạn chưa trả của lần mượn trước). Do đó, chúng ta tách dòng phụ này và use case "Mượn sách từ thư viện thành viên" và " "và tạo một liên kết <<extend>> từ use case này đến use case Xử lý mượn sách.

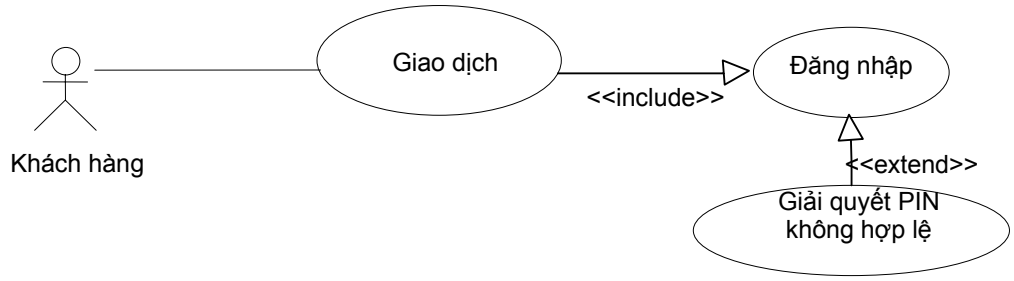


Liên kết sử dụng (<<include>>): được thành lập khi chúng ta có các use case mà tìm thấy một vài use case có những dòng hoạt động chung, và để tránh mô tả dòng hoạt động chung đó lặp lại trên những use case này, chúng ta có thể tách những dòng hoạt động chung đó ra thành một use case. Use case mới này có thể sử dụng bởi những use case khác. Quan hệ giữa những use case với use case được trích ra này gọi là quan hệ <<include>>. Quan hệ sử dụng giúp chúng ta tránh sự trùng lắp bằng cách cho phép một use case có thể được chia sẽ.



Trong ví dụ trên, use case *mượn sách* và *trả sách* đều phải thực hiện công việc kiểm tra thẻ thư viện của đọc giả, do đó chúng ta phát sinh một use case mới là *kiểm tra thẻ* bằng cách trích ra hoạt động kiểm tra thẻ thư viện từ hai use case trên và tạo một liên kết <<include>> tới use case từ hai use case đó tới use case mới. Các use case *Xử lý mượn sách* và *Xử lý trả sách* đều thừa hưởng tất cả hoạt động của use case của use case *kiểm tra thẻ*.

Trong hệ thống ATM, use case Giao dịch sẽ có mối liên kết <<include>> với use case đăng nhập.



Đăng nhập: khách hàng nhập vào mã số PIN gồm bốn ký số. Nếu mã số PIN hợp lệ, tài khoản của khách hàng sẽ sẵn sàng cho các giao dịch. Các bước như sau:

- Yêu cầu mật khẩu (mã số PIN)
- Nhập mật khẩu
- Kiểm tra mật khẩu

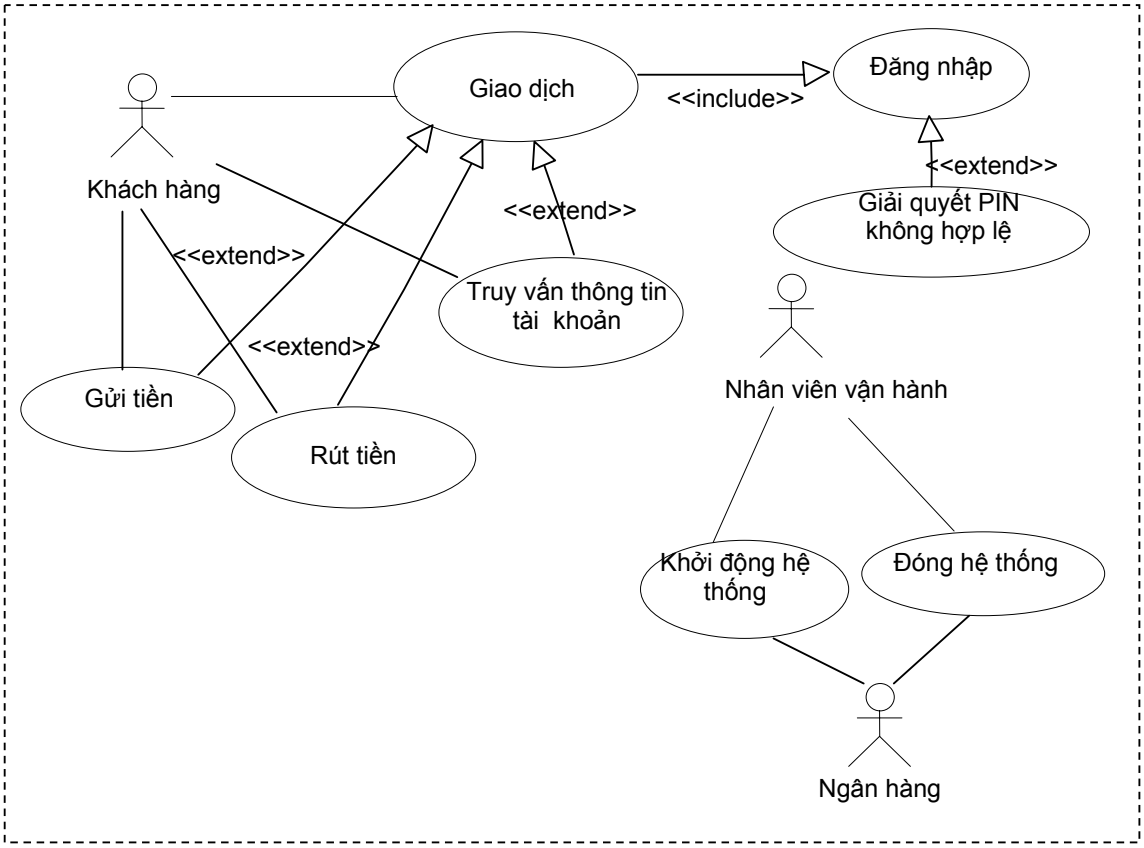
Giải quyết PIN không hợp lệ: nếu mã số PIN không hợp lệ, hệ thống sẽ hiển thị một thông báo tới khách hàng.

Sự giống nhau giữa liên kết <<extend>> và liên kết <<include>> là tất cả đều được xem như là một loại kế thừa. Khi chúng ta muốn chia sẽ một số hoạt động chung trong nhiều use case, dùng liên kết <<include>> bằng cách trích các hoạt động chia sẽ đó thành một use case mới. Khi chúng ta muốn thêm vào một ít khác biệt cho một use case để mô tả một tình huống đặc biệt trong một tình huống chung, chúng ta sẽ tạo một use case mới có liên kết <<extend>> với use case chung đó.

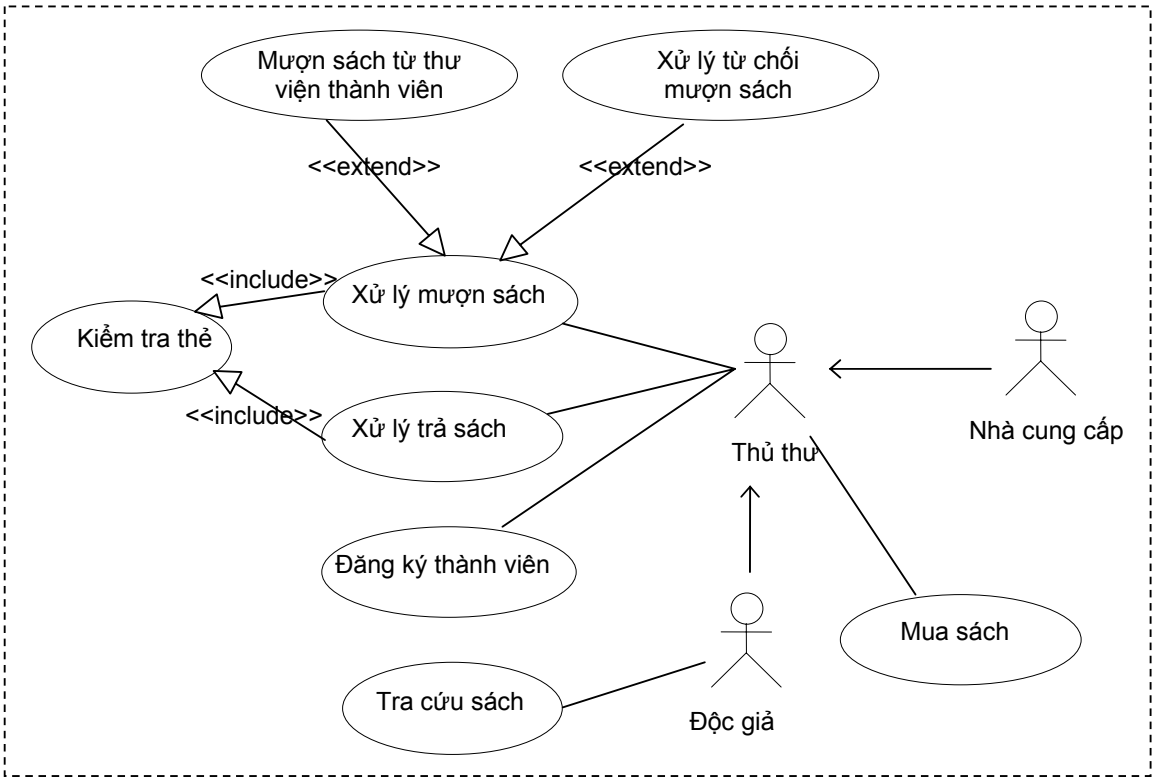
Dựa vào các liên kết được thiết lập cho các use case chúng ta phân use case thành hai loại:

Use case trừu tượng: là use case chưa hoàn hảo nghĩa là không tương tác với bất kỳ một tác nhân nào mà được sử dụng bởi một use case khác. Use case trừu tượng cũng có thể có liên kết <<extend>> hoặc liên kết <<include>> trong những mức độ khác. Ví dụ: các use case *Kiểm tra thẻ, Xử lý từ chối mượn sách,...* là các use case trừu tượng.

Use case cụ thể: là use case có tương tác trực tiếp với một tác nhân. Ví dụ: các use case *Xử lý mượn sách, Xử lý trả sách,* hoặc *Khởi động máy,Đóng máy,....*



Mô hình use case của hệ thống máy ATM

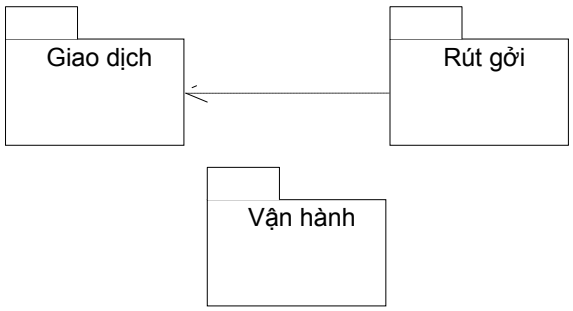


Mô hình use case hệ thống phần mềm quản lý thư viện

Phân chia các use case thành các gói (package)

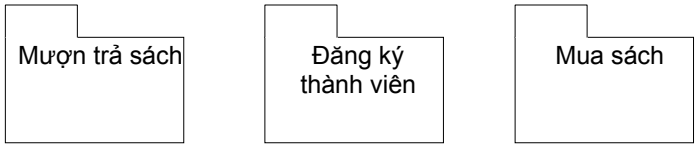
Mỗi use case minh hoạ một kịch bản trong hệ thống. Khi gặp những hệ thống tương đối phức tạp thì chúng ta nên thu hẹp tiêu điểm của các kịch bản trong hệ thống bằng cách phân chia thành các gói. Mỗi gói phản ánh một phạm vi của hệ thống mà chúng ta chỉ muốn quản lý nó khi chúng ta truy cập gói đó.

Ví dụ, có thể chia các se case của hệ thống máy ATM thành ba gói: Giao dịch, Rút gửi và Vận hành



Trong đó, gói Giao dịch gồm các use case: Giao dịch, Đăng nhập, Giải quyết PIN không hợp lệ; gói Rút gởi gồm các use case: Gởi tiền, Rút tiền, Truy vấn thông tin tài khoản; và gói Vận hành gồm các use case: Khởi động hệ thống, Đóng hệ thống

Hệ thống quản lý thư viện được chia thành ba gói như sau: Mượn trả sách, đăng ký thành viên, và Mua sách.



Trong đó, gói Mượn trả sách gồm các use case: Xử lý mượn sách, Xử lý trả sách, Kiểm tra thẻ, Mượn sách từ thư viện thành viên, Xử lý từ chối mượn sách; gói Đăng ký thành viên gồm use case: Đăng ký thành viên; gói mua sách gồm use case: Mua sách.

Câu hỏi và bài tập

Câu hỏi

9. Mô hình use case là gì?

10. Tại sao mô hình hoá use case là hữu dụng trong phân tích hệ thống?

11. Ai có thể là tác nhân?

12. Khi nào chúng ta sử dụng liên kết <<extend>> và <<include>> trong mô hình hoá use case?

13. Xác định tác nhân bằng cách nào?

14. Phân biệt sự khác nhau giữa người dùng và tác nhân?

15. Tai sao nên phân chia mô hình use case thành các gói?

Bài tập

1. Hãy xây dựng sơ đồ use case mô tả một hệ thống diễn đàn trao đổi học tập của khoa Công Nghệ Thông Tin. Hoạt động của diễn đàn được mô tả như sau:

   - Tất cả mọi người khi truy cập vào địa chỉ của diễn đàn đều có thể xem được thông tin nội dung trao đổi của diễn đàn, tin tức trong diễn đàn.
   - Khi một thành viên muốn gởi thông tin lên diễn đàn. Thông tin đó có thể là một chủ đề cần trao đổi, hoặc một ý kiến liên quan đến một chủ đề đã được đưa ra, hoặc là một tin tức. Trước tiên, thành viên phải đăng nhập vào hệ thống với một tên và mã đăng nhập. Sau khi đăng nhập thành công, thành viên có thể sử dụng chức năng soạn thảo của hệ thống để soạn thảo và gởi thông tin lên diễn đàn. Thành viên có thể xem được các thông báo kết quả duyệt tin từ quản trị về những lần gởi trước.
   - Một người dùng khi truy cập thì có thể đăng ký thành viên của diễn đàn. Khi đăng ký, người dùng sẽ phải nhập các thông tin liên quan như: họ tên, tên đăng nhập, mã sinh viên, mã đăng nhập. Việc đăng ký có hiệu lực sau khi quản trị duyệt và chấp nhận.
   - Khi một thành viên đăng nhập vào quyền quản trị (quản trị hệ thống). Hệ thống sẽ hiển thị các các thông tin được gởi bởi các thành viên. Ứng với mỗi thông tin, quản trị có thể chấp nhận hoặc từ chối đăng thông tin lên diễn đàn:
     - Khi thông tin được đánh dấu là chấp nhận, hệ thống sẽ cập nhật lại trạng thái của thông tin và sẽ hiển thị thông tin này tới diễn đàn.
     - Khi thông tin được đánh dấu là từ chối, hệ thống cũng sẽ cập nhật lại trạng thái thông tin và ghi chú về lý do từ chối thông tin.
   - Ngoài ra, người quản trị cũng có thể duyệt và chấp nhận hoặc từ chối một thành viên mới đăng ký. Huỷ một thành viên. Cập nhật thông tin của một thành viên.

2. Phân chia sơ đồ use case của hệ thống trên thành các gói. Mô tả sự liên kết giữa các gói (nếu có) và sơ đồ use case cho từng gói.

Chương 6

MÔ HÌNH HOÁ NGHIỆP VỤ (BUSINESS MODELING)

Mục tiêu

Cung cấp cho người học các kiến thức về:

- Như thế nào là mô hình hoá nghiệp vụ, mục tiêu và quy trình của mô hình hoá nghiệp vụ
- Các hoạt động trong phân tích, thiết kế qui trình nghiệp vụ
- Áp dụng UML vào mô hình hoá nghiệp vụ. Đặc biệt, sử dụng sơ đồ use case biểu diễn nội dung của hệ thống nghiệp vụ trong giai đoạn phân tích. Sử dụng sơ đồ đối tượng trong việc thiết kế nghiệp vụ.
- Xác định các yêu cầu tự động hoá từ hệ việc phân tích và thiết kế thống nghiệp vụ.

Giới thiệu

Mô hình hóa nghiệp vụ là một kỹ thuật để tìm hiểu quy trình nghiệp vụ của một tổ chức. Mô hình nghiệp vụ xác định các quy trình nghiệp vụ nào được hỗ trợ bởi hệ thống. Tóm lại, song song với quá trình khảo sát tìm hiểu về vấn đề hệ thống thì cách tiếp cận nghiệp vụ là phương pháp có hệ thống nhất để nắm bắt các yêu cầu của các ứng dụng nghiệp vụ.

Khi những hệ thống ngày càng phức tạp, việc mô hình hóa trực quan và cách vận dụng các kỹ thuật mô hình hóa ngày càng trở nên quan trọng hơn. Có nhiều nhân tố bổ sung cho sự thành công của một dự án, nhưng việc có một tiêu chuẩn ngôn ngữ mô hình hóa chặt chẽ là nhân tố quan trọng nhất. Một trong những mục đích đầu tiên của mô hình hoá nghiệp vụ là tạo ra các "đối tượng" (mô hình) nhằm để dễ hiểu hơn và để có thể thiết kế những chương trình máy tính bằng cách thông qua hiện tượng thế giới thực như: người, nguyên liệu làm việc và cách thức chúng thực hiện những nhiệm vụ của họ. Như vậy, việc mô hình hóa nghiệp vụ là lập mô hình những tổ chức thế giới thực.

Phạm vi ảnh hưởng của việc mô hình hóa nghiệp vụ có thể biến đổi tuỳ theo nhu cầu và hệ thống nghiệp vụ cụ thể. Có thể đơn giản chúng ta chỉ nhằm vào việc tăng năng suất bằng cách cải tiến những quy trình đã tồn tại, hoặc là đang tạo ra những sự cải tiến có ảnh hưởng lớn bằng cách thay đổi đáng kể những qui trình nghiệp vụ dựa trên sự phân tích kỹ lưỡng các mục tiêu và các khách hàng của tổ chức. Cho dù là bất kỳ trường hợp nào, những hệ thống thông tin hỗ trợ cho hệ thống nghiệp vụ đều bị ảnh hưởng bởi sự cải tiến của hoạt động nghiệp vụ.

Tại sao phải mô hình hoá nghiệp vụ?

Trong quá trình phát triển hệ thống phần mềm, đặc biệt là trong các hệ thống phức tạp, một vấn đề tồn tại rất lớn cho thấy đội ngũ phát triển hệ thống thường hiếm khi có một kiến thức hiểu biết đầy đủ về nghiệp vụ của tổ chức mà chính họ là người xây dựng hệ thống phần mềm thực hiện tự động hoá xử lý thông tin trong môi trường nghiệp vụ đó. Trong khi đó, người sử dụng phần mềm chính là các đối tượng xử lý nghiệp vụ thường hiếm khi am hiểu tường tận về các công nghệ và các kỹ thuật của phần mềm nhằm chọn lựa và áp dụng nó một cách phù hợp và hiệu quả với nhu cầu của mình. Điều này luôn tạo ra một khoảng cách giữa người xây dựng và người sử dụng hệ thống. Khoảng cách này là một trở ngại dẫn đến nhiều sự thất bại hoặc không hiệu quả của nhiều dự án tin học hoá hệ thống. Do đó, làm thế nào để các đối tượng này có thể nắm bắt và thống nhất được tốt nhất về cách giải quyết hệ thống trong quá trình tin học hoá. Mô hình hoá nghiệp vụ là một nỗ lực nhằm đưa ra các cách thức diễn tả những qui trình nghiệp vụ dưới dạng những đối tượng và hành động tương tác giữa chúng.

Nếu không mô hình hóa nghiệp vụ thì ta có thể gặp nhiều rủi ro do những người phát triển không có thông tin đầy đủ về cách thức mà nghiệp vụ được thực hiện. Họ chỉ làm những gì mà họ hiểu rõ, như là thiết kế và tạo ra phần mềm, mà không quan tâm đến những gì nghiệp vụ thực thi. Điều này gây ra một sự lãng phí do trước đó đã xây dựng các qui trình nghiệp vụ tốn kém. Rủi ro do những hệ thống được xây dựng không hỗ trợ các nhu cầu thực sự của tổ chức cũng có thể xảy ra rất cao.

Việc hiểu rõ những qui trình nghiệp vụ là quan trọng để có thể xây dựng những hệ thống đúng. Việc mô hình hóa nghiệp vụ có mục tiêu chính là sự phát triển hệ thống, trong đó công việc thực sự là xác định đúng các yêu cầu hệ thống.

Cơ sở để xây dựng hệ thống là sử dụng những vai trò và trách nhiệm của con người cũng như định nghĩa các công việc được xử lý bởi nghiệp vụ. Điều này được thể hiện trong một mô hình đối tượng nghiệp vụ, mà qua đó có thể thấy các vai trò đối tượng sẽ được làm rõ.

Một khi xác định được các mô hình nghiệp vụ, chúng ta cần phải thiết lập những mối quan hệ giữa các use case hệ thống và những mô hình nghiệp vụ. Điều này sẽ cho phép các nhà phân tích được thông báo khi có những thay đổi ở trong hệ thống.

Tóm lại, mục đích của mô hình hóa nghiệp vụ là:

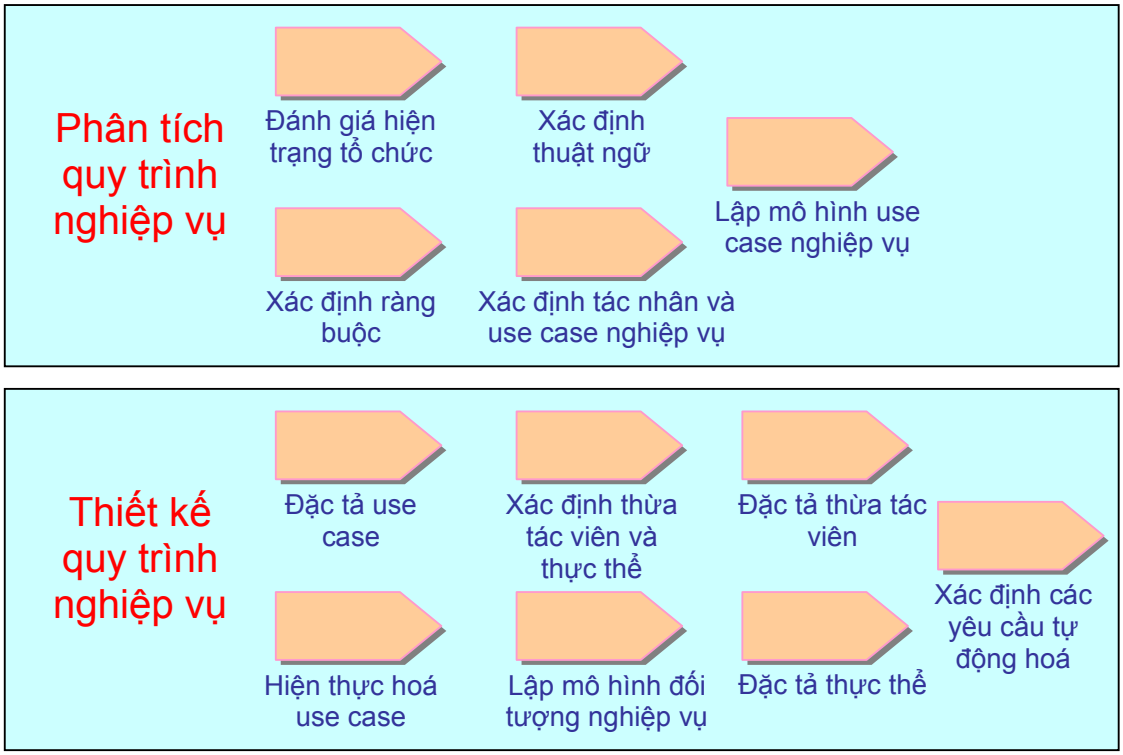
- Hiểu được cấu trúc và các hoạt động của tổ chức đang được hệ thống triển khai.
- Hiểu được các vấn đề hiện tại trong tổ chức và xác định các vấn đề cần cải tiến.
- Bảo đảm rằng các khách hàng, người dùng cuối, và các nhà phát triển có sự hiểu biết chung về tổ chức.
- Thiết lập các yêu cầu tự động hoá hệ thống nhằm hỗ trợ tổ chức.

Để đạt được những mục đích trên, luồng công việc mô hình hóa nghiệp vụ mô tả một bức tranh tổng quát về tổ chức. Từ đó xác định các qui trình (process), các vai trò (role), và các trách nhiệm của tổ chức này trong mô hình use-case nghiệp vụ (business use-case model) và mô hình đối tượng nghiệp vụ (business object model).

Luồng công việc trong mô hình hoá nghiệp vụ

Hệ thống nghiệp vụ là một loại hệ thống, do đó quá trình tiếp cận mô hình hoá cũng tuân theo quy trình chung qua nhiều giai đoạn. Tài liệu này sẽ giới thiệu hai giai đoạn mô hình hoá nghiệp vụ sử dụng UML.

- Phân tích quy trình nghiệp vụ: đây là giai đoạn đầu tiên của mô hình hóa nghiệp vụ giúp cho các nhà quản lý dự án hiểu rõ tình trạng tổ chức hiện tại và hoạt động của tổ chức, nắm bắt yêu cầu của người dùng và khách hàng từ đó phác thảo và giới hạn hệ thống phát triển.
- Thiết kế quy trình nghiệp vụ: đây là giai đoạn đặc tả chi tiết một bộ phận của tổ chức bằng cách mô tả luồng công việc của một hay nhiều nghiệp vụ, xác định các đối tượng làm việc và các thực thể nghiệp vụ trong biểu diễn hiện thực hóa nghiệp vụ và sắp xếp các hành vi của nghiệp vụ đồng thời xác định các trách nhiệm, thao tác, thuộc tính và mối quan hệ giữa các người làm việc và các thực thể trong nghiệp vụ.



Phân tích quy trình nghiệp vụ

Các công việc của quy trình phân tích nghiệp vụ bao gồm:

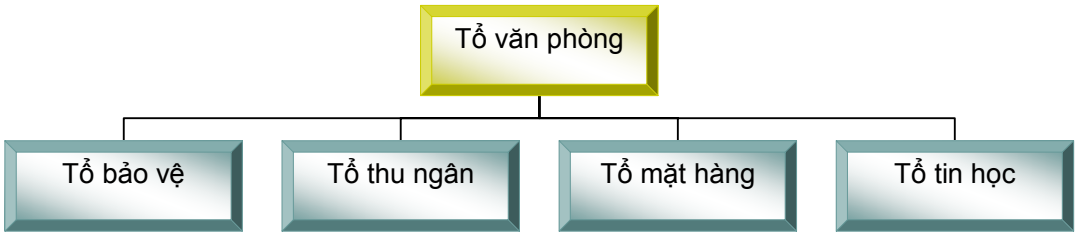
- Đánh giá và nắm bắt thông tin về tổ chức.
- Xác định các đối tượng liên quan (stakeholder) và khách hàng của hệ thống.
- Định nghĩa phạm vi của việc mô hình hóa nghiệp vụ.
- Tán thành những tiềm năng cải tiến và các mục tiêu mới của tổ chức.
- Mô tả những mục tiêu chính của tổ chức.

Nắm bắt thông tin về tổ chức

Để thiết kế hệ thống phù hợp với nhu cầu của khách hàng thì việc hiểu rõ thông tin về cấu trúc tổ chức sẽ được triển khai hệ thống là điều quan trọng. Tất cả các thành viên trong dự án đều cần phải nắm bắt rõ ràng các thông tin này. Chúng ta có thể mô tả ngắn gọn các bộ phận cấu thành tổ chức và mối quan hệ giữa các bộ phận này thông qua các sơ đồ tổ chức và trình bày ngắn gọn các thông tin liên quan.

Để minh hoạ cho nội dung mô hình nghiệp vụ, giáo trình này đưa ra một hệ thống nghiệp vụ về hoạt động của một siêu thị XYZ. Tất cả các ví dụ sẽ được minh hoạ dựa trên việc phân tích các hoạt động nghiệp vụ của siêu thị này.

Sơ đồ tổ chức của siêu thị XYZ



Tổ văn phòng: Gồm 1 Giám Đốc và 2 phó Giám Đốc có nhiệm vụ điều phối toàn bộ hoạt động của siêu thị. Tổ phải nắm được tình hình mua bán, doanh thu của siêu thị để báo cáo lại cho ban giám đốc. Việc báo cáo được thực hiện hàng tháng, hàng quý hoặc cũng có khi báo cáo đột xuất theo yêu cầu.

Tổ bảo vệ: Kiểm tra, bảo vệ an ninh của Siêu Thị, ghi nhận Hàng Hóa đổi lại của khách hàng.

Tổ thu ngân: Thực hiện việc bán hàng và lập hóa đơn cho khách hàng đồng thời ghi nhận lại số hàng hoá bán được của mỗi loại để báo cáo cho tổ quản lý sau mỗi ca làm việc.

Tổ mặt hàng: Nhiệm vụ của tổ là kiểm tra chất lượng hàng hoá và nắm tình trạng hàng hoá của siêu thị, đảm bảo hàng hoá luôn ở trong tình trạng tốt nhất khi đến tay khách hàng. Khi phát hiện hàng hư hỏng phải kịp thời báo ngay cho tổ văn phòng để có biện pháp giải quyết và điều phối hàng. Ngoài ra, thường xuyên thống kê số lượng hàng tồn trên quầy, báo cáo về tổ văn phòng

Tổ tin học: Thực hiện việc nhập liệu, kết xuất các báo cáo cần thiết phục vụ cho tổ Văn Phòng.

Xác định các đối tượng có liên quan và khách hàng

Việc tin học hóa công tác quản lý trong một tổ chức tạo sẽ ra những biến đổi, phần do việc tự động hóa công việc hành chính, phần do kiến thiết lại tổ chức và sự vận hành của hệ thống. Những thay đổi quan trọng phát sinh từ việc thiết kế hệ thống thông tin, chủ yếu tập trung vào việc tin học hóa, nếu không biết thực hiện dần dần sẽ có nguy cơ lớn chạm đến con người trong tổ chức dẫn đến thất bại ngay từ đầu. Bản thân công việc thiết kế hệ thống thông tin đã được thực hiện dưới nhiều góc độ khác nhau, thậm chí kể cả tâm lý, với những đặc thù riêng biệt và có độ phức tạp cao. Chính vì thế, cần phải tìm hiểu những đối tượng có liên quan và khách hàng của hệ thống là ai, đồng thời nắm bắt được nhu cầu của họ.

Nếu đánh giá tình trạng của tổ chức, ta nên xác định những đối tượng có liên quan trong nghiệp vụ. Nhưng khi xác định các mục tiêu của hệ thống thì cần xác định những đối tượng liên quan trong phạm vi dự án và điều đó cũng phụ thuộc vào phạm vi mô hình hóa nghiệp vụ, cũng như những phạm vi nào cần xác định đối với việc mô hình hóa.

Ví dụ: thể hiện thông tin các đối tượng liên quan và người dùng hệ thống qua bảng sau trong hệ thống siêu thị như sau:

Bảng mô tả tóm tắt các đối tượng liên quan:

Tên	Đại diện	Vai trò
Người quản lý	Giám đốc, người quản lý siêu thị	Theo dõi tiến trình phát triển của dự án và theo dõi tình hình hoạt động của siêu thị.
Nhân viên bán hàng	Người nhập các thông tin trong hệ thống.	Chịu trách nhiệm trong khâu bán hàng ở siêu thị, duy trì hoạt động của siêu thị.

Bảng mô tả tóm tắt các người dùng:

Tên	Mô tả	Đối tượng liên quan
Người quản lý	Đáp ứng các nhu cầu quản lý siêu thị như hàng hóa, khách hàng, doanh số.	Người quản lý
Nhân viên bán hàng	Đảm bảo rằng hệ thống sẽ đáp ứng các nhu cầu của công việc bán hàng.	Nhân viên bán hàng
Khách hàng	Đáp ứng nhu cầu tra cứu thông tin về hàng hóa có trong siêu thị.	

Nắm bắt nhu cầu của các đối tượng liên quan

Nhiệm vụ trao cho họ là những công việc thực sự như là xử lý thông tin, chứ không chỉ đơn thuần là thao tác với máy tính và các thiết bị, vì vậy ta không được phép bỏ qua các ý kiến, nhu cầu của họ đối với hệ thống tin học tương lai. Hãy liệt kê danh sách các nhu cầu chính bằng cách điền đủ thông tin vào bảng sau:

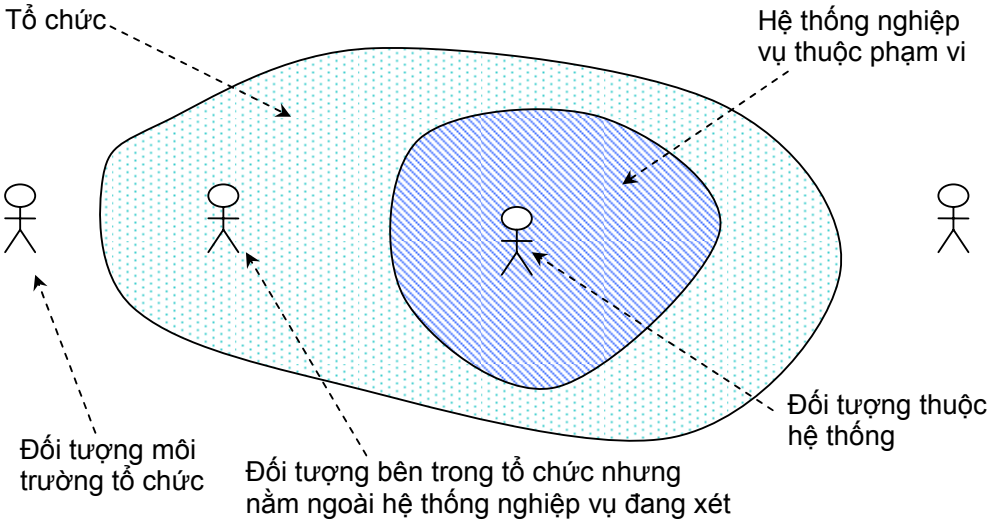
Tên đối tượng liên quan/ khách hàng	Độ ưu tiên	Nhu cầu	Giải pháp hiện hành	Giải pháp đề xuất

Ví dụ:

Tên đối tượng liên quan/ khách hàng	Độ ưu tiên	Nhu cầu	Giải pháp hiện hành	Giải pháp đề xuất
Người quản lý	Cao	Xem các báo cáo thống kê theo các yêu cầu khác nhau	Báo cáo thống kê doanh thu	Hiển thị báo cáo theo nhiều tiêu chí khác nhau, thông tin bố trí dễ nhìn và đơn giản nhưng đầy đủ.

Giới hạn hệ thống phát triển

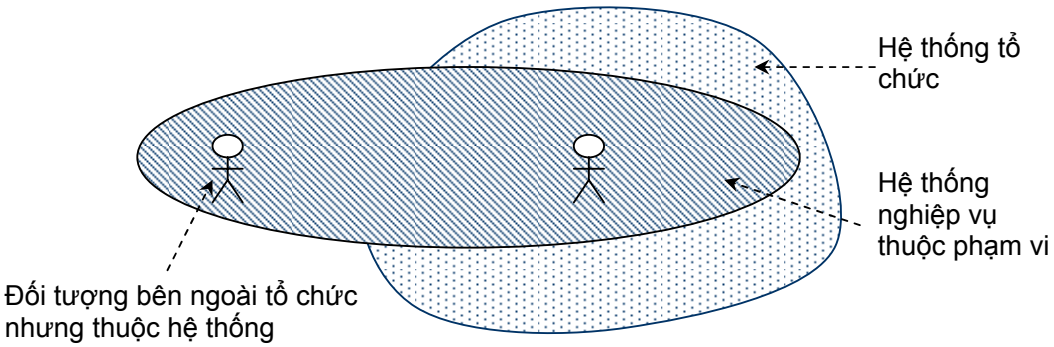
Cần phải đạt được sự thỏa thuận về những thực thể chính nằm ngoài hệ thống với các đối tượng liên quan và các thực thể này với nhau. Trong trường hợp mô hình hóa nghiệp vụ để xác định các yêu cầu cho một hệ thống cụ thể, có thể có những phần trong tổ chức sẽ không bị ảnh hưởng bởi hệ thống này, những phần đó có thể được xem như các thực thể nằm bên ngoài.



Những ranh giới đặt ra cho hệ thống có thể khác rất nhiều so với những gì có thể được xem là ranh giới của tổ chức.

Nếu mục đích là xây dựng một hệ thống mới là để hỗ trợ bán hàng, ta không cần quan tâm đến bất cứ việc gì trong kho hàng, nhưng cần xem kho hàng như là một tác nhân bởi vì chúng ta cần phải làm rõ ranh giới giữa chúng. Trong ví dụ này, các thực thể bên trong tổ chức được xem như là bên ngoài hệ thống đang xét và được mô hình hóa thành tác nhân nghiệp vụ.

Nếu mục tiêu xây dựng hệ thống là nhằm nâng cao khả năng trao đổi thông tin với các đối tác hay các nhà cung cấp (ứng dụng business-to-business) như quản lý đặt hàng thì các đối tác hay các nhà cung cấp này của tổ chức được mô hình hóa cần phải được quan tâm. Trong trường hợp này, các thực thể bên ngoài tổ chức sẽ nằm trong tổ chức. Điều này chỉ xảy ra khi sự cộng tác giữa các bên ảnh hưởng sâu sắc đến phương thức hoạt động của nhau. Nhưng nếu sự ảnh hưởng này không quá lớn hay nghiêm trọng thì các đối tác được xem như là các thực thể bên ngoài và được mô hình hóa thành tác nhân nghiệp vụ.



Nếu mục đích là xây dựng các ứng dụng tổng quát, tùy biến (như là ứng dụng kế toán tài chính, các ứng dụng đóng gói) thì chúng ta cần trình bày cách thức khách hàng sẽ sử dụng sản phẩm cuối như thế nào và nó là một thực thể trừu tượng.

Xác định và trình bày các vấn đề của hệ thống

Trong quá trình khảo sát hệ thống, có thể thu thập rất nhiều nhu cầu cần thay đổi của khách hàng. Đây được xem là các vấn đề của khách hàng cần chúng ta giải quyết trong hệ thống tương lai. Vì thế ta cần phải hiểu và trình bày rõ ràng các vấn đề này để mọi thành viên trong dự án nắm bắt tốt. Có thể áp dụng mẫu như sau:

Vấn đề	mô tả vấn đề
Đối tượng chịu tác động	các nhân vật bị ảnh hưởng bởi vấn đề
Ảnh hưởng của vấn đề	tác động ảnh hưởng của vấn đề
Một giải pháp thành công	liệt kê một vài lợi ích của một giải pháp thành công

Ví dụ:

Vấn đề	Cơ sở dữ liệu của các khách hàng thân thiết được lưu trữ ở nhiều nơi và không có sự đồng bộ .
Đối tượng chịu tác động	Khách hàng, người quản lý
Ảnh hưởng của vấn đề	Dịch vụ khách hàng thân thiết chỉ thiết lập được ở từng siêu thị. Điều này là bất hợp lý, làm rắc rối trong việc nâng cao dịch vụ khách hàng, làm giảm khả năng cạnh tranh của siêu thị.
Một giải pháp thành công	Nhân viên có thể sử dụng chung một tài khoản (account) cấp cho mỗi khách hàng được dùng ở tất cả siêu thị. Nâng cao khả năng chăm sóc khách hàng của siêu thị tốt hơn từ đó thu hút được khách hàng nhiều hơn, tăng doanh thu của siêu thị. Giúp người quản lý có thể làm tốt công tác quản lý khách hàng, theo dõi tình hình phục vụ khách hàng một cách dễ dàng.

Xác định những lãnh vực cần ưu tiên

Cần phải thảo luận và đạt được sự nhất trí về những lãnh vực cần được ưu tiên trong mô hình hóa nghiệp vụ. Sự thảo luận này có thể theo nhiều hướng khác nhau, tùy vào mục tiêu của mô hình hóa nghiệp vụ:

- Nếu mục đích mô hình hóa nghiệp vụ là tạo một mô hình để thực hiện sự cải tiến đơn giản, thì chỉ cần mô tả nghiệp vụ hiện tại. Khi đó, những lĩnh vực nào cần cải tiến phải xác định rõ.
- Nếu mục đích là tạo một nghiệp vụ mới hay thay đổi hoàn toàn nghiệp vụ hiện tại, thì phạm vi mô hình hóa sẽ lớn hơn. Lúc này, công việc tái cấu trúc các use case nghiệp vụ của một nghiệp vụ đã tồn tại hay thêm các use case nghiệp vụ mới - để tái thiết kế nghiệp vụ (business reengineering) hay thiết kế mới nghiệp vụ (business creation) là cần thiết.

Ví dụ: Trong Hệ thống quản lý nghiệp vụ bán hàng tại siêu thị, việc mô hình hóa nghiệp vụ nhằm mục đích để cải tiến nghiệp vụ nên chúng ta chỉ cần xác định những nghiệp vụ cần cải tiến.

Để cải tiến nghiệp vụ, một số câu hỏi được đặt ra như sau:

- Cấu trúc của tổ chức có thể được cải tiến không? Đó là cách thức tổ chức nhân viên làm việc trong các qui trình nghiệp vụ. Ta có thể xây dựng các nhóm nhân viên có nhiều năng lực khác nhau để thực hiện những công việc chính, giảm số lượng người giữ vai trò một công việc dẫn đến giảm chi phí, giảm các sai sót và để cho các nhân viên có nhiều trách nhiệm hơn, khi đó họ không phải chờ người khác quyết định.
- Có công việc nào không cần thiết không? Xác định những công việc không cần thiết trong tổ chức như: viết báo cáo mà không có ai đọc, lưu trữ những thông tin không bao giờ được sử dụng ....
- Có công việc nào giống hoặc tương tự nhau được thực hiện ở những nơi khác nhau không? Như công việc được làm lại, do người ta không tin tưởng vào kết quả hoặc không biết trước đó đã làm gì hay các kết quả được kiểm tra và chấp thuận nhiều lần.
- Có vấn đề nào về thời gian và chi phí không? Thời gian thực hiện có thể là một vấn đề thậm chí nếu mỗi thứ đều hoạt động tốt. Để xác định công việc nào có thời gian quá cấp bách, hãy phân tích mỗi use case nghiệp vụ sử dụng thời gian. Xác định mối quan hệ giữa thời gian sản xuất, thời gian chờ, và thời gian truyền.

Kết quả chính của hoạt động này là một bản mô tả tầm nhìn nghiệp vụ (Business Vision), trong đó mô tả tầm nhìn của hệ thống tương lai.

Bảng tầm nhìn nghiệp vụ xác định một tập hợp các mục tiêu của công việc mô hình hóa nghiệp vụ, cung cấp đầu vào cho qui trình kiểm chứng dự án, có liên quan mật thiết với các trường hợp nghiệp vụ (Business Case), cũng như tài liệu tầm nhìn của công nghệ phần mềm. Nó được sử dụng bởi các nhà quản lý, những người có thẩm quyền về ngân quỹ, những người làm việc trong mô hình hóa nghiệp vụ, và các nhà phát triển nói chung.

Sưu liệu này phải bảo đảm rằng:

- Nó phải được cập nhật và được phân phối.
- Nó phải giải quyết được đầu vào từ tất cả các đối tượng có liên quan.

Xác định và mô tả các thuật ngữ nghiệp vụ

Một trong những khó khăn của dự án phần mềm quản lý hệ thống thông tin là sự bất đồng ngôn ngữ diễn đạt vấn đề giữa khách hàng và quản trị dự án hay giữa các thành viên tham gia trong dự án. Điều này gây ra các khó khăn trong việc tìm hiểu hay hiểu lầm các quy trình nghiệp vụ trong tổ chức của các thành viên trong dự án. Nhằm tránh những rủi ro này, chúng ta cần phải xác định và thống nhất những thuật ngữ trong các quy trình nghiệp vụ của tổ chức.

Sưu liệu thuật ngữ này chỉ thực sự hữu ích khi cần phân biệt rõ những từ chuyên môn của nghiệp vụ được dùng trong việc mô hình hóa nghiệp vụ với các từ chuyên môn của nghiệp vụ được dùng trong quá trình phát triển phần mềm.

Thông thường, mỗi thuật ngữ được mô tả như một danh từ với định nghĩa của nó. Tất cả các bên tham gia phải thống nhất với nhau về định nghĩa của các thuật ngữ này.

Ví dụ: Bảng thuật ngữ của hệ thống quản lý siêu thị như sau

Thuật ngữ	Diễn giải
Người quản lý	Người quản lý siêu thị và cũng là người quản trị hệ thống. Nguoiquanly được gọi chung cho những người được cấp quyền là "Quản lý", có thể bao gồm giám đốc, phó giám đốc, kế toán, nhân viên tin học, …
Nhân viên bán hàng	Là nhân viên làm việc trong siêu thị. Nhân viên bán hàng, đứng ở quầy thu tiền và tính tiền cho khách hàng. Thông qua các mã vạch quản lý

	trên từng mặt hàng được nhân viên bán hàng nhập vào hệ thống thông qua một đầu đọc mã vạch.
Tên đăng nhập	Tên đăng nhập của người sử dụng hệ thống. Mỗi nhân viên khi vào làm trong siêu thị sẽ được đăng ký một tên đăng nhập nhằm để quản lý. Khi đăng nhập vào hệ thống, nhân viên đó sẽ sử dụng tên này để đăng nhập. Người quản lý chịu trách nhiệm quản lý tên đăng nhập của nhân viên. Tồn tại duy nhất.
Mật khẩu	Mật khẩu đăng nhập của người sử dụng hệ thống. Mỗi nhân viên khi sử dụng tên đăng nhập sẽ được đăng ký kèm theo một mật khẩu đăng nhập. Mỗi nhân viên chỉ được biết duy nhất một mật khẩu của mình. Mật khẩu có thể rỗng.
Quyền đăng nhập	Quyền đăng nhập vào hệ thống. Tùy theo quyền và chức vụ trong công ty, nhân viên có quyền đăng nhập tương ứng.
Khách hàng thân thiết	Khách hàng thân thiết của siêu thị hay khách hàng đăng ký tham gia chương trình khách hàng thân thiết của siêu thị.
Điểm thưởng	Số điểm của khách hàng thân thiết trong siêu thị được thưởng do mua vượt mức thanh toán của siêu thị.
Ngày cấp thẻ	Ngày cấp thẻ khách hàng thân thiết cho khách hàng khi họ đăng ký chương trình khách hàng thân thiết của siêu thị.
Hóa đơn thanh toán	Hóa đơn tính tiền của siêu thị khi khách hàng mua hàng tại siêu thị
Chủng loại hàng	Chủng loại hàng hóa trong siêu thị, được phân chia tương ứng theo quầy hàng trưng bày trong siêu thị.
Loại hàng	Loại hàng trong siêu thị được phân chia theo tiêu chí công ty sản xuất, đơn vị tính....
Hàng hóa	Hàng hóa được bày bán trong siêu thị.
Hàng tồn	Số lượng hàng hóa còn lại trong siêu thị chưa bán được cho khách hàng.
Mức giảm	Tỉ lệ phần trăm giảm đối với khách hàng thân thiết
Thống kê doanh thu	Báo cáo thống kê tình hình kinh doanh của siêu thị theo tiêu chí nào đó như: hàng hóa, quý, khoảng thời gian....
Thống kê hàng hóa	Báo cáo thống kê số lượng hàng hóa của siêu thị theo tiêu chí nào đó như: hàng hóa, quý, khoảng thời gian....

Xác định tác nhân và use case nghiệp vụ

Mục đích:

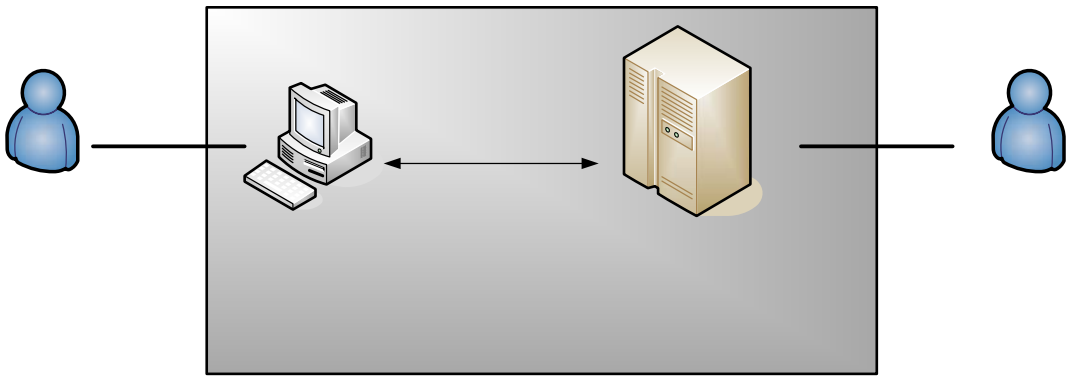
- Phác thảo các qui trình trong nghiệp vụ.
- Xác định ranh giới của nghiệp vụ cần được mô hình hóa.
- Xác định những gì sẽ tương tác với nghiệp vụ.
- Tạo ra các lược đồ của mô hình use-case nghiệp vụ.

Tác nhân (actor) trong môi trường nghiệp vụ

Để hiểu rõ được mục tiêu của nghiệp vụ, cần phải biết nghiệp vụ tương tác với những ai; nghĩa là ai đang yêu cầu hay quan tâm đến đầu ra của nó. Những ai đó này được biểu diễn như là các tác nhân nghiệp vụ (business actor).

Thuật ngữ **tác nhân** trong trường hợp này ám chỉ vai trò mà một người hay một thứ gì đó nắm giữ trong khi tương tác với nghiệp vụ. Những loại người dùng nghiệp vụ sau đây có khả năng được xem là những tác nhân nghiệp vụ: khách hàng, nhà cung cấp, đối tác, đồng nghiệp ở những nghiệp vụ không được mô hình hóa ...

Như vậy, một tác nhân thường tương ứng với con người. Tuy nhiên, có những tình huống, chẳng hạn như một hệ thống thông tin đóng vai trò của một tác nhân. Ví dụ, ngân hàng có thể quản lý hầu hết các giao dịch trực tuyến từ một máy tính thì các use case của hệ thống sẽ tương tác với ngân hàng, khi đó ngân hàng được xem là một tác nhân, điều đó có nghĩa tác nhân lúc này là một hệ thống thông tin.



Một tác nhân biểu diễn một loại người dùng cụ thể hơn là một người dùng thực tế. Nhiều người dùng thực tế của một nghiệp vụ có thể chỉ giữ một vai trò của tác nhân; nghĩa là, họ được xem như là các thể hiện của cùng một tác nhân. Hoặc một người dùng có thể giữ nhiều vai trò tác nhân khác nhau; nghĩa là cùng một người có thể là thể hiện của các tác nhân khác nhau.

Cách thức đặt tên các tác nhân nghiệp vụ: tên của một tác nhân nghiệp vụ cần phản ánh vai trò nghiệp vụ của nó, đồng thời có thể áp dụng được với bất cứ ai - hay bất cứ hệ thống thông tin nào - đóng vai trò ấy.

Tiêu chí đánh giá những thừa tác viên chuẩn:

Mỗi sự vật tương tác trong môi trường nghiệp vụ - cả con người và máy móc - đều được mô hình hóa bởi các tác nhân. Không thể chắc chắn tìm thấy tất cả tác nhân cho đến khi tất cả use case được tìm ra và được mô tả đầy đủ.

Mỗi tác nhân "người" diễn tả một vai trò, chứ không phải một người cụ thể. Chúng ta phải chỉ rõ ít nhất hai người có thể có vai trò của mỗi tác nhân. Nếu không, ta có thể đang mô hình hóa một người, chứ không phải một vai trò. Dĩ nhiên là có những tình huống chỉ tìm thấy một người có thể đóng một vai trò.

Mỗi tác nhân mô hình hóa một sự vật ở bên ngoài nghiệp vụ.

Khách hàng

Quầy bán hàng

Mỗi tác nhân có liên quan đến ít nhất một use case. Nếu một tác nhân không tương tác với ít nhất một use case, thì nên loại bỏ nó đi.

Một tác nhân cụ thể không tương tác với nghiệp vụ theo nhiều cách khác nhau hoàn toàn. Nếu một tác nhân tương tác theo nhiều cách khác nhau hoàn toàn, thì một tác nhân có thể có nhiều vai trò khác nhau. Trong trường hợp đó, tác nhân đó được chia thành nhiều actor, mỗi cái biểu diễn cho một vai trò khác nhau.

Mỗi tác nhân có một cái tên và mô tả rõ ràng. Tên của tác nhân cần trình bày vai trò nghiệp vụ của nó, tên này phải dễ hiểu cho những người không nằm trong nhóm mô hình hóa nghiệp vụ.

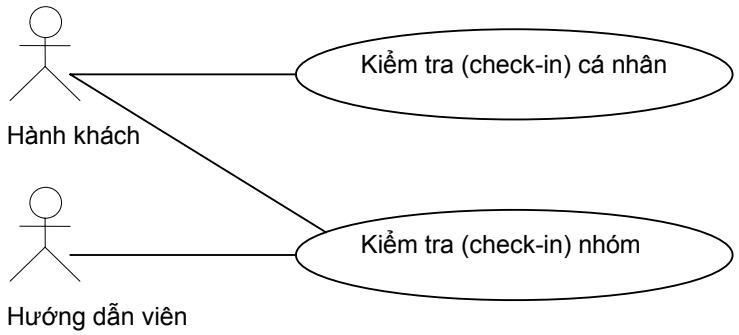
Xác định use case nghiệp vụ

Các qui trình của một nghiệp vụ được xác định thành một số các use case nghiệp vụ khác nhau, mỗi cái biểu diễn một luồng công việc cụ thể trong nghiệp vụ. Một use case nghiệp vụ xác định những gì xảy ra trong nghiệp vụ khi nó được thực hiện; nó mô tả sự thực thi một chuỗi các hành động nhằm tạo ra một kết quả có giá trị cho một tác nhân cụ thể.

Tên của use case cần diễn tả những gì xảy ra khi một thể hiện use case được thực hiện. Do đó, tên cần ở dạng chủ động, thông thường là một động từ kết hợp với một danh từ.

Tên có thể mô tả các hoạt động trong use case từ góc nhìn bên ngoài hoặc bên trong, ví dụ: *đặt hàng* hay *nhận đặt hàng*. Cho dù một use case mô tả những gì xảy ra bên trong nghiệp vụ, cách tự nhiên nhất vẫn là đặt tên use case từ góc nhìn của tác nhân chủ chốt trong use case đó. Một khi đã quyết định theo phong cách nào, ta nên áp dụng cùng một quy tắc cho tất cả use case trong mô hình nghiệp vụ.

Ví dụ:



Một hành khách hoặc có thể đi du lịch riêng lẻ hoặc cùng với một nhóm. Khi đi du lịch cùng với một nhóm, sẽ có một hướng dẫn viên du lịch cùng đi và việc check-in có thể được thực hiện cho một đoàn bởi hướng dẫn viên hoặc bởi một hành khách đại diện.

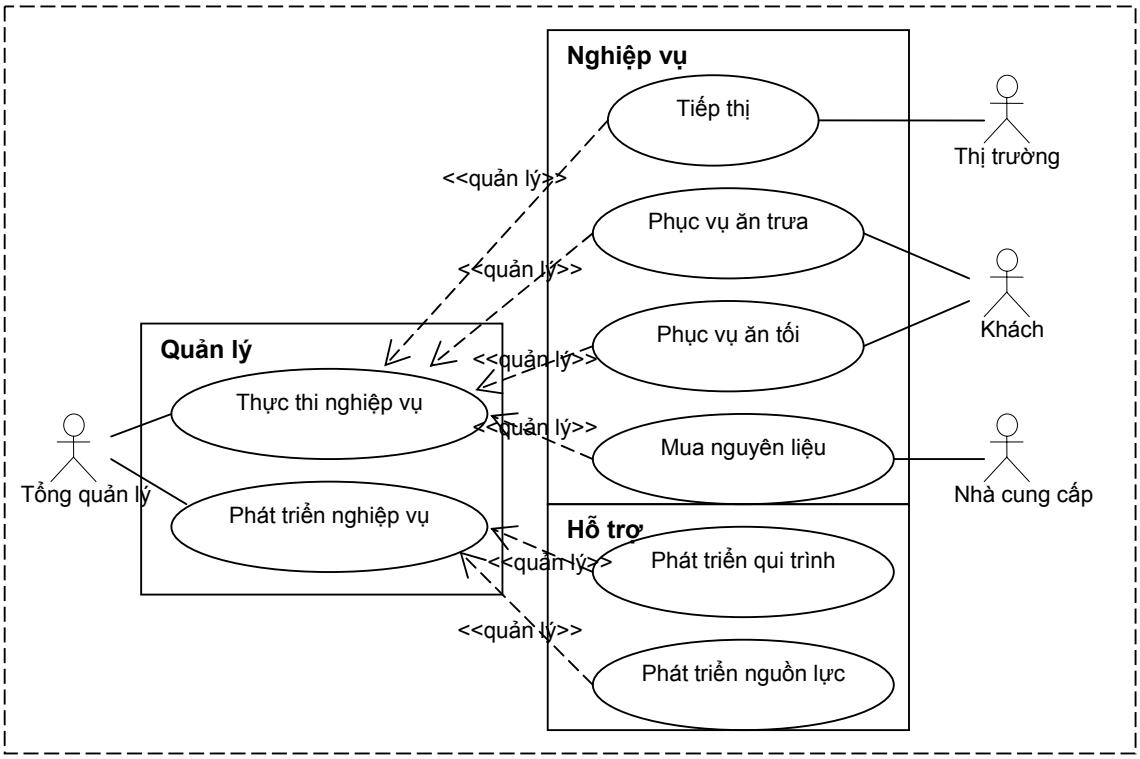
Phân loại use case nghiệp vụ

Khi nhìn vào các hoạt động trong một nghiệp vụ, ta có thể xác định tối thiểu ba loại công việc tương ứng với ba loại use case sau:

- Các hoạt động liên quan đến công việc của tổ chức, thường được gọi là các qui trình nghiệp vụ.
- Nhiều hoạt động không liên quan đến công việc của tổ chức, nhưng phải được thực hiện theo một cách nào đó để làm cho nghiệp vụ hoạt động. Ví dụ như quản trị hệ thống, dọn dẹp, an ninh,… Các use case này mang đặc điểm hỗ trợ.
- Công việc quản lý: các use case có đặc điểm quản lý cho thấy những loại công việc ảnh hưởng đến cách thức quản lý các use case khác và các mối quan hệ của nghiệp vụ với những chủ nhân của nó.

Thông thường, một use case quản lý mô tả tổng quan về các mối quan hệ giữa nhà quản lý với những nhân viên làm việc trong các use case. Nó cũng mô tả cách thức phát triển và khởi tạo các use case.

Ví dụ: các loại use case nghiệp vụ của một tổ chức nhà hàng



Lưu ý rằng một use case nghiệp vụ quan trọng đôi khi có thể là một use case nghiệp vụ hỗ trợ trong một nghiệp vụ khác. Ví dụ: phát triển phần mềm là một use case nghiệp vụ quan trọng của một công ty phát triển phần mềm, trong khi đó nó được phân loại thành một use case nghiệp vụ hỗ trợ trong một ngân hàng hay một công ty bảo hiểm.

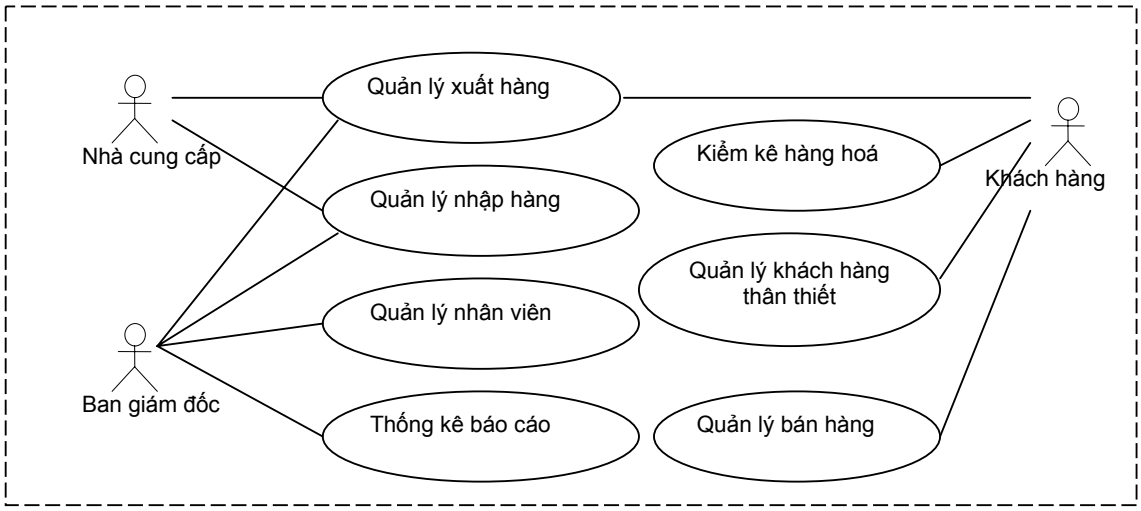
Qui mô của một use case nghiệp vụ

Đôi khi khó quyết định được một dịch vụ là một, hay nhiều use case nghiệp vụ. Áp dụng định nghĩa của một use case nghiệp vụ cho qui trình đăng ký chuyến bay. Một hành khách đưa vé và hành lý cho nhân viên đăng ký, nhân viên này sẽ tìm một chổ ngồi cho hành khách, in ra thẻ lên máy bay và bắt đầu xử lý hành lý. Nếu hành khách có một hành lý thông thường, nhân viên đăng ký sẽ in ra thẻ đánh dấu hành lý và thẻ kiểm soát hành khách, cuối cùng kết thúc use case nghiệp vụ bằng cách gắn thẻ đánh dấu cho hành lý, đưa thẻ kiểm soát cùng với thẻ lên máy bay cho hành khách. Nếu hành lý là một dạng đặc biệt hay chứa những thứ đặc biệt không thể vận chuyển một cách bình thường, hành khách phải mang nó đến một quầy hành lý đặc biệt. Nếu hành lý quá nặng, hành khách phải tiếp tục đến văn phòng vé máy bay để trả tiền, bởi vì các nhân viên đăng ký không xử lý việc đóng tiền.

Câu hỏi đặt ra là có cần một use case nghiệp vụ tại quầy đăng ký, một use case nghiệp vụ khác tại quầy hành lý đặc biệt và cái thứ ba ở văn phòng vé? Hay là chỉ cần một use case nghiệp vụ duy nhất? Chắc chắn là sự giao dịch này có liên quan đến ba loại hành động khác nhau. Nhưng câu hỏi ở đây là có một hành động nào đó sẽ có ý nghĩa đối với hành khách mang hành lý đặc biệt nếu hành khách này không thực hiện những hành động còn lại? Câu trả lời là không có, nó chỉ là một thủ tục hoàn chỉnh - từ lúc hành khách đến quầy đăng ký đến khi ông ta trả thêm phí phụ thu (chỉ có giá trị hay có ý nghĩa đối với hành khách). Như vậy, thủ tục hoàn chỉnh có liên quan đến ba quầy khác nhau chính là một trường hợp sử dụng hoàn chỉnh, tức là một use case nghiệp vụ.

Ngoài tiêu chí này, điều quan trọng là cần giữ mô tả của các dịch vụ có liên quan mật thiết này cùng với nhau, để sau này có thể xem lại chúng cùng một lúc, điều chỉnh, kiểm tra và viết hướng dẫn cho chúng, và nói chung là quản lý chúng như một đơn vị.

Kết quả của quá trình tiếp cận phân tích nghiệp vụ là (các) sơ đồ use case nghiệp vụ và các mô tả của use case.



Mô hình use case mô tả nghiệp vụ của siêu thị

Bài tập

1.1 Cho một hệ thống được mô tả như sau:

Government Solutions Company (GSC) là công ty chuyên bán các trang thiết bị cho các cơ quan chính phủ liên ban. Khi một cơ quan cần mua trang thiết bị từ GSC, cơ quan này sẽ phát sinh một đơn đặt hàng dựa trên một hợp đồng chuẩn trước đó đã được thoả thuận với với công ty. GSC quản lý các hợp đồng với các cơ quan liên bang. Khi một đơn đặt hàng gởi tới nhân viên quản lý hợp đồng của GSC, nhân viên này xem xét lại các thông tin về các điều khoản và điều kiện của hợp đồng đã ký với cơ quan bằng cách sử dụng mã số hợp đồng được tham khảo trong đơn đặt hàng để tìm kiếm trong cơ sở dữ liệu của GSC rồi so với thông tin của đơn đặt hàng để xác định đơn đặt hàng có hợp lệ hay không. Đơn đặt hàng lợp lệ nếu hợp đồng chưa hết hạn, danh sách trang thiết bị phải thuộc các thiết bị của hợp đồng, và tổng chi phí phải không vượt quá giới hạn xác định trước. Nếu đơn đặt hàng không hợp lệ, nhân viên quản lý hợp đồng sẽ gởi trả lại đơn đặt hàng cho cơ quan kèm theo một lá thư diễn giải lý do đơn đặt hàng không hợp lệ và lưu lại một bản sao của lá thư.

Nếu đơn đặt hàng hợp lệ, nhân viên quản lý hợp đồng lưu vào cơ sở dữ liệu đơn đặt hàng đó cùng với trạng thái là "chưa giải quyết". Sau đó, đơn đặt hàng sẽ được gởi đến bộ phận đáp ứng đơn hàng, bộ phận này sẽ kiểm tra tồn kho cho mỗi thiết bị dựa trên thông tin tồn kho lấy từ cơ sở dữ liệu. Nếu có bất kỳ một thiết bị nào không đủ số giao, bộ phận này sẽ tạo ra một báo cáo liệt kê tất cả các mặt hàng không đủ giao cùng với số lượng thiếu.

Tất cả các đơn đặt hàng sẽ được chuyển tới kho để thực hiện việc giao hàng cho cơ quan. Sau khi giao hàng, tại đây sẽ lập một hoá đơn giao hàng ghi nhận các thiết bị được giao gởi cho cơ quan, và đính kèm một bản sao hóa đơn này với đơn đặt hàng gởi trở lại cho nhân viên quản lý hợp đồng. Nhân viên này sẽ kiểm tra nếu đơn đặt hàng đã được giao hết thì cập nhật lại trạng thái đơn đặt hàng là "hoàn tất" và lưu thông tin hoá đơn giao hàng vào cơ sở dữ liệu.

Hàng tháng, nhân viên quản lý hợp đồng sẽ lập các báo cáo các thiết bị được đặt hàng trong tháng, các thiết bị đã được giao, các đơn đặt hàng đang "chưa giải quyết", các hợp đồng đã hết hạn gởi cho giám đốc để giúp cho giám đốc nắm được tình hình hoạt động của công ty.

- Hãy xác định các tác nhân nghiệp vụ của hệ thống
- Xác định tất cả các use case nghiệp vụ
- Xây dựng sơ đồ use case nghiệp vụ

1.2 Mô tả hệ thống "Quản lý cho thuê văn phòng cao ốc" như sau:

Một công ty địa ốc muốn tin học hóa hoạt động cho thuê cao ốc của mình cho các công ty làm văn phòng hoạt động kinh doanh. Công ty có nhiều cao ốc ở trong thành phố, mỗi cao ốc được quản lý bởi một tên, có một địa chỉ, mô tả đặc điểm và tổng diện tích sử dụng. Một cao ốc sẽ có nhiều tầng, mỗi tầng có nhiều phòng, mỗi phòng có các thông tin cần quản lý là: mã phòng, diện tích sử dụng, số chổ làm việc (theo tính toán của công ty), mô tả vị trí, giá cho thuê.

Hoạt động thuê phòng

Khách hàng muốn thuê phòng thì phải đến nơi quản lý tòa nhà để tham khảo vị trí, diện tích phòng, và tại đây khách hàng được nhân viên tiếp tân cung cấp thông tin tình trạng giá cả của phòng. Giá cả mỗi phòng được ấn định tùy theo độ cao, diện tích sử dụng,…

Khách hàng sau khi đồng ý thuê thì đến gởi và trình bày yêu cầu làm hợp đồng với bộ phận quản lý nhà, bộ phận này soạn thảo hợp đồng dựa trên yêu cầu thuê mướn được cung cấp bởi khách hàng, rồi tính toán giá thuê và lập hợp đồng. Hợp đồng sau khi được ký nhận bởi khách hàng và trưởng bộ phận quản lý nhà sẽ được lưu lại một bản, và một bản sẽ gởi cho khách hàng. Khách có thể làm hợp đồng thuê cùng lúc nhiều phòng. Nội dung của hợp đồng bao gồm: số hợp đồng, ngày hiệu lực hợp đồng, ngày thanh toán đầu tiên, khách hàng, thời gian thuê và chi tiết gồm các phòng cần thuê, giá thuê. Thời gian của đợt thuê ít nhất sáu tháng và sau đó có thể gia hạn thêm. Khách phải trả trước tiền thuê của sáu tháng đầu tiên, từ tháng thứ bảy nếu có thì phải trả vào đầu mỗi tháng. Giá thuê phòng không kể chi phí điện trong đó, do đó cuối tháng khách cũng phải thanh toán các chi phí điện.

Đầu mỗi tuần, phòng kế toán kiểm tra lại tất cả hợp đồng đến hạn phải thanh toán trong tuần. Sau đó lập thông báo gởi tới các khách hàng để chuẩn bị thanh toán. Mỗi lần thanh toán, khách hàng phải đến phòng kế toán của công ty để đề nghị thanh toán. Tại đây, phòng kế toán sẽ tìm kiếm hợp đồng từ hồ sơ lưu, kiểm tra thông tin đợt thanh toán của khách hàng. Sau đó lập hoá đơn thanh toán và lưu lại một bản, một bản gởi cho khách hàng. Các thông tin trên hoá đơn sẽ là: số phiếu, ngày thanh toán, lý do thanh toán (tiền phòng hay tiền điện nước), thanh toán cho hợp đồng nào và tổng số tiền thanh toán.

Gia hạn hợp đồng

Khi gần hết hạn hợp đồng, khách hàng muốn gia hạn thêm thi phải đến yêu cầu bộ phận quản lý xin gia hạn hợp đồng. Bộ phận này sẽ kiểm tra xem thực sự đã hến hạn chưa, đã thanh toán đầy đủ chưa. Nếu các điều kiện trên được đáp ứng thì sẽ lập một bản hợp đồng gia hạn. Nội dung bản gia hạn gồm có: số hợp đồng cần gia hạn, ngày ký gia hạn, mô tả và chi tiết gia hạn gồm các phòng gia hạn (các phòng này phải được thuê bởi hợp đồng) và thời gian gia hạn. Việc thanh toán gia hạn cũng sẽ được lập theo từng tháng và một hợp đồng có thể được gia hạn nhiều lần.

Hoạt động truy xuất

Các báo cáo hằng ngày được bộ phận quản lý lập:

- Danh sách phòng đang được thuê
- Danh sàch phòng đang trống
- Danh sách nhân viên đang làm việc
- Danh sách công ty hết hạn thuê trong tháng

Yêu cầu

- Hãy xác định các tác nhân nghiệp vụ của hệ thống
- Xác định tất cả các use case nghiệp vụ
- Xây dựng sơ đồ use case nghiệp vụ

Thiết kế qui trình nghiệp vụ

Đặc tả các use case nghiệp vụ:

Bước đầu tiên trong giai đoạn thiết kế này chính là đặc tả các use case nghiệp vụ nhằm làm rõ nội dung của các use case này. Chú ý rằng chúng ta chỉ mô tả các nội dung xử lý thứ tự luận lý giữa các xử lý này, về kết quả, cách mô tả này độc lập tương đối với một một trường thực tế xử lý nó có nghĩa rằng chúng ta chỉ làm rõ phần nội dung của use case mà chưa mô tả rõ các vai trò thực hiện và các đối tượng bị tác động bởi use case.

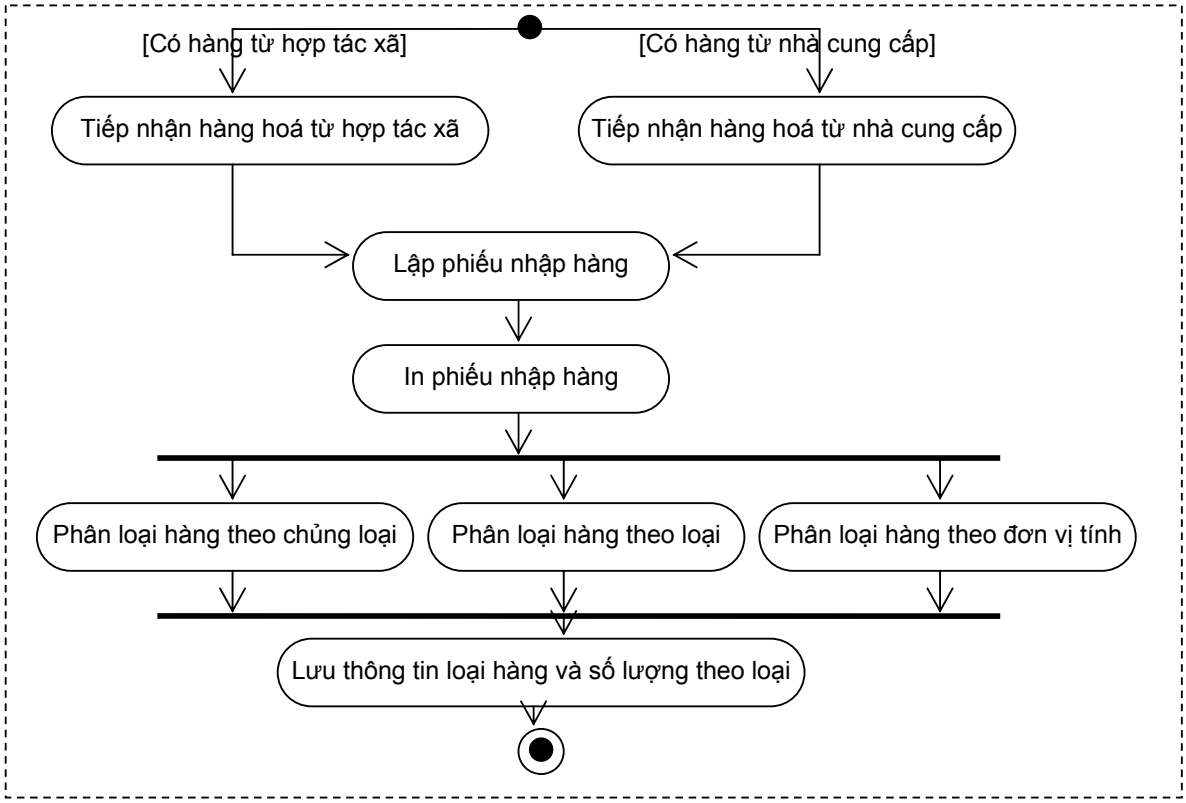
Hãy bắt đầu mô tả luồng công việc bình thường trong use case nghiệp vụ, xác định sự tương tác giữa các tác nhân và use case. Sau đó, khi luồng công việc bình thường ổn định, ta bắt đầu mô tả các luồng công việc thay thế khác. Một luồng công việc use-case nghiệp vụ được trình bày theo các cách thức đã nhất trí và tham khảo bảng chú giải chung khi viết những văn bản mô tả.

Mô tả tất cả những luồng sự kiện bất thường và luồng sự kiện tùy chọn. Mô tả một luồng sự kiện con trong phần bổ sung của luồng công việc đối với các trường hợp:

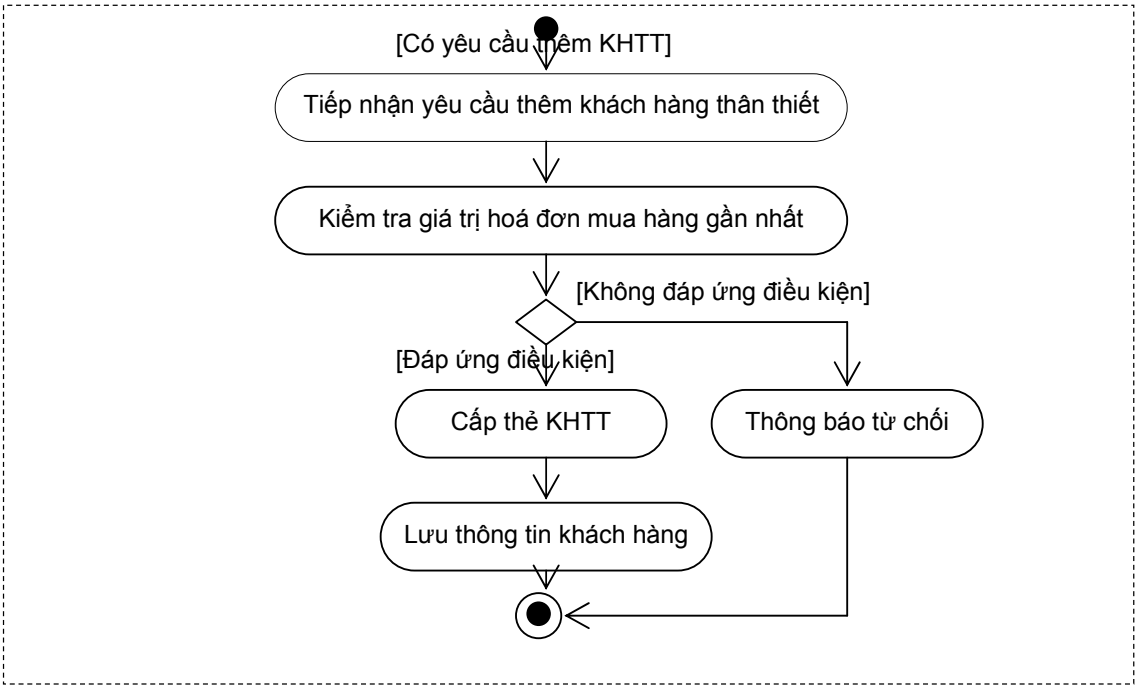
- Những luồng sự kiện con tham gia phần lớn luồng công việc chính.
- Những luồng công việc bất thường giúp luồng công việc chính rõ ràng hơn.
- Những luồng sự kiện con xảy ra ở những khoảng thời gian khác nhau trong cùng một luồng công việc và chúng có thể được thực thi.

Ngoài ra, có thể đặc tả cấu trúc luồng công việc trong một sơ đồ hoạt động.

Ví dụ: sơ đồ hoạt động đặc tả use case nhập hàng



Ví dụ: sơ đồ hoạt động đặc tả use case Quan lý khách hàng thân thiết



Trong sơ đồ hoạt động trên, luồng công việc chính của use case bao gồm bốn hoạt động được vẽ trên một luồng thẳng đứng. Sau đó luồng công việc phụ như *Thông báo từ chối* được xác định bổ sung thành các nhánh.

Xác định các thừa tác viên nghiệp vụ (business worker) và các thực thể chịu tác động bởi nghiệp vụ (business entity)

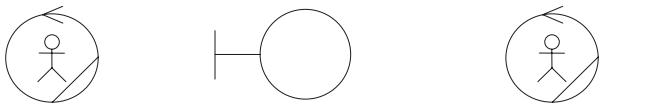
Xác định thừa tác viên nghiệp vụ

Một thừa tác viên biểu diễn sự trừu tượng của một người hoạt động trong nghiệp vụ. Một đối tượng thừa tác viên tương tác với các đối tượng thừa tác viên khác đồng thời thao tác với các đối tượng thực thể để hiện thực hóa một thể hiện use-case.

Một thừa tác viên được khởi tạo khi luồng công việc của thể hiện use-case tương ứng bắt đầu hay ngay vào lúc người có vai trò của thừa tác viên bắt đầu thực hiện vai trò đó trong thể hiện use-case. Một đối tượng thừa tác viên thường "sống" khi use case thực thi.

UML phân chia thừa tác viên thành hai loại: thừa tác viên thực hiện các công việc bên trong hệ thống và thừa tác viên tương tác trực tiếp với các tác nhân bên ngoài hệ thống. Thực sự việc phân chia này chỉ muốn chuyên biệt hơn nữa vai trò của các thừa tác viên trong việc giao tiếp với tác nhân bên ngoài.

Ký hiệu:



Trong các ký hiệu trên cho thấy, Nhân viên bán hàng được mô hình hoá như là một thừa tác viên giao tiếp trực tiếp với các tác nhân bên ngoài. Quản trị hệ thống được mô hình hoá là môt thừa tác viên làm việc bên trong hệ thống. Thủ thư là một thừa tác viên có thể vừa làm việc bên trong và vừa chịu trách nhiệm giao tiếp với tác nhân ben ngoài, tuy nhiên, nếu được mô hình hoá theo như trên thì chúng ta muốn nhấn mạnh vài trò hoạt động bên trong hệ thống của Thủ thư hơn là vai trò giao tiếp với tác nhân bên ngoài.

Xác định các thực thể nghiệp vụ

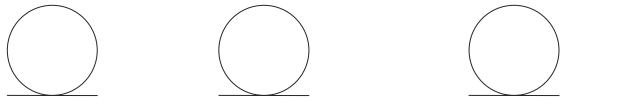
Các thực thể biểu diễn những thứ được xử lý hoặc sử dụng bởi các thừa tác viên khi chúng thực thi một use case nghiệp vụ. Một thực thể thường biểu diễn một sự vật có giá trị cho một số thể hiện use case hoặc thể hiện use case, vì vậy đối tượng thực thể sống lâu hơn. Nói chung, thực thể không nên giữ thông tin nào về cách thức nó được sử dụng bởi ai.

Một thực thể biểu diễn một tài liệu hoặc một phần thiết yếu của sản phẩm. Đôi khi nó là một thứ gì đó mơ hồ, như kiến thức về một thị trường hay về một khách hàng. Ví dụ về các thực thể tại nhà hàng là Thực đơn và Thức uống; tại phi trường là Vé và Thẻ lên máy bay (Boarding Pass) là những thực thể quan trọng.

Mô hình hóa các hiện tượng thành những thực thể chỉ khi những lớp khác trong mô hình đối tượng phải tham chiếu đến các hiện tượng này. Những thứ khác có thể được mô hình hóa thành các thuộc tính của các lớp thích hợp, hay chỉ cần được mô tả bằng văn bản trong những lớp này.

Tất cả mỗi sự vật trong nghiệp vụ như: sản phẩm, tài liệu, hợp đồng, ... đều được mô hình hóa thành các thực thể nghiệp vụ, và nó tham gia vào tối thiểu một use case nghiệp vụ.

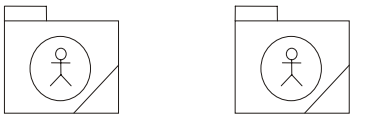
Ký hiệu:



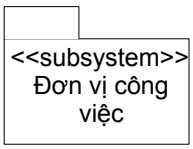
Các khái niệm UML hỗ trợ thêm cho quá trình mô hình hoá nghiệp vụ

Ngoài ra, UML (phiên bản 1.5) còn bổ sung thêm một số stereotype cho phép mô hình hoá đầy đủ hơn về hệ thống nghiệp vụ:

Đơn vị tổ chức (organization unit)
Mô tả : tập hợp các thừa tác viên, thực thể, use case nghiệp vụ, các sơ đồ, và các đơn vị tổ chức khác. Dối tượng này được dùng để phân chia mô hình nghiệp vụ thành nhiều phần khác nhau.



Đơn vị công việc (WorkUnit)
Mô tả : là một loại hệ thống con có thể chứa một hoặc nhiều thực thể. Nó là một tập đối tượng hướng nhiệm vụ nhằm hình thành một tổng thể có thể nhận thức được bởi người dùng cuối và có thể có một giao diện xác định cách nhìn các thực thể công việc thích hợp tới nhiệm vụ đó.



Hiện thực hoá use case nghiệp vụ

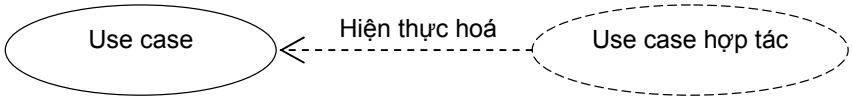
Trong một dự án mô hình hóa nghiệp vụ hướng use-case, hãy phát triển hai khung nhìn nghiệp vụ.

Use case nghiệp vụ trình bày khung nhìn bên ngoài của nghiệp vụ, qua đó xác định những gì thiết yếu cần thực hiện cho nghiệp vụ để phân phối các kết quả mong muốn cho tác nhân. Nó cũng xác định trong nghiệp vụ cần có những tương tác nào với tác nhân khi use case được

thực thi. Khung nhìn này được phát triển khi đang lựa chọn và nhất trí về những gì cần được thực hiện trong mỗi use case. Một tập hợp các use case cung cấp một cái nhìn tổng quan về nghiệp vụ, nó rất hữu ích để thông báo cho các nhân viên về những thay đổi, những điểm khác biệt của nghiệp vụ đang thực hiện, và những kết quả nào được mong muốn.

Mặt khác, một hiện thực hóa use-case cung cấp một khung nhìn bên trong về use case, qua đó xác định cách thức công việc cần được tổ chức và thực hiện như thế nào nhằm đạt được những kết quả mong muốn như trên. Một hiện thực hóa bao gồm các thừa tác viên và thực thể có liên quan đến sự thực thi một use case và các mối quan hệ giữa chúng. Những khung nhìn như vậy cần thiết cho công việc lựa chọn và thống nhất về cách thức tổ chức các công việc trong mỗi use case nhằm đạt được những kết quả mong muốn.

Cả hai khung nhìn của use case đều chủ yếu dành cho những nhân viên bên trong nghiệp vụ - khung nhìn bên ngoài dành cho những người hoạt động bên ngoài use case, khung nhìn bên trong dành cho những người hoạt động bên trong use case.



Mô hình hoá hiện use case hiện thực hoá qua việc lập cấu trúc mô hình đối tượng nghiệp vụ (business object)

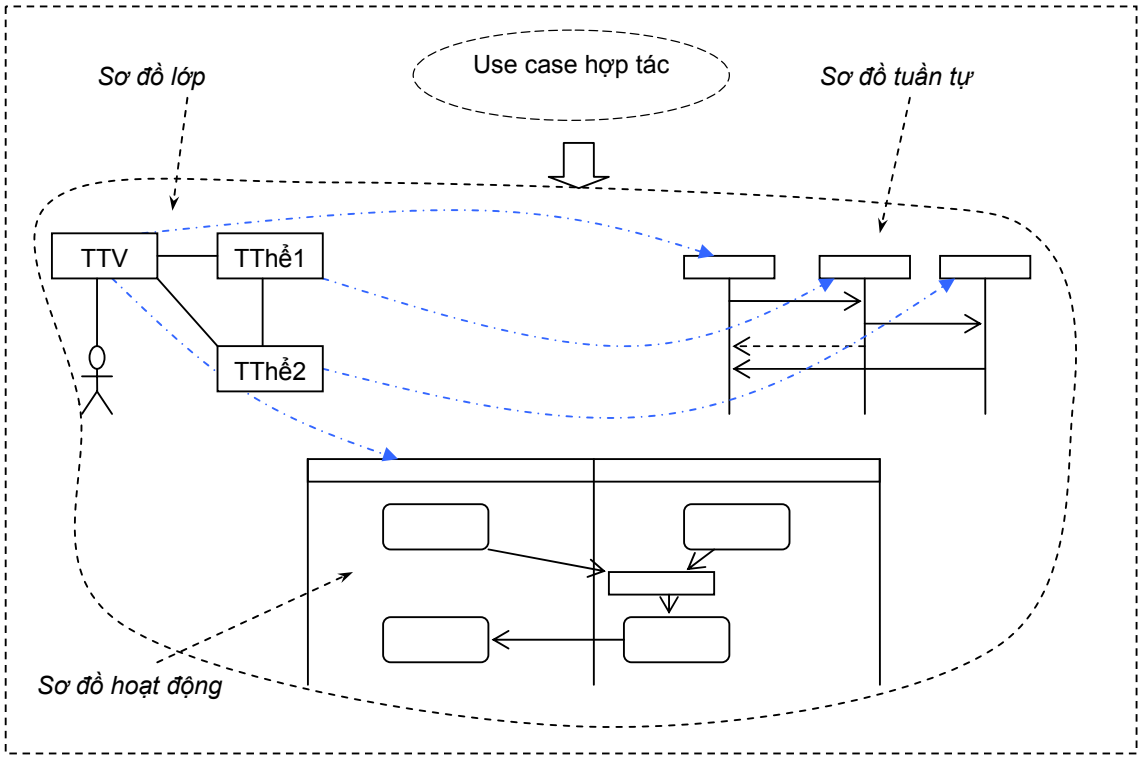
Sơ đồ đối tượng nghiệp vụ là một tập các sơ đồ nhằm trình bày sự hiện thực hóa của các use case nghiệp vụ. Nó mô tả trừu tượng cách thức các thừa tác viên và thực thể liên kết và cộng tác với nhau để thực hiện nghiệp vụ.

- *Giải thích*

Sơ đồ đối tượng xác định các use case từ góc nhìn bên trong của các thừa tác viên. Mô hình định nghĩa cách thức các nhân viên nghiệp vụ với những gì họ xử lý liên hệ với nhau để tạo ra các kết quả mong muốn. Nó nhấn mạnh vào các vai trò được thực hiện trong lĩnh vực nghiệp vụ và các trách nhiệm của nhân viên. Các đối tượng của các lớp trong mô hình cần có khả năng thực hiện tất cả use case nghiệp vụ.

Các thành phần chính của mô hình đối tượng nghiệp vụ là:

- Các thừa tác viên (worker): cho thấy các trách nhiệm của một nhân viên
- Các thực thể (entity): biểu diễn đầu ra, tài nguyên, sự vật được sử dụng
- Các hiện thực hóa use-case nghiệp vụ: cho thấy các thừa tác viên cộng tác và các thực thể thực hiện luồng công việc như thế nào. Các hiện thực hóa use-case nghiệp vụ được đặc tả với:
  - Các lược đồ lớp: là các thừa tác viên và thực thể tham gia
  - Các lược đồ hoạt động: trong đó các swimlane cho thấy các trách nhiệm của các thừa tác viên, các luồng đối tượng cho thấy cách sử dụng các thực thể trong luồng công việc.
  - Các lược đồ tuần tự: mô tả chi tiết sự tương tác giữa các thừa tác viên, tác nhân, và cách truy xuất các thực thể khi thực hiện một use case nghiệp vụ.



- *Mục đích của mô hình đối tượng nghiệp vụ*

- Là một thành phần trung gian để làm rõ các ý kiến về nghiệp vụ theo cách suy nghĩ của các nhà phát triển phần mềm, mà vẫn giữ được nội dung nghiệp vụ. Nó là sự thống nhất về lĩnh vực nghiệp vụ được mô tả dưới dạng các đối tượng, thuộc tính, trách nhiệm.
- Khảo sát bản chất của lĩnh vực nghiệp vụ nhằm chuyển tiếp lối tư duy về các vấn đề nghiệp vụ sang lối tư duy về các ứng dụng phần mềm.
- Làm rõ những yêu cầu được hỗ trợ bởi hệ thống thông tin đang xây dựng.
- Thống nhất các định nghĩa về đối tượng nghiệp vụ, các mối quan hệ giữa các đối tượng, tên các đối tượng và quan hệ. Qua đó, cho phép trình bày chính xác các kiến thức về lĩnh vực nghiệp vụ sao cho các chuyên gia về lĩnh vực nghiệp vụ có thể hiểu được.

- *Lập cấu trúc mô hình đối tượng nghiệp vụ:*

Phân tích chu kỳ sống của mỗi thực thể. Mỗi thực thể nên được tạo ra và hủy đi bởi một người nào đó trong đời sống của nghiệp vụ. Hãy bảo đảm rằng mỗi thực thể được truy xuất và sử dụng bởi một thừa tác viên hay một thực thể khác.

Cần giảm bớt số lượng các thừa tác viên. Khi phát triển các mô hình, có thể ta sẽ thấy có quá nhiều thừa tác viên. Hãy bảo đảm rằng mỗi thừa tác viên tương ứng với một tập hợp các tác vụ mà một người thường thực hiện.

Mỗi thực thể nên có một người chịu trách nhiệm cho nó. Điều này có thể được mô hình hóa bằng một mối kết hợp từ thừa tác viên đến các thực thể mà thừa tác viên đó chịu trách nhiệm. Một số thực thể có thể do những người ngoài nghiệp vụ chịu trách nhiệm. Mô tả điều này trong bản mô tả vắn tắt của thực thể đó.

- *Lược đồ lớp (class diagram)*

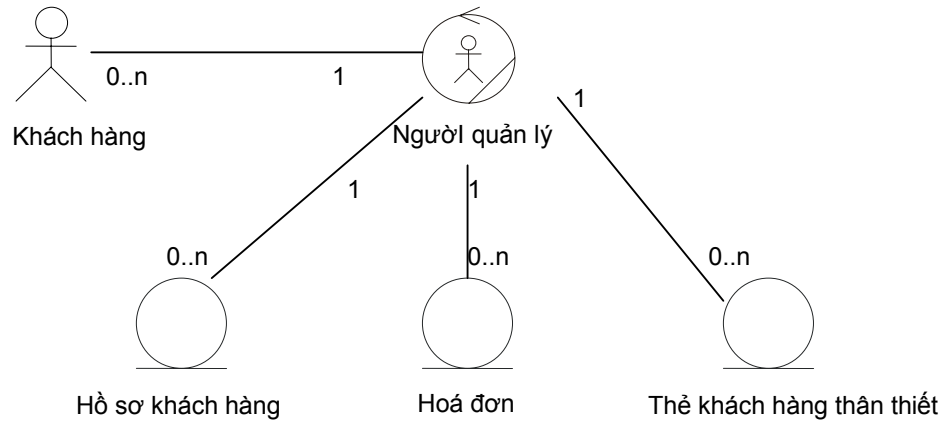
Một lược đồ lớp cho thấy một tập hợp các thành phần (tĩnh) của mô hình, như lớp, gói, nội dung của chúng và các mối quan hệ.

Các lược đồ lớp cho thấy các mối kết hợp, kết tập và tổng quát hóa giữa thừa tác viên và thực thể. Những lược đồ lớp được quan tâm về:

- Sự phân cấp kế thừa
- Các mối kết tập của thừa tác viên và thực thể.
- Cách thức các thừa tác viên và thực thể liên quan đến nhau thông qua các mối kết hợp.

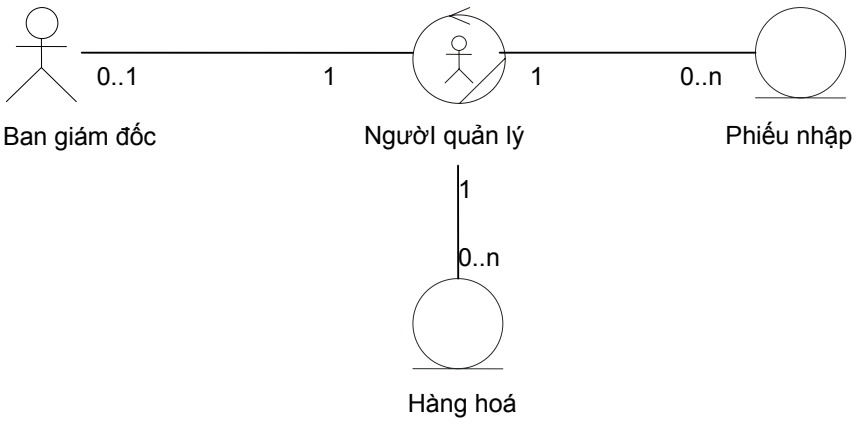
Các lược đồ lớp cho thấy các cấu trúc chung trong mô hình đối tượng nghiệp vụ, nhưng cũng có thể là một phần của tài liệu mô tả một hiện thực hóa use case bằng cách cho thấy các thừa tác viên và thực thể tham gia.

Ví dụ: sơ đồ lớp cho use case Quản lý khách hàng thân thiết cho biết các thừa tác viên, các thực thể và tác nhân liên kết với nhau trong việc thực hiện của use case này.



Trong đó, ***Người quản lý*** là thừa tác viên thực hiện use case. ***Hồ sơ khách hàng, Thẻ khách hàng thân thiết*** và ***Hoá đơn*** là ba thực thể được sử dụng trong use case này bởi thừa tác viên.

Hoặc sơ đồ lớp cho use case Quản lý nhập hàng



Trong đó, Người quản lý là thừa tác viên thực hiện use case. Phiếu nhập và hàng hoá là các thực thể bởi thừa tác viên này trong việc thực hiện hoạt động của use case.

Đặc tả luồng công việc hiện thực hoá use case nghiệp vụ

Sử dụng sơ đồ hoạt động

Đầu tiên, để lập tài liệu hiện thực hóa cho một use case nghiệp vụ chính là vẽ một lược đồ hoạt động, trong đó các luồng (swimlane) biểu diễn các thừa tác viên tham gia. Đối với mỗi hiện thực hóa use-case, có thể có một hoặc nhiều lược đồ hoạt động để minh họa luồng công

việc. Một cách phổ biến là sử dụng một lược đồ tổng quan không có các swimlane để mô tả toàn bộ luồng công việc, trong đó trình bày các "hoạt động vĩ mô" ở mức cao. Sau đó, đối với mỗi hoạt động vĩ mô sẽ có một lược đồ hoạt động chi tiết, trình bày các luồng (swimlane) và các hoạt động ở cấp độ thừa tác viên. Mỗi lược đồ nên được gói gọn trong một trang giấy.

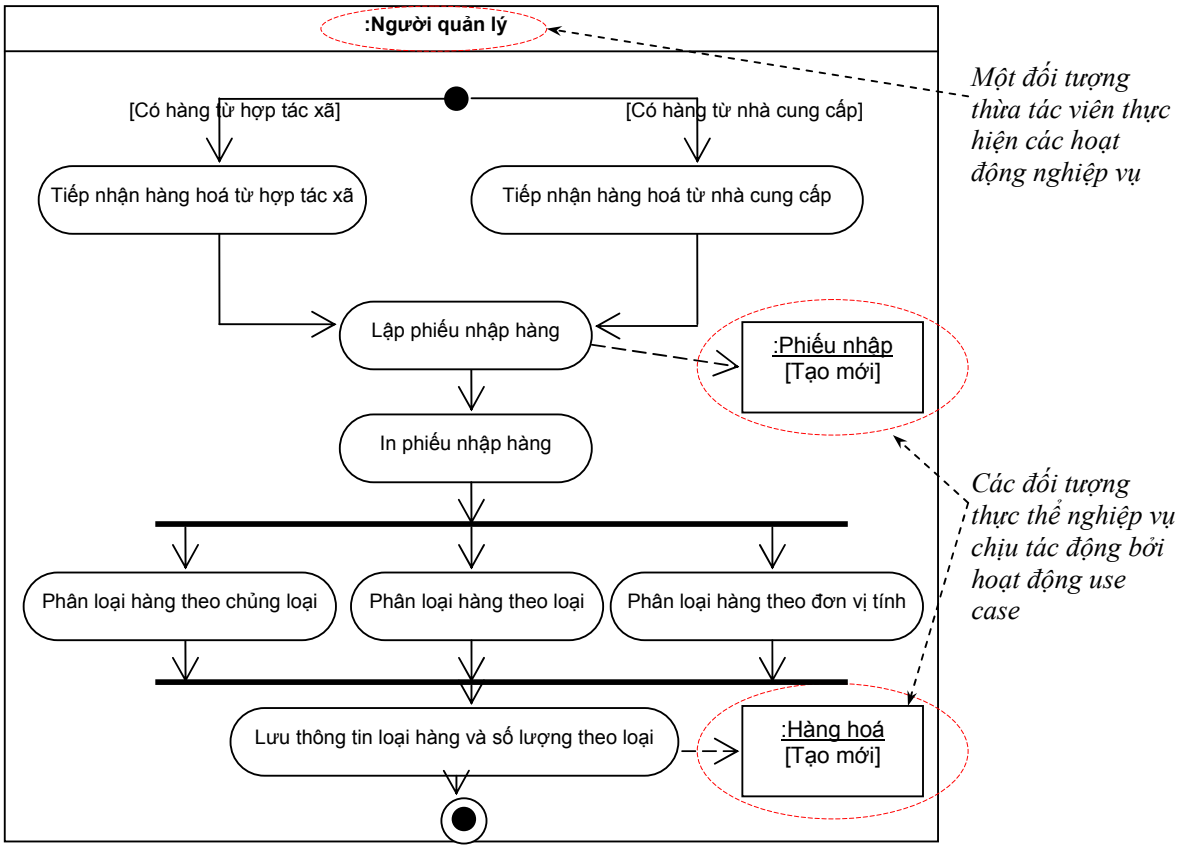
Lược đồ hoạt động trong mô hình đối tượng minh họa luồng công việc của một hiện thực hóa use-case nghiệp vụ. Lược đồ hoạt động của một hiện thực hóa use-case minh hoạ việc sắp xếp các công việc theo một thứ tự nhằm đạt được các mục tiêu của nghiệp vụ, cũng như thỏa mãn nhu cầu giữa các tác nhân bên ngoài và các thừa tác viên bên trong. Một hoạt động trong lược đồ hoạt động có thể là một công việc thủ công hoặc tự động hóa để hoàn thành một đơn vị công việc.

Các lược đồ hoạt động giúp:

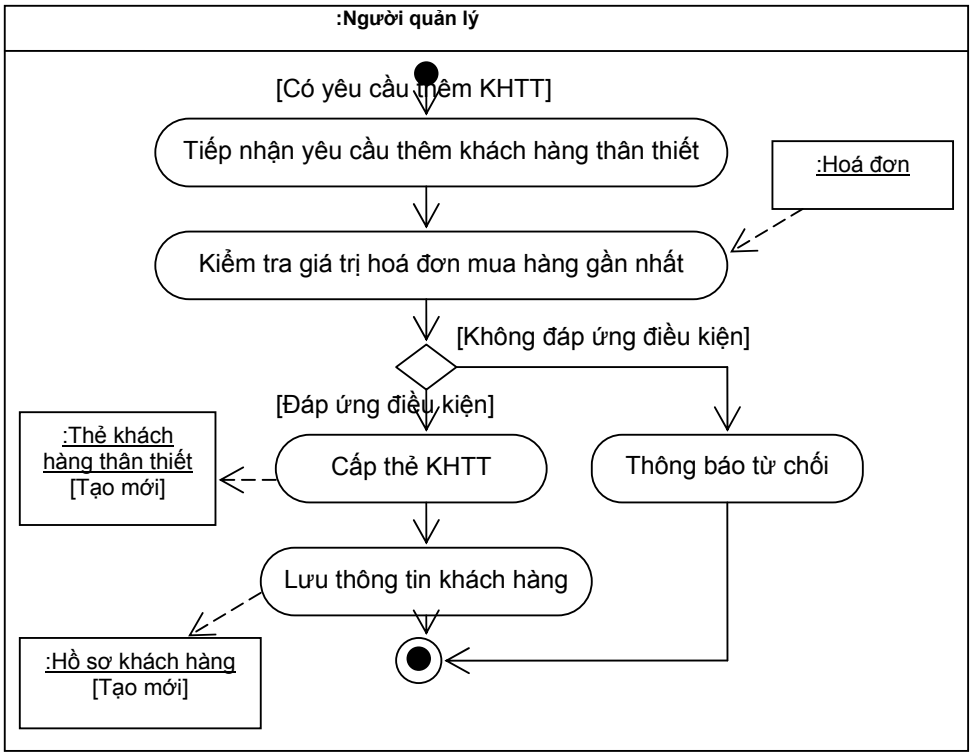
- Cung cấp cơ sở để giới thiệu các hệ thống thông tin đến doanh nghiệp một cách dễ hiểu hơn.
- Thiết lập các mục tiêu cho các dự án phát triển hệ thống nhằm cải tiến nghiệp vụ.
- Điều chỉnh mức độ đầu tư vào việc tự động hóa quy trình dựa trên các thông tin đo lường qui trình nghiệp vụ đó.

So sánh với lược đồ tuần tự có cùng mục đích thì lược đồ hoạt động tập trung mô tả cách thức phân chia trách nhiệm thành các lớp, trong khi đó, lược đồ tuần tự mô tả cách thức các đối tượng tương tác theo trình tự. Lược đồ hoạt động tập trung vào luồng công việc, trong khi lược đồ tuần tự tập trung vào việc xử lý các thực thể. Chúng bổ sung cho nhau, như lược đồ tuần tự cho thấy những gì xảy ra trong một trạng thái hoạt động.

Ví dụ: sơ đồ hoạt động hiện thực hoá use case Quản lý nhập hàng



Sơ đồ hoạt động hiện thực hoá use case Quản lý khách hàng thân thiết



Sử dụng các swimlane (làn bơi)

- Nếu các luồng (swimlane) được sử dụng và được nhóm thành các lớp (chủ yếu là các thừa tác viên) trong mô hình đối tượng, thì chúng ta đang sử dụng lược đồ hoạt động để trình bày các hiện thực hóa use-case nghiệp vụ.
- Lược đồ hoạt động cung cấp chi tiết về những gì xảy ra trong nghiệp vụ bằng cách khảo sát những người có các vai trò cụ thể (các thừa tác viên) và các hoạt động mà họ thực hiện. Đối với các dự án phát triển ứng dụng, các lược đồ này giúp ta hiểu một cách chi tiết về lĩnh vực nghiệp vụ sẽ được hỗ trợ hay chịu tác động của ứng dụng mới. Các lược đồ hoạt động giúp ta hình dung hệ thống mới được đề nghị rõ ràng hơn đồng thời xác định các use case của hệ thống đó.

Sử dụng các luồng đối tượng

- Trong ngữ cảnh này, các luồng đối tượng được sử dụng để cho thấy cách thức các thực thể được tạo ra và sử dụng trong một luồng công việc. Các luồng đối tượng mô tả các đầu vào và đầu ra từ các trạng thái hoạt động trong một biểu đồ hoạt động. Có hai thành phần ký hiệu sau:
  - Trạng thái luồng đối tượng (object flow state): biểu diễn một đối tượng của một lớp tham gia vào luồng công việc được biểu diễn trong biểu đồ hoạt động. Đối tượng này có thế là đầu ra của một hoạt động và là đầu vào của nhiều hoạt động khác.
  - Luồng đối tượng (object flow) là một kiểu luồng điều khiển với một trạng thái luồng đối tượng làm đầu vào/đầu ra.
- Ký hiệu luồng đối tượng nói lên sự tồn tại của một đối tượng trong một trạng thái cụ thể, chứ không là chính đối tượng đó. Cùng một đối tượng này có thể được thao tác bởi một số các hoạt động kế tiếp nhau làm thay đổi trạng thái của đối tượng. Sau đó, nó có thể được hiển thị nhiều lần trong một biểu đồ hoạt động,

mỗi lần xuất hiện sẽ biểu diễn một trạng thái khác nhau trong đời sống của nó. Trạng thái của đối tượng tại mỗi thời điểm có thể được đặt trong ngoặc và viết thêm vào tên của lớp.

- Một trạng thái luồng đối tượng có thể xuất hiện như là trạng thái kết thúc của một luồng đối tượng (sự chuyển tiếp) và là trạng thái bắt đầu của nhiều luồng đối tượng (những sự chuyển tiếp).
- Các luồng đối tượng có thể được so sánh với các luồng dữ liệu bên trong luồng công việc của một use case. Không giống như các luồng dữ liệu truyền thống, các luồng đối tượng tồn tại ở một thời điểm xác định trong một biểu đồ hoạt động.

Sử dụng các lược đồ hợp tác (collaboration) và tuần tự (sequence)

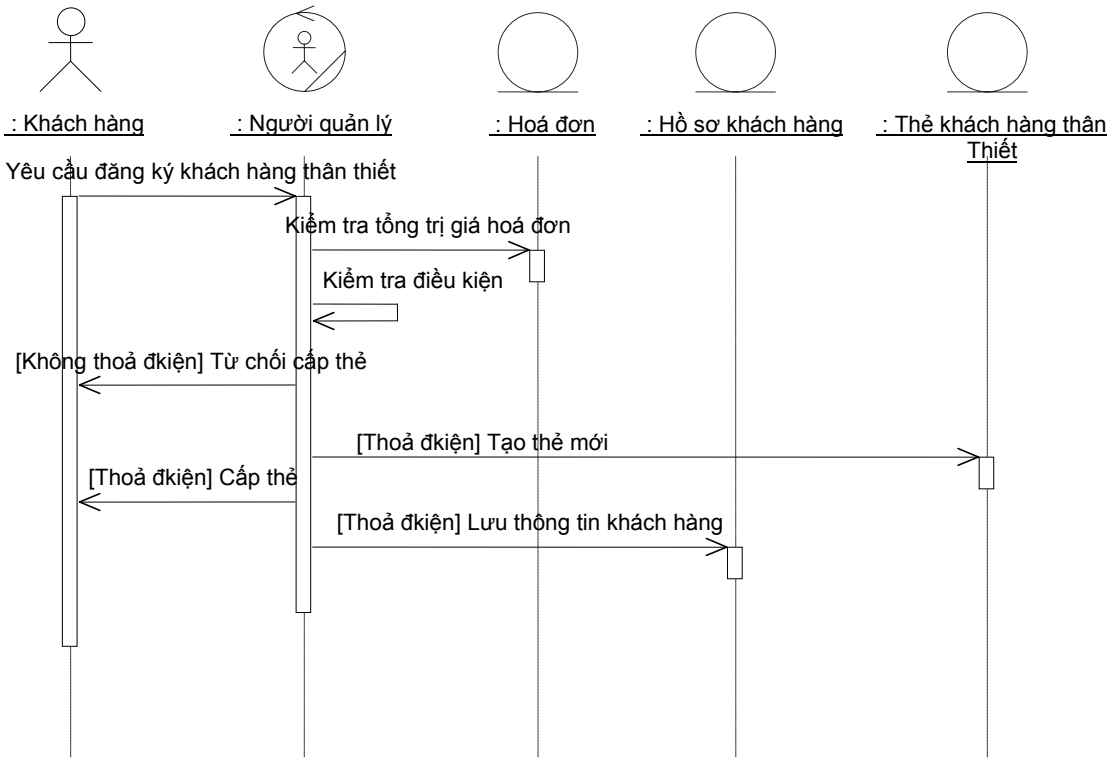
Đối với mỗi hiện thực hóa use-case, có thể có một hoặc nhiều lược đồ tương tác để mô tả các thừa tác viên và thực thể tham gia, cùng với những tương tác của chúng. Có hai loại lược đồ tương tác là: lược đồ tuần tự và lược đồ hợp tác. Chúng diễn tả những thông tin tương tự nhau, nhưng trình bày những thông tin này theo những cách khác nhau:

- Lược đồ tuần tự mô tả rõ ràng trình tự các sự kiện. Với các kịch bản phức tạp, thì lược đồ tuần tự thích hợp hơn so với các lược đồ hoạt động.
- Lược đồ hợp tác trình bày các mối liên kết giao tiếp và những thông điệp giữa các đối tượng. Chúng phù hợp hơn trong việc giúp ta hiểu được tất cả các tác động trên một đối tượng cho trước.
- Nếu có ít luồng rẽ nhánh, nhưng có nhiều thực thể tham gia, thì lược đồ tương tác thường là một sự lựa chọn tốt hơn so với lược đồ hoạt động nhằm để trình bày hiện thực hóa của luồng công việc.

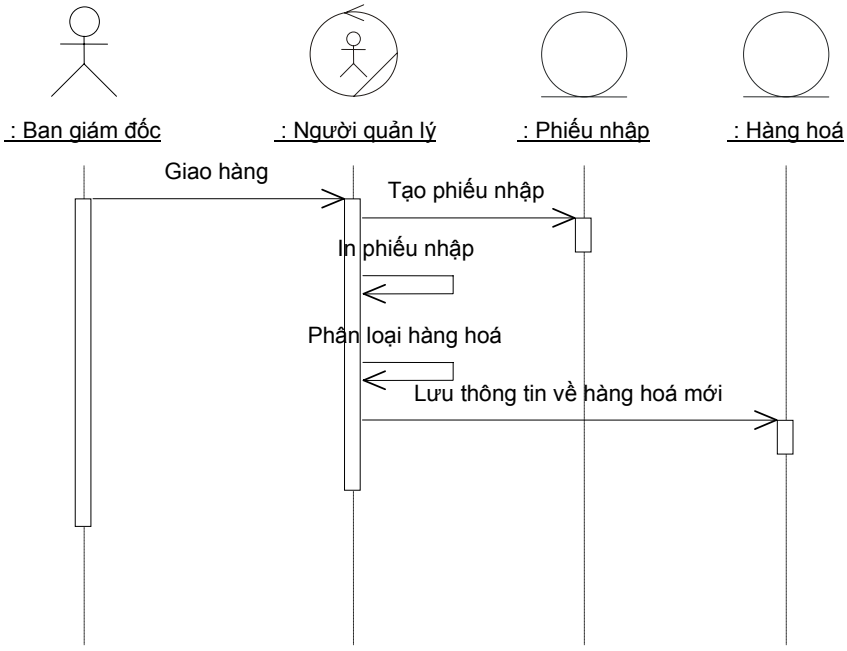
Lược đồ tuần tự mô tả một mẫu tương tác giữa các đối tượng, được sắp xếp theo thứ tự thời gian; nó cho thấy các đối tượng tham gia vào sự tương tác theo những "lifeline" và những thông điệp mà chúng gửi cho nhau.

Về mặt đồ họa, một lược đồ tuần tự mô tả chi tiết sự tương tác giữa các thừa tác viên, tác nhân, và cách thức các thực thể được truy xuất khi một use case được thực thi. Một lược đồ tuần tự mô tả vắn tắt các thừa tác viên tham gia làm những gì, và cách thức các thực thể được thao tác thông qua những sự kích hoạt, và cách thức chúng giao tiếp bằng cách gửi thông điệp cho nhau.

Ví dụ: Sơ đồ tuần tự của use case Quản lý khách hàng thân thiết



Sơ đồ tuần tự của use case Quản lý nhập hàng

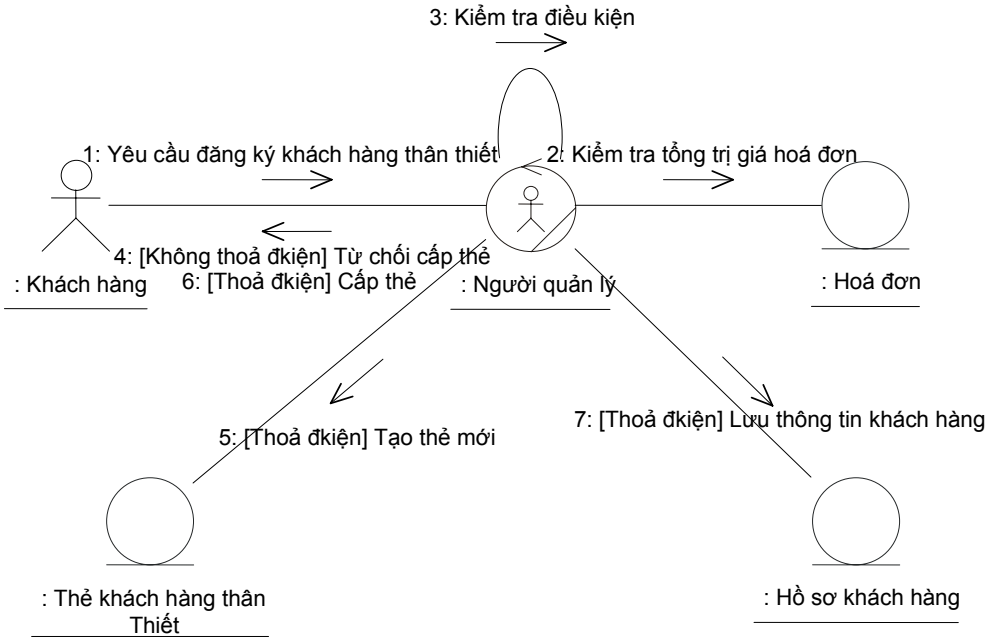


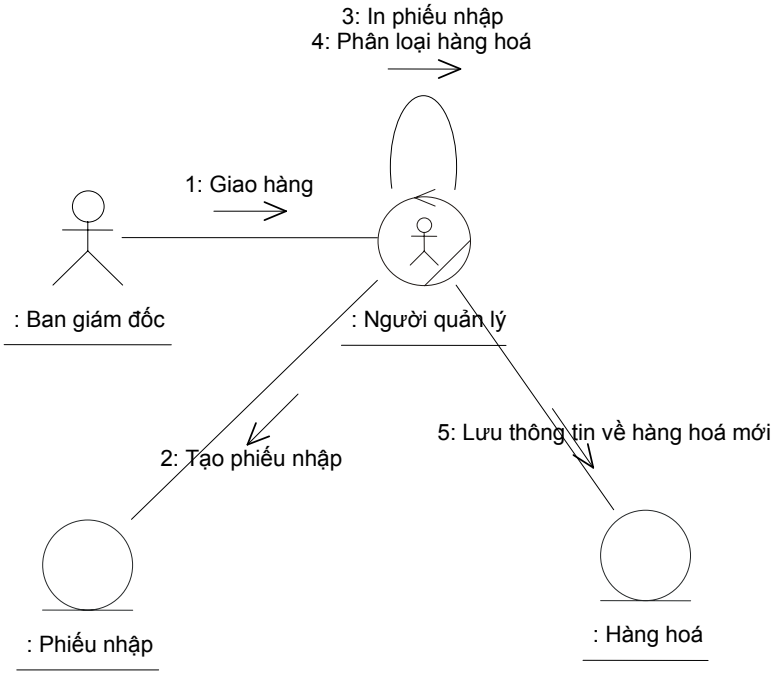
Những thông tin được tìm thấy trong một lược đồ tuần tự cũng có thể được biểu diễn trong một lược đồ hợp tác.

Một lược đồ hợp tác mô tả một mẫu tương tác giữa các đối tượng; nó cho thấy các đối tượng tham gia vào sự tương tác thông qua những mối liên kết giữa chúng và những thông điệp mà chúng gửi cho nhau.

Một lược đồ hợp tác về mặt ngữ nghĩa cũng tương tự như một lược đồ tuần tự, nhưng tập trung chủ yếu vào các đối tượng, trong khi lược đồ tuần tự tập trung vào các tương tác. Một lược đồ tuần tự trình bày một tập con các đối tượng có liên quan đến chuỗi công việc bị ảnh hưởng, bao gồm các mối liên kết giữa chúng, các thông điệp và các chuỗi thông điệp.

Ví dụ: lược đồ hợp tác tương ứng



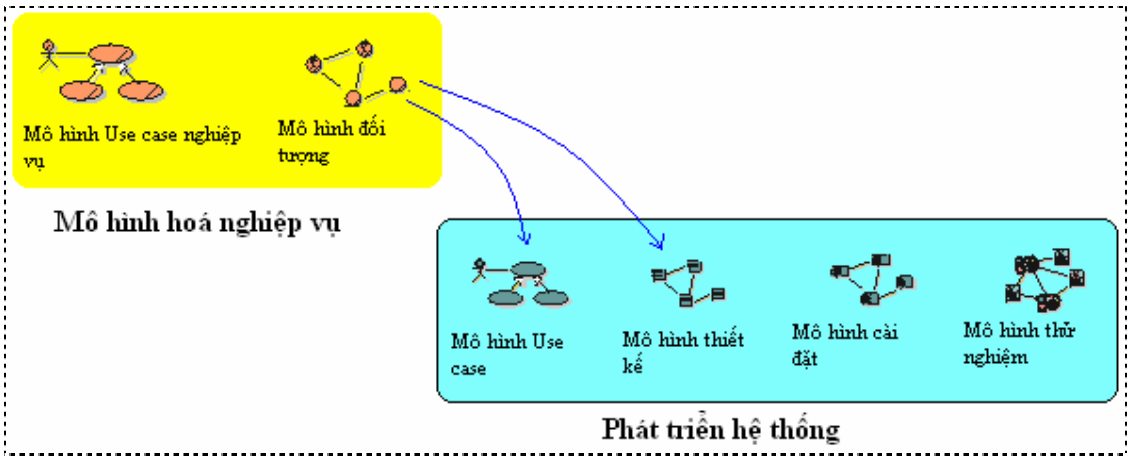


Xác định yêu cầu tự động hoá

Mục đích:

- Hiểu được cách thức sử dụng các công nghệ mới cải thiện hoạt động hiệu quả của tổ chức.
- Xác định mức độ tự động hóa trong tổ chức.
- Thiết lập các yêu cầu hệ thống từ những kết quả mô hình hóa nghiệp vụ.

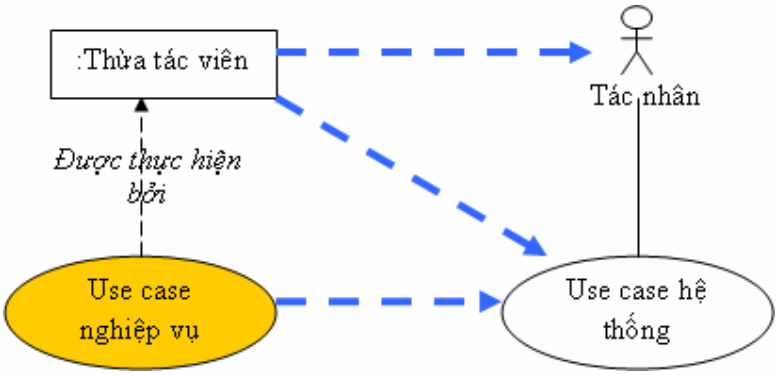
Xác định tác nhân và use case hệ thống phần mềm



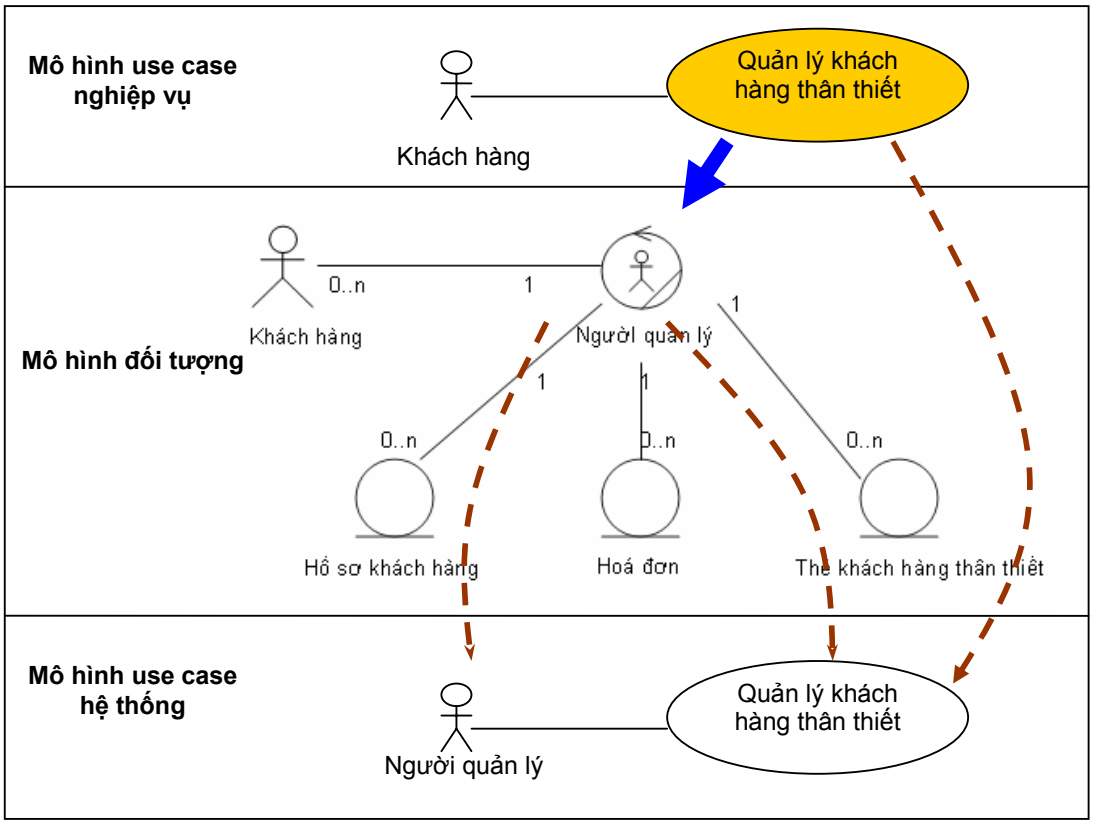
Để xây dựng các hệ thống, cần phải hiểu rõ các qui trình nghiệp vụ. Thậm chí sẽ hữu ích hơn nếu sử dụng các vai trò và trách nhiệm của nhân viên, cũng như những gì được xử lý bởi nghiệp vụ làm nền tảng để xây dựng hệ thống. Điều này được xác định từ góc nhìn bên trong nghiệp vụ dựa vào mô hình đối tượng, trong đó có thể thấy được mối liên kết chặt chẽ nhất đến hình thức thể hiện các mô hình của hệ thống.

Để xác định các use case trong hệ thống thông tin, hãy bắt đầu từ các thừa tác viên trong mô hình đối tượng nghiệp vụ. Đối với mỗi thừa tác viên, thực hiện những bước sau đây:

- Xác định xem thừa tác viên sẽ sử dụng hệ thống thông tin không?
- Nếu có, xác định một tác nhân cho thừa tác viên đó của hệ thống thông tin trong mô hình use-case của hệ thống thông tin. Đặt tên tác nhân với tên của thừa tác viên.
- Đối với mỗi use case nghiệp vụ mà thừa tác viên tham gia, tạo một use case hệ thống và mô tả vắn tắt.
- Xem xét các mục tiêu về tốc độ thực thi hay những thông tin bổ sung cho thừa tác viên cần được chú thích như là một yêu cầu đặc biệt của use case hệ thống, hoặc nhập vào sưu liệu đặc tả bổ sung của hệ thống.
- Lập lại những bước này cho tất cả các thừa tác viên.



Ví dụ: Xác định tác nhân và use case hệ thống phần mềm cho use case nghiệp vụ Quản lý khách hàng thân thiết



Trong trường hợp này, khách hàng giao tiếp với người quản lý để giải quyết các yêu cầu về khách hàng thân thiết. Người quản lý sẽ sử dụng phần mềm như là một công cụ nhằm trợ giúp trong các xử lý đáp ứng cho yêu cầu của khách hàng. Do đó, người quản lý lúc này sẽ là tác nhân của hệ thống phần mềm và các chức năng được tự động hoá trong use case nghiệp vụ Quản lý khách hàng thân thiết sẽ trở thành use case mô tả chức năng phần mềm hệ thống.

Nếu mục đích của việc xây dựng một hệ thống là tự động hóa hoàn toàn các qui trình nghiệp vụ (chẳng hạn như việc xây dựng một ứng dụng thương mại điện tử) thì thừa tác viên sẽ không trở thành tác nhân hệ thống nữa. Thay vào đó, chính tác nhân của môi trường nghiệp vụ giao tiếp trực tiếp với hệ thống và hoạt động như một tác nhân hệ thống.

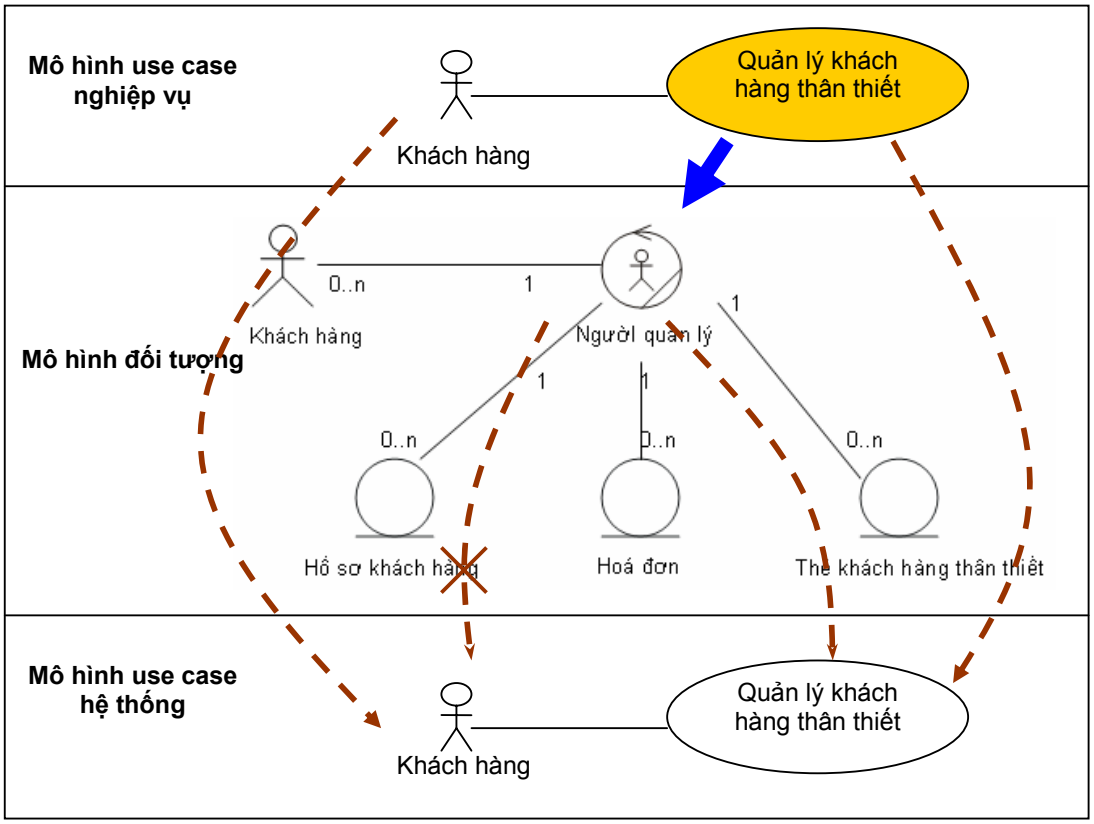
Khi đó, cách thức thực hiện nghiệp vụ sẽ bị thay đổi khi xây dựng một ứng dụng thuộc loại này. Các trách nhiệm của thừa tác viên sẽ chuyển sang tác nhân nghiệp vụ.

Ví dụ: giả sử xây dựng một site thương mại điện tử cho một việc quản lý khách hàng thân thiết, ta sẽ thay đổi cách thức mà qui trình được hiện thực hóa. Lúc này, Khách hàng sử dụng trực tiếp hệ thống thông qua việc truy cập ứng dụng web.

Các trách nhiệm của thừa tác viên Người quản lý sẽ chuyển sang tác nhân nghiệp vụ Khách hàng.

Tạo ra tác nhân hệ thống Khách hàng tương ứng với tác nhân nghiệp vụ Khách hàng.

Loại bỏ đi Người quản lý.

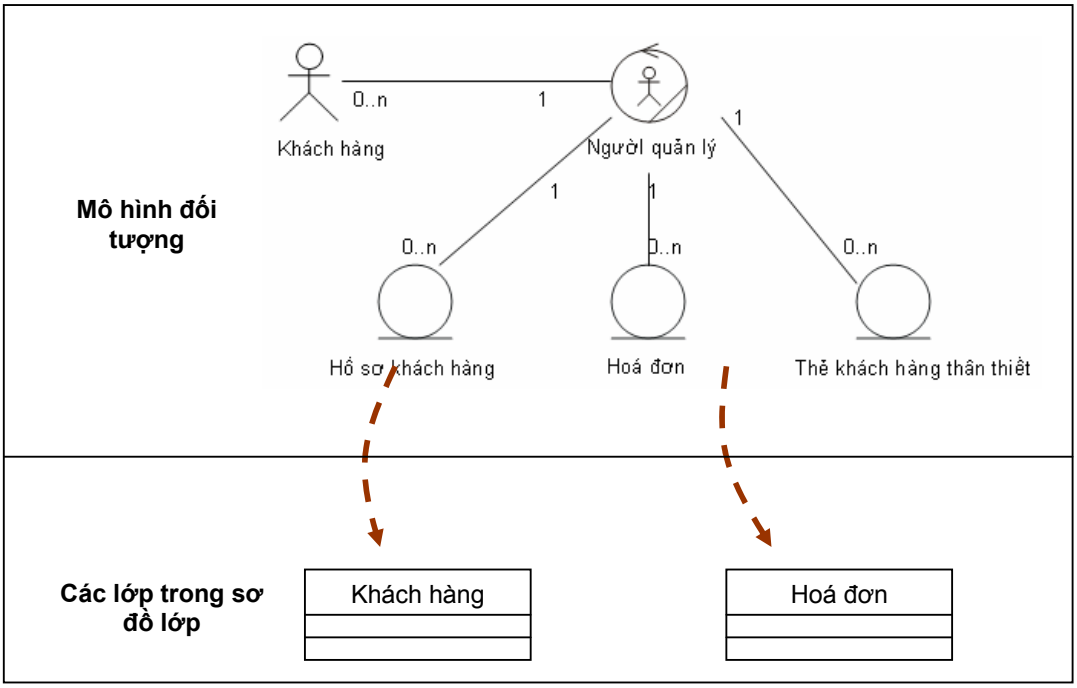


Xác định các lớp đối tượng thông tin trong hệ thống phần mềm

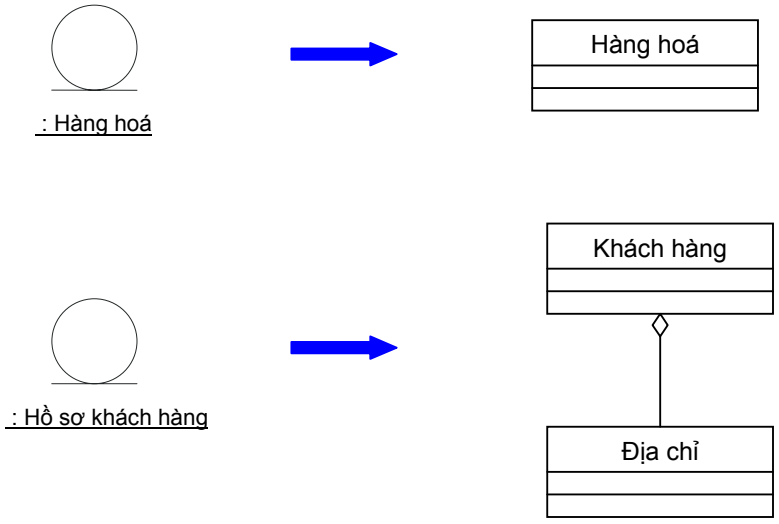
Một thực thể nghiệp vụ được quản lý bởi hệ thống thông tin sẽ tương ứng với một thực thể trong mô hình phân tích của hệ thống thông tin. Tuy nhiên, trong một số trường hợp, sẽ thích hợp nếu để các thuộc tính của thực thể nghiệp vụ tương ứng với các thực thể trong mô hình hệ thống thông tin.

Nhiều thừa tác viên có thể truy xuất một thực thể nghiệp vụ. Do đó, các thực thể tương ứng trong hệ thống có thể tham gia vào một số use case hệ thống thông tin.

Ví dụ: các lớp đối tượng hệ thống phần mềm cho từ sơ đồ lớp của use case nghiệp vụ "Quản lý khách hàng thân thiết"



Việc xác định các lớp đối tượng phần mềm chính là việc xác định cấu trúc thông tin của đối tượng nghiệp vụ để quản lý trong hệ thống thông tin. Đây cũng có thể xem như công đoạn chi tiết hoá về mặt cấu trúc nhằm đạt được một cách nhìn luận lý về cấu trúc của một đối tượng nghiệp vụ theo góc độ tin học hoá cho đối tượng đó. Tuỳ theo độ phức tạp của cấu trúc thực thể mà chúng ta có những cách chuyển đổi đơn giản (một thực thể nghiệp vụ → một lớp) hoặc phức tạp (một thực thể nghiệp vụ → nhiều lớp). Cuối cùng, sau khi đã xác định được các lớp thì để đạt được một mô hình đầy đủ, chúng ta còn phải xác định các mối kết hợp giữa các lớp. Chi tiết của phần này được đề cập ở chương tiếp theo.



Bài tập

Mô hình hoá việc hiện thực hoá các use case nghiệp vụ đã được tìm ra trong hệ thống "Quản lý thuê văn phòng cao ốc" bằng cách:

- Đặc tả nội dung use case dùng mô hình hoạt động (không sử dụng swimlane)
- Xác định các thừa tác viên và các thực thể
- Xây dựng sơ đồ lớp cho use case nghiệp vụ
- Thiết kế chi tiết use case dùng:
    - Sơ đồ hoạt động (dùng swimlane để mô tả vai trò của thừa tác viên và các thực thể được tác động bởi các hoạt động)
    - Xây dựng các sơ đồ tương tác (sơ đồ tuần tự, sơ đồ hợp tác, sơ đồ trạng thái)
- Đề xuất một sơ đồ use case mô tả chức năng hệ thống phần mềm giúp cho việc tự động hoá hệ thống nghiệp vụ trên.

Chương 7

XÂY DỰNG SƠ ĐỒ LỚP ĐỐI TƯỢNG HỆ THỐNG

Mục tiêu:

- Các khái niệm về sự phân loại
- Tìm các lớp đối tượng với các phương pháp: cụm danh từ, phân loại đối tượng và sử dụng sơ đồ use case
- Xác định liên kết giữa các lớp
- Xác định thuộc tính và phương thức của lớp

Giới thiệu

Phân tích hướng đối tượng là một tiến trình mà qua đó chúng ta có thể định dạng được các lớp đóng một vai trò quan trọng nhằm đạt được mục tiêu và yêu cầu hệ thống. Mô hình hoá đối tượng là một tiến trình mà trong đó, các đối tượng trong một hệ thống thực được thể hiện bởi các đối tượng luận lý trong các sơ đồ và trong chương trình. Sự thể hiện trực quan này của các đối tượng và quan hệ giữa chúng cho phép dễ dàng hiểu về đối tượng của hệ thống. Tuy nhiên, việc xác định lớp là một công việc khó nhất bởi vì không có một cấu trúc lớp nào hoàn hảo cũng như không có một cấu trúc nào hoàn toàn đúng.

Trong phần dưới đây sẽ trình bày về cách để phát triển các mô hình đối tượng bằng cách xây dựng các sơ đồ lớp mô tả việc phân loại đối tượng hệ thống. Trước tiên, chúng ta sẽ ôn lại các khái niệm cơ bản của sơ đồ lớp. Sau đó, chúng ta sẽ được giới thiệu xây dựng sơ đồ lớp thông qua việc giới thiệu lần lượt về các cách tiếp cận để phân loại đối tượng và xác định lớp, cách xác định liên kết giữa các lớp cũng như thuộc tính và phương thức của lớp.

Sơ đồ lớp (Class diagram)

Các khái niệm

Đối tượng

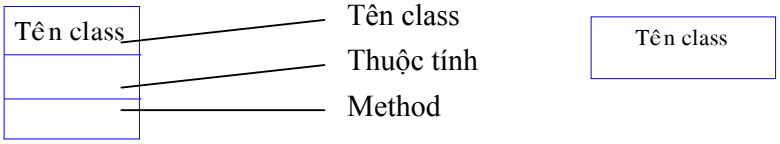
Trong tiếp cận hướng đối tượng, chúng ta mô hình hoá hệ thống bằng các đối tượng, nghĩa là nhìn hệ thống như là một đối tượng . Do đó, trước khi tiếp cận để mô hình hoá hệ thống. Chúng ta cần phải hiểu như thể nào là một đối tượng (object). Có nhiều nguồn mô tả hoặc định nghĩa về đối tượng, tuy nhiên trong tài liệu này chúng ta có thể tổng hợp lại như sau: một đối tượng là một thực thể có một vai trò xác định rõ ràng trong lãnh vực ứng dụng, có trạng thái, hành vi và định danh. Một đối tượng là một khái niệm, một sự trừu tượng hoá hoặc một sự vật có ý nghĩa trong phạm vi ngữ cảnh của hệ thống.

Đối tượng được có thể là một thực thể hữu hình, trực quan (như là: con người, vị trí, sự vật,…); có thể là một khái niệm, sự kiện (ví dụ: bộ phận, đặng ký, …); có thể là một khái niệm trong tiến trình thiết kế (như là: User interface, Controller, Scheduler,…)

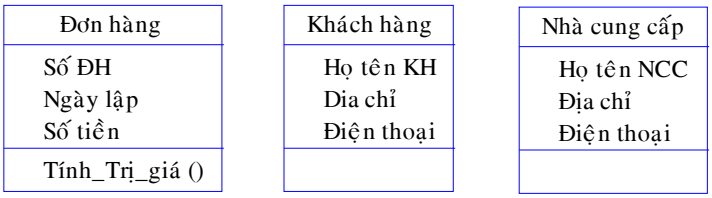
Lớp (class)

Là một tập hợp các đối tượng chia sẽ chung một cấu trúc và hành vi (cùng thuộc tính, hoạt động, mối quan hệ và ngữ nghĩa). Cấu trúc được mô tả bởi các thuộc tính và các mối quan hệ, còn hành vi được mô tả bởi các hoạt động. Một lớp là một sự trừu tượng hoá của các đối tượng thế giới thực, và các đối tượng tồn tại trong thế giới thực được xem như là các thể hiện của lớp.

Ký hiệu: lớp được trình bày gồm ba phần: tên lớp, danh sách các thuộc tính (attribute), danh sách các hoạt động (operation). Trong đó, phần thuộc tính và phần hoạt động có thể bị che dấu đi trong mức độ trình bày tổng quan.



Ví dụ: biểu diễn tập hợp các đơn hàng NGK, khách hàng mua NGK, nhà cung cấp NGK,… cùng chia sẽ chung thuộc tính, hoạt động, mối quan hệ và ngữ nghĩa thành các lớp:



Mối kết hợp (association)

Mối kết hợp nhị phân: là quan hệ ngữ nghĩa được thiết lập giữa hai hay nhiều lớp, biểu diễn bởi những thành phần sau:

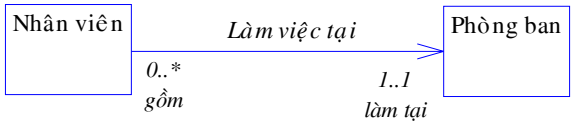
- Tên quan hệ: thường là cụm động từ phản ánh mục đích của mối kết hợp
- Vai trò quan hệ (role): là một phần của mối kết hợp dùng để mô tả ngữ nghĩa tham gia của một lớp vào mối kết hợp đó (không phải một phần của lớp). Mỗi quan hệ có thể có 2 vai trò (quan hệ nhị phân) hoặc nhiều hơn (quan hệ đa phân).
    - Tên vai trò: dùng động từ hoặc danh từ (cụm danh từ) để biểu diễn vai trò của các đối tượng. Trong mối kết hợp *làm việc tại* có hai vai trò, *làm tại* và *gồm* cho biết: nhân viên làm việc tại phòng ban và phòng ban gồm có các nhân viên trực thuộc.
    - Bản số: là cặp giá trị (mincard, maxcard) xác định khoảng giá trị cho phép một đối tượng của một lớp có thể tham gia bao nhiêu lần vào mối kết hợp với các đối tượng của các lớp khác.

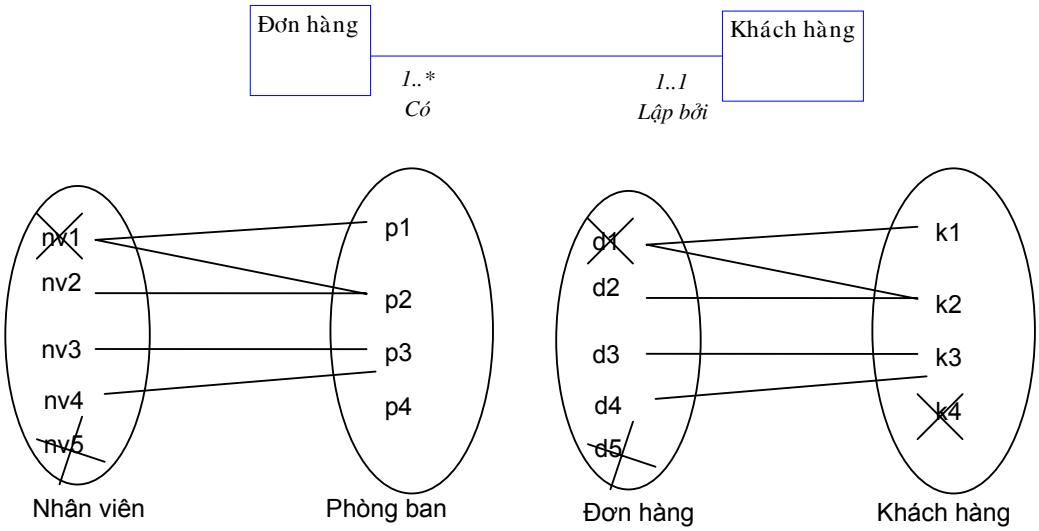
    Giá trị mincard: qui định về ràng buộc tối thiểu của một đối tượng tồn tại trong lớp phải tham gia vào mối kết hợp với một số lượng lớn hơn hoặc bằng.

    Giá trị maxcard: qui định số lượng tối đa mà một đối tượng của lớp nếu tồn tại trong lớp đó không được tham gia vào mối kết hợp vượt giá trị này.

    Bản số mối kết hợp dưới cho biết một nhân viên phải thuộc ít nhất và nhiều nhất (duy nhất) một phòng ban, tuy nhiên mỗi đối tượng phòng ban có thể tồn tại mà không có nhân viên làm việc thuộc phòng. Các mẫu bản số thường là:

    0..1, 1..1, 3..5, 0..*, 1..*, 2..*





Tổng quát, cho hai lớp C1, C2 và mối kết hợp A giữa chúng. Tùy theo giá trị bản số tối thiểu chúng ta có những trường hợp sau:

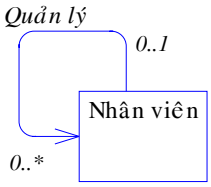
- Nếu mincard (C1,A) = 0 thì chúng ta nói rằng lớp C1 có sự tham gia tùy ý trong mối kết hợp bởi vì một đối tượng của lớp C1 có thể không tham gia kết hợp với đối tượng lớp C2 trong mối kết hợp A.
- Nếu mincard (C1,A) >0 thì chúng ta nói rằng lớp C1 có sự tham gia bắt buộc vào mối kết hợp bởi vì một đối tượng của lớp C1 phải bắt buộc tham gia kết hợp với ít nhất một phần tử của lớp C2 trong mối kết hợp A.

Tùy theo giá trị của bản số tối đa mà chúng ta có các trường hợp sau:

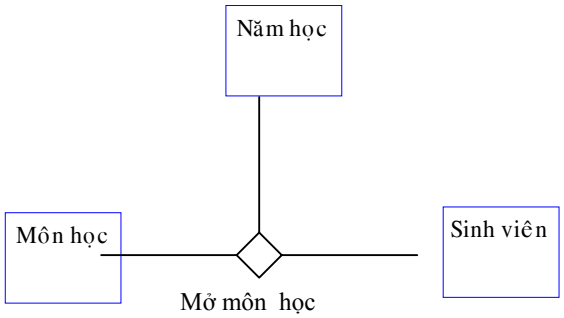
- Nếu maxcard(C1,A) = 1 và maxcard(C2,A) = 1 thì ta gọi là mối kết hợp một - một (one- to – one)
- Nếu maxcard(C1,A) = 1 và maxcard(C2,A) = n thì ta gọi là mối kết hợp một - nhiều (one- to – many)
- Nếu maxcard(C1,A) = n và maxcard(C2,A) = 1 thì ta gọi là mối kết hợp nhiều - một (many- to – one)
- Nếu maxcard(C1,A) = n và maxcard(C2,A) = n thì ta gọi là mối kết hợp nhiều - nhiều (many- to – many)

o Tính khả điều hướng (navigability): được mô tả bởi một mũi tên chỉ ra hướng truy xuất trong mối kết hợp từ một đối tượng của lớp đến một đối tượng của lớp còn lại. Tính khả điều hướng có thể là không có, hoặc chỉ một, hoặc cả hai. Ví dụ trên cho thấy chiều mũi tên trong mối kết hợp *làm việc tại* cho biết chúng ta có thể truy cập lớp *phòng ban* từ mối kết hợp, tuy nhiên, chúng ta không thể truy xuất tới lớp *nhân viên* từ mối kết hợp này. Hoặc trong mối kết hợp giữa lớp *Đơn hàng* và *Khách hàng*, hướng truy xuất là có thể cho cả hai lớp (không có chiều mũi tên)

- Mối kết hợp phản thân: một mối kết hợp có thể được thiết lập từ một lớp đến chính nó. Ví dụ mối kết hợp *quản lý* được thiết lập giữa lớp *Nhân viên* tới chính nó cho biết một nhân viên quản lý những nhân viên khác.

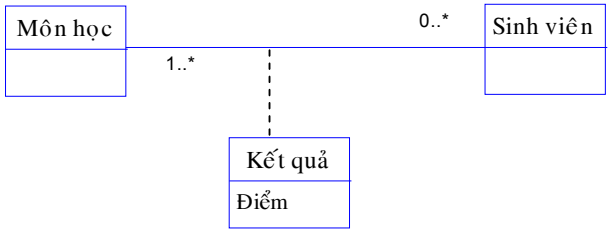


- Mối kết hợp đa phân: là mối kết hợp được thiết lập từ ba lớp trở lên. Ký hiệu mối kết hợp đa phân là một hình thoi với hơn ba vai trò nối tới các lớp tham gia. Đây là mối kết hợp được thừa hưởng từ cách tiếp cận truyền thống của mô hình thực thể - kết hợp (ER). Tuy nhiên, mối kết hợp đa phân không cho phép quan hệ thu nạp, bản số phức tạp. Do đó, trong cách sử dụng chúng ta thường thay thế nó bằng một lớp và đưa về mối kết hợp nhị phân.

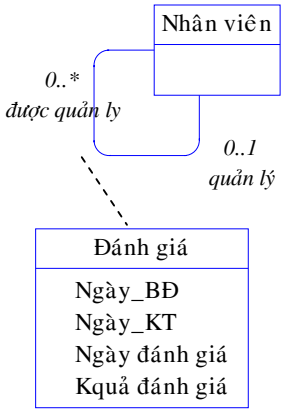


Lớp kết hợp

Khi một mối kết hợp có các đặc trưng (thuộc tính, hoạt động, và các mối kết hợp), chúng ta tạo một lớp để chứa các thuộc tính đó và kết nối với mối kết hợp, lớp này được gọi là lớp kết hợp. Tên của lớp này chính là tên của mối kết hợp, trong trường hợp lớp này có thuộc tính nhưng không có hoạt động hoặc bất kỳ mối kết hợp nào khác, thì tên của mối kết hợp vẫn duy trì trên mối kết hợp và để trống phần tên của lớp này để duy trì tính tự nhiên của nó.

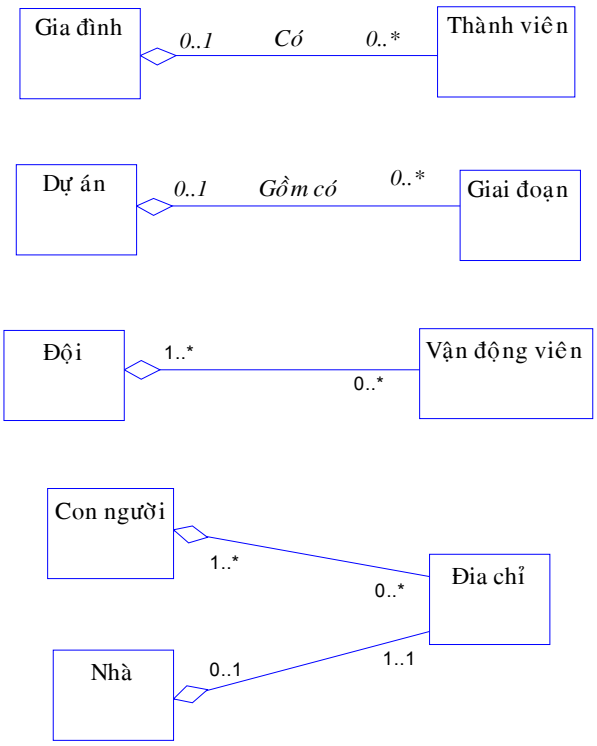


Trường hợp phổ biến nhất của lớp kết hợp là mối kết hợp nhiều - nhiều. Ví dụ trên cho thấy, sinh viên tham gia các môn học khác nhau và mỗi lần đăng ký học, sinh viên sẽ có một kết quả được ghi nhận bởi điểm thi. Vậy điểm thi là một thuộc tính được hình thành thông qua việc tham gia học tập của một sinh viên trên một môn học nên nó là thuộc tính của mối kết hợp.



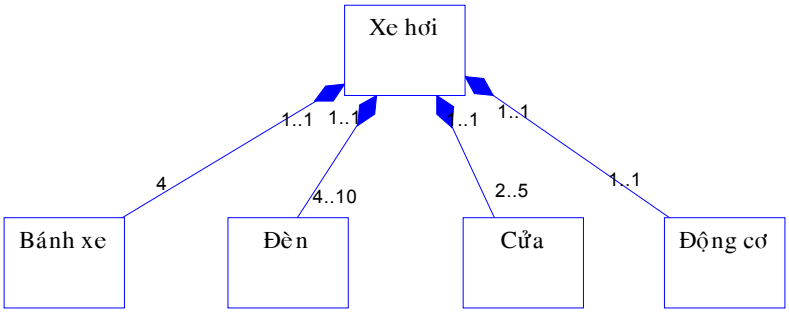
Quan hệ thu nạp (aggregation) và quan hệ thành phần (composition, a-part-of)

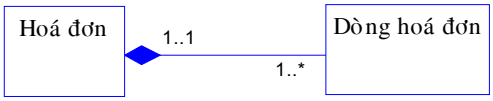
- **Quan hệ thu nạp (aggregation):** mô tả mối quan hệ giữa một đối tượng lớn hơn được tạo ra từ những đối tượng nhỏ hơn. Một loại quan hệ đặc biệt này là quan hệ "*có*", nó có nghĩa là một đối tượng tổng thể có những đối tượng thành phần. Ví dụ dưới đây cho thấy, *Gia đình* là một đối tượng tổng thể có những *Thành viên* trong gia đình.



Một đối tượng thành phần cũng có thể tham gia kết hợp với nhiều đối tượng tổng thể khác nhau, trường hợp này gọi là chia sẽ. Ví dụ một vận động viên có quan hệ tới một đội với ý nghĩa là một phần tử của đội, tuy nhiên vận động viên này cũng có thể thành viên của một đội khác, trường hợp này gọi là sự chia sẽ. Do đó, nếu một đội bị hủy bỏ, thì không nhất thiết phải huỹ bỏ vận động viên này.

- **Quan hệ thành phần (composition)** là một loại đặc biệt của quan hệ thu nạp, nó có một sự liên hệ mạnh mẽ hơn để trình bày thành phần của một đối tượng phức hợp. Quan hệ thành phần cũng được xem như là quan hệ thành phần - tổng thể (part-whole), và đối tượng tổng hợp sẽ quản lý việc tạo lập và huỷ bỏ của những đố tượng thành phần của nó.



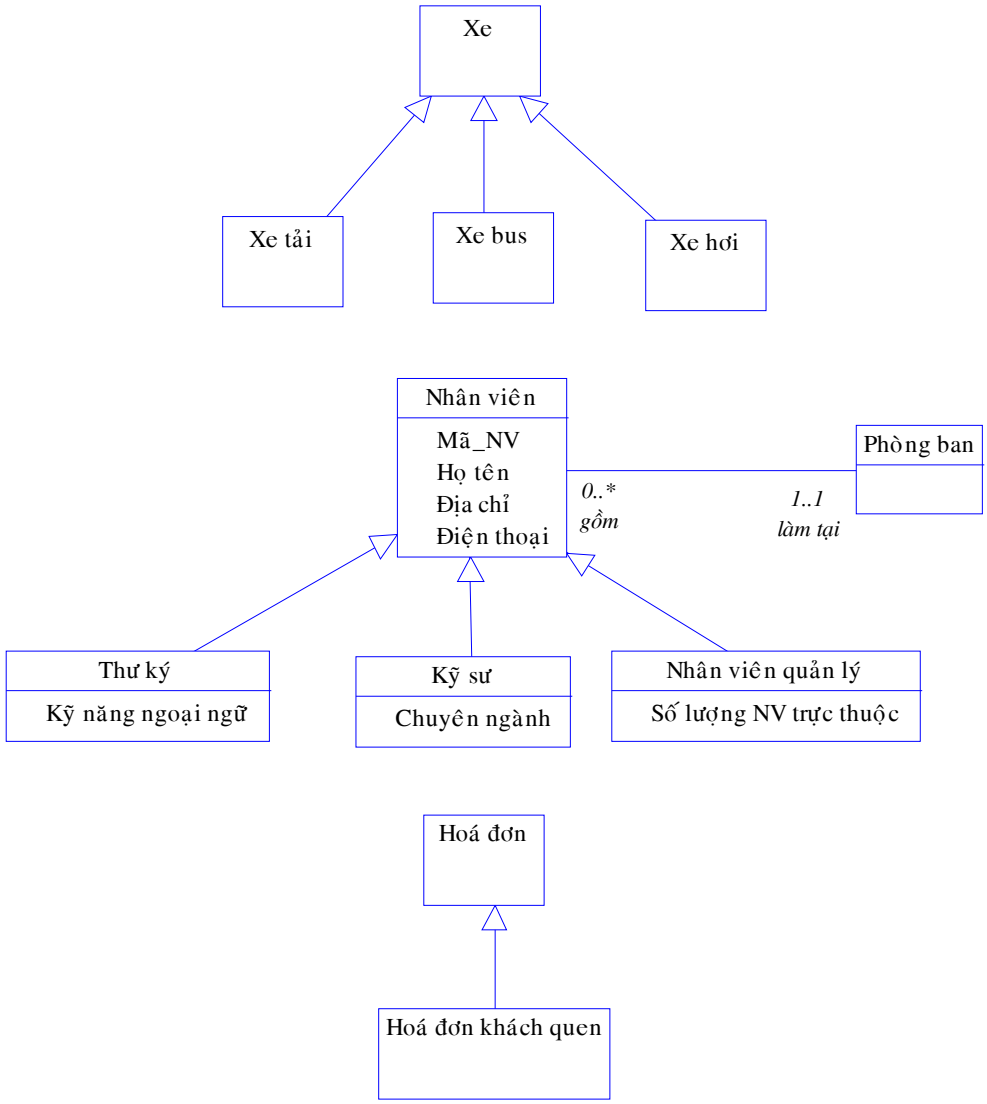


Như vậy, quan hệ thành phần mô tả sự phụ thuộc rất chặt chẽ giữa lớp tổng thể đến lớp thành phần về sự phụ thuộc. Nghĩa là các lớp thành phần là một bộ phận cấu tạo nên lớp tổng thể và thể hiện vật lý của nó là nằm trong lớp tổng thể.

Ví dụ trên cho thấy, một chiếc xe hơi được làm nên bởi những bánh xe, đèn, cửa, động cơ, … việc tạo thành một chiếc xe hơi là việc lắp ráp các thành phần này. Cũng như hoá đơn chứa các dòng hoá đơn trong đó, một hoá đơn bị huỷ nghĩa là các đòng của hóa đơn đó cũng sẽ bị huỷ theo.

Quan hệ tổng quát hóa

Là quan hệ được thiết lập giữa một lớp tổng quát hơn đến một lớp chuyên biệt. Quan hệ này dùng để phân loại một tập hợp đối tượng thành những loại xác định hơn mà hệ thống cần làm rõ ngữ nghĩa.



Ví dụ trên đây chỉ ra rằng, tất cả các lớp chuyên biệt Thư ký, Kỹ sư, Nhân viên quản lý đều có thể kế thừa các thuộc tính (Mã_NV, Họ tên, Địa chỉ, Điện thoại) của lớp tổng quát Nhân viên và mối kết hợp giữa lớp Nhân viên với Phòng ban.

Trong mối kết hợp tổng quát hoá, một thể hiện của lớp chuyên biệt cũng là một thể hiện của lớp tổng quát. Ví dụ trên cho thấy một đối tượng Kỹ sư, hoặc Thư ký, hoặc Nhân viên quản lý đều là một đối tượng của lớp Nhân viên. Vì lý do đó, đặc trưng của loại kết hợp này là tính **kế thừa**, một lớp chuyên biệt có thể kế thừa tất cả các đặc trưng (thuộc tính, mối kết hợp, hoạt động) của lớp tổng quát.

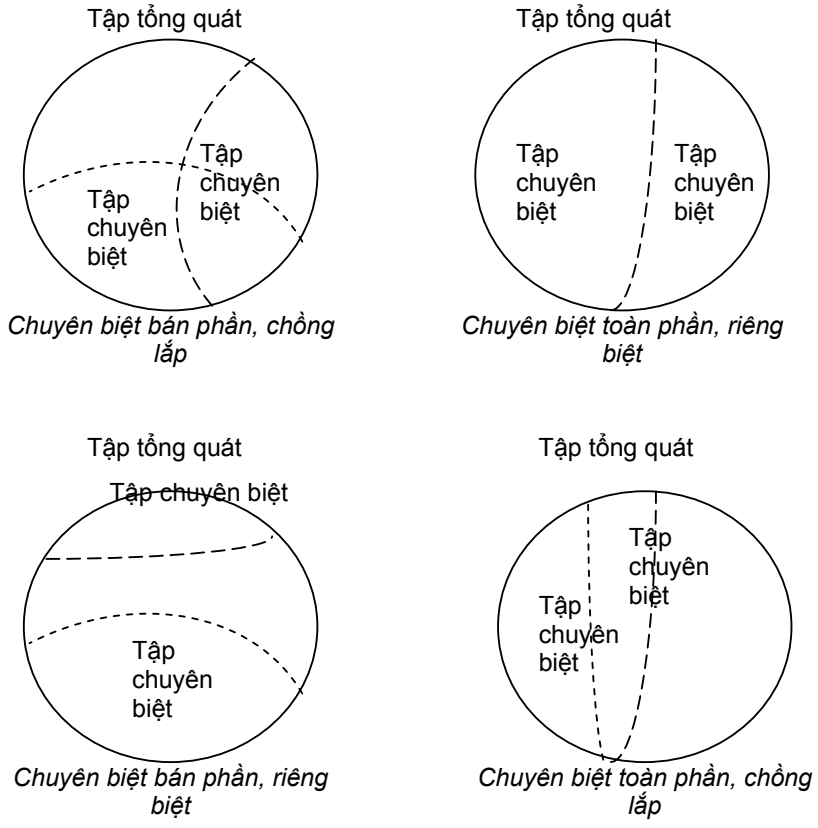
Sự tương quan của các lớp trong quan hệ tổng quát hoá

Sự tương quan giữa các lớp chuyên biệt với lớp tổng quát:

- Tập hợp các đối tượng của tất cả các lớp chuyên biệt phủ toàn bộ tập đối tượng của lớp tổng quát thì gọi là toàn phần (complete).
- Tập hợp các đối tượng của tất cả các lớp chuyên biệt không phủ toàn bộ tập đối tượng của lớp tổng quát thì gọi là bán phần (incomplete).

Sự tương quan giữa các lớp chuyên biệt:

- Không tồn tại một đối tượng của lớp tổng quát thuộc hai lớp chuyên biệt trở lên thì gọi là riêng biệt (disjoint).
- Tồn tại một đối tượng của lớp tổng quát thuộc hai lớp chuyên biệt trở lên thì gọi là chồng lắp (overlapping).



Hình 3. Sự tương quan giữa các lớp trong quan hệ tổng quát hoá

Như vậy, trong quan hệ tổng quát hoá, sự tương quan giữa các lớp được biểu diễn quan bốn trường hợp (bán phần - chồng lắp, bán phần - riêng biệt, toàn phần - chồng lắp, toàn phần – riêng biệt). Sự tương quan này phản ánh ràng buộc ngữ nghĩa trong tập hợp các đối tượng của quan hệ: một đối tượng của lớp chuyên biệt này có thể là đối tượng trong lớp chuyên biệt khác hay không? Và một đối tượng trong lớp tổng quát có thể không thuộc một lớp chuyên biệt nào hay không?.

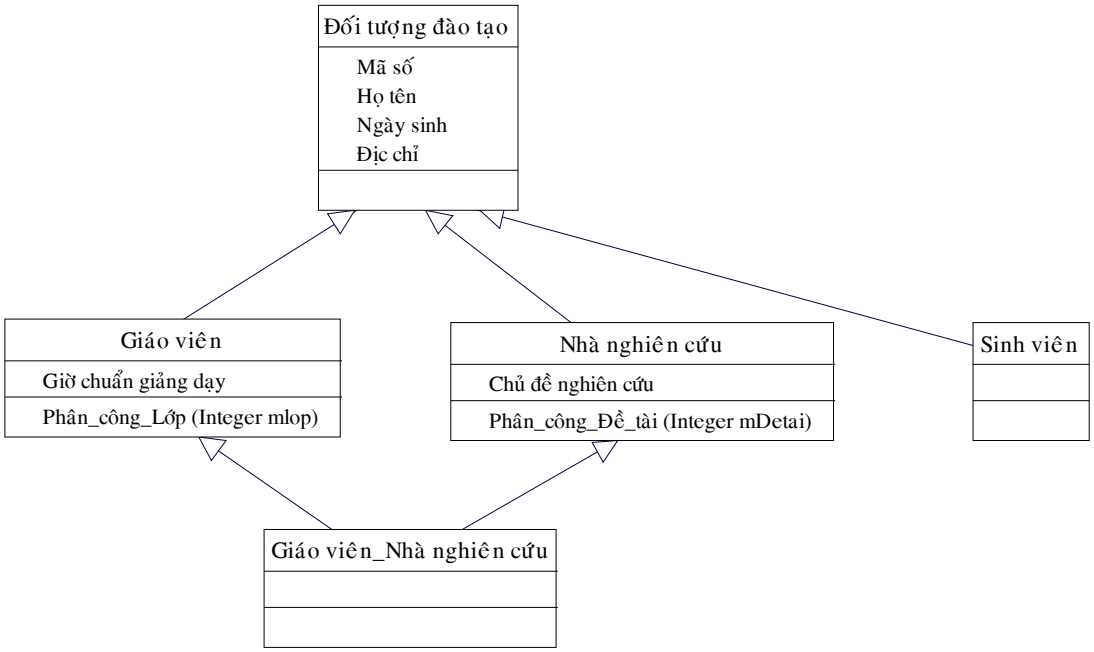
Ví dụ, quan hệ tổng quát hoá giữa Xe – Xe tải, Xe bus, Xe hơi có sự tương quan là bán phần – riêng biệt (incomplete, disjoint). Quan hệ giữa Nhân viên – Thư ký, Kỹ sư, Nhân viên quản lý có sự tương quan là bán phần - chồng lắp (incomplete, overlapping).

Đa kế thừa

Đa số các trường hợp trong quan hệ tổng quát hoá là đơn kế thừa, nơi mà một lớp là chuyên biệt duy nhất cho một lớp tổng quát. Trong một số trường hợp đặc biệt chúng ta cũng thấy một lớp chuyên biệt có thể kế thừa từ hai hoặc nhiều lớp tổng quát. Trường hợp này gọi là đa kế thừa.

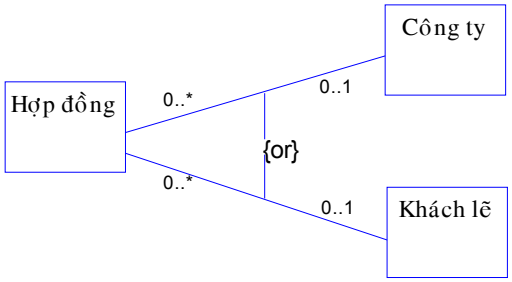
Ví dụ sau cho thấy, lớp Giáo viên_Nhà nghiên cứu là lớp đa kế thừa từ hai lớp Giáo viên và lớp Nhà nghiên cứu. Lớp này sẽ thừa kế tất cả các đặc trưng như: Giờ chuẩn giảng dạy, Chủ đề nghiên cứu, Phân_công_Lớp và Phân_công_Đề_tài của hai lớp trên.

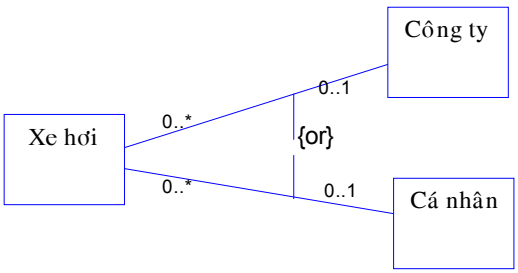
Tuy nhiên, theo lời khuyên của các chuyên gia thì không nên sử dụng đa kế thừa vì tính chất phức tạp của nó. Do đó, đa kế thừa không được đưa vào ngôn ngữ UML gốc và một số ngôn ngữ hướng đối tượng khác.



Quan hệ hoặc (OR)

Là mối quan hệ xác định một tình huống mà trong đó hai (hoặc nhiều) lớp tham gia mối kết hợp với một lớp thứ ba với ràng buộc loại trừ. Một thể hiện của lớp thứ ba sẽ tham gia kết hợp loại trừ với các đối tượng của hai lớp kia (hoặc là không kết hợp, hoặc kết hợp chỉ các đối tượng của một trong hai lớp) tại một thời điểm. Ví dụ dưới đây cho thấy, một hợp đồng có thể được lập bởi một công ty hoặc bởi một khách hàng lẽ. Hoặc một chiếc xe hơi thì được sở hữu bởi một cá nhân hoặc bởi một công ty, không sở hữu một lúc bởi cả hai.





Thực chất của mối kết hợp OR cũng chính là một ràng buộc ngữ nghĩa giữa các lớp tham gia kết hợp với một lớp thứ ba: sự hiện diện tham gia của đối tượng này sẽ không cho phép sự tham gia của đối tượng kia và ngược lại.

Thuộc tính (attribute)

Thuộc tính dùng để mô tả đặc trưng của đối tượng, người ta có thể chia thuộc tính thành ba loại sau:

- Thuộc tính đơn trị: là thuộc tính chỉ có một giá trị duy nhất cho một đối tượng, đây là thuộc tính phổ biến nhất. Ví dụ: họ tên, ngày sinh, lương,…
- Thuộc tính đa trị: là thuộc tính có thể có nhiều giá trị cho một đối tượng. Ví dụ: nếu chúng ta muốn lưu nhiều số điện thoại của một nhân viên, chúng ta có thể đặt thuộc tính số điện thoại trong lớp nhân viên là đa trị.
- Thuộc tính tham chiếu.

Biểu diễn một thuộc tính

Thuộc tính được biểu diễn gồm những thành phần như sau:

<phạm vi> <tên thuộc tính> : <biểu thức kiểu> = <giá trị khởi tạo>

<phạm vi>: nhận một trong những giá trị sau:

+ toàn cục (có thể truy cập bởi tất cả lớp)

bảo vệ (có thể truy cập bởi lớp và lớp chuyên biệt của nó)

- riêng (chỉ được truy cập bởi lớp)

Bản số: là một cặp (số tối thiểu, số tối đa) mà thuộc tính có thể có giá trị.

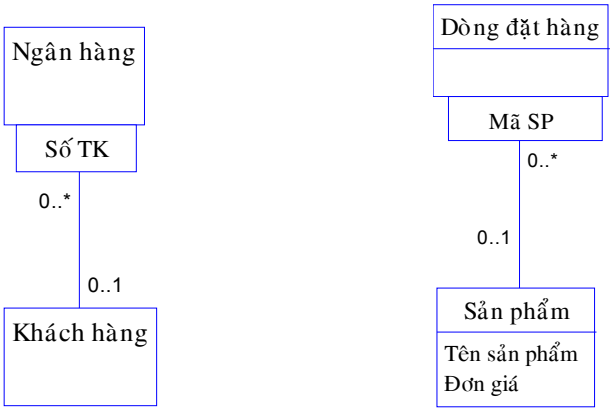
- số tối thiểu= 0 → thuộc tính được gọi là không bắt buộc.
- số tối thiểu = 1 → thuộc tính được gọi là bắt buộc.
- số tối đa = 1 → thuộc tính đơn trị.
- số tối đa > 1 → thuộc tính đa trị

Ví dụ: *Số điện thoại[0..*]*: *string*, *Địa chỉ[0..1]: string*,…

Trong giai đoạn phân tích việc xác định thuộc tính thường chỉ bao gồm xác định tên thuộc tính (có thể thêm kiểu dữ liệu), các đặc điểm khác của thuộc tính sẽ được xác định lại trong giai đoạn thiết kế.

Thuộc tính quan hệ (Qualifier)

Là một thuộc tính quan hệ (không phải của lớp). Nghĩa là thuộc tính này sẽ hình thành từ một quan hệ giữa các lớp. Nhằm thực hiện việc thiết lập mối liên kết giữa một tập thể hiện với một thể hiện khác. Một đối tượng và một giá trị của thuộc tính qualifier sẽ xác lập một định danh duy nhất, hình thành nên khoá phức hợp.



Xem xét mối kết hợp giữa lớp Sản phẩm và lớp Dòng đặt hàng của đơn đặt hàng. Một dòng trong đơn hàng sẽ liên quan đến một sản phẩm sẽ được đặt, mỗi dòng đơn hàng tham chiếu đến duy nhất một sản phẩm, trong khi đó mỗi sản phẩm có thể được đặt trong nhiều dòng đơn hàng của những đơn hàng khác nhau. Mối liên kết qualifier có thuộc tính Mã SP cho biết: mỗi sản phẩm có duy nhất một Mã SP và Dòng đơn hàng liên kết với Sản phẩm sử dụng Mã SP.

Định danh (identifier)

Định danh là một hoặc nhiều thuộc tính của lớp có giá trị xác định duy nhất cho một đối tượng của lớp.

Các cách tiếp cận xác định lớp đối tượng

Gần như không có một phương thức chung để xác định các lớp của một hệ thống. Thông thường đây là một quá trình sáng tạo và lặp lại qua nhiều vòng lặp và cần phải có sự thống nhất với các chuyên gia trong lãnh vực ứng dụng nghiệp vụ. Có nhiều phương pháp tiếp cận để xác định lớp. Có phương pháp đề nghị tiến hành mô hình hoá nghiệp vụ, chỉ ra phạm vi bài toán nghiệp vụ sẽ được tự động hoá mà kết quả của nó sẽ cung cấp các lớp cho hệ thống tương ứng với việc tự động hoá việc quản lý, lưu trữ các đối tượng thông tin (thực thể) đã được thống nhất trên các yêu cầu với người dùng xử lý nghiệp vụ. Có phương pháp để xuất xác định tất cả các lớp thuộc phạm vi bài toán, mối quan hệ của chúng. Sau đó, sẽ phân tích use case và phân bổ trách nhiệm các lớp theo use case. Có phương pháp đề xuất lấy mô hình use case làm nền tảng để tìm lớp (use case - driven), và trong quá trình xác định trách nhiệm thực hiện của use case thì các lớp sẽ được tìm thấy. Vì quá trình xác định lớp trong giai đoạn này là một quá trình lặp lại mà kết quả của bước sau có thể làm thay đổi các kết quả của bước trước, cho nên các lớp được tìm thấy thường được gọi là lớp ứng viên (candidate class).

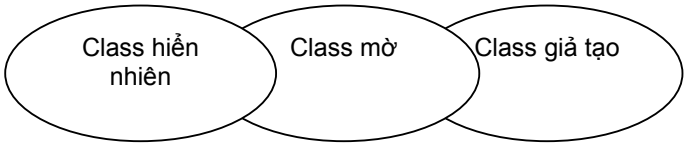
Dưới đây chúng ta đề cập đến một số kỹ thuật tiếp cận để xác định lớp: tiếp cận theo cụm danh từ; tiếp cận theo mẫu chung; và tiếp cận theo use case.

Tiếp cận theo cụm danh từ (noun phrase)

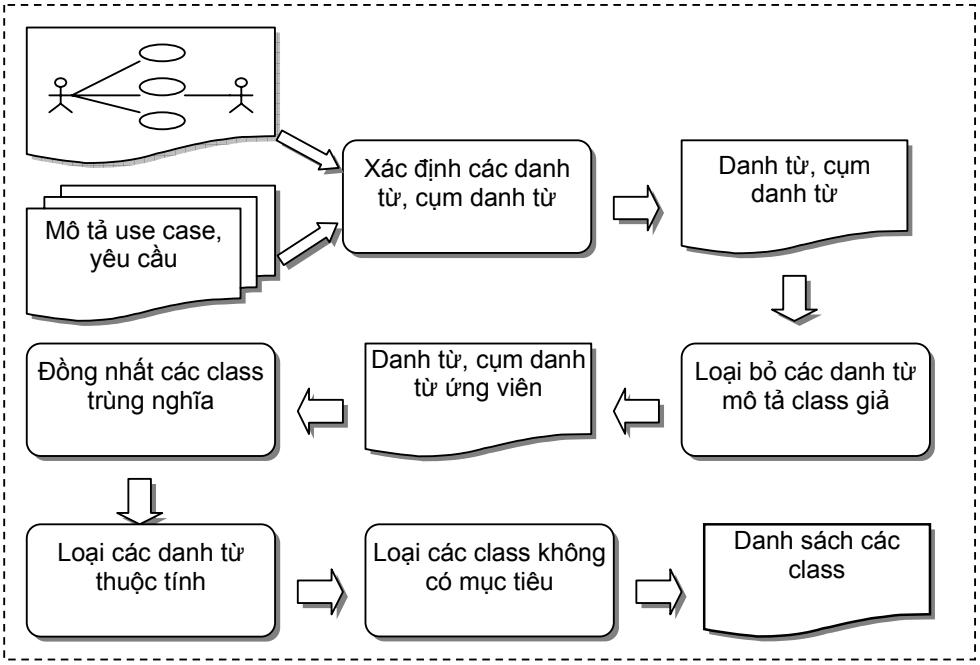
Phương pháp tiếp cận theo cụm danh từ được đề xuất bởi Rebecca Wirfs-Brock, Brian Wilkerson, và Lauren Wiener. Phương pháp đề xuất việc xác định các lớp thông qua việc đọc trong các văn bản mô tả use case hoặc các mô tả yêu cầu để tìm kiếm và trích lọc các cụm danh từ. Các cụm danh từ có thể được xem là các ứng viên của các lớp và các động từ là các ứng viên của phương thức (method) của lớp. Tất cả danh từ hoặc cụm danh từ tìm được sẽ được phân thành ba loại:

- Các lớp hiển nhiên

- Các lớp mờ
- Các lớp giả tạo



Đầu tiên tất cả lớp thuộc loại lớp giả sẽ bị loại bỏ, vì nó không có mục đích hoặc không cần thiết để sử dụng. Các lớp thuộc hai loại còn lại sẽ trở thành các ứng viên. Quy trình xác định như sau:



Khởi tạo danh sách các lớp ứng viên

- Tìm các danh từ hoặc các cụm danh từ trong các mô tả use case, yêu cầu
- Tất cả các lớp phải có ý nghĩa trong lãnh vực ứng dụng, tránh đưa vào các lớp cài đặt được mô tả trong giai đoạn thiết kế.
- Đặt tên cho lớp

Trích lọc trong use case và mô tả use case của hệ thống ATM, chúng ta có những danh từ và cụm danh từ sau:

Tài khoản	Bao thư
Số dư tài khoản	Bốn ký số
Số tiền	Ngân quỹ
Tiến trình đăng nhập	Tiền
Thẻ ATM	PIN
Máy ATM	PIN không hợp lệ

Ngân hàng	Thông điệp
Khách hàng ngân hàng	Mật khẩu
Thẻ	Mã PIN
Tiền mặt	Mẫu tin
Khách hàng	Bước
Tài khoản khách hàng	Hệ thống
VND	Giao dịch
	Lịch sử giao dịch

Loại bỏ các lớp giả

Các lớp ứng viên phải thuộc loại lớp hiển nhiên và lớp mờ. Các lớp giả sau đây sẽ bị loại khỏi danh sách: Bao thư, Bốn ký số, Bước.

Tài khoản	~~Bao thư~~
Số dư tài khoản	~~Bốn ký số~~
Số tiền	Ngân quỹ
Tiến trình đăng nhập	Tiền
Thẻ ATM	PIN
Máy ATM	PIN không hợp lệ
Ngân hàng	Thông điệp
Khách hàng ngân hàng	Mật khẩu
Thẻ	Mã PIN
Tiền mặt	Mẫu tin
Khách hàng	~~Bước~~
Tài khoản khách hàng	Hệ thống
VND	Giao dịch
	Lịch sử giao dịch

Đồng nhất các lớp ứng viên trùng lắp

Cần rà soát lại danh sách để tìm kiếm các danh từ, cụm danh từ trùng lắp về ý nghĩa mặc dù cách dùng từ có khác nhau. Chúng ta chọn lựa danh từ, hoặc cụm danh từ chứa đầy ngữ nghĩa nhất và loại những danh từ, cụm danh từ khác.

Khách hàng, Khách hàng ngân hàng	= Khách hàng
Tài khoản, Tài khoản khách hàng	= Tài khoản
PIN, Mã PIN	= PIN
Tiền, Ngân quỹ	= Ngân quỹ
Thẻ ATM, Thẻ	= Thẻ ATM
Tài khoản	~~Bao thư~~
Số dư tài khoản	~~Bốn ký số~~

Số tiền	Ngân quỹ
Tiến trình đăng nhập	~~Tiền~~
Thẻ ATM	PIN
Máy ATM	PIN không hợp lệ
Ngân hàng	Thông điệp
~~Khách hàng ngân hàng~~	Mật khẩu
~~Thẻ~~	~~Mã PIN~~
Tiền mặt	Mẫu tin
Khách hàng	~~Bước~~
~~Tài khoản khách hàng~~	Hệ thống
VND	Giao dịch
	Lịch sử giao dịch

Xác định các danh từ, cụm danh từ có thể là các thuộc tính

Các danh từ hoặc cụm danh từ là các thuộc tính khi:

- Chỉ được sử dụng như là giá trị
- Không có nhiều hơn một đặc trưng riêng, hoặc chỉ mô tả một đặc trưng của đối tượng khác.

Xem xét các danh từ, cụm danh từ có thể là thuộc tính của danh sách trên ta có:

Số tiền: một giá trị, không phải một lớp

Số dư tài khoản: thuộc tính của lớp Tài khoản

PIN không hợp lệ: một giá trị, không phải một lớp

Mật khẩu: một thuộc tính (có thể của lớp Khách hàng)

Lịch sử giao dịch: một thuộc tính (có thể của lớp Giao dịch)

PIN: một thuộc tính (có thể của lớp Khách hàng)

Sau đây là danh sách các ứng viên còn lại:

Tài khoản	~~Bao thư~~
~~Số dư tài khoản~~	~~Bốn ký số~~
~~Số tiền~~	Ngân quỹ
Tiến trình đăng nhập	~~Tiền~~
Thẻ ATM	~~PIN~~
Máy ATM	~~PIN không hợp lệ~~
Ngân hàng	Thông điệp
~~Khách hàng ngân hàng~~	~~Mật khẩu~~
~~Thẻ~~	~~Mã PIN~~
Tiền mặt	Mẫu tin
Khách hàng	~~Bước~~

~~Tài khoản khách hàng~~

VND

Hệ thống

Giao dịch

~~Lịch sử giao dịch~~

Loại bỏ các lớp ứng viên không có mục tiêu hoặc không thuộc phạm vi hệ thống

Mỗi lớp phải có một mục tiêu khi thuộc hệ thống, mục tiêu này phải thật rõ ràng trong ngữ cảnh mục tiêu chung hệ thống. Nếu chúng ta không thể diễn đạt mục tiêu của lớp trong hệ thống thì loại ra khỏi danh sách. Hoặc các lớp mặc dù có tham gia vào hoạt động của hệ thống, tuy nhiên nó không thuộc phạm vi quản lý của hệ thống sẽ bị loại ra. Các lớp ứng viên là:

Máy ATM: cung cấp một giao diện tới ngân hàng

Thẻ ATM: cung cấp một khách hàng với một khoá tới một tài khoản

Khách hàng: một khách hàng là một cá nhân sử dụng máy ATM, có một tài khoản.

Ngân hàng: các khách hàng phụ thuộc vào ngân hàng. Nó là một nơi tập trung các tài khoản và xử lý các giao dịch tài khoản.

Tài khoản: nó mô hình hoá một tài khoản của khách hàng và cung cấp các dịch vụ về tài khoản cho khách hàng

Giao dịch: mô tả một giao tác của khách hàng khi sử dụng thẻ ATM. Một giao tác được lưu trữ với thời gian, ngày, loại, số tiền, và số dư.

Các danh từ, cụm danh từ không có mục đích hoặc không thuộc phạm vi quản lý của hệ thống:

Thông điệp

Hệ thống

Mẫu tin

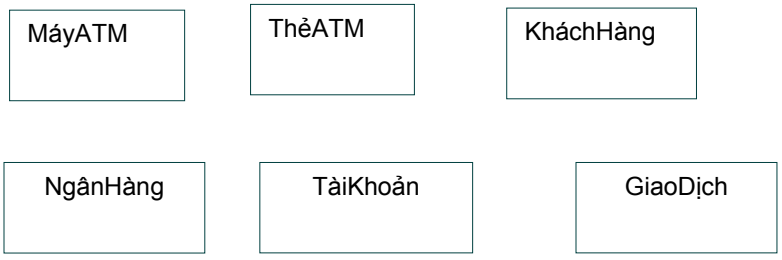
Ngân quỹ

VND

Tiền mặt

Tiến trình đăng nhập

Kết quả của quá trình chọn lựa gồm các lớp ứng viên sau hệ thống ATM:



Nhận xét: một hạn chế chính của cách tiếp cận cụm danh từ là nó phụ thuộc vào tính đúng và đầy đủ của các tài liệu mô tả. Điều này trên thực tế để có được những tài liệu này thì quả là khó. Hoặc chăng một văn bản lớn của hệ thống có thể dẫn đến quá nhiều lớp ứng viên! Dầu vậy, cách tiếp cận này rất có tính sư phạm và hữu dụng khi kết hợp với các cách tiếp cận khác.

Tiếp cận phân loại

Phương pháp thứ hai được gọi là phương pháp sử dụng mẫu lớp chung, phương pháp này dựa trên một cơ sở tri thức về việc phân loại lớp theo những mẫu chung. Các mẫu chung đó là:

Lớp khái niệm (concept)

Một khái niệm là một quan niệm hoặc sự hiểu biết riêng biệt về thế giới. Lớp khái niệm bao gồm các nguyên lý được dùng để tổ chức hoặc để lưu trữ các hoạt động và các trao đổi về mặt quản lý. Thông thường các khái niệm là các ý tưởng, sự hiểu biết được chia sẽ trong cộng đồng và dùng để trao đổi.

Ví dụ: phương pháp, phương pháp luận, mô hình,… là ví dụ của đối tượng lớp khái niệm.

Lớp sự kiện (event)

Lớp sự kiện là các điểm thời gian cần được lưu trữ. Các sự việc xảy ra tại một thời điểm, hoặc một bước trong một dãy tuần tự các bước. Liên quan tới các sự việc được lưu trữ là các thuộc tính (và các đối tượng chứa thuộc tính) như là: ai, cái gì, khi nào, ở đâu, như thế nào, hoặc tại sao.

Ví dụ: đăng ký, kết quả, hoá đơn, đơn hàng…

Lớp tổ chức (organization)

Một lớp tổ chức là một tập hợp con người, tài nguyên, phương tiện, hoặc những nhóm xác định chức năng người dùng,….

Ví dụ: đơn vị, bộ phận, phòng ban, chức danh,…

Lớp con người (people)

Lớp con người thể hiện các vai trò khác nhau của người dùng trong việc tương tác với ứng dụng. Những đối tượng này thường là người dùng hệ thống hoặc những người không sử dụng hệ thống nhưng thông tin về họ được lưu trữ bởi hệ thống (đa số là những đối tượng mà hệ thống có trao đổi thông tin nhưng không sử dụng hệ thống)

Ví dụ: sinh viên, khách hàng, giáo viên, nhân viên,…

Lớp vị trí (place)

Các vị trí vật lý mà hệ thống cần mô tả thông tin về nó.

Ví dụ: toà nhà, kho, văn phòng, chi nhánh, đại lý, …

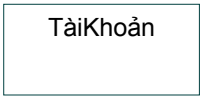
Sự vật hữu hình và lớp thiết bị

Các đối tượng vật lý hoặc các nhóm của đối tượng hữu hình mà có thể cảm nhận trực quan và các thiết bị mà hệ thống tương tác.

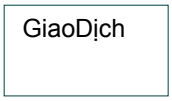
Ví dụ: xe hơi, máy bay, … là các sự vật hữu hình; thiết bị cảm ứng nhiệt là một lớp thiết bị.

Ví dụ: chúng ta cố gắng xác định lại các lớp trong hệ thống ATM dùng phương pháp tiếp cận:

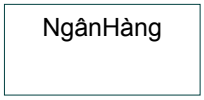
- Các lớp khái niệm:



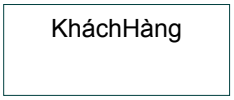
- Các lớp sự kiện:



- Các lớp tổ chức:



- Các lớp con người:



- Các lớp sự vật hữu hình và thiết bị:

Cách tiếp cận theo use case

Như chúng ta đã được giới thiệu, use case được dùng để mô hình hoá các kịch bản trong hệ thống và xác định cách thức các tác nhân tương tác với kịch bản. Kịch bản có thể được mô tả bằng văn bản hoặc thông qua một thứ tự các bước. Một khi hệ thống được mô tả trong ngữ nghĩa các kịch bản. Chúng ta có thể kiểm tra đoạn mô tả văn bản hoặc các bước của mỗi kịch bản để xác định các đối tượng nào cần thiết để cho kịch bản được thực hiện. Chúng ta có thể mô hình hoá các kịch bản của use case sử dụng sơ đồ tuần tự (sequence diagram) hoặc sơ đồ hợp tác (collaboration diagram). Các mô hình này cho phép chúng ta mô hình hoá một cách trực quan hơn ở giai đoạn phân tích và trợ giúp thiết kế hệ thống thông qua việc mô hình hoá sự tương tác giữa các đối tượng trong hệ thống.

Tuy nhiên, việc mô hình hoá kịch bản của use case một cách quá cụ thể sẽ dễ dẫn đến mô tả hoạt động phần mềm hệ thống nơi mà các đối tượng phần mềm có thể sẽ được xác định (mà đúng ra nó phải được xác định ở giai đoạn thiết kế). Do đó, cách tiếp cận này nên kết hợp với cách tiếp cận phân tích cụm danh từ hoặc cách tiếp cận phân loại để xác định đúng các đối tượng trong giai đoạn phân tích.

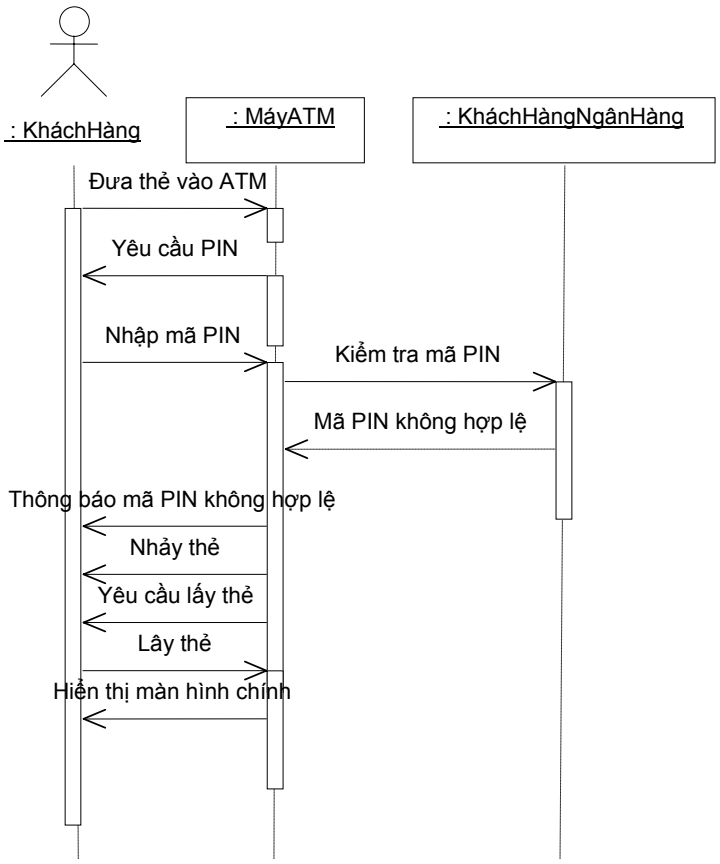
Trước tiên, chúng ta xác định các dòng tương tác của tác nhân với hệ thống trong một use case. Sau đó, chúng ta sẽ đặt câu hỏi "đối tượng nào của hệ thống sẽ chịu trách nhiệm tiếp nhận sự tương tác này?". Trả lời câu hỏi này giúp chúng ta tìm ra đối tượng đầu tiên của use case. Nếu đối tượng này chuyển giao toàn bộ hoặc một phần trách nhiệm xử lý cho đối tượng khác nào đó, thì chúng ta tiếp tục tiếp tục xác định đối tượng đó. Quá trình này cứ tiếp tục cho đến khi hết tất cả các dòng tương tác đã được kiểm tra.

Ví dụ: trong hệ thống ATM chúng ta xem hoạt động của use case "Giải quyết PIN không hợp lệ". Ở đây chúng ta cần nghĩ về tuần tự các hoạt động mà một khách hàng có thể thực hiện:

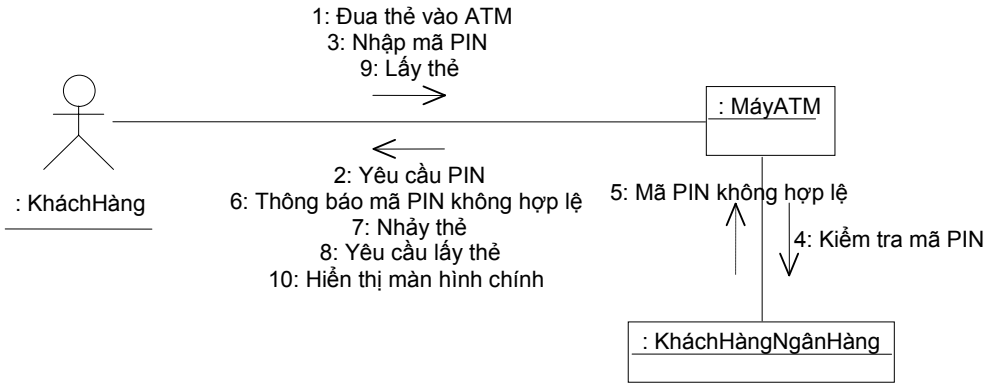
- Đưa vào thẻ ATM
- Nhập mã PIN
- Rút thẻ ATM

Dựa trên các hoạt động này, phản ứng của hệ thống hoặc chấp nhận quyền truy cập của tài khoản tương ứng hoặc từ chối. Kế tiếp chúng ta cần xác định một cách tường minh hơn về hệ thống: Chúng ta đang tương tác với cái gì (của hệ thống)? Máy ATM. Tiếp tục với kịch bản tiếp theo: máy ATM sẽ sử dụng đối tượng nào để kiểm tra mã PIN? Khách hàng ngân hàng. Một khách hàng trong trường hợp này là bất kỳ người nào muốn truy cập đến một tài khoản thông qua máy ATM, và có thể có hoặc có thể không có tài khoản. Ngược lại, một khách hàng ngân hàng có một tài khoản.

Sơ đồ tuần tự

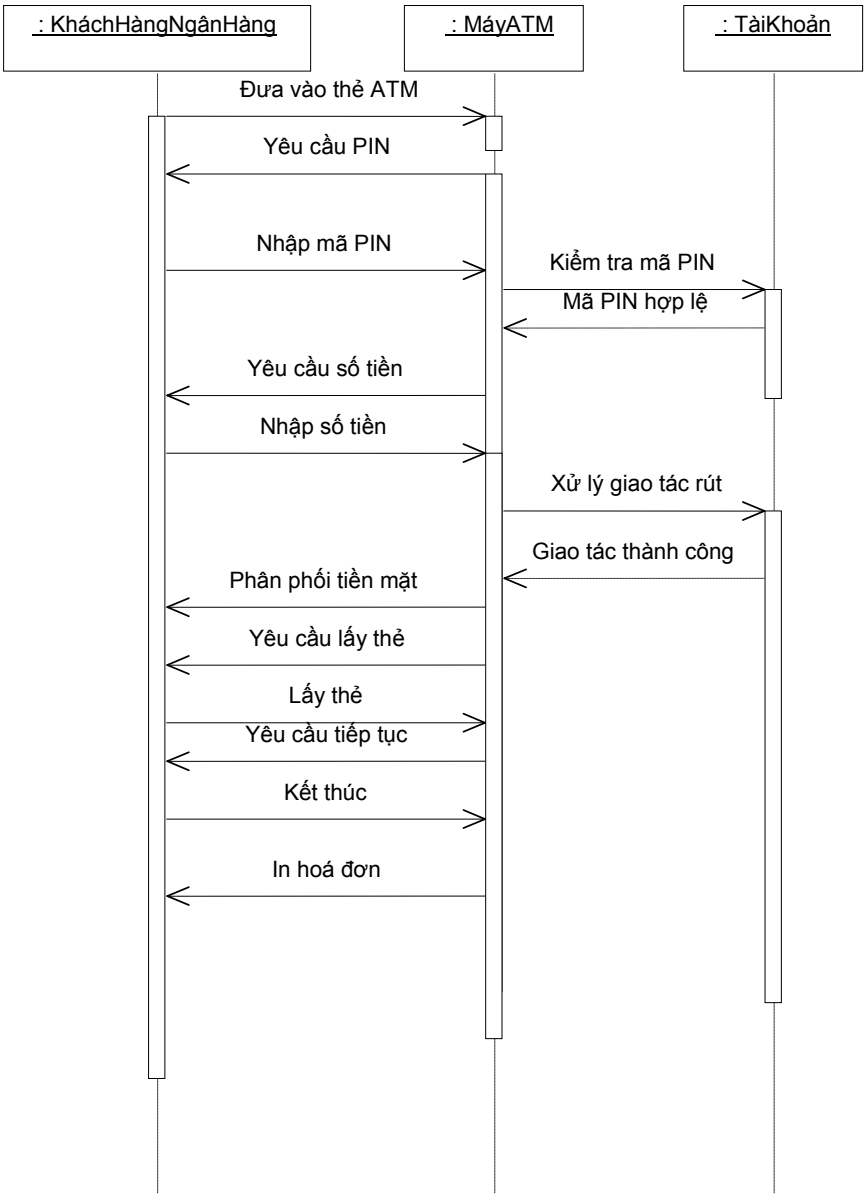


Sơ đồ hợp tác

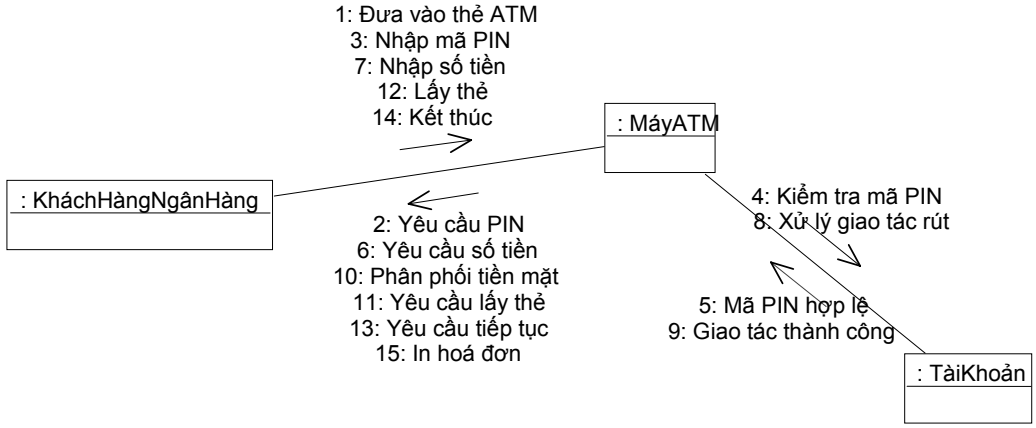


Hoặc xét use case “Rút tiền”

Sơ đồ tuần tự



Sơ đồ hợp tác



Như vậy, dựa vào hai sơ đồ tuần tự của hai use case chúng ta đã xác định được các lớp: MáyATM, KháchHàngNgânHàng (KháchHàng), TàiKhoản. Tiếp tục mô hình hoá với sơ đồ tuần tự hoặc hợp tác với các use case còn lại của hệ thống ATM, chúng ta sẽ xác định được các lớp còn lại.

Xác định mối quan hệ giữa các lớp

Trong môi trường hướng đối tượng, đối tượng đảm nhiệm một vai trò chủ động trong một hệ thống. Đối tượng không tồn tại một cách độc lập mà luôn tương tác với những đối tượng khác. Sự tương tác này thể hiện thông qua mối kết hợp bao gồm luôn các hoạt động và hành vi của các đối tượng. Sau đây chúng ta sẽ được giới thiệu các hướng dẫn giúp xác định ba loại liên kết: mối kết hợp, mối kết hợp thu nạp (thành phần) và mối kết hợp tổng quát hoá.

Xác định mối kết hợp (association)

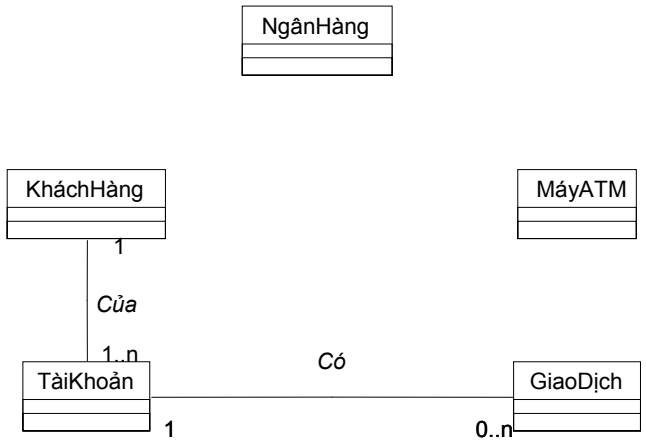
Xác định mối kết hợp bắt đầu bằng việc phân tích sự tương tác giữa các lớp. Thông thường thì bất kỳ có một liên kết phụ thuộc nào giữa hai hay nhiều lớp đều có thể thiết lập mối kết hợp. Một cách làm điều này chính là kiểm tra trách nhiệm của đối tượng để thiết lập mối kết hợp. Nói cách khác, nếu một đối tượng được xác nhận để thi hành một nhiệm vụ và lại thiếu thông tin để thi hành nhiệm vụ đó, thì đối tượng này phải ủy quyền thực hiện lại cho đối tượng khác sở hữu thông tin đó. Có nhiều cách tiếp cận để xác định mối kết hợp, đầu tiên trích lọc các mối kết hợp ứng viên từ việc tham khảo mô tả hệ thống và yêu cầu hệ thống và những tài liệu khác liên quan đến hệ thống. Sau đó, tinh chế dần để chọn ra mối kết hợp có ý nghĩa nhất. Chú ý rằng mối kết hợp và mối kết hợp thành phần có ngữ nghĩa rất gần nhau, do đó có những trường hợp khó phân biệt, mối kết hợp thành phần chỉ là một trường đặc biệt của mối kết hợp. Tùy theo lãnh vực ứng dụng nếu chúng ta tìm được một cách tự nhiên để biểu diễn mối kết hợp thành phần thì hãy chọn nó, ngược lại hãy chọn mối kết hợp để biểu diễn. Sau đây là các hướng dẫn và các mẫu chung cho phép xác định mối kết hợp:

Hướng dẫn xác định mối kết hợp

- Một sự phụ thuộc giữa hai hay nhiều lớp có thể thiết lập thành mối kết hợp. Mối kết hợp thường tương ứng với một động từ hoặc cụm giới từ như là *thành phần của*, *làm việc cho*, *chứa trong*, …
- Một tham chiếu từ một lớp đến một lớp khác là một mối kết hợp.

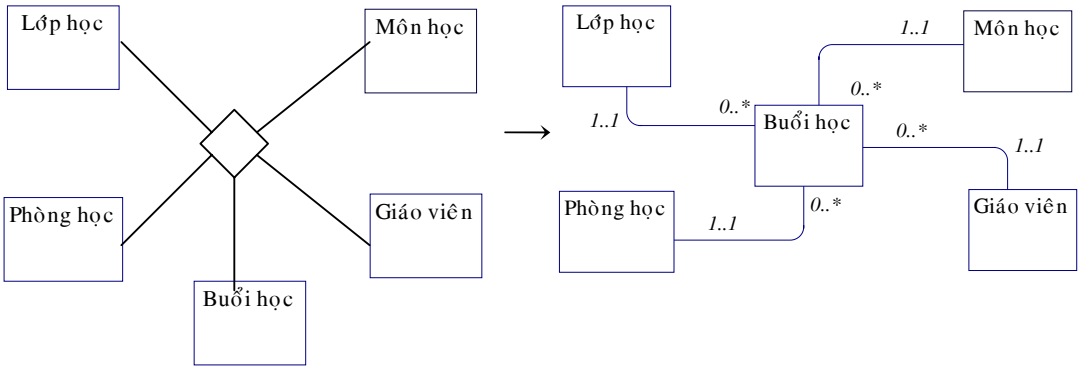
Mẫu chung xác định mối kết hợp

- Mối kết hợp vị trí (location): *liên kết tới, thành phần của, chứa trong, làm việc tại, ….*
- Mối kết hợp sở hữu: *của, có, thuộc,…* Mối kết hợp *có* giữa lớp TàiKhoản và lớp GiaoDịch, mối kết hớp *của* giữa lớp TàiKhoản và lớp KháchHàng
- Mối kết hợp truyền thông, liên lạc (communication): *đặt tới, trao đổi với, gởi cho, tiếp nhận từ,…*

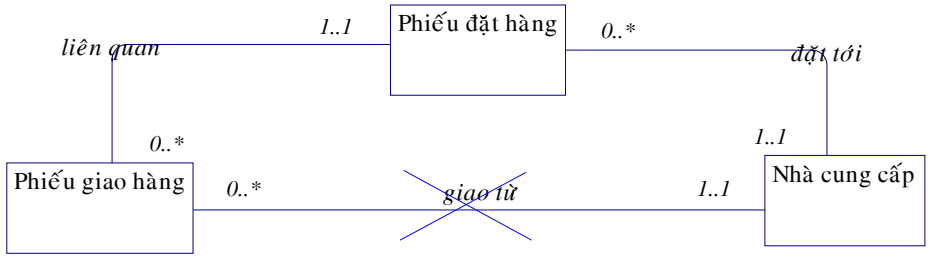


Loại bỏ những mối kết hợp không cần thiết

- **Mối kết hợp cài đặt**: mối kết hợp cài đặt nên được đưa vào trong quá trình thiết kế và cài đặt hệ thống. Mối kết hợp cài đặt là mối kết hợp mô tả sự liên quan giữa các lớp trong giai đoạn thiết kế cài đặt hệ thống bên trong môi trường phát triển hoặc ngôn ngữ lập trình cụ thể và không phải là môi liên kết giữa các đối tượng mô tả nghiệp vụ.
- **Mối kết hợp đa phân**: là mối kết hợp giữa ba lớp trở lên, mối kết hợp này phức tạp trong cách thể hiện. Nếu có thể, phát biểu lại nó dùng mối kết hợp nhị phân.



- **Mối kết hợp trực tiếp dư thừa (directed action)**: là các mối kết hợp được định nghĩa trong ngữ nghĩa của những mối kết hợp khác (còn gọi là mối kết hợp suy diễn hoặc bắc cầu). Những mối kết hợp loại này là dư thừa do đó cần loại bỏ nó khỏi mô hình. Trong các mối kết hợp dưới đây *liên quan*, *đặt tới*, *giao từ* thì mối kết hợp *giao từ* có thể được suy ra từ mối kết hợp *liên quan* và *đặt tới*. Vậy mối kết hợp *giao từ* là dư thừa và có thể loại bỏ nó khỏi mô hình.

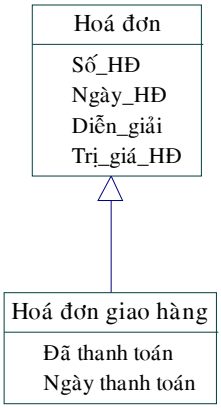


Xác định mối kết hợp tổng quát – chuyên biệt (generalization)

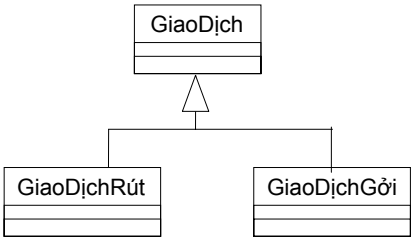
Mối kết hợp tổng quát hoá – chuyên biệt thể hiện quan hệ kế thừa giữa các lớp và một cấu trúc phân cấp xác định những dòng kế thừa này. Quan hệ này cho phép các đối tượng có thể xây dựng từ những đối tượng khác. Một thuận lợi chính khi sử dụng mối kết hợp này là chúng ta có thể xây dựng trên những gì đang có và quan trọng hơn là sử dụng lại cái đã có. Sau đây là các tiếp cận hướng dẫn xác định mối kết hợp tổng quát hoá:

Tiếp cận trên xuống (top-down): Từ một lớp chúng ta tìm kiếm cụm danh từ chứa tên lớp và tính từ (hoặc danh từ). Đánh giá xem cụm danh từ này có thể là một trường hợp đặc biệt cần được quản lý trong hệ thống không; tìm kiếm xem có những đặc trưng riêng của lớp này mà hệ thống cần chỉ ra hay không. Nếu quyết định là một lớp thì nó là một loại chuyên biệt của lớp ban đầu.

Ví dụ: từ lớp Hoá đơn chúng ta tìm thấy một cụm danh từ Hoá đơn giao hàng có chứa từ Hoá đơn, và trường hợp dưới đây quyết định Hoá đơn giao hàng là một lớp chuyên biệt của lớp Hóa đơn.



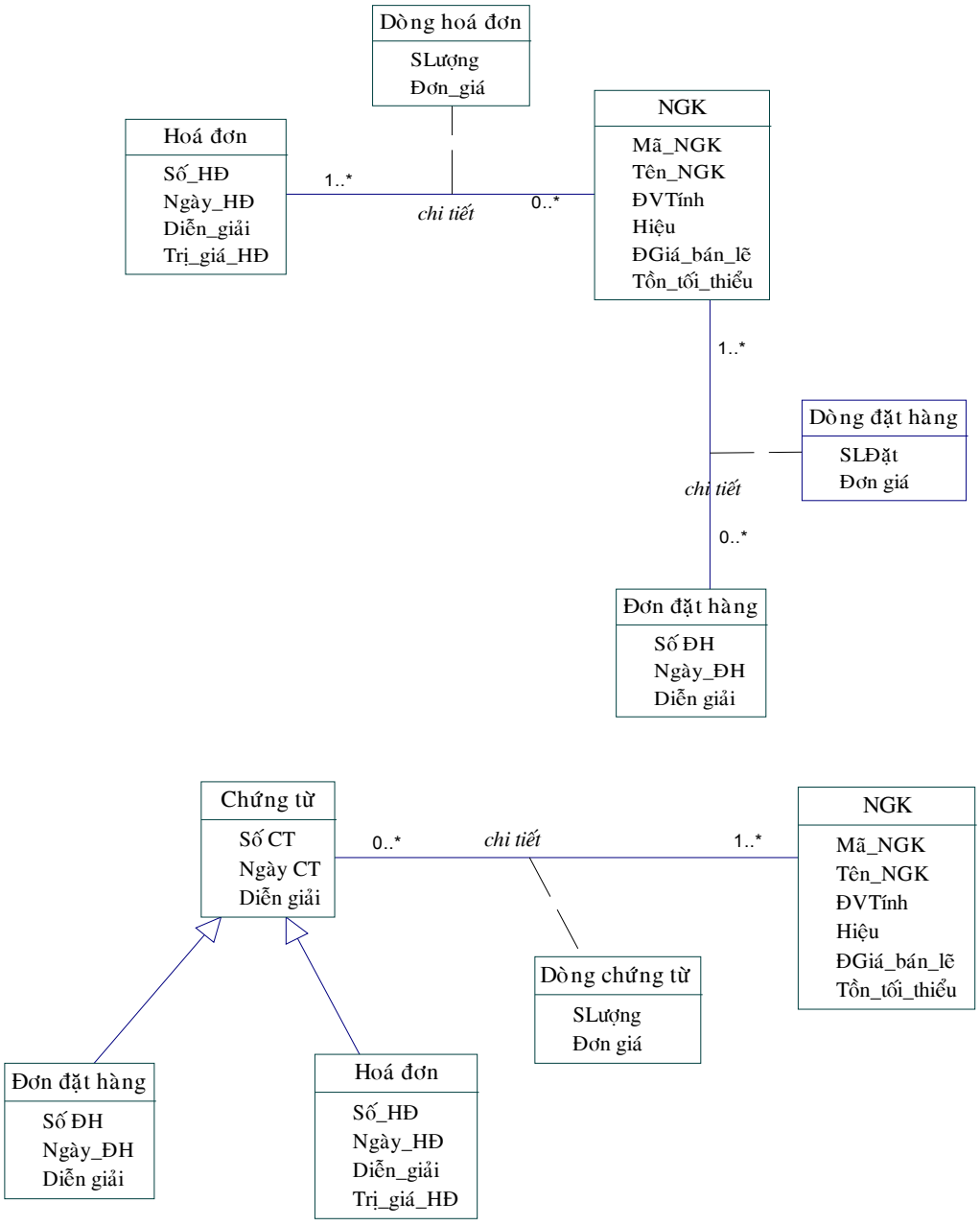
Hoặc trong hệ thống ATM, từ lớp GiaoDịch chúng ta có thể chuyên biệt hoá xuống hai lớp con GiaoDịchRút và GiaoDịchGởi.



Trong quá trình tạo cấu trúc phân cấp tổng quát – chuyên biệt, chúng ta không nên quá lạm dụng việc chuyên biệt hóa mà đưa vào tất cả các lớp chuyên biệt không cần thiết. Các lớp chuyên biệt cần thiết là các lớp có những đặc trưng (thuộc tính, hành vi và mối kết hợp) riêng được biểu diễn trong hệ thống. Ví dụ, có rất nhiều loại hoá đơn có thể tạo lớp chuyên biệt như là: hoá đơn bán lẽ, hoá đơn bán sĩ, hoá đơn giao hàng. Trong khi đó, chỉ có Hoá đơn giao hàng là có những thuộc tính riêng nên nó được biểu diễn trong hệ thống, còn hai lớp còn lại không cần thiết vì không tìm ra đặc trưng riêng của nó.

Tiếp cận dưới lên (bottom-up): tìm kiếm trong các lớp để xác định xem có các thuộc tính và phương thức giống nhau. Sau đó chúng ta có thể gom nhóm và đưa các thuộc tính và phương thức chung này lên một lớp tổng quát (trừu tượng). Tạo mối kết hợp tổng quát hoá từ các lớp này đến lớp tổng quát mới xác định.

Mô hình dưới đây cho thấy, các lớp Hoá đơn và Đơn đặt hàng có ba thuộc tính giống nhau (Hoá đơn: Số HĐ, Ngày HĐ, Diễn giải. Đơn đặt hàng: Số ĐH, Ngày ĐH, Diễn giải) và mối kết hợp với lớp NGK giống nhau. Do đó, chúng ta tạo ra một lớp tổng quát Chứng từ và di chuyển cả ba thuộc tính và mối kết hợp lên lớp này và tạo ra mối kết hợp tổng quát hoá từ lớp Hoá đơn và Đơn đặt hàng đến lớp Chứng từ.



Tiếp cận tái sử dụng: di chuyển các thuộc tính và phương thức lên càng cao càng tốt trong cấu trúc phân cấp.

Đa kế thừa: tránh sử dụng quá mức đa kế thừa trong mô hình vì đa kế thừa sẽ phức tạp trong việc xác định đường dẫn kế thừa một hành vi từ những lớp nào.

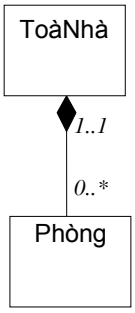
Xác định mối quan hệ thành phần (a-part-of, aggregation)

Mối quan hệ thành phần được sử dụng trong tình huống một lớp hình thành bao gồm những lớp thành phần. Hai đặc trưng chính của mối quan hệ thành phần là tính bắc cầu và tính phản đối xứng.

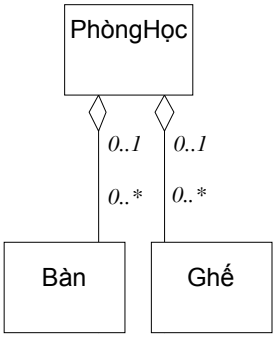
- *Tính bắc cầu*: nếu lớp A là một thành phần của lớp B và lớp B là thành phần của lớp C, thì lớp A là thành phần của lớp C.
- *Tính phản đối xứng*: nếu lớp A là thành phần của lớp B thì lớp B không phải là thành phần của lớp A.

Việc xác định mối quan hệ thành phần có thể dựa trên các mẫu chỉ dẫn sau:

- **Tập hợp**: một đối tượng vật lý được hình thành từ các đối tượng vật lý thành phần khác.

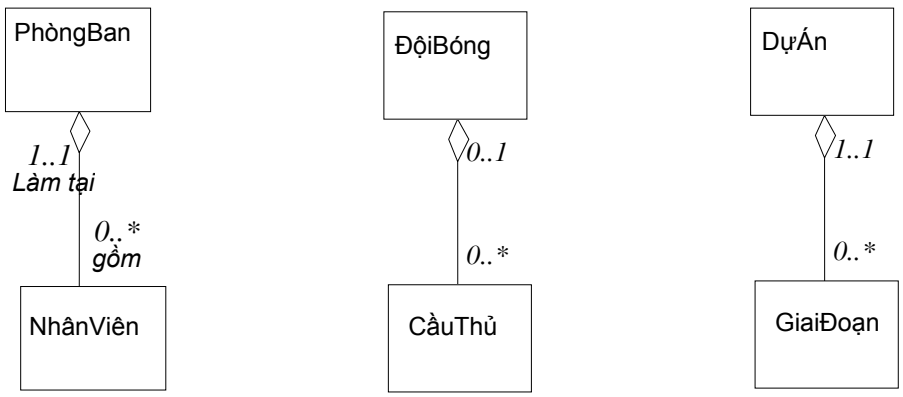


- **Vật chứa**: một đối tượng vật lý chứa đựng các thành phần nhưng không được cấu tạo bởi các thành phần.



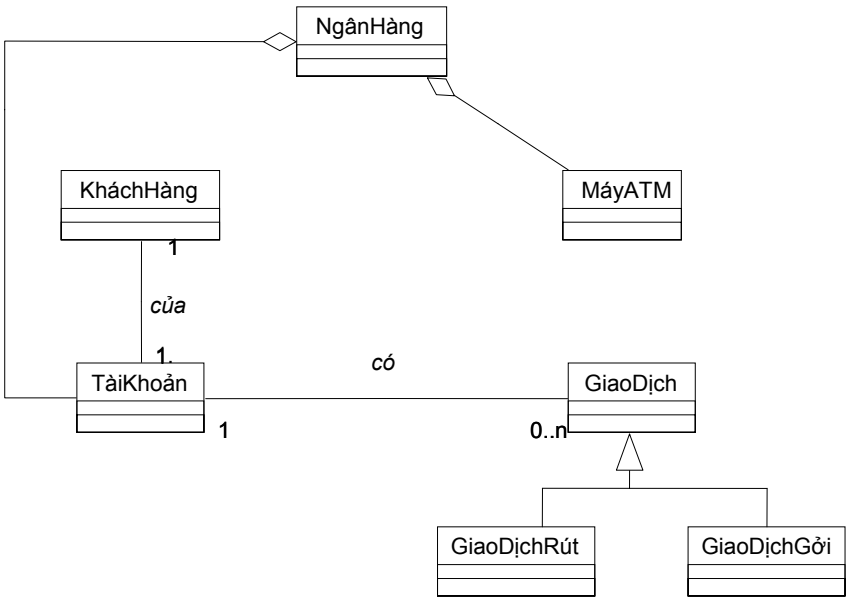
Một phòng học chứa đựng tất cả các đối tượng bàn ghế nhưng không được cấu tạo từ những đối tượng này.

- **Tập hợp – thành viên:** một đối tượng quan niệm chứa các thành phần có thể vật lý hoặc quan niệm.



Các lớp PhòngBan, ĐộiBóng, DứAn, GiaiĐoạn là các lớp quan niệm; các lớp NhânViên, CầuThủ là các lớp vật lý.

Trong hệ thống ATM, một ngân hàng bao gồm các máy ATM, các tài khoản, các toà nhà, các nhân viên,.v.v… Tuy nhiên, các đối tượng toà nhà, nhân viên,… không thuộc phạm vi hệ thống đang xét. Do đó, chúng ta định nghĩa mối kết hợp thành phần giữa lớp NgânHàng và các lớp: MáyATM, TàiKhoản.



Xác định thuộc tính (attribute) và phương thức (method) của lớp

Xác định thuộc tính và phương thức cũng là một công việc khó như là xác định lớp, và đây cũng là một tiến trình lặp. Thông thường người ta cũng dựa vào use case và các sơ đồ UML khác để xác định các thuộc tính và phương thức của lớp.

Thuộc tính là các thành phần mà một đối tượng phải ghi nhớ như là màu sắc, trị giá, hảng sản xuất,… Xác định thuộc tính của một lớp hệ thống bắt đầu với việc tìm hiểu về các trách nhiệm của hệ thống. Và chúng ta đã xác định răng, trách nhiệm của hệ thống có thể được nhận định qua việc phát triển các use case và các đặc điểm mong muốn của ứng dụng như là xác định thông tin gì mà người dùng cần cho hệ thống. Các câu hỏi sau đây có thể giúp xác định nhiệm vụ của lớp và các thành phần dữ liệu mà hệ thống muốn lưu trữ.

- Thông tin gì về đối tượng sẽ được lưu trữ?
- Dịchvụ gì mà một lớp phải cung cấp?

Trả lời câu hỏi thứ nhất giúp chúng ta xác định các thuộc tính của một lớp. Trả lời câu hỏi thứ hai giúp chúng ta xác định các phương thức của lớp.

Xác định thuộc tính

Bằng việc phân tích các use case, các yêu cầu, các mô tả và các sơ đồ chúng ta có thể bắt đầu hiểu trách nhiệm của lớp và cách thức mà các lớp tương tác để thi hành công việc. Mục tiêu chính ở đây là để hiểu những gì mà một lớp có trách nhiệm về tri thức.

Sau đây à một số hướng dẫn giúp xác định lớp trong các use case:

- Thuộc tính thường tương ứng tới các danh từ đi theo bởi các cụm phó từ như là: *chi phí của* sản phẩm. Các thuộc tính cũng có thể tương ứng tới các tính từ hoặc các phó từ.
- Giữ cho lớp đơn giản: chỉ dùng đủ thuộc tính để diễn đạt trạng thái đối tượng
- Các thuộc tính ít có thể được mô tả đầy đủ trong mô tả vấn đề. Do đó, chúng ta phải sử dụng tri thức về lãnh vực ứng dụng và thực tế để tìm chúng.
- Không nên quan tâm quá về việc phải khám phá hết thuộc tính. Chúng ta có thể bổ sung thêm các thuộc tính trong các vòng lặp tiếp theo.

Ví dụ: xác định các thuộc tính cho các lớp của hệ thống ATM.

Xác định thuộc tính cho lớp KháchHàng

Bằng việc phân tích use case, các sơ đồ tuần tự và hợp tác và sơ đồ hoạt động, rõ ràng rằng, với lớp KháchHàng, lãnh vực bài toán và hệ thống đưa ra một vài thuộc tính. Tìm kiếm trong sơ đồ tuần từ của use case "Xứ lý PIN không hợp lệ" chúng ta tìm thấy rằng lớp KháchHàng phải có một mã PIN (hay password) và số thẻ. Do đó, *mãPIN* và *sốThẻ* là hai thuộc tính thích hợp của lớp KháchHàng. Các thuộc tính khác của KháchHàng là các biểu diễn tri thức chung về khách hàng, do đó các thuộc tính của lớp KháchHàng là:

tênKháchHàng

họKháchHàng

mãPIN

sốThẻ

trong thời điểm này chúng ta chỉ quan tâm đến chức năng của đối tượng khách hàng mà không quan tâm đến các thuộc tính cài đặt

Xác định thuộc tính cho lớp TàiKhoản

Tương tự các thuộc tính của lớp TàiKhoản được xác định là:

sốTàiKhoản

loạiTàiKhoản

sốDư

Xác định thuộc tính cho lớp GiaoDịch

giaoDịchID

ngàyGiaoDịch

thờiGianGiaoDịch

loạiGiaoDịch

sốTiền

sốDư

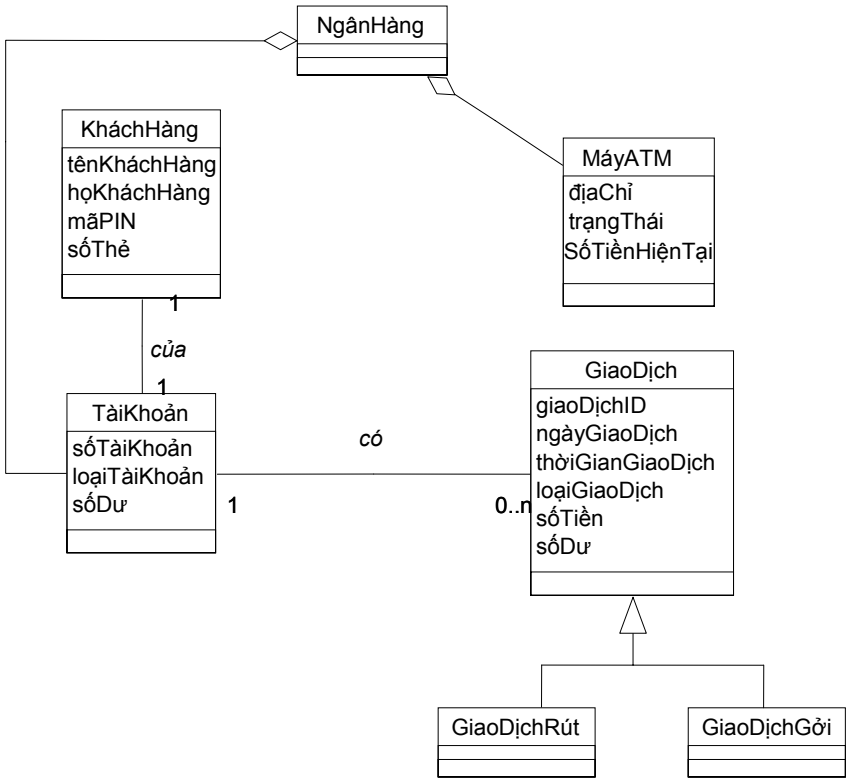
Xác định thuộc tính cho lớp MáyATM

Máy ATM là một đối tượng vật lý và hữu hình, do đó thuộc tính của nó dùng để mô tả vị trí và trạng thái của máy.

địaChỉ

trạngThái

sồTiềnHiệnTại



Xác định phương thức

Phương thức (method) và thông điệp (message) là các thành phần gánh vác công việc xử lý hệ thống hướng đối tượng. Trong một môi trường hướng đối tượng, mỗi phần dữ liệu hoặc đối tượng được bao quanh bởi một tập hợp các thường trình gọi là các phương thức. Những phương thức này chính là các dịch vụ và các toán tử của một lớp định nghĩa để cài đặt hànhvi của các đối tượng thành viên của lớp. Các phương thức chính là cách thức mà đối tượng thực hiện tương tác với các đối tượng khác trong hệ thống. Phát hiện các phương thức để cài đặt các hành vi đối tượng là một hoạt động quan trọng trong giai đoạn phân tích.

Cũng như thuộc tính, một lớp sẽ có những phương thức nội bộ (private) và toàn cục (puplic). Trong giai đoạn phân tích, chúng ta chỉ quan tâm phát hiện các phương thức toàn cục mà ít khi quan tâm đến các phương thức nội bộ của đối tượng. Các phương thức thường tương ứng với các truy vấn về các thuộc tính của đối tượng. Nói cách khác, các phương thức chịu trách nhiệm quản lý các trị của thuộc tính như là truy vấn, cập nhật, đọc và ghi. Chú ý rằng trong giai đoạn phân tích, chú ta đang làm việc ở mức cao của sử trừu tượng hoá. Do đó, các phương thức như là tạo (constructor) hoặc các phương thức mô tả chi tiết việc cài đặt sẽ được phát hiện trong giai đoạn thiết kế.

Trong phần này chúng chúng ta xem xét cách thức xác định phương thức sử dụng các sơ đồ UML như là: sơ đồ trạng thái, sơ đồ hoạt động, sơ đồ tuần tự/hợp tác và sơ đồ use case.

Xác định phương thức bằng việc phân tích các sơ đồ UML và use case

Trong sơ đồ tuần tự, các đối tượng được đặt trong sơ đồ với các đường đứt quảng thẳng đứng. Do đó, các sự kiện xảy ra giữa các đối tượng được đặt theo dòng nằm ngang. Một sự kiện được xem như là một hành động để chuyển thông tin. Mặt khác, những hành động này là các toán tử mà đối tượng phải thực thi.

Ví dụ, để xác định các phương thức của lớp TàiKhoản, chúng ta xem xét các sơ đồ tuần tự ứng với các use case:

Rút tiền

Gởi tiền

Xem thông tin tài khoản

Sơ đồ tuần tự có thể trợ giúp chúng ta xác định các dịch vụ mà các đối tượng phải cung cấp. Ví dụ, qua việc nghiên cứu sơ đồ tuần tự của use case *Rút tiền*, chúng ta thấy rằng lớp TàiKhoản phải cung cấp dịch vụ *rútTiền*. Qua việc nghiên cứu use case *Gởi tiền*, lớp TàiKhoản phải cung cấp dịch vụ *gởiTiền*.

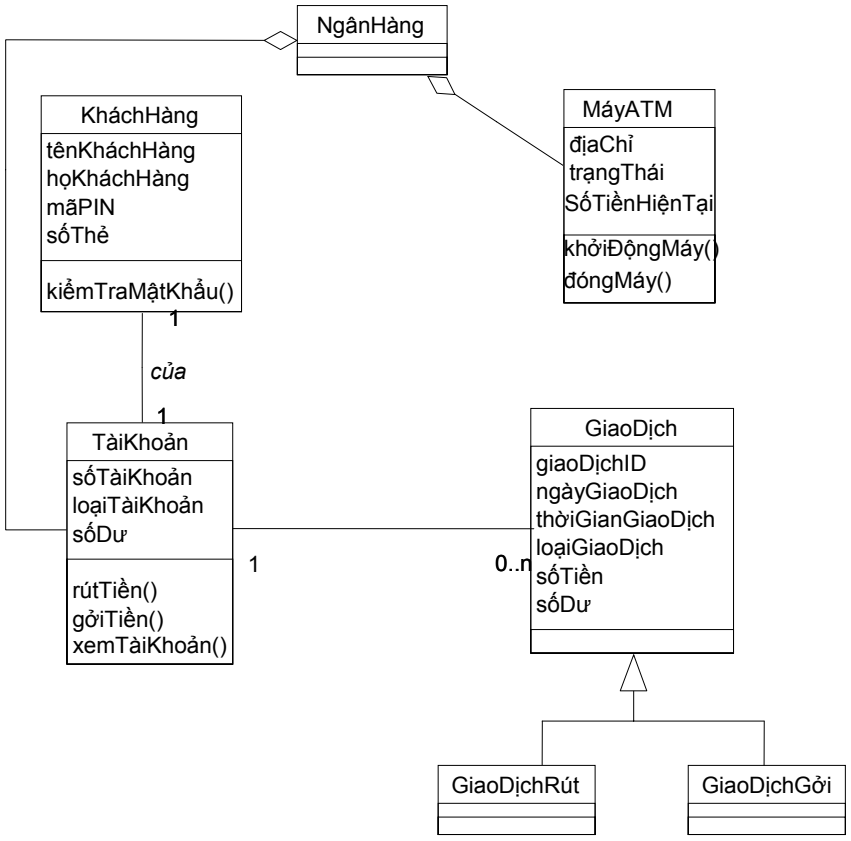
Tương tự, các dịch vụ lớp KháchHàng:

kiểmTraMậtKhẩu (kiểm tra mật khẩu của khách hàng)

Các dịch vụ của lớp MáyATM

Khởi động máy

Đóng máy



Chú ý, các dịch vụ được đưa vào lớp tổng quát sẽ được thừa kế trong các lớp chuyên biệt.

Câu hỏi và bài tập

Câu hỏi

1. Các đối tượng của hệ thống trong giai đoạn phân tích có thể xác định từ đâu?
2. Mô tả chiến lược "cụm danh từ" trong việc xác định các lớp ứng viên trong một lãnh vực vấn đề?

3. Chiến lược phân loại là gì?
4. Các lớp khái niệm là gì?
5. Các lớp sự kiện là gì?
6. Các lớp tổ chức là gì?
7. Các lớp con người là gì?
8. Các lớp vị trí là gì?
9. Các lớp sự vật hữu hình và thiết bị là gì?
10. Tại sao xây dựng các sơ đồ tuần tự/hợp tác là một hoạt động hữu dụng cho việc xác định các lớp?
11. Tại sao việc xác định lớp là một quá trình gia tăng?
12. Tại sao việc xác định sự phân cấp của các lớp là quan trọng trong phân tích hướng đối tượng?
13. Liên kết kết hợp (association), tổng quát hoá (generalization) là gì?
14. Thuộc tính và phương thức của lớp được xác định bằng cách nào?

Bài tập

1. Hãy xây dựng sơ đồ lớp của hệ thống "Diễn đàn trao đổi học tập của khoa Công Nghệ Thông Tin".

PHẦN 3

THIẾT KẾ HỆ THỐNG

Mục tiêu

Cung cấp các nội dung về:

- Các nguyên lý căn bản về thiết kế hướng đối tượng
- Quá trình thiết kế lớp đối tượng: thiết kế thuộc tính, method và mối kết hợp
- Kiến trúc ba tầng trong thiết kế phần mềm
- Thiết kế use case
    - Quá trình thiết kế các lớp tầng truy cập dữ liệu: xác định các lớp, thuộc tính, method và mối kết hợp qua việc phân tích use case
    - Thiết kề các lớp tầng giao diện: xác định các lớp, xây dựng bản mẫu (prototype), xác định thuộc tính và method qua việc phân tích use case
    - Mô hình hoá use case hiện thực hoá dùng sơ đồ lớp, sơ đồ tương tác
- Nâng cấp kiến trúc bằng việc phân chia hệ thống thành các gói (package)
- Xây dựng mô hình thiết kế vật lý hệ thống sơ đồ thành phần và sơ đồ triển khai nhằm chuẩn bị cho cài đặt phần mềm hệ thống

Giới thiệu

Mục tiêu chính của giai đoạn phân tích việc phát triển phần mềm là tập trung vào xác định những gì cần được thực hiện. Các đối tượng được phát hiện trong giai đoạn phân tích có thể phục vụ như là bộ khung (framework) cho giai đoạn thiết kế. Các thuộc tính, phương thức và mối liên kết của lớp được xác định trong giai đoạn phân tích phải được thiết kế cho việc cài đặt như là một thành phần được mô tả theo ngôn ngữ cài đặt. Trong phần này, chúng ta tập trung chi tiết hoá khung nhìn luận lý (logical view) của hệ thống phần mềm bằng cách xác định thêm các lớp phần mềm (tầng giao diện và tầng truy cập CSDL) và thiết kế chúng và kết quả là một sơ đồ lớp hoàn chỉnh mô tả đầy đủ các đối tượng phần mềm hệ thống chuẩn bị cho cài đặt. Mặt khác, dựa trên kết quả này chúng ta phát triển thiết kế vật lý hệ thống bằng cách xây dựng thêm vào các khung nhìn cài đặt (implementation view) và khung nhìn triển khai (deployment view) nhằm chuyển giao kết quả thiết kế hệ thống gần với một ngôn ngữ và công cụ lập trình xác định cho giai đoạn lập trình và sau đó có thể cài đặt phù hợp với các thiết bị tài nguyên trong một môi trường hệ thống thực tế một cách hiệu quả. Phần này bao gồm bốn chương: các chương về thiết kế luận lý: chương 8 Thiết kế lớp, chương 9 Thiết kế use case, chương 10 Thiết kế gói và hệ thống con; chương về thiết kế vật lý: chương 11 Thiết kế cài đặt.

Chương 8 THIẾT KẾ LỚP

Mục tiêu

Cung cấp các kiến thức về:

- Một số các tiên đề và hệ quả trong thiết kế hệ thống hướng đối tượng nhằm giúp người thiết kế đạt được một kết quả thiết kế tốt.
- Bước đầu tiên tinh chế các lớp trong giai đoạn phân tích thành các lớp có thể cài đặt trong hệ thống phần mềm bao gồm: thiết kế thuộc tính, mối kết hợp, và method

Các tiên đề và hệ luận trong thiết kế hướng đối tượng

Các tiên đề

Theo Suh, trong tiếp cận thiết kế hướng đối tượng có hai tiên đề được áp dụng, các tiên đề là:

Tiên đề 1: tiên đề độc lập.

Duy trì sự độc của các thành phần: Trong quá trình thiết kế, chúng ta đi từ yêu cầu và use case đến một thành phần hệ thống, mỗi thành phần phải thoã mãn yêu cầu mà không ảnh hưởng đến những thành phần khác.

Tiên đề 2: tiên đề thông tin.

Giảm tối đa nội dung thông tin thiết kế. Tiên đề này nói về tính đơn giản hoá trong thiết kế. Mục đích chính là giảm tối đa tính phức tạp, như vậy các đối tượng sẽ dễ bảo trì và nâng cấp hơn. Trong thiết kế hướng đối tượng, cách tốt nhất để giảm độ phức tạp là sử dụng tính thừa kế. (xem hệ luận 6)

Các hệ luận

Từ hai tiên đề trên, có nhiều hệ luận được trích dẫn. Những hệ luận này sẽ mang lại lợi ích hơn trong việc quyết định thiết kế các tình huống cụ thể, vì chúng có thể áp dụng tới các tình huống thực tế dễ dàng hơn so với tiên đề gốc ban đầu.

Hệ luận 1: thiết kế độc lập, giảm tối đa thông tin trao đổi (Uncouple Design with Less Information)

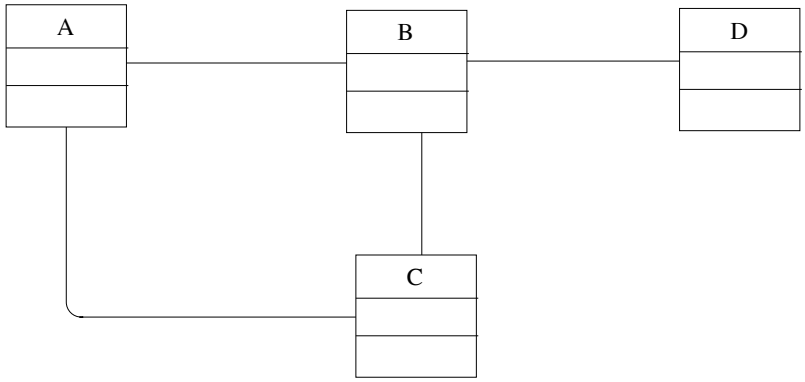
Sự cố kết giữa các đối tượng cao có thể làm giảm tính liên kết giữa chúng. Bởi vì chỉ một lượng nhỏ các thông tin cần thiết trao đổi giữa các đối tượng đó.

Coupling: Ở giai đoạn thiết kế, coupling được dùng để đo mức độ liên kết giữa các đối tượng hoặc giữa thành phần phần mềm. Coupling là một mối kết hợp nhị phân, và là một khái niệm quan trọng khi đánh giá một thiết kế bởi vì nó giúp chúng ta tập trung đúng vào vấn đề quan trọng của thiết kế. Ví dụ, một thành phần trong hệ thống thay đổi sẽ có một tác động tối thiểu đến những thành phần khác. Coupling trong các đối tượng càng mạnh càng mạnh thì hệ thống càng phức tạp. Mức độ của coupling có thể được đánh giá trên:

- Mức độ phức tạp của kết nối
- Kết nối tham chiếu đến chính bản thân đối tượng hoặc bên ngoài đối tượng
- Các thông điệp nhận và gửi đi

Mức độ coupling giữa hai thành phần được xác định bằng mức độ phức tạp của thông tin trao đổi giữa chúng. Coupling càng gia tăng thì càng làm gia tăng độ phức tạp hoặc mơ hồ, tối nghĩa của giao diện. Coupling càng giảm khi sự kết nối được thiết lập ở thành phần giao diện thay vì ở thành phần bên trong. Trong thiết kế hướng đối tượng, có hai loại coupling là coupling tương tác và coupling thừa kế:

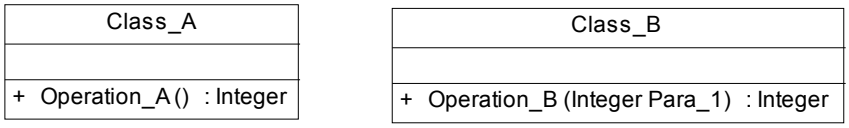
Coupling tương tác: thể hiện qua số lượng và độ phức tạp của các thông điệp giữa những thành phần, sự tương tác càng ít thì càng được ưu tiên. Coupling cũng áp dụng tới mức độ phức tạp của thông điệp. Một chỉ dẫn chung là giữ các thông điệp càng đơn giản và càng ít xảy ra thì càng tốt. Tổng quát, nếu một kết nối thông điệp gồm ba tham số trở lên thì kiểm tra xem có thể làm đơn giản hoá nó nữa được không. Một đối tượng được kết nối với nhiều thông điệp phức tạp thì gọi là liên kết chặt (*tightly coupled)*, có nghĩa là bất kỳ có sự thay đổi nào có thể tác động đến sự thay đổi trong những cái khác.



Hình 4. Ví dụ về biểu diễn coupling và B là liên kết chặt

Năm mức độ coupling tương tác

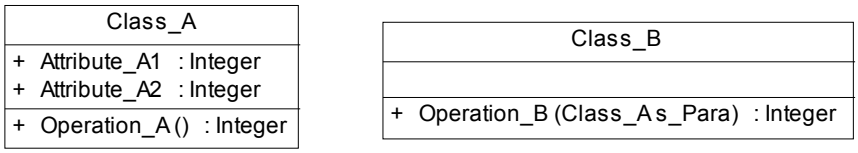
Data coupling: kết nối giữa các thành phần hoặc là dữ liệu dạng nguyên tố hoặc là hoặc cấu trúc tổng hợp tất cả yếu tố được dùng bởi đối tượng nhận. Đây là loại coupling tốt nhất và là mục đích của thiết kế kiến trúc. Các đối tượng trao đổi với nhau thông qua các dữ liệu thành tố hoặc các cờ hiệu thông tin. Thành phần này không quan tâm đến bất cứ gì bên trong thành phần khác mà chỉ quan tâm đến dữ liệu gì nó cần và dữ liệu gì nó gởi trả.




```
integer Operation_A()
{
int x,y;
Class_B cB;
….
y = cB.Operation_B(x);
…
}
```

Operation_A không quan tâm đến nội dung thực hiện của Operation_B mà chỉ quan tâm đến dữ liệu gởi đến và nhận từ Operation_B

Stamp coupling: trong stamp coupling, dữ liệu trao đổi là một phần của cấu trúc hoặc toàn bộ cấu trúc. Loại coupling này được đánh giá là thấp hơn so với data coupling bởi vì sự trao đổi là một cấu trúc thay vì một yếu tố đơn nên làm cho nó phức tạp hơn. Một thay đổi trong cấu trúc sẽ ảnh hưởng đến tất cả đối tượng sử dụng nó. Stamp coupling làm cho các thành phần phụ thuộc nhiều hơn đến thành phần khác, bởi vì, để tránh những lỗi có thể xảy ra, những thành phần sẽ phải có hiểu biết về hoạt động bên trong của thành phần khác sử dụng cùng cấu trúc dữ liệu.




```
integer Operation_A()
{
int x,y;
Class_B cB;
….
y = cB.Operation_B(this);
…
}
```

Operation_B nhận dữ liệu là cấu trúc Class_A để thực hiện. Do đó, nội dung của nó sẽ sử dụng dữ liệu từng thành phần của cấu trúc Class_A. Một sử thay đổi trong cấu trúc này sẽ ảnh hưởng đến Operation_B

Control coupling: khi một thành phần thành phần gởi một thông tin điều khiển tới một thành phần khác, hai thành phần này được gọi là có control coupling. Thông tin điều khiển có thể xuất hiện dưới dạng cờ hiệu thông báo cho thành phần khác hành động sẽ thực hiện. Khi thông tin điều khiển được gởi đi, thành phần gởi phải biết về hoạt động bên trong của thành phần nhận, do đó nó tạo ra một sự phụ thuộc giữa các thành phần.

Common coupling: khi hai thành phần tham khảo đến cùng một vùng dữ liệu chung toàn cục thì có liên kết common coupling. Liên kết này cũng nên tránh bởi vì nó tạo ra một cơ hội lớn về lỗi trên toàn bộ hệ thống. Một lỗi phát sinh trong bất kỳ một thành phần nào sử dụng dữ liệu toàn cục này có thể gây ra lỗi trong bất kỳ thành phần khác sử dụng cùng dữ liệu này.

Content coupling: loại coupling được xếp thấp nhất là content coupling. Bởi vì loại này giới thiệu một thành phần tham khảo trực tiếp hoạt động bên trong của một thành phần khác. Ví dụ, một method có thể thay đổi dữ liệu trong một method khác hoặc thay đổi một đoạn code trong method khác. Hiện nay, các ngôn ngữ lập trình (đặc biệt là ngôn ngữ lập trình hướng đối tượng) đều không cho phép tạo ra content coupling.

Tên coupling	Xếp hạng phụ thuộc
Data coupling	Rất thấp
Stamp coupling	Thấp
Control coupling	Trung bình
Common coupling	Cao
Content coupling	Rất cao

Coupling thừa kế: là coupling giữa lớp tổng quát và lớp chuyên biệt. Một lớp chuyên biệt liên kết với lớp tổng quát của nó thông qua thuộc tính và method. Không như coupling tương tác, coupling thừa kế càng cao thì càng ưu tiên. Tuy nhiên, để đạt được mức độ cao coupling thừa kế trong một hệ thống. Mỗi lớp chuyên biệt không nên thừa kế những method và thuộc tính không liên quan hoặc không cần thiết. Ví dụ, nếu một lớp chuyên biệt chồng lên hầu hết các method hoặc không sử dụng nó từ lớp tổng quát, điều này cho thấy coupling thừa kế là thấp và lúc đó thiết kế viên phải thay đổi lại cách xây dựng tổng quát hoá và chuyên biệt hoá này.

Cohesion

Trong khi coupling đề cập đến mức độ tương tác giữa các đối tượng thành phần thì cohesion lại đề cập đến sự tương tác bên trong một đối tượng thành phần. Cohesion phản ánh tính "đơn mục đích" của một đối tượng. Tất cả các lệnh, chức năng trong một thành phần được định nghĩa gắn liền tới một nhiệm vụ. Trong hệ thống nếu các thành phần đạt được mức độ cohesion càng cao thì mức độ liên kết coupling giữa các thành phần càng yếu, do đó để đạt được đỉnh cao của cohesion thì chúng ta phải làm giảm mức độ coupling. Cohesion cũng trợ giúp trong thiết kế lớp bằng việc giúp xác định mục đích và mục tiêu của lớp một cách rõ ràng.

Method cohesion: một method chỉ nên đảm nhận một chức năng hoặc một nhiệm vụ. Thông thường cách đặt tên của method cũng phải ngụ ý nói lên chức năng liên quan của nó. Ví dụ: Chon_nha_cung_cap(), Tinh_ton(),…

Class cohesion: các method và các thuộc tính trong một lớp phải có mức độ cohesion cao, nghĩa là phải được dùng bởi các method bên trong lớp hoặc chính là method phục vụ cho mục đích của lớp.

Cohesion thừa kế: các lớp chuyên biệt và lớp tổng quát phải cùng mô tả về một loại đối tượng của hệ thống. Các thuộc tính và hành vi của lớp tổng quát phải được thừa hưởng tối đa trong lớp chuyên biệt theo cách hoặc là phục vụ cho hành vi của lớp chuyên biệt hoặc trở thành một hành vi của lớp chuyên biệt.

Hệ luận 2: đơn mục đích (single purpose)

Mỗi lớp phải có một mục đích xác định trong việc đạt được mục tiêu hệ thống. Nếu chúng ta mô tả một lớp phức tạp thì hãy phân chia nó thành những lớp nhỏ hơn, đơn giản và xác định hơn. Mỗi method chỉ cung cấp một dịch vụ, kích thước không quá lớn (không nên vượt quá một trang coding)

Hệ luận 3: nhiều lớp đơn. Tạo các lớp đơn để cho phép tái sử dụng

Kết quả thiết kế hệ thống tốt chính là tạo ra một tập lớn các lớp đơn giản hơn là một tập nhỏ các lớp phức tạp. Các lớp càng ít chuyên biệt thì các vấn đề trong tương lai càng khó giải quyết hơn thông qua việc kết nối các lớp đang có và tái sử dụng chúng.

Thiết kế hướng đối tượng luôn đề xuất việc sản sinh thư viện các thành phần tái sử dụng và nhấn mạnh trên tính bao bọc (encapsulation), đơn vị hoá (modularization), và tính đa hình (polymorphism) để tái sử dụng khi xây dụng một đối tượng hơn là làm mới.

Hệ luận 4: ánh xạ kết quả giữa các giai đoạn phải chặt chẽ (strong mapping)

Giai đoạn phân tích và thiết kế dựa trên cùng mô hình, kết quả mô hình. Từ quá trình phân tích đến cài đặt, các chi tiết sẽ được đưa thêm vào nhưng vẫn duy trì về cơ bản giống nhau. Ví dụ, trong giai đoạn phân tích chúng ta xác định lớp Hoá đơn. Trong giai đoạn thiết kế chúng ta thiết kế lớp Hoá đơn: method, dữ liệu,…

Hệ luận 5: chuẩn hoá (standardization).

Đề xuất sự chuẩn hoá bằng việc thiết kế các thành phần có thể thay thế và tái sử dụng các thành phần có sẵn.

Để tái sử dụng, chúng ta phải có một hiểu biết sâu về các lớp trong môi trường lập trình hướng đối tượng chúng ta đang dùng. Hầu hết các hệ thống hướng đối tượng đều có các các lớp thư viện xây dựng sẵn và ngày càng gia tăng khi chúng ta tạo các ứng dụng mới. Tri thức về các lớp tồn tại giúp chúng ta xác định những gì một lớp mới cần có do thừa kế hơn là phải xây dựng mới. Mặc dù vậy, tài liệu mô tả các lớp thư viện thường không được biên tập tốt hoặc ít được cập nhật thường xuyên khi có sự điều chỉnh, nâng cấp. Do đó, trong quá trình xây dựng các hệ thống chúng ta nên tạo ra các lớp thư viện và tích lũy dần nó từ việc xây dựng các hệ thống và luôn luôn xem xét việc thừa kế nó khi xây dựng một hệ thống mới.

Hệ luận 6: thiết kế thừa kế (design inheritance).

Các đặc tính chung phải được chuyển lên lớp tổng quát. Cấu trúc lớp tổng quát – lớp chuyên biệt phải tạo ra ý nghĩa luận lý.

Thiết kế lớp

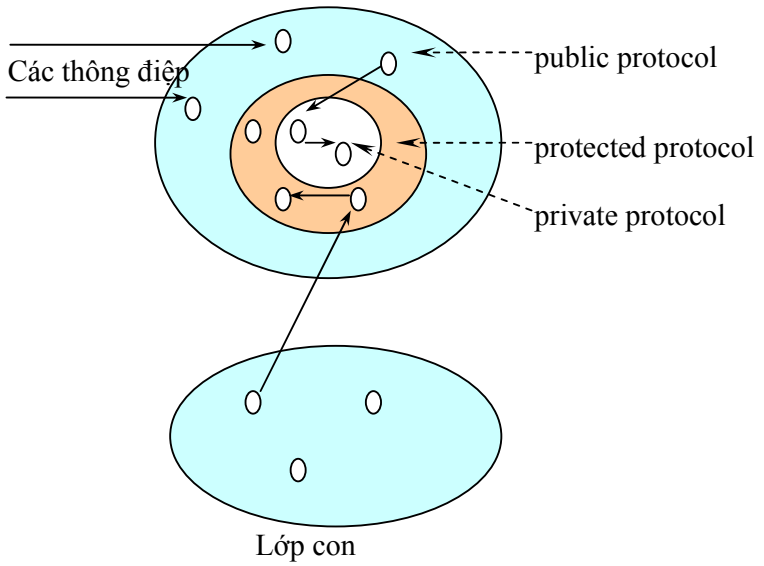
Một tiến trình thiết kế lớp bao gồm:

- Tinh chế thuộc tính
- Thiết kế hành vi (method) và nghi thức (protocol) sử dụng sơ đồ trong UML
- Tinh chế quan hệ giữa các lớp
- Tinh chế sự phân cấp và thiết kế sự kế thừa

Phạm vi ảnh hưởng của lớp

Trong việc thiết kế hành vi và thuộc tính cho các lớp, chúng ta gặp phải hai vấn đề. Thứ nhất là ***nghi thức (protocol)***, hoặc giao diện tới các toán tử và sự có thể thấy (visibility) của nó; thứ hai là cách thức cài đặt nó. Nghi thức được xem là cách thức xác định các đặc trưng của lớp có thể "nhìn thấy" từ bên ngoài hoặc cỉ được xem là "nội bộ" bên trong. Các nghi thức đó là: toàn cục (public protocol), riêng (private protocol), và bảo vệ (protected protocol).

- pupblic protocol: định nghĩ các hành vi trạng thái của lớp
- private protocol: dùng để xác định các đặc trưng của lớp chỉ được truy cập bởi chính lớp đó
- Protected protocol: xác định đặc trưng của một lớp có thể sử dụng bởi chính nó và lớp con của nó.



Tinh chế thuộc tính

Các thuộc tính trong giai đoạn phân tích hướng đối tượng phải được tinh chế theo cách nhìn về việc cài đặt nó trong môi trường tin học. Trong giai doạn phân tích, chúng ta chỉ cần tên của thuộc tính là đủ. Tuy nhiên, trong giai đoạn thiết kế, thông tin chi tiết về thuộc tính đó phải được thêm vào. Mục tiêu chính của hoạt động này là tinh chế các thuộc tính tồn tại (được xác định trong giai đoạn phân tích) và đưa thêm các thuộc tính dùng cho việc cài đặt nhằm đặc tả lớp như một thành phần của môi trường tin học.

Kiểu thuộc tính

Có ba kiểu cơ bản của thuộc tính:

- Thuộc tính đơn trị
- Thuộc tính đa trị
- Thuộc tính dùng để tham chiếu tới các đối tượng khác hoặc tới một thể hiện kết nối

Các thuộc tính thể hiện cho trạng thái của đối tượng. Khi một trạng thái của đối tượng thay đổi, sự thay đổi này được phản ánh qua giá trị của các thuộc tính. Thuộc tính đơn trị là loại phổ biến nhất, nó chỉ có một giá trị hoặc một trạng thái tương ứng với một đối tượng. Ví dụ: *Tên, Ngày sinh, Mức lương*,…

Thuộc tính đa trị có thể có một tập các giá trị tương ứng cho một đối tượng. Ví dụ: chúng ta muốn lưu trữ nhiều số điện thoại của một khách hàng, chúng ta có thể đặt thuộc tính *Số điện thoại* là đa trị.

Thuộc tính tham chiếu được dùng để cung cấp việc tham khảo cần thiết để một đối tượng đáp ứng các trách nhiệm của nó. Nói cách khác, thuộc tính tham chiếu hiện thực hoá mối kết hợp của các đối tượng.

Hiển thị thuộc tính

Trong UML việc trình bày thuộc tính được đề nghị như sau:

<Phạm vi> <tên> : <kiểu thuộc tính> = <giá trị khởi tạo>

Trong đó, *<phạm vi>* sẽ là:

+ : toàn cục (public protocol)

: bảo vệ (protected protocol)

- : cục bộ (private protocol)

<kiểu thuộc tính> là một đặc tả cài đặt thuộc tính độc lập ngôn ngữ.

<giá trị khởi tạo> là một biểu thức độc lập ngôn ngữ xác định giá trị khởi tạo khi một đối tượng được tạo mới. tham số này là tuỳ chọn.

Thuộc tính đa trị được xác định bằng việc thêm vào chỉ số mảng theo sau tên thuộc tính. Ví dụ:

Địa_chỉ[3]: string

Tập_hợp_điểm[2..*]: điểm

Ví dụ: tinh chế thuộc tính các lớp của hệ thống ATM

Lớp KháchHàng

#tênKháchHàng: String

#họKháchHàng: String

#mãPIN: String

#sốThẻ: String

#tàiKhoản: TàiKhoản (thuộc tính tham chiếu)

Trong đó, thuộc tính #tàiKhoản dùng để mô tả mối quan hệ giữa lớp KháchHàng và lớp TàiKhoản. Việc thêm thuộc tính này cho phép chúng ta tham khảo đến một đối tượng tài khoản từ một đối tượng khách hàng. Tất cả thuộc tính đều được gán cho một phạm vi truy cập nội bộ dạng nghi thức protected nhằm đảm bảo tính bao bọc và có thể thừa kế nếu sau này các phát triển thêm các lớp con.

Lớp TàiKhoản

#sốTàiKhoản: String

#loạiTàiKhoản: String

#sốDư: float

#giaoTác: GiaoTác (cài đặt mối kết hợp giữa lớp TàiKhoản và lớp GiaoTác)

#kháchHàng: KháchHàng (cài đặt mối kết hợp giữa lớp TàiKhoản và lớp KhácHàng)

Lớp GiaoTác

#giaoDịchID: String

#ngàyGiaoDịch: Date

#thờiGianGiaoDịch: Time

#loạiGiaoDịch: String

#sốTiền: float

#sốDư: float

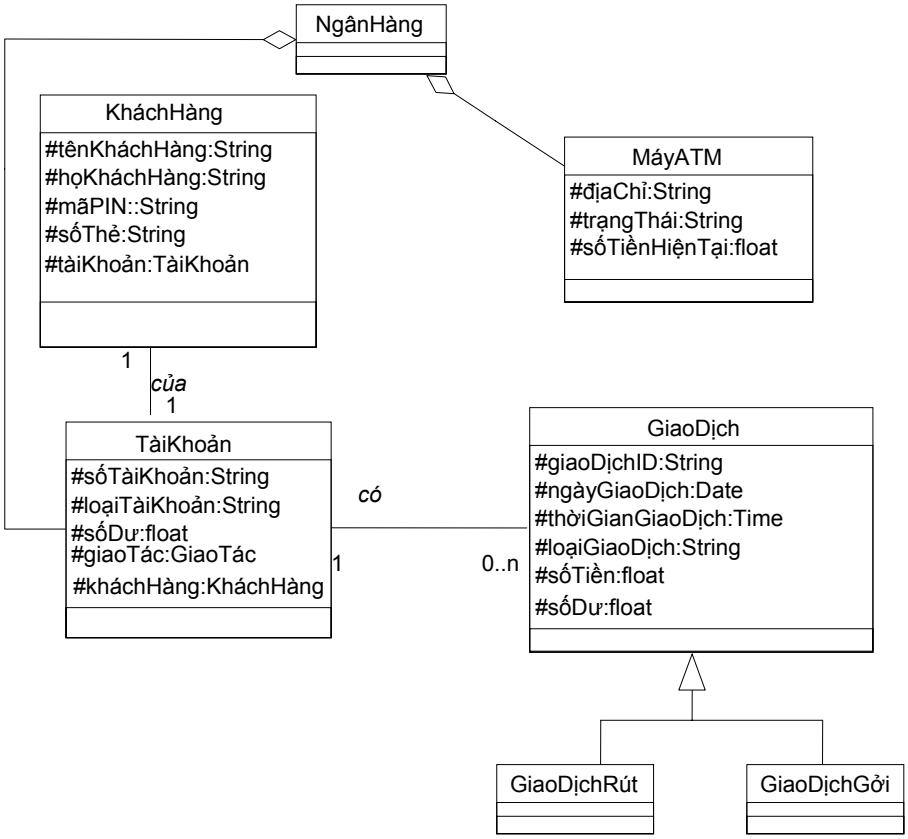
Lớp GiaoTác và lớp TàiKhoản có một mối kết hợp. Khi tính chế thuộc tính cho lớp TàiKhoản, chúng ta đã thêm vào thuộc tính #giaoTác dùng để cài đặt mối kết hợp này. Vậy khi tính chế thuộc tính cho lớp GiaoTác chúng ta cũng có thể thêm vào một thuộc tính cũng để cài đặt cho mối kết hợp này nếu chúng ta có nhu cầu biết thông tin về tài khoản từ một giao tác. Tuy nhiên, với bài toán của chúng ta đây không có nhu cầu đó, vậy nên chúng ta không thêm vào lớp GiaoTác thuộc tính tham chiếu tới lớp TàiKhoản.

Lớp MáyATM

#địaChỉ: String

#trạngThái: String

#sốTiềnHiệnTại:float



Sơ đồ lớp của hệ thống ATM sau khi đã tính chế thuộc tính

Tinh chế hành vi (method)

Mục tiêu chính của hoạt động này là mô tả thuật toán cho các hành vi đã được xác định ở giai đoạn phân tích. Việc mô tả thuật toán cũng có thể được thực hiện trên nhiều cách, hoặc bằng văn bản (mã giả) hoặc bằng sơ đồ. Việc sử dụng sơ đồ (trong UML có thể dùng sơ đồ hoạt động, tuần tự,…) cho phép chúng ta dễ dàng hơn trong việc chuyển đổi chúng sang một ngôn ngữ lập trình một cách thủ công hoặc tự động (thông qua một CASE Tool).

Trong giai đoạn này chúng cũng nên giảm tối đa mức độ phức tạp các kết nối thông điệp giữa các lớp số lượng thông điệp được gởi và nhận bởi một đối tượng. Mục tiêu nhằm gia tăng tính đoàn kết (cohension) trong số các đối tượng và các thành phần phần mềm để cải tiến tính liên kết (coupling), bởi vì chúng ta phải giữ một số lượng tối thiểu các thông tin cần thiết chuyển đổi giữa các thành phần. Áp dụng các tiên đề và hệ luận thiết kế chúng ta có các hướng dẫn sau:

- Một tập lớn các lớp đơn giản sẽ tốt hơn một tập nhỏ các lớp phức tạp
- Tạo một lớp tổng quát cho các lớp mà chúng ta tìm thấy có một số nội dung giống nhau. Mục tiêu là tăng tối ta việc tái sử dụng
- Luôn tập trung vào mục tiêu của lớp khi định nghĩa nhằm tránh việc thiết kế lạc đề hoặc mở rộng vượt khỏi phạm vi ý nghĩa của lớp: điều này thường xảy ra khi chúng ta tạo ra các thay đổi trên lớp đang tồn tại khi bài toán có sự thay đổi và càng ngày tạo ra các lớp với nội dung phức tạp không còn đúng với mục tiêu ban đầu của lớp.

Một lớp có thể có những loại hành vi sau:

- *Constructor*: methode tạo thể hiện (đối tượng) của lớp
- *Destructor*: methode huỷ thể hiện của lớp
- *Conversion*: method chuyển đổi một đơn vị đo lường này sang một đơn vị đo lường khác
- *Copy*: method sao chép nội dung của một thể hiện sang một thể hiện khác
- *Attribute set*: method gán giá trị cho một hoặc nhiều thuộc tính
- *Attribute get*: method trả về giá trị của một hoặc nhiều thuộc tính
- *I/O method*: method cung cấp tới hoặc nhằn dữ liệu từ một thiết bị
- *Domain specific*: method xác định tới các ứng dụng của đối tượng

Hiển thị hành vi

Theo UML một hành vi của lớp được trình bày theo cú pháp:

<phạm vi> <tên> <(danh sách tham số)> : <kiểu trả về>

Trong đó,

<phạm vi> được qui định giống như của thuộc tính

<danh sách tham số>: mỗi tham số cách nhau bởi dấu phẩy và có cú pháp như sau:

<tên tham số>: <kiểu dữ liệu> = <giá trị mặc định>.

<kiểu trả về> là một đặc tả độc lập ngôn ngữ về cài đặt giá trị trả về của một hành vi. Nếu mục này bị bỏ qua thi hành vi không trả về giá trị

Ví du:

+get_Tên(): String

+get_SốTàiKhoản(vtàiKhoản : TàiKhoản): String

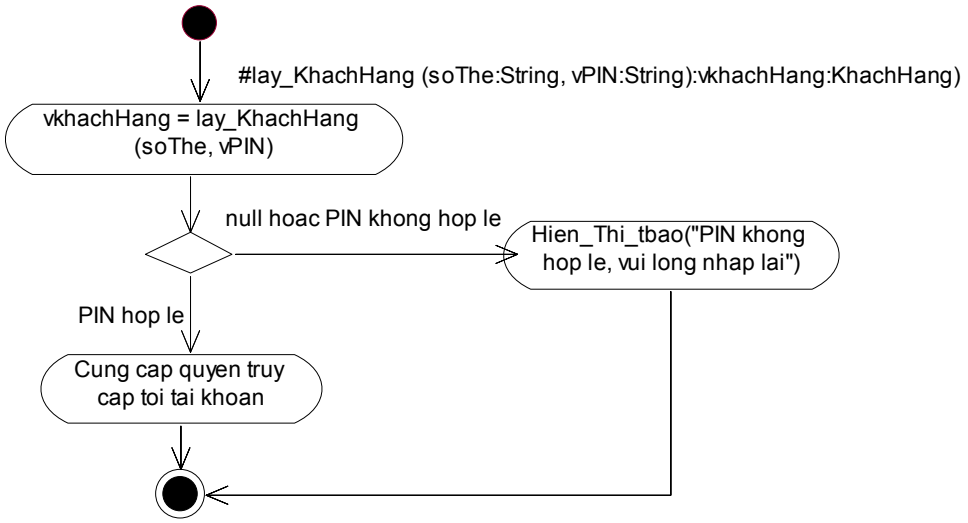
Thiết kế hành vi cho hệ thống ATM

Tại thời điểm này, chúng ta đã xác định các đối tượng hình thành tầng nghiệp vụ của hệ thống, cũng như là các dịch vụ mà các đối tượng này cung cấp. Công việc còn lại là thiết kế hành vi, giao diện người dùng và xử lý truy cập cơ sở dữ liệu. Với hệ thống ATM, chúng ta đã xác định các toán tử ở giai đoạn phân tích bao gồm:

Lớp KháchHàng

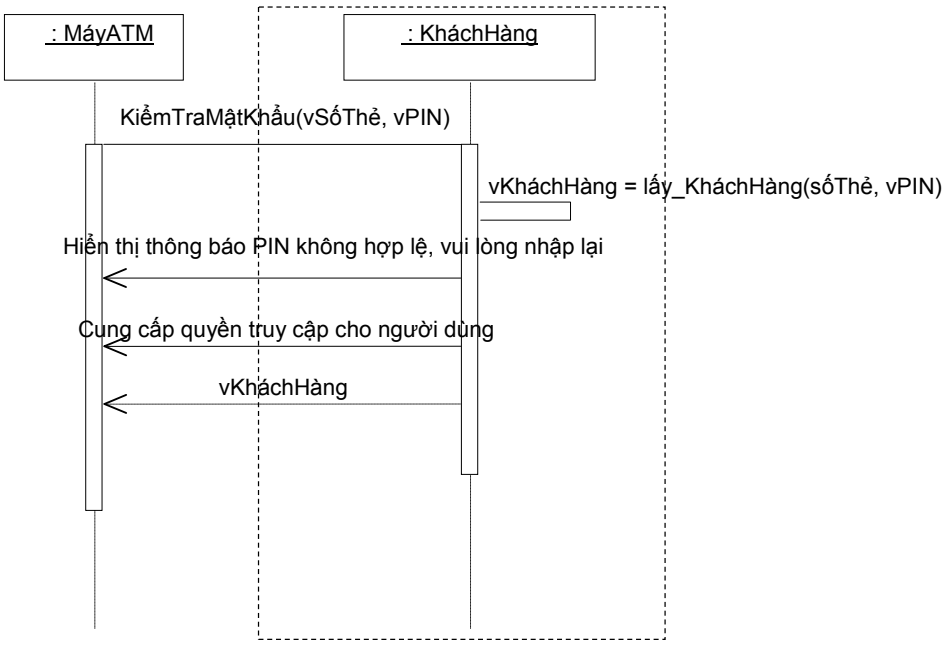
KháchHàng::+kiểmTraMậtKhẩu(sốThẻ:String, vPIN:String): vkháchHàng: KháchHàng

Thiết kế thuật giải cho hành vi dùng sơ đồ hoạt động



Hành vi *kiểmTraMậtKhẩu()* trước hết sẽ thi hành tạo một đối tượng khách hàng và thực hiện lấy thông tin về khách hàng dựa trên dựa trên một số thẻ và mã PIN. Tại đây, chúng ta lại nhận thấy rằng cần phải có một hành vi khác để thực hiện điều này đó là *lấy_KháchHàng()* và có phạm vi nội bộ dạng protected (#). Hành vi này sẽ lấy tham số đầu vào là số thẻ và mã PIN, kết quả trả về là một đối tượng khách hàng tìm thấy hoặc là "null" nếu ngược lại. Nếu giá trị trả về là "null", sẽ gởi một thông điệp tới hệ thống để thực hiện thông báo "PIN không hợp lệ, vui lòng nhập lại", ngược lại, sẽ gán quyền truy cập cho người dùng.

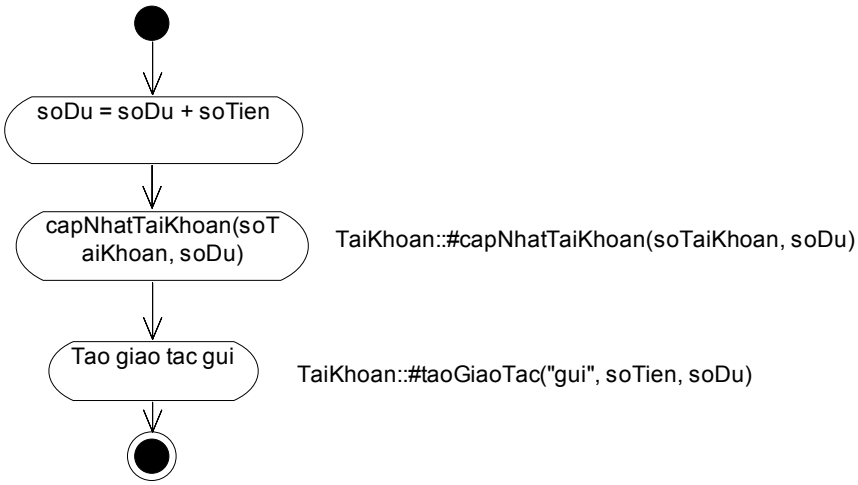
Thiết kế thuật giải dùng sơ đồ tuần tự



Với đồ trên chúng ta chỉ quan tâm đến các lớp (nghiệp vụ) sẽ cung cấp các dịch vụ gì để thực hiện *kiểmTraMậtKhẩu()*. Chúng ta chưa quan tâm đến các đối tượng ở các tầng khác như là: truy cập cơ sở dữ liệu hoặc giao diện hiển thị. Ví dụ như hành vi *lấy_KháchHàng(sốThẻ, vPIN)* sẽ phải truy cập cơ sở dữ liệu để thực hiện, cũng như đối tượng: MáyATM chỉ là giả lập giao diện giữa khách hàng và hệ thống, chúng ta chưa xác định đối tượng giao diện cụ thể nào (form, trang web,…) sẽ thực hiện. Phần này sẽ được đề cập chi tiết trong những chương sau.

Lớp TàiKhoản

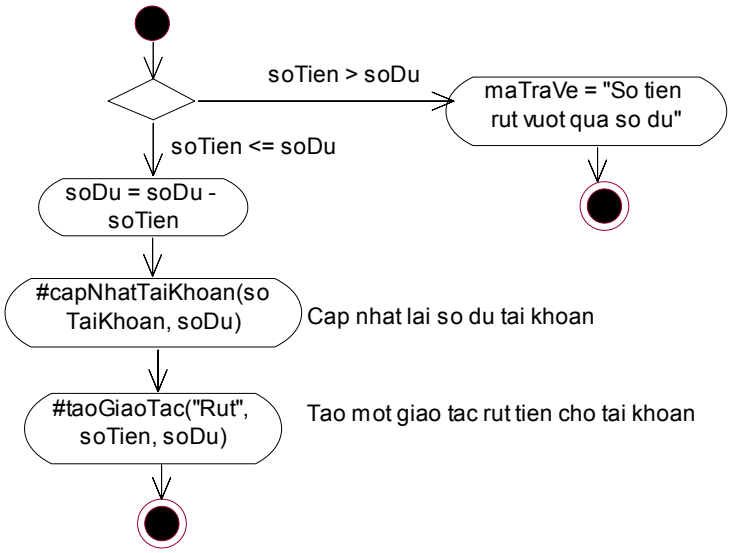
TàiKhoản::+gửiTiền(sồTiền: foat)



Khi thực hiện một việc gửi tiền, số tiền được gửi được chuyển đến một đối tượng tài khoản và được dùng như là một đối số cho hành vi *gửiTiền()*. Tài khoản này điều chỉnh số dư hiện hành của nó bằng cách cộng thêm với số tiền gửi. Sau đó, tài khoản này sẽ lưu thông tin lần gửi bằng cách tạo ra một đối tượng giao tác.

Một lần nữa chúng ta lại khảm phá ra những hành vi khác: *cậpNhậtTàiKhoản()*, hành vì này là cục bộ (#) cho phép cập nhật dữ liệu tài khoản. *tạoGiaoTác()*, cũng là hành vi cục bộ cho phép tạo một giao tác tương ứng với tài khoản này.

TàiKhoản::+rútTiền(sốTiền:float): mãTrảVề:String

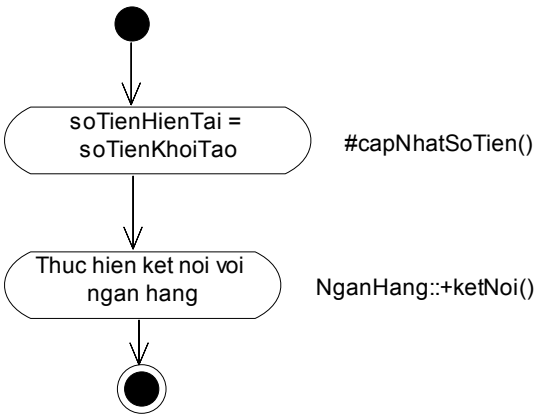


Một số tiền mà khách hàng muốn rút sẽ được chuyển đến một đối tượng tài khoản như là một tham số đầu vào. Tài khoản này sẽ kiểm tra số dư hiện hành của nó so với số tiền này. Nếu vẫn lớn hơn hoặc bằng số tiền rút thì tài khoản sẽ cập nhật lại số dư và tạo một giao tác rút tiền, ngược lại thông báo lỗi với mãTrảVề "Số tiền rút vượt quá số dư".

Một lần nữa chúng ta thấy hành vi *rútTiền* lại sử dụng *cậpNhậtTàiKhoản* và *tạoGiaoTác* đã đề cập đến trong khi thiết kế hành vi *gửiTiền*.

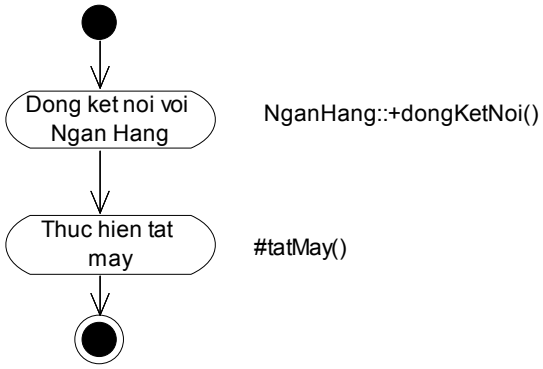
Lớp MáyATM

MáyATM::+khởiĐộngMáy(sốTiềnKhởiTạo:float)



Sau khi bật máy ATM, một số tiền khởi tạo được nhập từ nhân viên vận hành sẽ được chuyển đến đối tượng máyATM như là một tham số đầu vào. Đối tượng máyATM sẽ cập nhật lại số tiền ban đầu cho máy và sau đó thực hiện việc kết nối tới ngân hàng nhằm thực hiện việc liên kết truy cập cơ sở dữ liệu. Quá trình này chúng ta lại có nhu cầu phát sinh thêm hành vi cục bộ *#cậpNhậtSốTiền* và hành vi toàn cục *NgânHàng::+kếtNối()*.

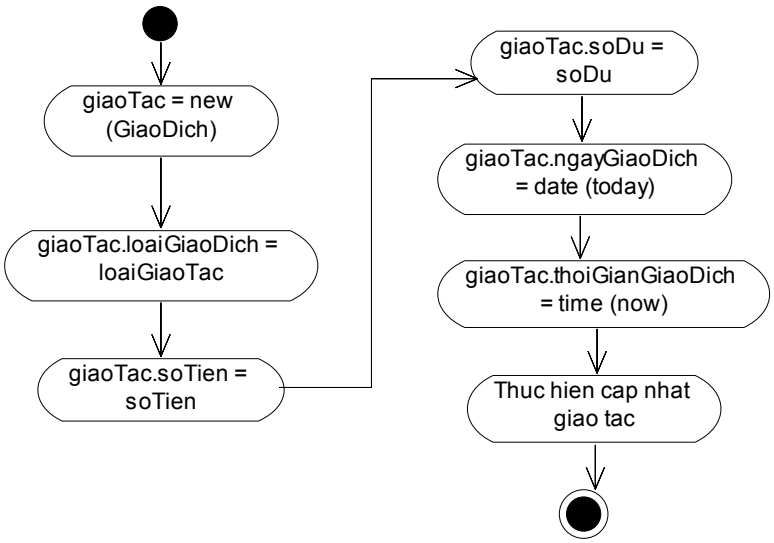
MáyATM::+đóngMáy()



Đối tượng máyATM thực hiện đóng máy bằng cách gọi thực hiện việc đóng kết nối với ngân hàng và gọi thực hiện tắt máy. Quá trình này phát sinh thêm hai hành vi: *+đóngKếtNối()*do đối tượng NgânHàng đảm nhận và *#tắtMáy()* là hành vi cục bộ của đối tượng MáyATM.

Như vậy, chúng ta đã thiết kế xong các hành vi đã được xác định ở giai đoạn phân tích quá trình thiết kế này lại phát sinh các hành vi: *#lấy_KháchHàng(), #cậpNhậtTàiKhoản(), #tạoGiaoTác(), +kếtNối(),+đóngKếtNối(), #cậpNhậtSốTiền(), #tắtMáy()*. Chúng ta lặp lại quá trình thiết kế cho những hành vi này. Chú ý rằng các hành vi liên quan đến việc truy cập dữ liệu hoặc xử lý giao diện sẽ được thiết kế ở những giai đoạn tiếp theo trong những chương sau. Các hành vi đó là: *#lấy_KháchHàng(), #cậpNhậtTàiKhoản(),....* Sau đây chúng ta tiếp tục thiết kế thuật giải cho các hành vi *tạoGiaoTác(), +kếtNối(), +đóngKếtNối(), #cậpNhậtSốTiền()*:

TàiKhoản::#tạoGiaoTác(loạiGiaoTác:String, sốTiền:float, soDu:float)



Một đối tượng tài khoản tạo một giao tác liên quan đến nó bằng cách truyền các tham số: loại giao tác (gửi, rút), số tiền, và số dư sau khi thực hiện giao tác. Đối tượng tài khoản sẽ tạo mới một đối tượng giao tác ứng với thuộc tính giaoTác của nó và gán các thông tin về giao tác đó, sau đó, lưu lại giao tác này vào cơ sở dữ liệu. Quá trình này lại phát sinh mới hai hành vi: hành vi +gánThôngTinGiaoDịch() của đối tượng giao dịch có phạm vi toàn cục; hành vi cập nhật vào cơ sở dữ liệu mà chúng ta sẽ thiết kế chi tiết trong các chương sau. Tạm thời ở đây là vấn đề nội bộ của đối tượng tài khoản.

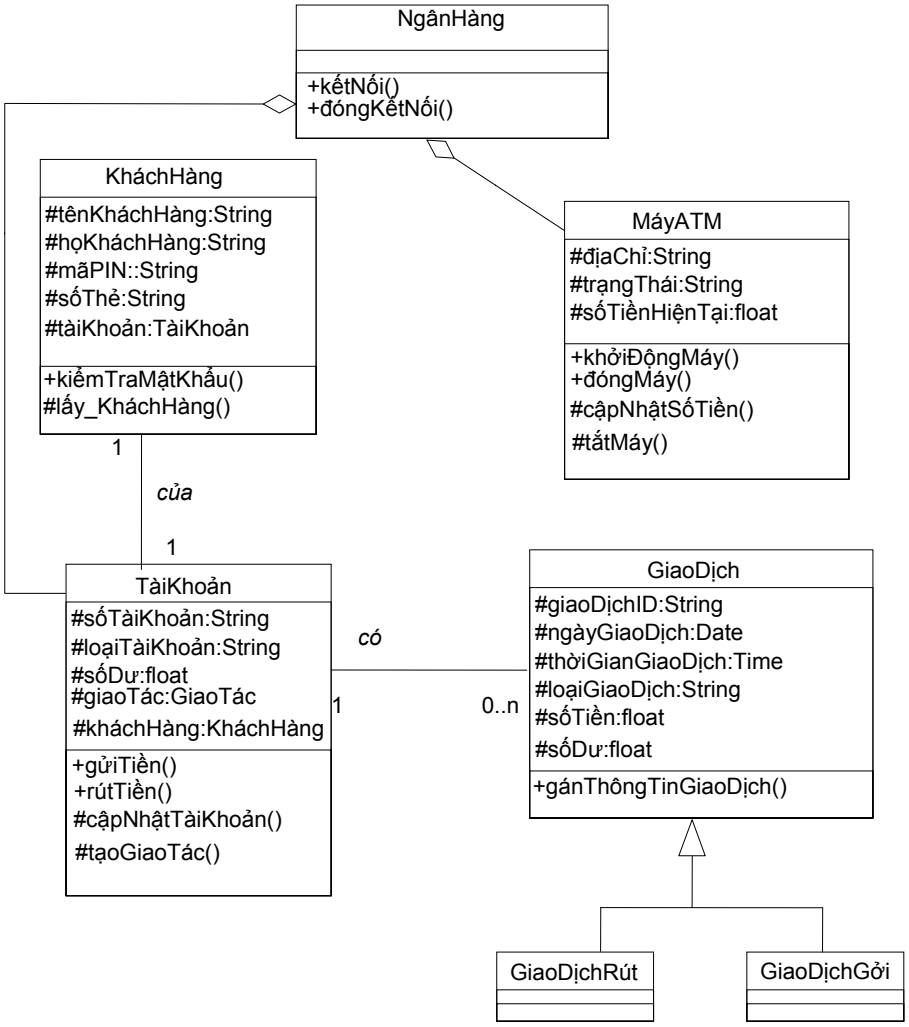
GiaoDịch:: +gánThôngTinGiaoDịch(loạiGD:String, sốTiền:float, sốDư:float, ngàyGD:Date, giờGD:Time)

Các hành vi *+kếtNối(), +đóngKếtNối(), +cậpNhậtSốTiền()* thì đơn giản do đó chúng ta chỉ mô tả khai báo nó:

NgânHàng::+kếtNối()

NgânHàng::+đóngKếtNối()

MáyATM::#cậpNhậtSốTiền(sốTiền:float)



Sơ đồ lớp của hệ thống máy ATM với kết quả tinh chế đầu tiên về thuộc tính và hành vi

Tổng kết

Bước đầu tiên trong tiến trình thiết kế hướng đối tượng là việc áp dụng các tiên đề và hệ luận để thiết kế các lớp, thuộc tính, hành vi, mối kết hợp, cấu trúc, phạm vi; quá trình này là một quá trình lặp và tinh chế. Trong giai đoạn phân tích, chỉ cần tên thuộc tính là đủ. Tuy nhiên, trong giai đoạn thiết kế, các thông tin chi tiết sẽ phải được thêm vào mô hình (đặc biệt là các định nghĩa của thuộc tính và hành vi).

Thiếu đi một nghi thức thiết kế tốt có thể phát sinh các kẽ hở về tính bao bọc (encapsulation). Vấn đề này xảy ra khi khi chi tiết về cài đặt bên trong của một lớp được phơi bày ra ở giao diện. Càng nhiều thông tin chi tiết bên trong của lớp bị phơi bày, sự uyển chuyển trong các thay đổi hệ thống sau này càng giảm. Bởi vì nó làm giảm đi tính độc lập của đối tượng trong hệ thống. Do đó, chúng ta sử dụng khai báo phạm vi cục bộ (private) hoặc bảo vệ (protected) để xác định việc cài đặt đối tượng; sử dụng khai báo phạm vi toàn cục (public) để xác định chức năng của đối tượng.

Thiết kế hướng đối tượng là một quá trình lặp. Do đó, chúng ta đừng ngại việc thay đổi tới một lớp, mỗi sự thay đổi hãy tin rằng sẽ là một sự cải tiến mô hình hiện tại. Một điều ghi nhớ rằng các vấn đề cần được giải quyết càng sớm càng tốt để khỏi phải trả giá lớn trong các giai đoạn sau.

Câu hỏi và bài tập

1. Các nghi thức public và private là gì? Ý nghĩa của việc sử dụng những nghi thức này trong thiết kế?
2. Khi thiết kế mối kết hợp, chúng ta cần thêm một thuộc tính tham chiếu tới một lớp khi chúng ta cần xác định điều gì?
3. Tại sao chúng ta lại dựa vào sơ đồ hoạt động hoặc sơ đồ tuần tự để xác định hành vi của một lớp?

Chương 9 THIẾT KẾ USE CASE

Mục tiêu

Mục tiêu của chương này cung cấp các kiến thức về:

- Cung cấp cơ bản về kiến trúc ba tầng (tree-layer)
- Xác định các lớp đối tượng ở tầng nghiệp vụ
- Xác định các đối tượng ở tầng truy cập dữ liệu và thiết kế method
- Xác định các đối tượng ở tầng giao diện và thiết kế method
- Mô tả hiện thực hoá nội dung thiết kế một use case

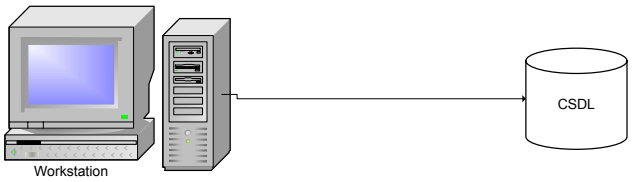
Nội dung

Mô hình use case chúng ta đạt được ở giai đoạn phân tích mô tả tính năng hệ thống từ phía người sử dụng. Các chức năng này được xem như là sự biểu diễn các yêu cầu chức năng mà hệ thống cần đáp ứng. Trong giai đoạn thiết kế, các chức năng này phải được mô tả ở gốc độ để cài đặt. Như vậy đối với một người thiết kế viên, chúng ta phải đặt câu hỏi là làm sao để biểu diễn sơ đồ use case này đủ chi tiết và trên một ngôn ngữ để có thể cài đặt được. Một cách tiếp cận là tách biệt diễn đạt chức năng thành hai phần: phần mô tả chức năng nhìn từ bên ngoài (từ phía người dùng: hệ thống có những chức năng gì cung cấp cho người sử dụng – sơ đồ use case) và phần mô tả chức chức nhìn từ bên trong (từ phía người phát triển: làm sao để hiện thực hoá các chức năng để cài đặt nó – mô tả hiện thực hoá use case).

Việc hiện thực hoá được đảm nhận bởi một các use case cộng tác và do đó thiết kế nội dung bên trong một use case chính tập trung vào thiết kế use case cộng tác. Nội dung thiết kế use case cộng tác bao gồm: xác định thêm các đối tượng hệ thống phần mềm cùng cộng tác trong việc thực hiện use case và sự tương tác giữa chúng. Trong tiếp cận kiến trúc ba tầng (tree-layer), các đối tượng hệ thống phần mềm là các đối tượng ở tầng giao diện và tầng truy cập dữ liệu cũng như các đối tượng điều khiển phần mềm. Sau đó, xác định sự tương tác giữa các đối tượng trong use case nhằm phát hiện các method cho các đối tượng của hai tầng này cũng như hoàn thiện việc tinh chế method cho các lớp. Kết quả của giai đoạn này là một sơ đồ lớp đầy đủ để chuẩn bị cho việc cài đặt hệ thống phần mềm trong giai đoạn sau.

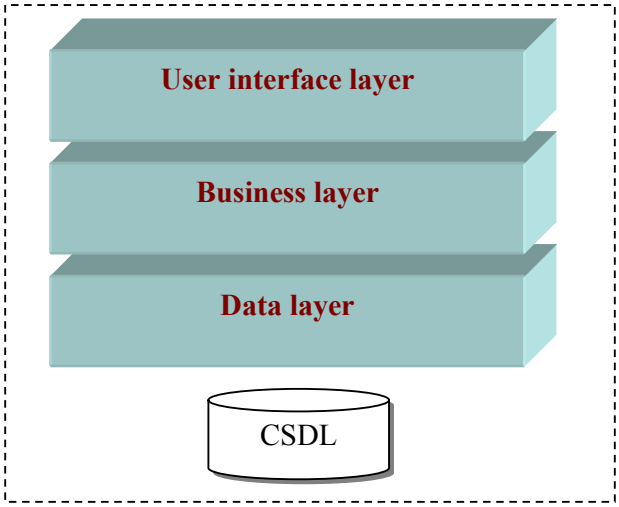
Kiến trúc 3 tầng (three - layer)

Hầu hết các hệ thống được phát triển sử dụng các công cụ CASE ngày nay hoặc trong các môi trường phát triển ứng dụng client – server có xu hướng xây dựng một kiến trúc hai tầng (two – layer): giao diện (interface) và dữ liệu (data). Trong một hệ thống hai tầng, các màn hình giao diện người dùng liên kết để truy cập dữ liệu thông qua các đoạn chương trình được cài trực tiếp trên các giao diện. Ví dụ, một chương trình viết trên Visual Basic có một form giao diện, một thủ tục xử lý biến cố trong button "Update" của form này có tên bUpdate_Click() có thể thực hiện luôn việc truy cập và cập nhật CSDL trực tiếp, như vậy thủ tục này cài đặt luôn các ngữ nghĩa về tác nghiệp (business). Việc thiết kế theo mô hình này tạo ra một sự phụ thuộc rất lớn giữa giao diện và CSDL và do đó, rất khó để cải tiến, bảo trì và tái sử dụng.



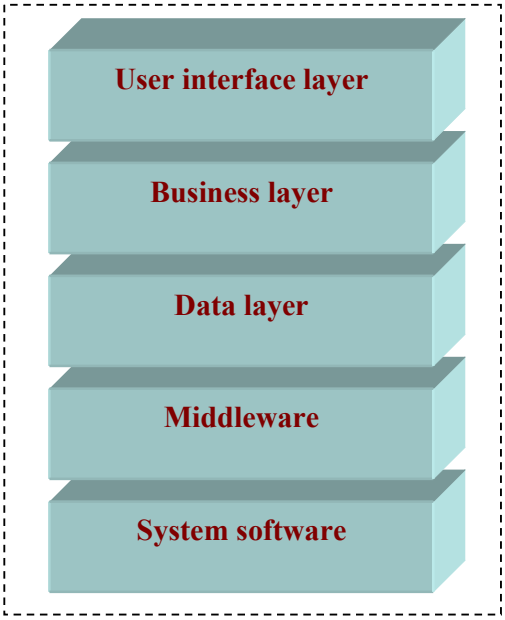
Hình 5. Kiến trúc hai lớp: giao diện và dữ liệu

Một cách tiếp cận kiến trúc khác tốt hơn chính là tạo ra sự độc lập giữa giao diện và người sử dụng bằng cách cô lập các chức năng của giao diện với các chức năng tác nghiệp (business), và cô lập các chức năng tác nghiệp với các chi tiết về truy cập CSDL, đó là cách tiếp cận ba tầng (three-layer). Từ cách tiếp cận này cho phép chúng ta tạo ra được các đối tượng đại diện các đối tượng hữu hình trong thực tế nhưng hoàn toàn độc lập với cách thức mà các đối tượng này trình bày tới người dùng hoặc là với cách mà dữ liệu của nó được lưu trữ vật lý trong CSDL. Do đó, ba tầng trong cách tiếp cận này là: tầng giao diện người dùng (user interface layer), tầng tác nghiệp (business layer), và tầng truy cập dữ liệu (data layer).



Hình 6. Sơ đồ biểu diễn tiếp cận ba tầng

Một tiếp cận khác đầy đủ hơn của một kiến trúc hệ thống có thể được trình bày như sơ đồ sau:



Hình 7. Một sơ đồ khác của cách tiếp cận ba tầng (nhiều tầng)

Trong đó,

Tầng Middleware: chứa các thành phần xây dựng giao diện (ví dụ: thành phần dạng ActiveX), thành phần giao diện tới các hệ quản trị CSDL (ví dụ: ODBC, JDBC driver), các dịch vụ hệ điều hành độc lập với platform, các thành phần nhúng OLE (ví dụ: các công cụ soạn thảo sơ đồ nhúng, các bảng tính nhúng,…).

Tầng System software: chứa các thành phần về hệ điều hành, CSDL, giao diện tới các phần cứng (ví dụ: các driver phần cứng cụ thể), v.v…

Xác định lớp ở tầng dịch vụ tác nghiệp (business layer)

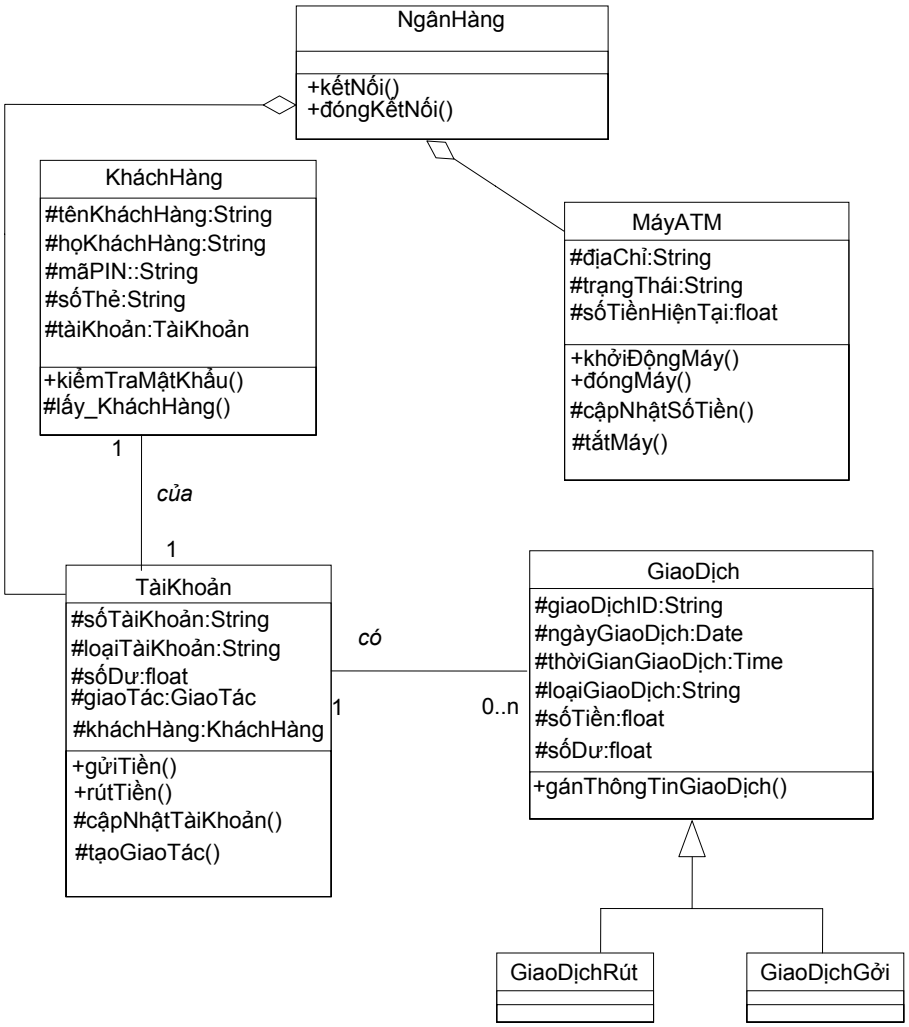
Tầng này chứa đựng tất cả đối tượng mô tả tác thành phần nghiệp vụ hệ thống (bao gồm cả dữ liệu và hành vi). Nó diễn đạt các đối tượng tồn tại trong thực tế vào trong hệ thống cần quản lý. Ví dụ, đơn đặt hàng, khách hàng, hoá đơn, nhà cung cấp,… hầu hết các phương pháp luận phân tích thiết kế đều đưa ra phương pháp xác định đối tượng này trong giai đoạn phân tích (tham khảo chương 6). Tuy nhiên, khi xác định các đối tượng ở lớp này chúng ta phải luôn nhớ hai điều sau:

- Các đối tượng ở tầng tác nghiệp không nên quan tâm đến cách thức nó được hiển thị và bởi ai. Các đối tượng này được thiết kế để độc lập với bất kỳ một giao diện cụ thể, và vì vậy cách thức chi tiết để hiển thị một đối tượng nên tồn tại trong tầng giao diện thay vì trong tầng tác nghiệp.
- Các đối tượng ở tầng tác nghiệp cũng không nên quan tâm đến nguồn gốc của nó hình thành. Có nghĩa là các đối tượng này sẽ độc lập về dữ liệu của nó được lấy từ truy cập CSDL hay là từ truy xuất tập tin.

I.1.1. Xác định các lớp tầng tác nghiệp

Các lớp ở tầng tác nghiệp đã được xác định trong giai đoạn phân tích (xem chương 6). Trong phần này chúng tả mô tả lại theo từng use case để cho phép chúng ta có một cách nhìn về những gì mà các đối tượng ở tầng tác nghiệp kế hợp với nhau trong hoạt động đáp ứng yêu cầu sử dụng được mô tả thông qua use case. Chú ý rằng một lớp trong tầng này đều có thể tham gia xử lý trong nhiều use case khác nhau.

Các kết quả thiết kế lớp trong chương 8 đã cho chúng ta một sơ đồ thiết kế về tầng nghiệp vụ này.



Sơ đồ lớp của hệ thống ATM tầng nghiệp vụ

Xác định lớp ở tầng truy cập dữ liệu (data layer)

Tầng này chứa các đối tượng nhằm mục đích cung cấp các dịch vụ về dữ liệu cho tầng tác nghiệp. Nói chung, tất cả các nhu cầu truy cập CSDL hoặc tập tin để thao tác trên dữ liệu được lưu trữ của hệ thống đều phải thông qua tầng này. Do đó, các đối tượng tầng này phải truy cập vật lý CSDL ở các vị trí (CSDL quan hệ, tập tin, internet,…) và xử lý nó. Hai nhiệm vụ chính của tầng này là:

- Chuyển dịch các yêu cầu: chuyển dịch tất cả các yêu cầu liên quan đến dữ liệu từ tầng tác nghiệp đến một phương thức truy cập dữ liệu thích hợp (dạng SQL, truy xuất file,…)
- Chuyển dịch kết quả: chuyển dịch tất cả dữ liệu truy cập được tới các đối tượng tác nghiệp thích hợp.

Xác định các đối tượng lưu trữ và persistence

Một chương trình sẽ tạo ra một số lượng dữ liệu trong quá trình thực thi. Mỗi dữ liệu sẽ có một thời gian sống (lifetime) khác nhau. Dựa vào thời gian sống này chúng ta có thể phân thành những loại dữ liệu sau:

- Là kết quả tạm thời để đánh giá một biểu thức
- Các biến trong quá trình thực thi một thủ tục (các tham số và biến trong phạm vi cục bộ)

- Các biến toàn cục và các biết cấp phát một cách tự động
- Dữ liệu tồn tại giữa các lần thực thi một chương trình
- Dữ liệu tồn tại giữa các phiên bản của một chương trình
- Dữ liệu tồn tại vượt ngoài phạm vi sống của một chương trình

Ba loại dữ liệu đầu tiên đều gọi là dữ liệu tạm thời (transient data). Là dữ liệu có thời gian sống phụ thuộc vào thời gian sống của tiến trình sử dụng nó. Khi tiến trình kết thúc thì dữ liệu này bị giải phóng. Ngược lại, ba loại dữ liệu cuối gọi là dữ liệu persistent (persistent data). Các dữ liệu này tồn tại lâu hơn tiến trình sử dụng nó và có thể độc lập với tiến trình này. Các ngôn ngữ lập trình cung cấp sự hỗ trợ hoàn hảo cho các loại dữ liệu tạm thời. Các loại dữ liệu persistent sẽ được quản lý bởi hệ quản trị cơ sở dữ liệu hoặc hệ thống lưu trữ tập tin.

Đối với các đối tượng cũng được áp dụng một cách tương tự. Các đối tượng cũng có một thời gian sống. Chúng được tạo ra một cách tường minh và có thể tồn tại trong một khoảng thời gian ngắn (thời gian của một ứng dụng, của một phiên,…). Tuy nhiên, cũng có đối tượng tồn tài lâu hơn thời gian của một ứng dụng, để làm điều này thì đối tượng phải được lưu trữ ra tập tin hoặc cơ sở dữ liệu. Việc lưu trữ này sẽ cung cấp cho đối tượng thời gian sống lâu hơn quá trình của tiến trình. Đặc tính này người ta gọi là *persistent*.

Trong hệ thống ATM, các lớp persistent gồm: KháchHàng, TàiKhoản, GiaoTác, GiaoTácGửi, GiaoTácRút. Các lớp như MáyATM, NgânHàng được xác định không phải là lớp persistent bởi vì chúng ta không có nhu cầu lưu trữ thông tin của các đối tượng các lớp này.

Chuyển đổi đối tượng sang mô hình quan hệ

Trong cơ sở dữ liệu quan hệ, một lược đồ được hình thành bởi các bảng (table) gồm các cột và dòng. Trong mô hình đối tượng, tương ứng tới một bảng là một lớp (hoặc nhiều lớp). Các thành phần tương ứng như sau:

- Một cột ứng với một thuộc tính (persistent) lớp
- Một dòng của bảng ứng với một đối tượng (thể hiện của một lớp)
- Một thủ tục lưu trữ nội (stored procedure) có thể tương ứng với một method.

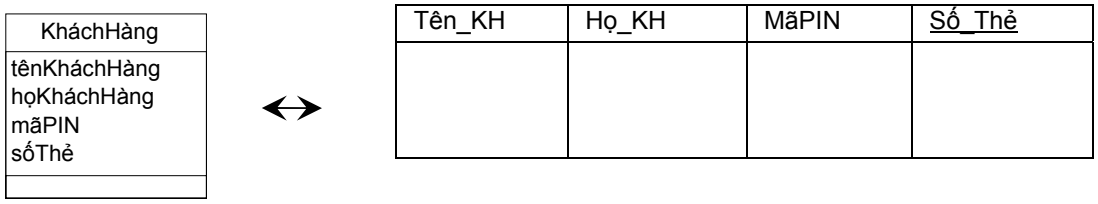
Như vậy, việc chuyển đổi sơ đồ lớp sang lược đồ quan hệ gồm có những công việc sau:

- Chuyển đổi lớp – bảng
- Chuyển đổi mối liên kết
    - Chuyển đổi liên kết kết hợp
    - Chuyển đổi liên kết kế thừa

Chuyển đổi lớp – bảng

Trong đa số các trường hợp thì việc chuyển đổi này là một – một. Một lớp sẽ chuyển thành một bảng cụ thể như sau:

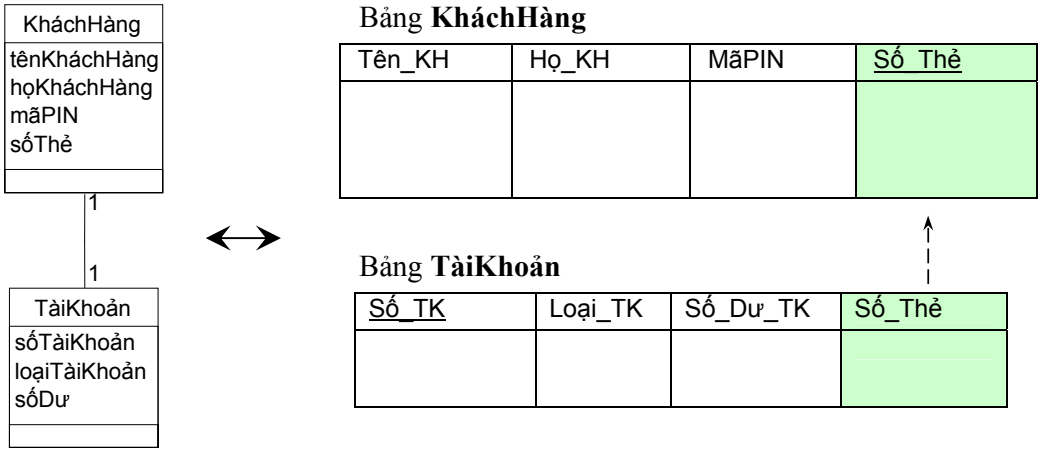
- Một lớp → một bảng
- Một thuộc tính (persistent) → một cột: chỉ có các thuộc tính có nhu cầu lưu trử và được đòi hỏi bởi ứng dụng sẽ được chuyển thành cột của bảng.
- Một đối tượng (thể hiện) → một dòng



Chuyển đổi mối liên kết

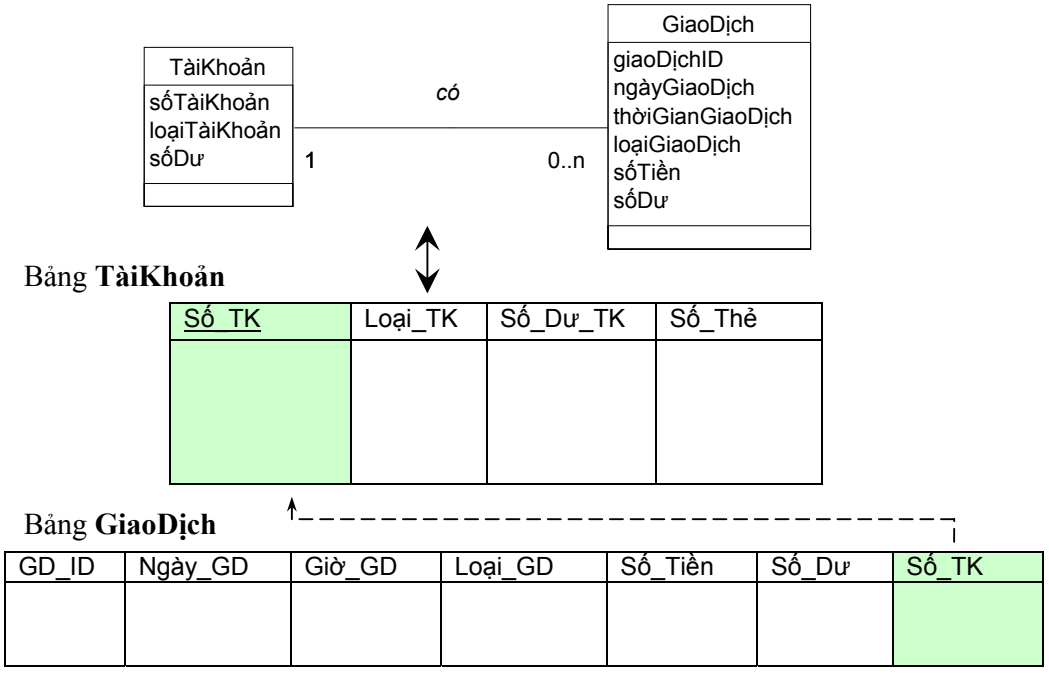
Chuyển đổi liên kết kết hợp(association)

Trường hợp 1



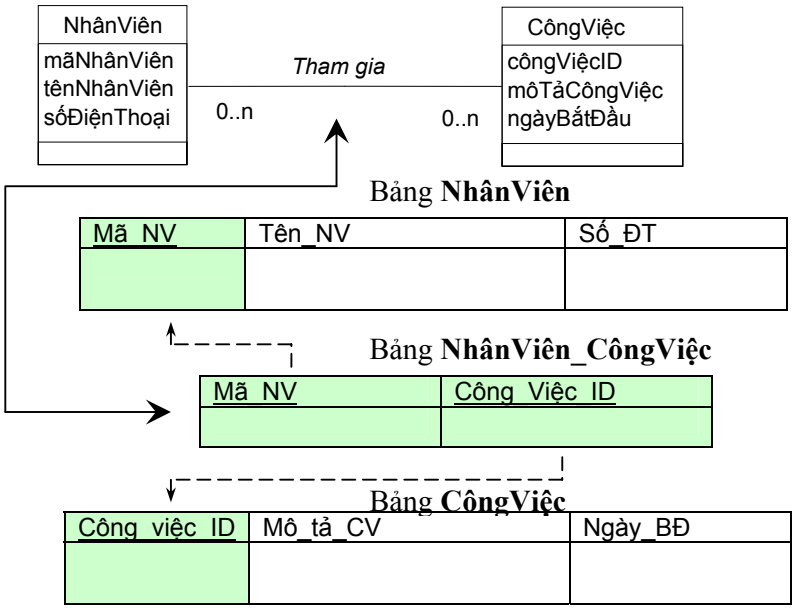
Trong chuyển đổi mối kết hợp dạng 1 – 1, chúng ta có thể lấy cột khoá chính trong một bảng chuyển qua bảng khác làm khoá ngoại. Trong mối kết hợp 1 – 1 trên (giữa lớp KháchHàng và TàiKhoản), chúng ta có thể lấy khoá của bảng KháchHàng (Số_Thẻ) đưa vào bảng TàiKhoản là khoá ngoại. Hoặc ngược lại, chúng ta có thể lấy khoá của bảng TàiKhoản (Số_TK) đưa vào bảng KháchHàng làm khoá ngoại. Hoặc thực hiện cả hai trường hợp.

Trường hợp 2



Trường hợp 3

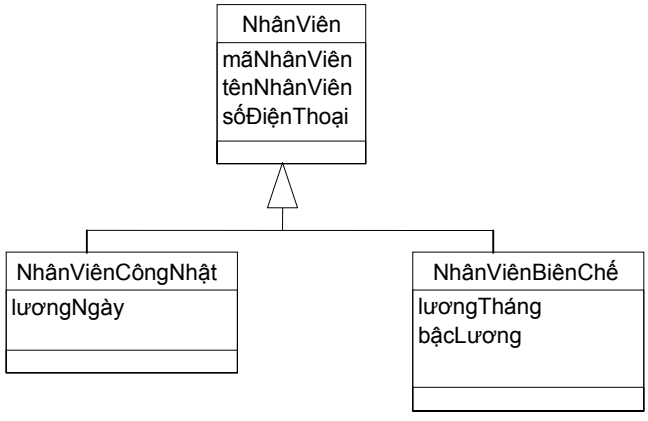
Trong chuyển đổi mối kết hợp dạng 1 – n, chúng ta lấy cột khoá chính của bảng ứng với lớp phía 1 trong mối kết hợp đưa vào bảng ứng với lớp phía n làm khoá ngoại. Trong ví dụ trên, bảng GiaoDịch sẽ lấy thuộc tính Số_TK trong bảng TàiKhoản làm khoá ngoại.



Trong chuyển đổi mối kết hợp n – n, chúng ta tạo ra một bảng cho mối kết hợp đó bằng cách lấy các khoá chính của các bảng đưa vào bảng mới này như là các khoá ngoại. Trong ví dụ trên, mối kết hợp *Tham gia* giữa lớp NhânViên và lớp CôngViệc sẽ được mô tả bởi một bảng NhânViên_CôngViệc bao gồm các cột Mã_NV và Công_Việc_ID lần lượt là khoá chính của bảng NhânViên và bảng CôngViệc.

Chuyển đổi liên kết kế thừa

Trong lược đồ quan hệ không có khái niệm kế thừa mà chúng ta thường dùng liên kết khoá chính – khoá ngoại để diễn đạt điều này. Sau đây là các trường hợp mà chúng ta có thể quyết định chọn một:



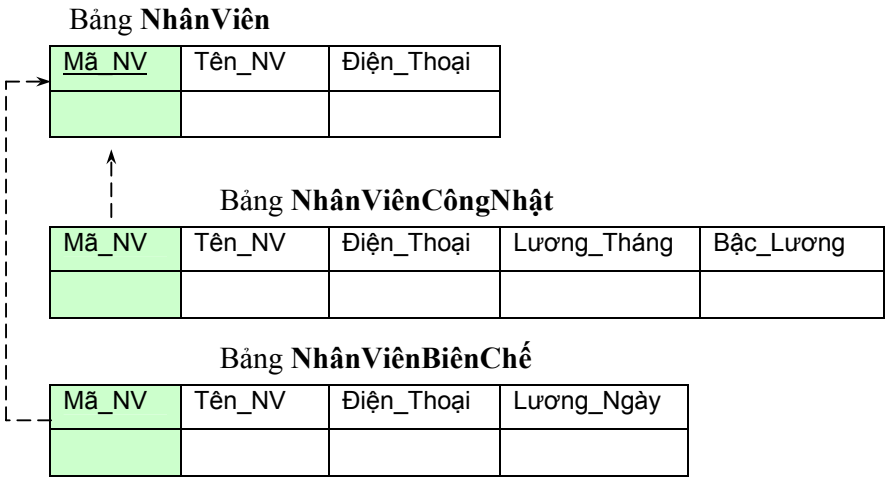
Trường hợp 1

Bảng **NhânViên**

Mã_NV	Tên_NV	Điện_Thoại	Lương_Ngày	Lương_Tháng	Bậc_Lương	***Loại_NV***

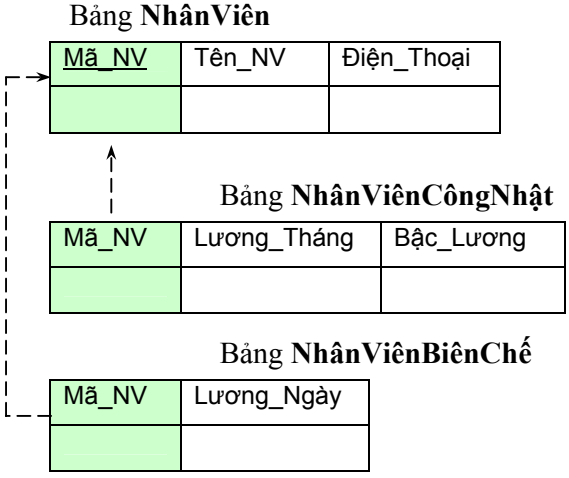
Chỉ sử dụng một bảng lưu trữ tất cả các loại nhân viên. Do đó, các thuộc tính của bảng được hình thành từ các thuộc tính của lớp NhânViên, NhânViênCôngNhật và Nhân ViênBiênChế. Ngoài ra chúng ta cũng đưa vào thêm một thuộc tính nhằm phân loại dòng này thuộc đối tượng của lớp nào.

Trường hợp 2



Sử dụng ba bảng tương ứng cho ba lớp. Tuy nhiên, nhằm mô tả sự thừa kế trong các bảng NhânViênCôngNhật và NhânViênBiênChế chúng ta thêm vào tất cả các thuộc tính của bảng nhân viên. Các thể hiện tương ứng của các nhân viên công nhật hay biên chế sẽ chỉ lưu trong bảng tương ứng.

Trường hợp 3



Giống như trường hợp 2. Ở hai bảng NhânViênCôngNhật và NhânViênBiênChế chỉ lưu khoá ngoại tham chiếu đến bảng nhân viên để tham khảo thông tin thừa kế.

Trường hợp 4

Bảng **NhânViênCôngNhật**

Mã_NV	Tên_NV	Điện_Thoại	Lương_Tháng	Bậc_Lương

Bảng **NhânViênBiênChế**

Mã_NV	Tên_NV	Điện_Thoại	Lương_Ngày

Chỉ dùng hai bảng NhânViênCôngNhật và NhânViênBiênChế. Tuy nhiên, tất cả các thuộc tính của lớp NhânViên sẽ được đưa vào hai bảng này nhằm mô tả sự thừa kế. Nếu chúng ta muốn truy xuất thông tin về lớp NhânViên thì chúng ta có thể tạo một khung nhìn (view) hội của hai bảng này.

Sơ đồ cài đặt các đối tượng persistent của hệ thống ATM dùng mô hình quan hệ như sau:

Bảng **KháchHàng**

Tên_KH	Họ_KH	MãPIN	Số_Thẻ

Bảng **TàiKhoản**

Số_TK	Loại_TK	Số_Dư_TK	Số_Thẻ

Bảng **GiaoDịch**

GD_ID	Ngày_GD	Giờ_GD	Loại_GD	Số_Tiền	Số_Dư	Số_TK

Trong lược đồ trên của máy ATM, chúng ta áp dụng trường hợp 1 cho cấu trúc các lớp GiaoDịch, GiaoDịchGửi và GiaoDịchRút.

Xác định lớp ở tầng truy cập dữ liệu

Mục tiêu chính của việc tạo ra một tầng truy cập dữ liệu là để tạo ra các lớp có nhiệm vụ truy cập tới các vị trí mà dữ liệu thực sự được lưu trữ nhằm giúp cho tầng nghiệp vụ không quan tâm đến vị trí cũng như hình thức lưu trữ của dữ liệu này (dạng tập tin, cơ sở dữ liệu quan hệ, cơ sơ dữ liệu đối tượng, internet, DCOM,…). Các đối tượng ở tầng này phải có trách nhiệm cung cấp một liên kết giữa nghiệp vụ (cách nhìn theo đối tượng) và dữ liệu lưu trữ. Tiến trình xác định các lớp ở tầng này gồm các bước sau:

- Với mỗi lớp persistent ở tầng nghiệp vụ, tạo một lớp tương ứng ở tầng truy cập dữ liệu. Ví dụ, nếu chúng ta có ba lớp ở tầng nghiệp vụ là Class1, Class2, Class3 thì chúng ta tạo ra ba lớp tương ứng ở tầng truy cập dữ liệu là: ClassDB1, ClassDB2, ClassDB3.
- Xác định mối kết hợp: tương tự như xác định mối kết hợp với các lớp ở tầng nghiệp vụ. Tạo mối kết hợp giữa lớp tầng truy cập dữ liệu và lớp của nó tương ứng ở tầng

nghiệp vụ là mối kết hợp dạng thành phần (aggregation). Tạo thuộc tính tham chiếu cho các lớp của tầng nghiệp vụ tham chiếu đến lớp ở tầng truy cập dữ liệu dựa trên mối liên kết vừa xác định.

- Đơn giản hoá các lớp và mối kết hợp: mục tiêu chính là để loại các lớp và cấu trúc dư thừa hoặc không cần thiết. Thông thường, chúng ta kết hợp nhiều lớp thành một và đơn giản hoá cấu trúc lớp cha – lớp con.
    - Các lớp dư thừa: nếu chúng ta có hai lớp trở lên cùng cung cấp các dịch vụ tương tự nhau, chúng ta giữ lại một và loại đi một.
    - Các method: xem lại các lớp chỉ có một hoặc hai method có thể kết hợp với các lớp khác? Thông thường, chúng ta chỉ quan tâm đến các method có nhu cầu truy cập đến dữ liệu lưu trữ. Các method đó là: đọc dữ liệu từ dữ liệu lưu trữ của đối tượng, xoá dữ liệu của đối tượng khỏi dữ liệu lưu trữ và cập nhật các thay đổi của đối tượng xuống dữ liệu lưu trữ.
- Lặp lại và tinh chế tiến trình này

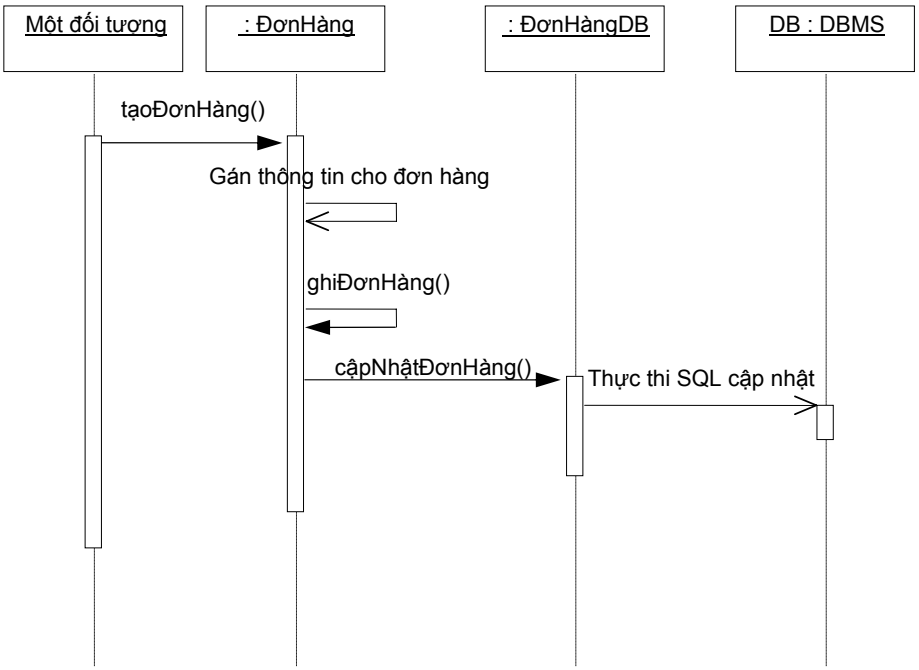
Xác định method các lớp tầng truy cập dữ liệu

Trong thiết kế hướng đối tượng nhằm đảm bảo tính bao bọc trong các cài đặt chi tiết, chúng ta mong muốn các đối tượng persistent được quan sát giống như một đối tượng tạm thời (transient) trong hệ thống. Nghĩa là chúng ta phải tạo được một cách nhìn trong suốt cho các đối tượng trong hệ thống mà không phân biệt và xử lý khác nhau giữa đối tượng persistent và bất kỳ đối tượng nào khác. Tuy nhiên, đây cũng là một vấn đề của hệ thống bởi vì dữ liệu và trạng thái của các đối tượng persistent được quản lý tách biệt khỏi chương trình ứng dụng. Do đó, sự nhất quán giữa đối tượng trong ứng dụng và trạng thái của nó trong cơ sở dữ liệu phải luôn luôn được đặt ra trong thiết kế hệ thống hướng đối tượng. Để đảm bảo điều này thì có nhiều khía cạnh của đối tượng một ứng dụng phải kiểm soát:

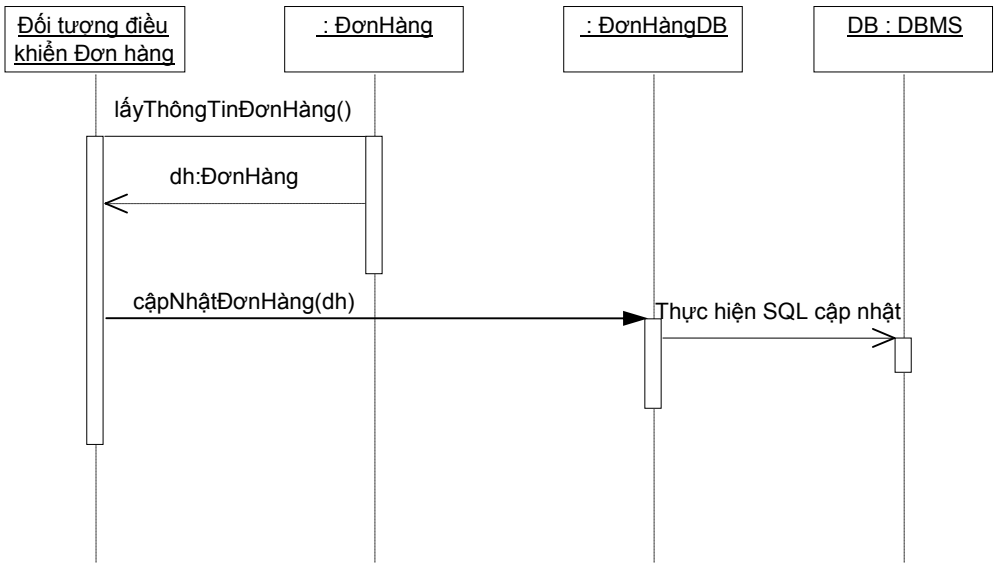
- Đọc và lưu các đối tượng persistent
- Xoá các đối tượng persistent
- Quản lý giao tác trên các đối tượng persistent
- Kiểm soát cơ chế khoá và truy cập đồng hành

Ghi đối tượng persistent

Có hai trường hợp cần xem xét: thời điểm ban đầu khi đối tượng được tao ra và phải ghi vào cơ sở dữ liệu; các thời điểm tiếp theo khi chương trình cập nhật trạng thái của đối tượng và thay đổi này cũng được ghi vào cơ sở dữ liệu.



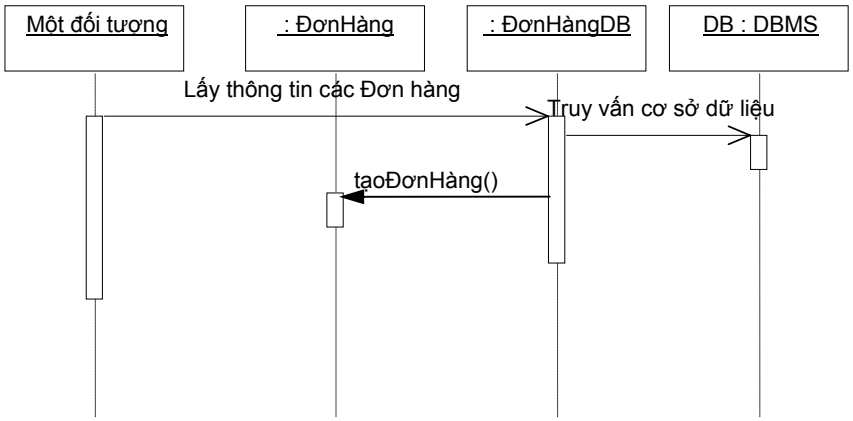
Sơ đồ tuần tự trên mô tả một trong những giải pháp cập nhật đối tượng persistent. Khi một đối tượng đơn hàng được tạo ra trong ứng dụng, đối tượng này ngay lập tức được ghi vào cơ sở dữ liệu bởi lớp ĐơnHàngDB của tầng truy cập dữ liệu. Tương tự cho hoạt động cập nhật, khi một đối tượng trong ứng dụng thay đổi trạng thái của đối tượng đơn hàng. Ngay lập tức sự thay đổi này được cập nhật vào cơ sở dữ liệu bởi ĐơnHàngDB.



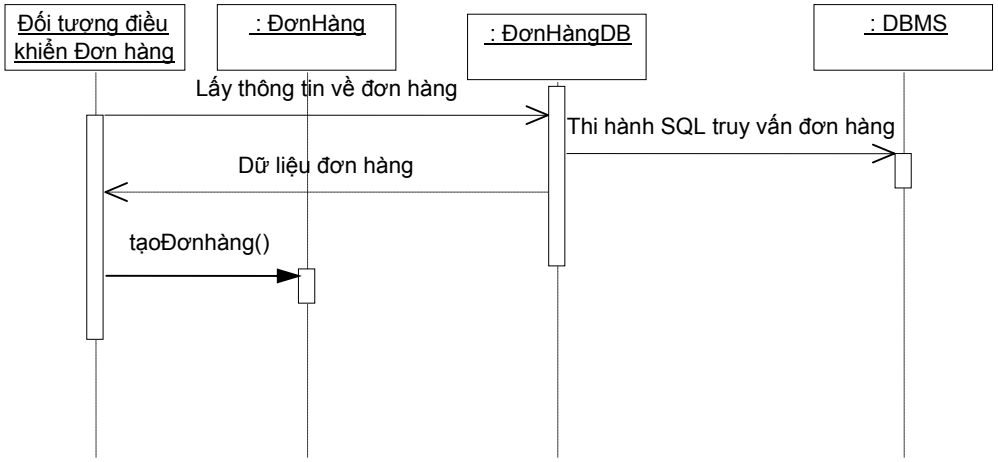
Một giải pháp khác cho cập nhật đối tượng persistent. Chúng ta xây dựng một đối tượng điều khiển và khi nào thông tin đối tượng persistent đuợc cập nhật sẽ được quản lý bởi đối tượng điều khiển. Method cậpNhậtĐơnHàng() sẽ được thiết kế thông minh để nhận ra một đơn hàng là tạo mới hay cập nhật. Trong giải pháp này, việc cập nhật được quản lý một cách tường minh và hệ thống chấp nhận các khoảng thời điểm thiếu sự nhất quán giữa đối tượng và dữ liệu lưu trữ về đối tượng.

Đọc đối tượng persistent

Việc lấy thông tin về các đối tượng persistent trong sơ sở dữ liệu cần thiết trước khi một ứng dụng gởi các thông điệp tới đối tượng. Vấn đề là khi gửi thông điệp cho một đối tượng thì phải đảm bảo đối tượng đó đã tồn tại trong bộ nhớ. Do vậy, chúng ta cần thiết kế một method truy vấn cơ sở dữ liệu để nhận đúng thông tin về đối tượng. Thông thường các đối tượng trong hệ thống liên kết với nhau qua mối kết hợp và vì vậy chúng ta đọc thông tin đầu tiên cho đối tượng **gốc**, rồi sau đó chúng ta sẽ đọc tiếp theo các đối tượng còn lại dựa theo mối kết hợp với đối tượng gốc.

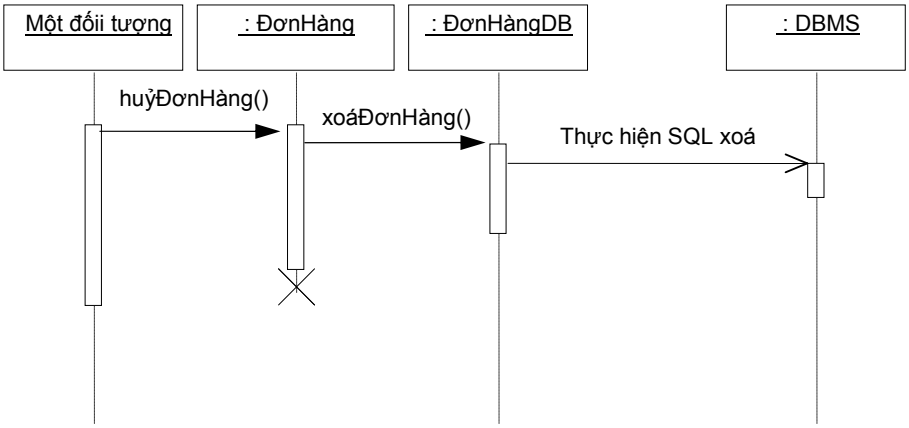


Một giải pháp khác là dùng một đối tượng điều khiển quản lý việc độc và tạo đối tượng đơn hàng được minh hoạ theo sơ đồ sau:

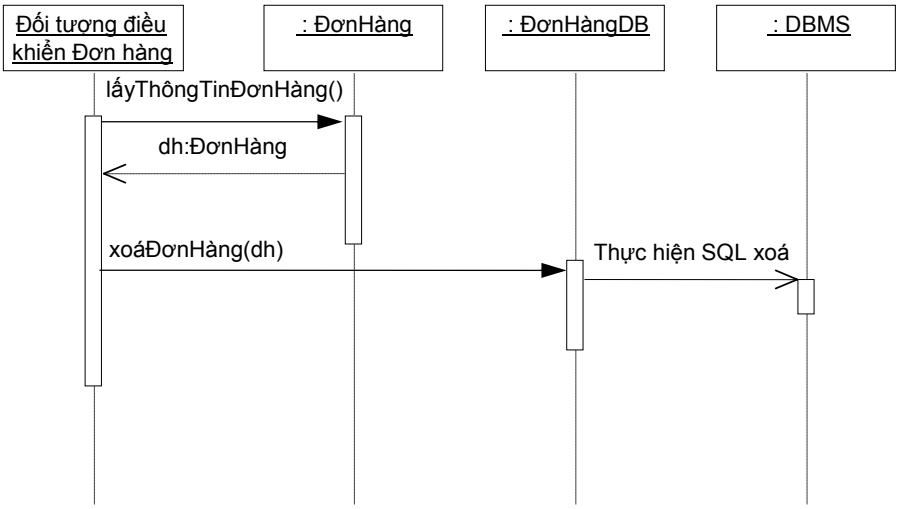


Xoá đối tượng persistent

Không giống như các đối tượng tạm thời (transtient) trong hệ thống, việc kết thúc sự hiện diện của đối tượng trong ứng dụng sẽ kết thúc vòng đời của đối tượng đó. Ngược lại, đối với đối tượng persistent, việc kết thúc vòng đời của nó đòi hỏi phải huỷ bỏ dữ liệu của nó khỏi nơi lưu trữ (hoặc ít nhất đánh dấu nó không còn hoạt động).



Một giải pháp khác dùng một đối tượng điều khiển

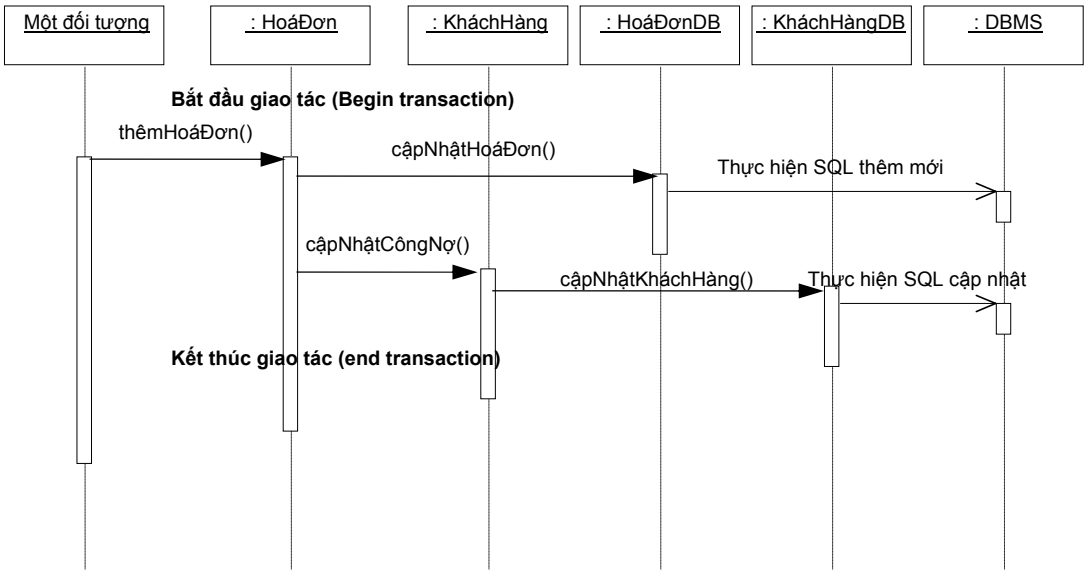


Quản lý giao tác (transaction)

Một giao tác xác định một tập các phép toán cập nhật trong cơ sở dữ liệu thành một đơn vị nhằm đáp ứng cho một yêu cầu cập nhật nghiệp vụ mà trong đó, hệ thống đòi hỏi phải cập nhật nhiều nguồn dữ liệu của nhiều đối tượng persistent khác nhau. Tất cả các phép toán cập nhật một giao tác hoặc thành công (thì giao tác thành công) hoặc không một phép toán nào thành công (thì giao tác không thành công).

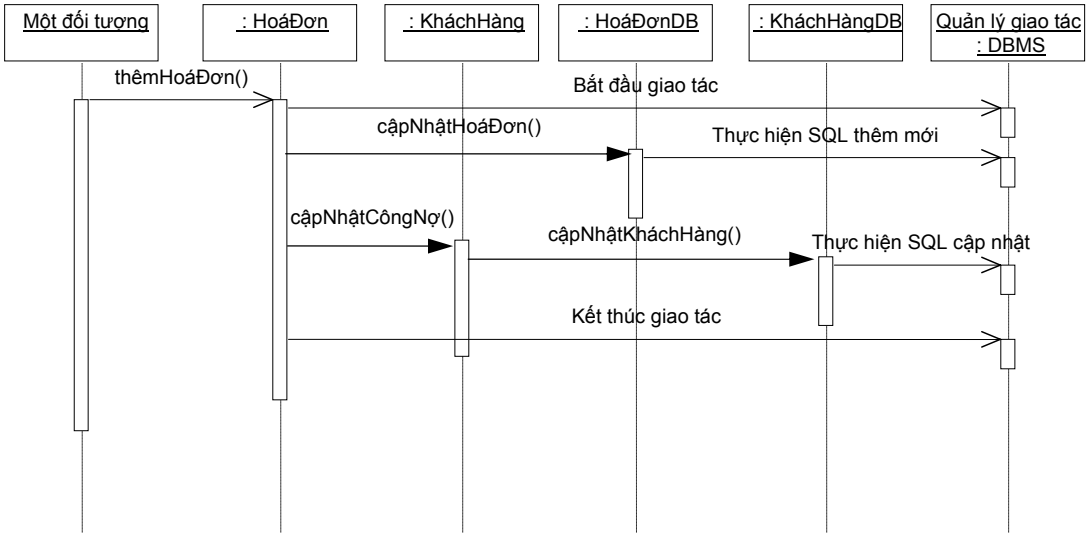
Có hai cách mô hình hoá một giao tác như sau:

Dùng văn bản để chỉ ra bắt đầu và kết thúc một giao tác



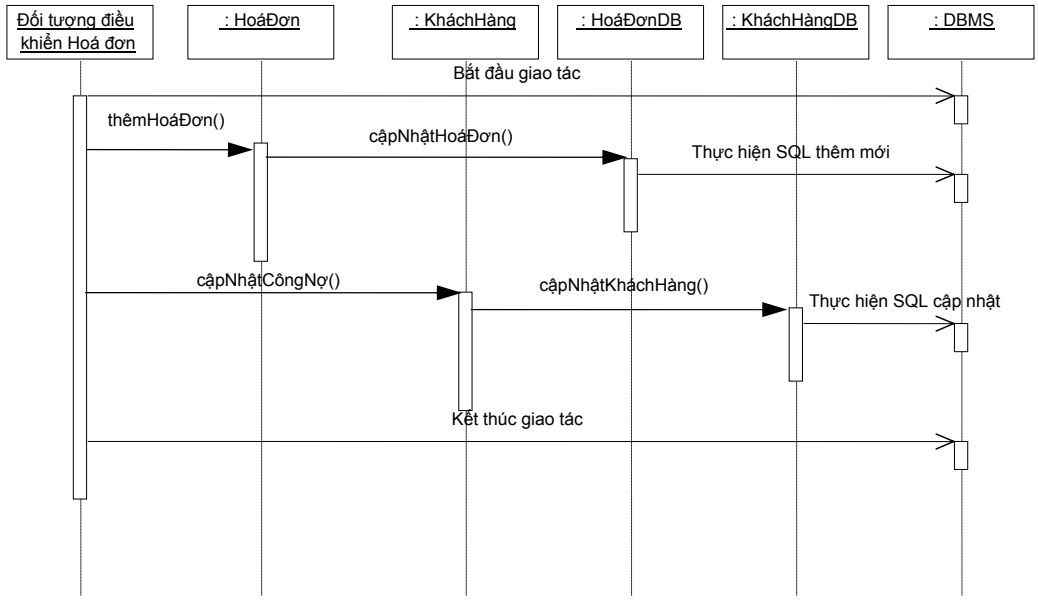
Sơ đồ trên minh hoạ việc quản lý một giao tác thêm một hoá đơn thanh toán. Giao tác này bao gồm hai phép toán cập nhật. Một là thêm mới một hoá đơn và hai là cập nhật lại thông tin công nợ của khách hàng. Giao tác kết thúc thành công nếu cả hai phép toán cập nhật đều thành công. Nếu một trong hai phép toán thực hiện không thành công thì giao tác sẽ không thành công và lúc này tất cả hai phép toán trên xem như chưa được thực hiện.

Dùng đối tượng quản lý giao tác



Chúng ta tạo một đối tượng Quản lý giao tác, tuỳ thuộc vào ngôn nhữ lập trình mà đối tượng này có kiểu khác nhau. Đối với một số ngôn ngữ thì nó là một đối tượng kiểu kết nối (connection) hoặc đối tượng kiểu thành phần hệ quản trị cơ sở dữ liệu trong ngôn ngữ lập trình đó. Sơ đồ minh hoạ trên chọn đối tượng quản lý giao tác là một đối tượng kiểu thành phần hệ quản trị cơ sở dữ liệu.

Một giải pháp dùng đối tượng điều khiển để quản lý giao tác được minh hoạ như sơ đồ dưới đây:



Trong sơ đồ, đối tượng điều khiển sẽ quản lý việc bắt đầu và kết thúc giao tác cũng như nội dung của một giao tác bằng việc quản lý các lệnh cập nhật của giao tác. Giải pháp này so với giải pháp trên có ưu điểm sau: làm cho đối tượng hoá đơn độc lập với đối tượng khách hàng. Thay vì làm cho đối tượng hoá đơn phụ thuộc vào đối tượng khách hàng qua việc gọi cậpNhậtCôngNợ() ở giải pháp trên. Giải pháp này chuyển sự phụ thuộc của hoá đơn và khách hàng vào đối tượng điều khiển và giữa đối tượng hoá đơn và khách hàng lúc này hoàn toàn độc lập.

Xác định lớp truy cập dữ liệu cho hệ thống ATM

Từ các lớp persistent của hệ thống ATM được xác định là: KháchHàng, TàiKhoản, GiaoDịch, GiaoDịchRút, GiaoDịchGửi chúng ta tạo ra các lớp truy cập dữ liệu tương ứng:

KháchHàngDB

TàiKhoảnDB

GiaoDịchDB (GiaoDịch, GiaoDịchRút, GiaoDịchGửi)

Các method được xác định:

KháchHàngDB::+đọcKháchHàng()

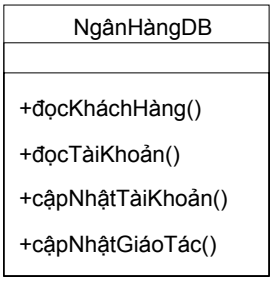
KháchHàng::+đọcTàiKhoản()

TàiKhoảnDB::+cậpNhậtTàiKhoản()

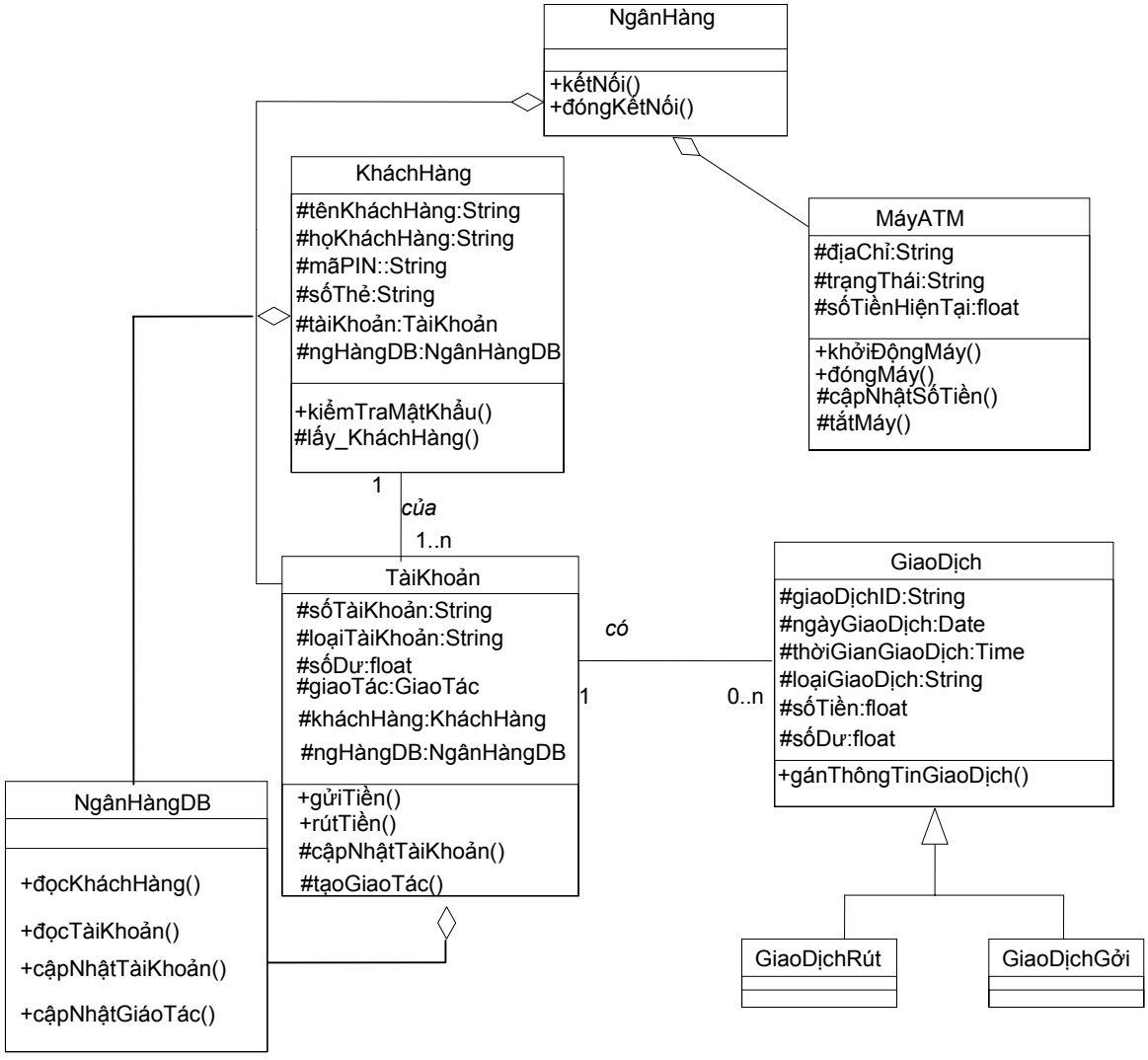
GiaoDịchDB::+cậpNhậtGiaoDịch()

Chúng ta gán mặc định cho các nghi thức ở tầng này là toàn cục (public), vì đa số đều được truy cập bởi những lớp khác, rồi trong quá trình thiết kế chi tiết cho từng method chúng ta sẽ xác định lại nghi thức này cho phù hợp nhằm đảm bảo tính bao bọc.

Bởi vì số lượng các method cho mỗi lớp này là ít (<= 2), cho nên chúng ta kết hợp tất cả lại thanh một lớp chung và gọi là NgânHàngDB.

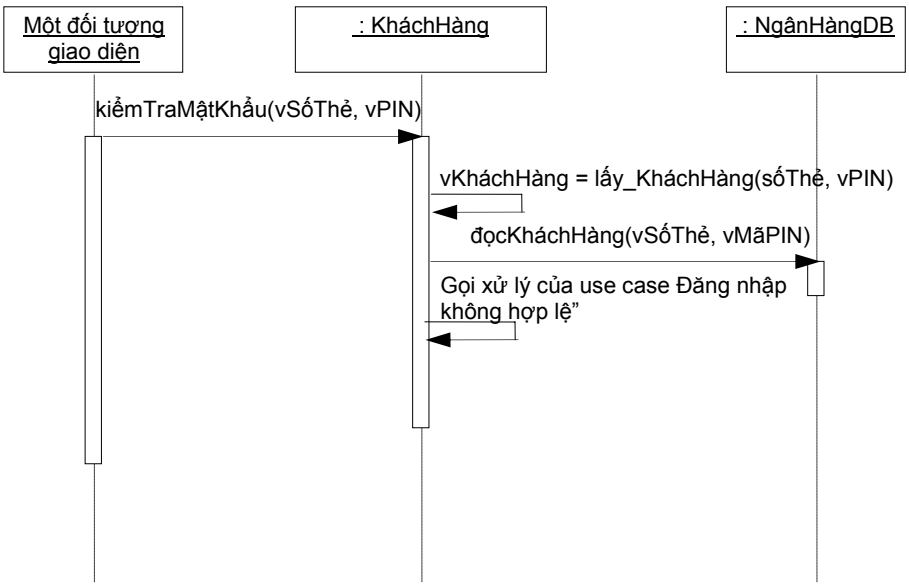


Sau đó chúng ta tạo liên kết aggregation từ lớp NgânHàngDB tới lần lượt các lớp: KháchHàng, TàiKhoản. Ứng với mỗi lớp vừa mới tạo liên kết tới lớp NgânHàngDB chúng ta thêm vào một thuộc tính tham chiếu đến lớp này.



Các method của lớp NgânHàngDB lần lượt sẽ được thiết kế chi tiết như sau:

NgânHàngDB::+đọcKháchHàng (vSốThẻ:String, vMãPIN:String): KháchHàng



Tinh chế chi tiết cho use case "Đăng Nhập" liên quan đến truy cập cơ sở dữ liệu theo sơ đồ trên, chúng ta đưa thêm vào một đối tượng NgânHàngDB và việc đọc dữ liệu thực sự về khách hàng sẽ do đối tượng này đảm nhận qua method đọcKháchHàng(). Sau đây là đoạn mã minh hoạ tương ứng:

```
KháchHàng::#lấy_KháchHàng(sốThẻ:String, mãPIN:String): KháchHàng
        vNgânHàngDB: NgânHàngDB
        return vNgânHàngDB.đọcKháchHàng (sốThẻ, mãPIN)
```

Ở đây chúng ta cần tạo ra một thể hiện của lớp truy cập NgânHàngDB và rồi gởi thông điệp tới đối tượng này để lấy thông tin về đối tượng khách hàng. Method đọcKháchHàng của NgânHàngDB sẽ thực sự đọc dữ liệu từ cơ sở dữ liệu. Có thể minh hoạ method này truy cập cơ sở dữ liệu quan hệ bằng SQL như sau:

```
NgânHàngDB::+đọcKháchHàng (vSốThẻ:String, vMãPIN:String): KháchHàng
        -- kết nối cơ dữ liệu ở đây
        return
        SELECT * FROM KháchHàng
        WHERE Số_Thẻ = vSốThẻ AND Mã_PIN = vMãPIN
```

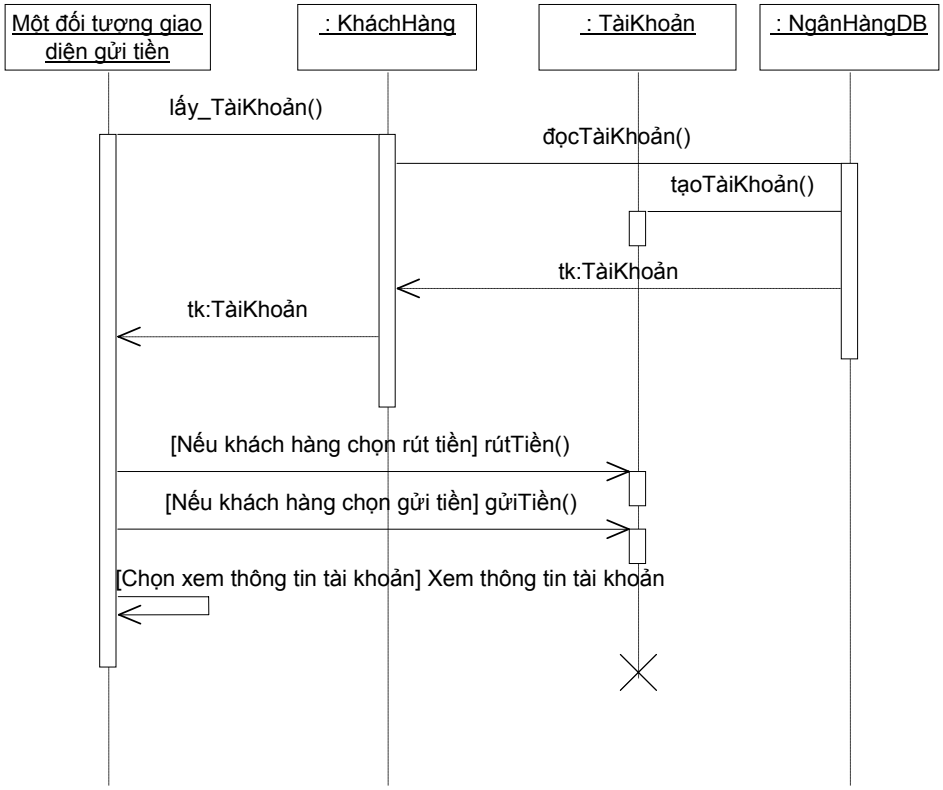
Giả sử rằng, câu truy vấn sẽ trả về dữ liệu khách hàng vào một đối tượng khách hàng và trả về cho method này. Tuỳ thuộc vào ngôn ngữ cài đặt cụ thể được chọn mà chúng ta thay đổi cho phù hợp, đoạn mã lệnh trên đây chỉ minh hoạ ý tưởng và cách viết độc lập ngôn ngữ.

NgânHàngDB::+đọcTàiKhoản(sốThẻ:String): TàiKhoản

Đây là một method đọc dữ liệu từ cơ sở dữ liệu để tạo một đối tượng persistent. Do đó, chúng ta có thể áp dụng một trong hai mẫu sơ đồ đọc đối tượng persistent ở phần trên.

Các use case liên quan đến method đọcTàiKhoản để tạo một đối tượng tài khoản phục vụ cho hoạt động của use case bao gồm: Giao dịch, Gửi tiền, Rút tiền, Truy vấn thông tin tài khoản. Sau đây chúng ta minh hoạ sơ đồ tuần tự thiết kế chi tiết các use case này qua sự tương tác giữa tầng nghiệp vụ và tầng truy cập dữ liệu, đặc biệt là tương tác nhằm tạo đối tượng tài khoản.

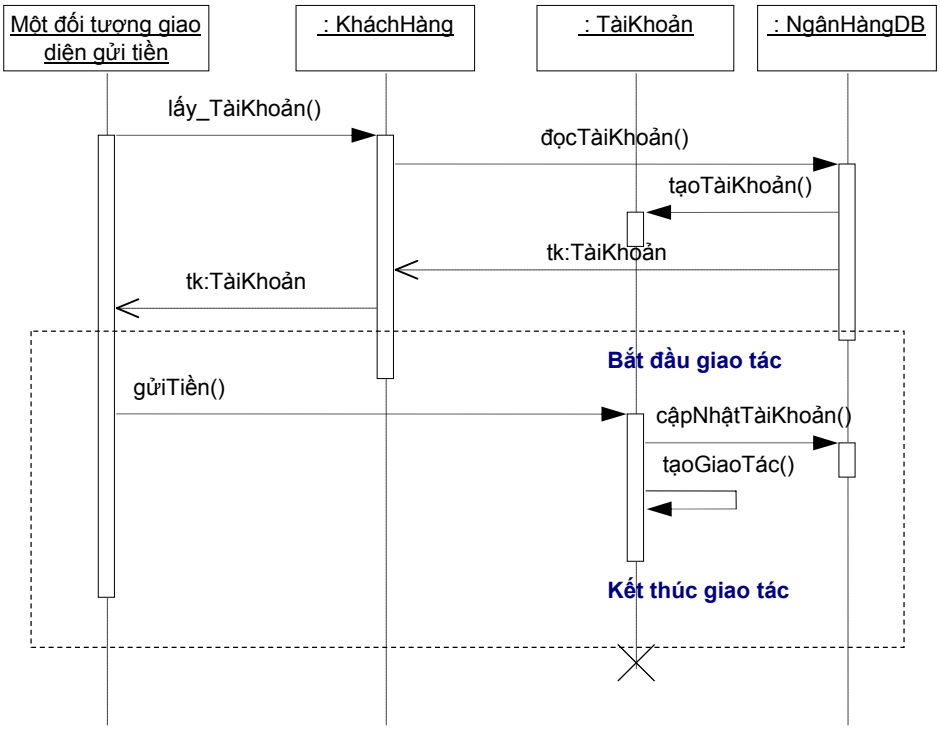
Use case Giao dịch



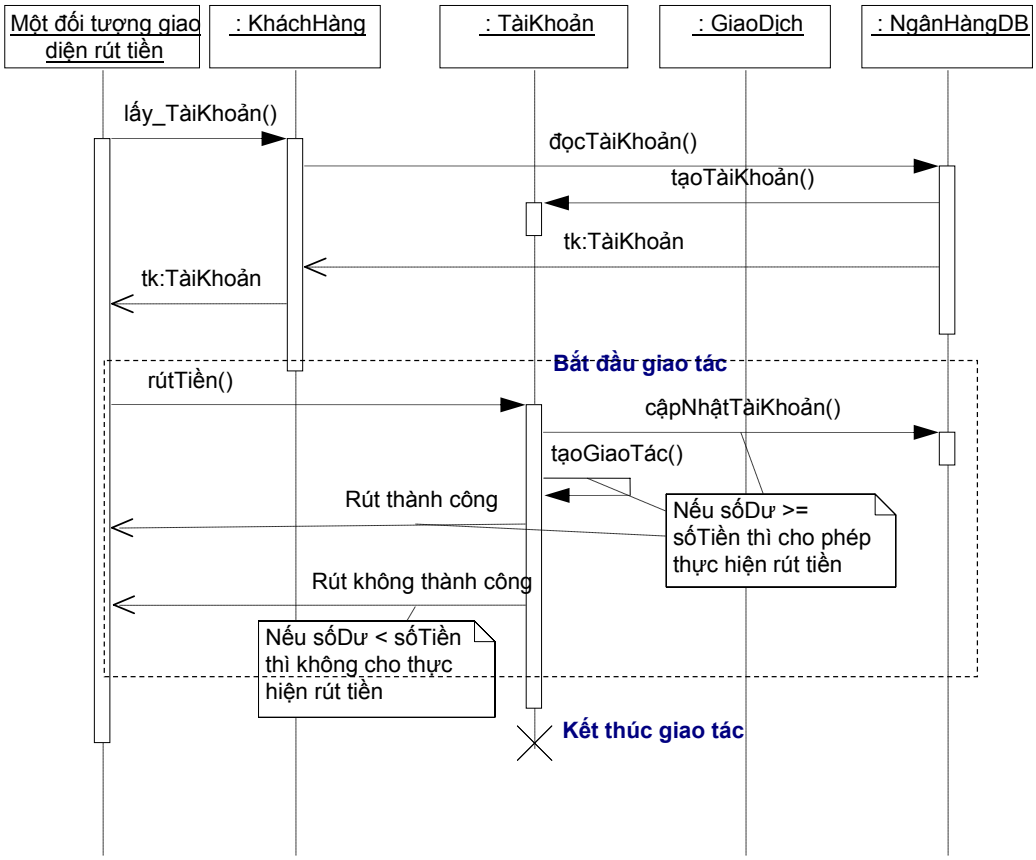
Sơ đồ tuần tự về use case Giao dịch được thiết kế: đầu tiên chúng ta giả lập một đối tượng tầng giao diện sẽ tương tác tới một đối tượng khách hàng (đã được tạo ra khi đăng nhập thành công) để tham khảo đến tài khoản của khách hàng bằng cách gởi thông điệp đến đối tượng truy cập dữ liệu là NgânHàngDB để thực hiện việc đọc dữ liệu từ cơ sở dữ liệu và tạo ra đối tượng tài khoản. Sau đó, đối tượng này sẽ được trả về cho đối tượng giao diện như một đối tượng để tham chiếu. Đối tượng giao diện sẽ được tinh chế trong phần sau của chương này.

Tuỳ theo loại giao dịch mà khách hàng chọn thì use cae này sẽ gọi thực hiện các use case mở rộng tương ứng. Các use case Rút tiền và Gửi tiền sẽ thực hiện dựa trên đối tượng tài khoản vừa đọc đọc được; use case Truy vấn thông tin tài khoản sẽ hiển thị thông tin về tài khoản vừa đọc được, do đó, nó được thực hiện bởi một đối tượng giao diện.

Use case Gửi tiền



Use case Rút tiền



Việc thực hiện gửi tiền hoặc rút tiền sẽ do đối tượng tài khoản đảm nhận, đối tượng này được tạo ra do thừa hưởng từ use case Giao dịch. Sau đó, được thiết kế bằng một giao tác gồm hai

hoạt động: cập nhật lại số dư tài khoản và tạo một giao tác mới. Việc tạo một giao tác mới sẽ được thực hiện bởi method tạoGiaoTác() sẽ được thiết kế trong phần tiếp theo sau.

Sau đây là đoạn mã minh hoạ cho các method:

```
KháchHàng::-lấy_TàiKhoản(sốThẻ:String): TàiKhoản
        vNgânHàngDB: NgânHàngDB
        return vNgânHàngDB.đọaTàiKhoản(sốThẻ)

NgânHàngDB::+đọcTàiKhoản(vSốThẻ:String): TàiKhoản
        -- kết nối cơ sở dữ liệu ở đây
        Return
        SELECT * FROM TàiKhoản
        WHERE Số_Thẻ = vSốThẻ

TàiKhoản::+gửiTiền(sốTiền:float)
        vNgânHàngDB: NgânHàngDB
        NgânHàngDB.cậpNhậtTàiKhoản (sốTàiKhoản, sốDư + sốTiền)
        tạoGiaoTác("gửi", sốTiền, sốDư + sốTiền)

TàiKhoản::+rútTiền(sốTiền:float): String
        vNgânHàngDB: NgânHàngDB
        if sốDư < sốTiền
                return "Số tiền rút vượt quá số dư tài khoản"
        else
                NgânHàngDB.cậpNhậtTàiKhoản (sốTàiKhoản, sốDư + sốTiền)
                tạoGiaoTác("gửi", sốTiền, sốDư + sốTiền)
                return ""
        endif
```

NgânHàngDB::+cậpNhậtTàiKhoản(vSốTàiKhoản:String, vSốDư:float)

Việc sử dụng method này được minh hoạ theo sơ đồ của use case Gửi tiền và Rút tiền. Sau đây là đoạn mã minh hoạ:

```
NgânHàngDB::+cậpNhậtTàiKhoản(vSốTàiKhoản:String, vSốDư:float)
        UPDATE TàiKhoản
        SET sốDư = vSốDư
        WHERE Số_TK = vSốTàiKhoản
```

NgânHàngDB::+cậpNhậtGiaoDịch()

Để thiết kế chi tiết method này chúng ta tiếp tục với các use case Gửi tiền và Rút tiền qua việc mô tả chi tiết method tạoGiaoTác() của đối tượng tài khoản qua sự tương tác giữa tầng nghiệp vụ và tầng truy cập dữ liệu.

TàiKhoản::#tạoGiaoTác(loạiGD:String, sốTiền:float, sốDư: float)

vNgânHàngDB: NgânHàngDB

vGiaoDịch = tạoMới(GiaoDịch)

vGiaoDịch.gánThôngTin(Date(today), Time(now), loạiGD, sốTiền, sốDư, sốTàiKhoản)

vNgânHàngDB.cậpNhậtGiaoDịch(sốTàiKhoản, loạiGD, sốTiền, sốDư)

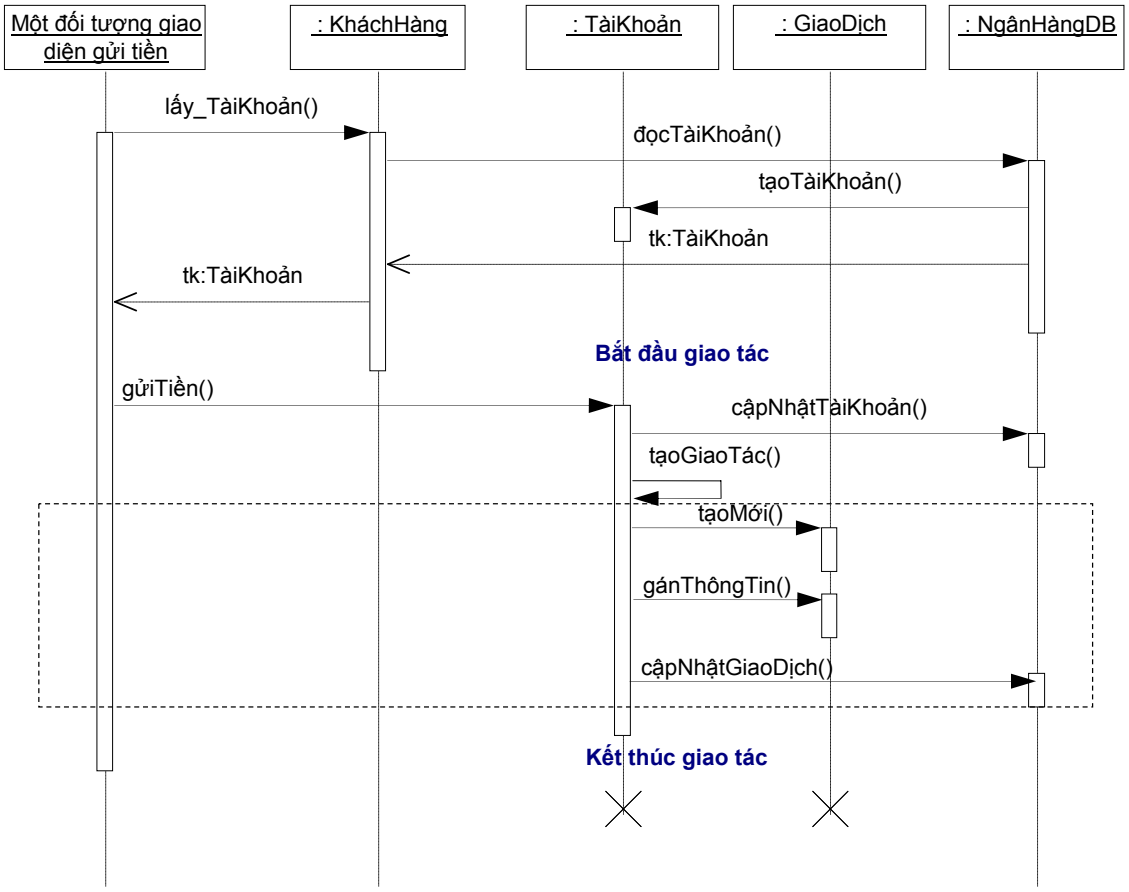
NgânHàngDB::+cậpNhậtGiaoDịch(vSốTàiKhoản:String, vLoạiGD:String, vSốDư:float)

--kết nối cơ sở dữ liệu ở đây

vGD_ID: String

vGD_ID = Tạo_GD_ID()

INSERT INTO GiaoDịch (GD_ID, Ngày_GD, Giờ_GD, Loại_GD, Số_Tiền, Số_Dư, Số_TK) VALUES (vGD_ID, Date(today), Time(now), vLoạiGD, vSốTiền, vSốDư, vSốTàiKhoản)



Xác định lớp tầng giao diện người dùng (user interface layer)

Bao gồm các đối tượng hệ thống mà người dùng sẽ tương tác và các đối tượng được dùng để quản lý hoặc điều khiển giao diện. Các class ở tầng này đảm nhận hai công việc chính:

- *Trả lời tương tác người dùng*: các đối tượng mức này phải được thiết kế để chuyển dịch những hành động của người dùng tới một tình huống xử lý phù hợp. Có thể là mở hoặc đóng một giao diện khác, hoặc gởi một thông điệp xuống tầng tác nghiệp hoặc khởi động một vài tiến trình ở tầng tác nghiệp

- *Hiển thị các đối tượng tác nghiệp*: tầng này phải trình bày một hình ảnh tốt nhất các đối tượng tác nghiệp tới người dùng trong một giao diện. Điều này có nghĩa là một hình ảnh trực quan thao tác được có thể bao gồm: các textbox, listbox, commbobox, page, form, menu, toolbar, ….

Tiến trình thiết kế tầng giao diện

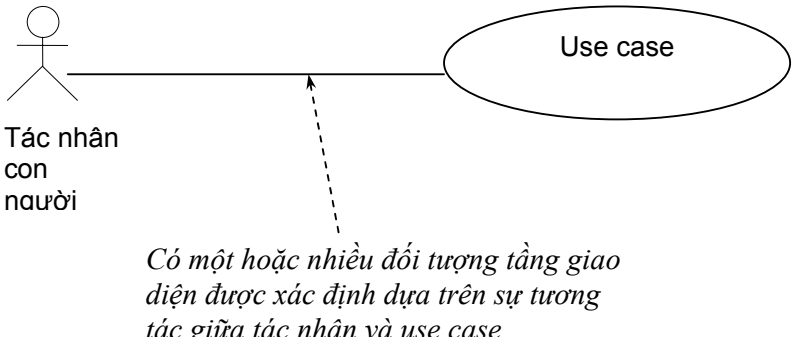
Một tiến trình thiết kế tầng giao diện bao gồm 4 bước sau:

1. Xác định các đối tượng ở tầng giao diện: đây là một hoạt động diễn ra luôn cả trong giai đoạn phân tích hệ thống. Mục tiêu chính là để xác định các lớp tương tác với các tác nhân con người thông qua việc phân tích các use case trong giai đoạn phân tích. Như đã mô tả trong các chương trước, mỗi use case bao gồm các tác nhân và các công việc mà tác nhân muốn hệ thống thực hiện. Do đó, nó mô tả một bức tranh đầy đủ về các yêu cầu giao diện của hệ thống. Trong giai đoạn này chúng ta cũng cần thiết tập trung vào cách thức cài đặt giao diện. Các sơ đồ tuần tự và hợp tác của use case cũng giúp chúng ta xác định chính xác yêu cầu về giao diện.
2. Xác định các lớp tầng giao diện
    a. Xác định các đối tượng tầng giao diện: xây dựng sơ đồ lớp mô tả các đối tượng giao diện
    b. Tạo bản mẫu giao diện (prototype): sau khi xác định mô hình thiết kế, chúng ta chuẩn bị cho một bản mẫu của giao diện. Thông thường các bản mẫu này đã được xác định trong những giai đoạn trước, nếu vậy thì chúng ta hãy cố gắng tích hợp với các đối tượng đã được xác định nhằm thống nhất các lớp giao diện trong sơ đồ lớp và các giao diện thiết kế.
    c. Xác định hành vi và thuộc tính cho các lớp giao diện: phân tích lại hành động của người dùng trong use case và tinh chế lại sơ đồ tuần tự để phát hiện các method cũng như xem xét lại các mối quan hệ của các đối tượng để xác định các thuộc tính (chủ yếu là các thuộc tính tham chiếu) cho lớp tầng giao diện.
3. Thử nghiệm giao diện.
4. Tinh chế và lặp lại các bước trên

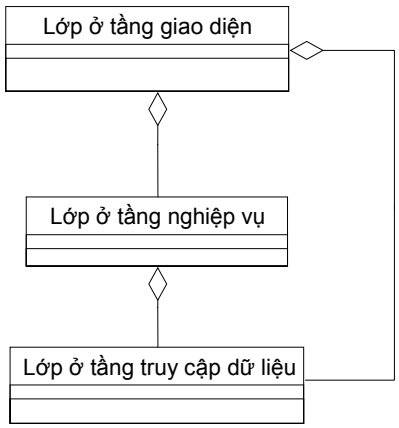
Xác định các lớp tầng giao diện qua việc phân tích use case

Trong một use case, đối tượng tầng giao diện xử lý tất cả trao đổi với tác nhân. Thực sự, các đối tượng này hoạt động như một vùng đệm giữa người dùng và phần còn lại của hệ thống là các đối tượng nghiệp vụ. Một đối tượng giao diện có thể tham gia vào nhiều use case. Bắt đầu từ use case, chúng ta xác định được các mục tiêu và nhiệm vụ của người dùng. Những người dùng khác nhau sẽ có những nhu cầu khác nhau trên giao diện, ví dụ, các người dùng chuyên nghiệp thì cần một giao diện có tính hiệu quả trong khi các người dùng bình thường thì cần có giao diện dễ sử dụng. Do đó, với các đối tượng người dùng khác nhau mà các giao diện được thiết kế sẽ khác nhau về mục tiêu, trách nhiệm, cách vận hành và hình thức trình bày. Việc xác định các lớp tầng giao diện gồm hai bước sau:

- Với mỗi lớp (tầng nghiệp vụ), nếu lớp đó có tương tác với một tác nhân con người trong một use case, chúng ta thức hiện như sau:
    - Xác định các đối tượng giao diện cho lớp đó, các trách nhiệm cũng như các yêu cầu của các đối tượng này. Việc này được thực hiện bằng cách phân tích lại sơ đồ tuần tự hoặc hợp tác của use case.



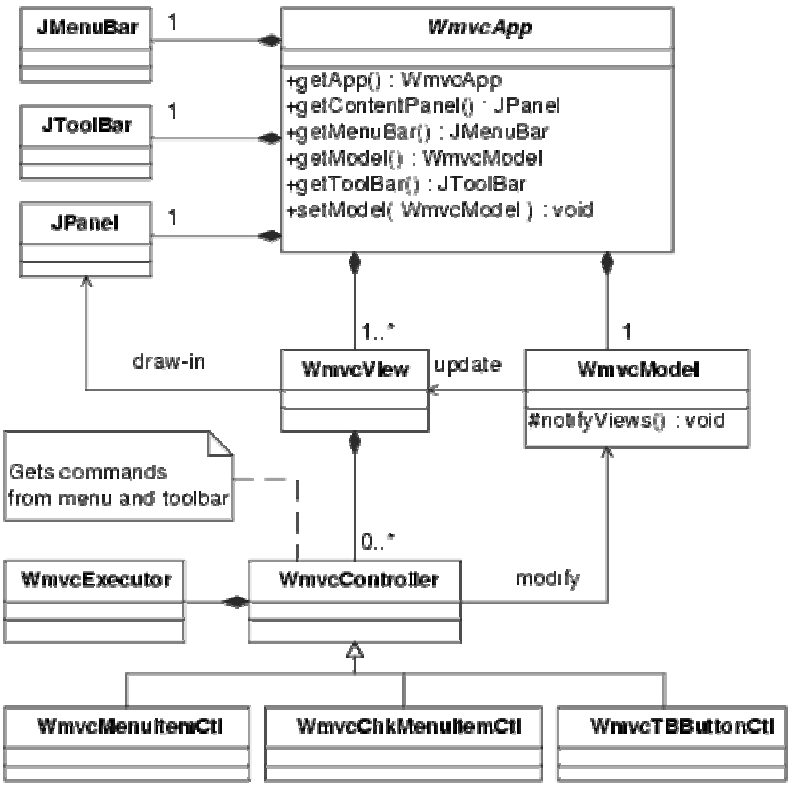
- Xác định sự liên kết giữa các đối tượng giao diện: các lớp giao diện cũng giống như các lớp khác, đều có mối quan hệ với các lớp ở tầng nghiệp vụ cùng tham gia trong một use case với nó. Các mối liên kết này cũng được xác định tương tự như của các lớp trong tầng nghiệp vụ. Sự liên kết này thường theo sơ đồ dưới đây.



- Lặp lại các bước trên và tinh chế

Ưu điểm của của việc sử dụng use case để xác định các đối tượng ở tầng giao diện là nó tập trung vào người dùng và đưa người dùng vào như một phần của kế hoạch thiết kế nhằm tìm ra một giao diện tốt nhất cho người dùng. Khi các đối tượng này đã được xác định, chúng ta phải xác định các thành phần cơ bản hoặc các đối tượng được dùng trong các công việc cũng như là các hành vi và các đặc điểm tạo ra sự khác biệt của mỗi loại đối tượng, bao gồm luôn cả mối quan hệ giữa các đối tượng và giữa đối tượng và người dùng.

Tuy nhiên, trong thiết kế giao diện khi chúng ta đã xác định môi trường phát triển thì chúng ta cũng nên tìm hiểu các khung mẫu (framework) cũng như các thư viện mà môi trường đó hỗ trợ để phát huy việc tái sử dụng trong thiết kế giao diện và tận dụng tối đa các hỗ trợ từ môi trường.



Một dạng khung mẫu Wmvc (Model – View - Controller) được hỗ trợ trong môi trường Java áp dụng cho các thiết kế giao diện ứng dụng với Java.

Xây dựng bản mẫu (prototype) cho giao diện

Việc xây dựng bản mẫu giao diện thường được thiết kế trong giai đoạn xác định yêu cầu và phân tích hệ thống. Công việc này được thực hiện sử dụng một công cụ gọi là CASE Tool hoặc các công cụ phát triển. Bao gồm ba bước sau:

- Tạo các đối tượng giao diện (như là các button, các vùng nhập liệu,…)
- Liên kết và gán các hành vi hoặc hành động thích hợp tới các đối tượng giao diện này và các sự kiện của nó.
- Thử nghiệm và lặp lại bước một

Thiết kế tầng giao diện cho hệ thống ATM

Xác định các đối tượng ở tầng giao diện, các yêu cầu và trách nhiệm của nó

Đối với mỗi lớp ở tầng nghiệp vụ: NgânHàng, MáyATM, KháchHàng, TàiKhoản, GiaoDịch, GiaoDịchGửi, GiaoDịchRút chúng ta xác định các lớp giao diện của nó bằng việc quan sát các sơ đồ tuần tự và hợp tác của các use case: Đăng nhập, Đăng nhập không hợp lệ, Giao dịch, Rút tiền, Gửi tiền, Truy vấn thông tin tài khoản, Khởi động hệ thống, Đóng hệ thống. Chúng ta xác định được các lớp tầng giao diện sau đây:

KháchHàngGD: biểu diễn giao diện tương tác giữa khách hàng và use case Đăng nhập, Đăng nhập không hợp lệ

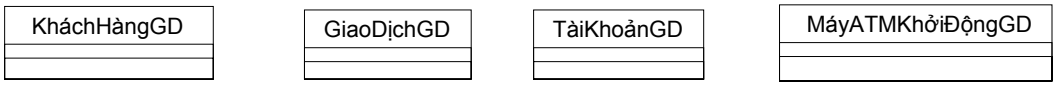
GiaoDịchRútGD: biểu diễn tương tác giữa khách hàng và use case Rút tiền

GiaoDịchGửiGD: biểu diễn tương tác giữa khách hàng và use case Gửi tiền

Tài khoảnGD: biểu diễn tương tác giữa khách hàng và use case Truy vấn thông tin tài khoản

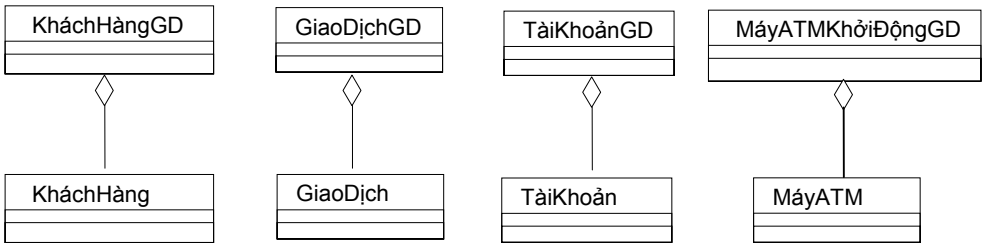
MáyATMKhởiĐộngGD: biểu diễn tương tác giữa nhân viên vận hành và use case Khởi động hệ thống

Các đối tượng giao diện GiaoDịchGửiGD, GiaoDịchRútGD có một hình thức trình bày chung của giao dịch và chỉ thay đổi loại giao dịch. Do đó, chúng ta gom lại thành một lớp và gọi là: GiaoDịchGD.



Xác định sự liên kết các lớp của tầng giao diện với các lớp của tầng nghiệp vụ

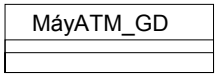
Mối kết hợp được xác lập là mối kết hợp dạng aggregation như sau: với mỗi lớp giao diện chúng ta tạo liên kết aggregation tới lớp tương ứng trong tầng nghiệp vụ. Mối liên kết này cho biết trong các thể hiện của lớp tầng giao diện sẽ sử dụng đối tượng ở tầng nghiệp vụ như là một thành phần để thực hiện gởi thông điệp.



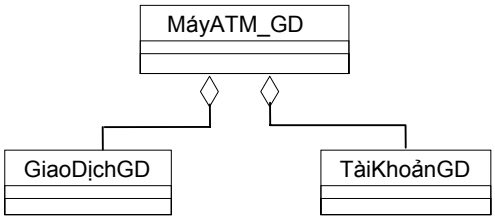
Xác định các đối tượng giao diện điều khiển như là: toolbar, menu, form điều khiển,....

Ngoài các lớp được xác định dựa vào use case, chúng ta xác định thêm các đối tượng giao diện có nhiệm vụ điều khiển chính, giao diện chính, thực đơn, tool bar,…

Với hệ thống ATM chúng ta có thêm một đối tượng giao diện điều khiển chính hoạt động giao diện của máy ATM gọi là MáyATM_GD.



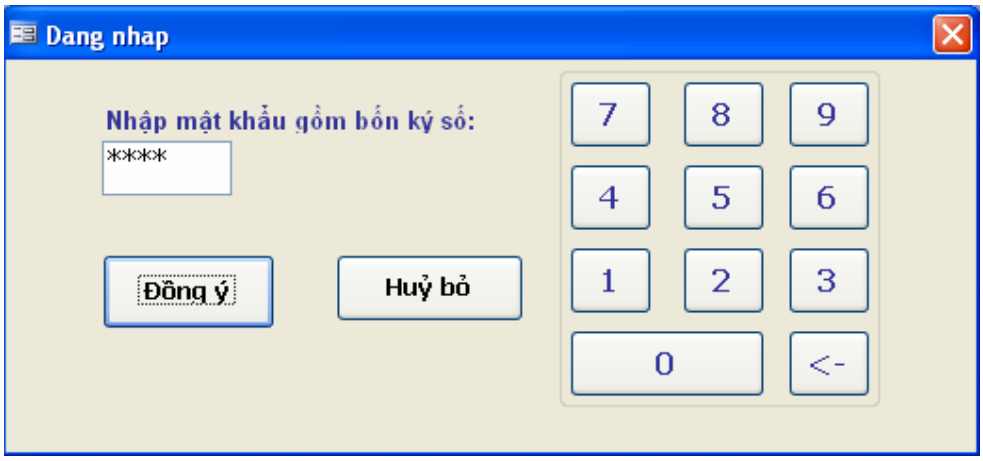
Vì đối tượng của lớp MáyATM_GD sẽ gởi thông điệp đến tất cả các đối tượng giao diện rút, gửi và truy vấn thông tin nhằm điều khiển các giao diện này. Do đó, trước khi gởi thông điệp thì đối tượng lớp MáyATM_GD sẽ phải tạo ra các đối tượng kia như là một thành phần của nó. Chính vì vậy mà chúng ta sẽ xác lập mối kết hợp aggregation từ lớp MáyATM_GD đến các lớp: GiaoDichGD, TaiKhoanGD



Thiết kế giao diện mẫu (prototype)

Chúng ta lần lượt thiết kế bản mẫu cho các đối tượng giao diện:

KháchHàngGD



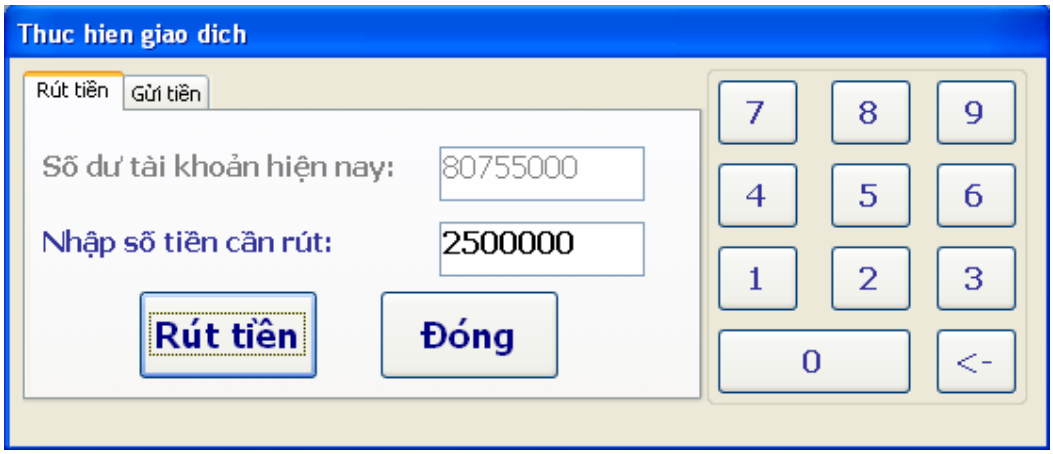
Giao diện đăng nhập tài khoản cho phép khách hàng chọn bốn ký số bằng cách nhấn vào các nút số được thiết kế ngay trên màn hình. Khách hàng có thể chọn đồng ý hoặc huỷ bỏ đăng nhập.

MáyATM_GD

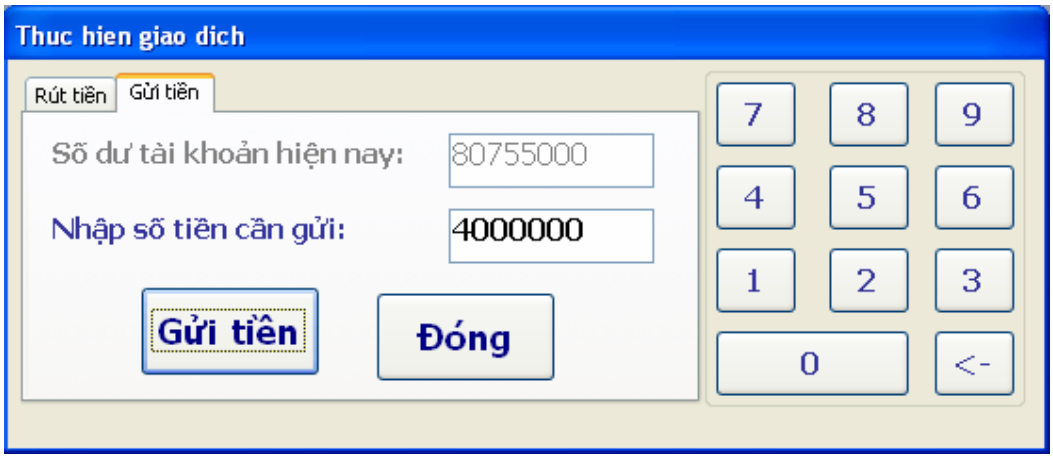
Sau khi đăng nhập thành công, giao diện chính điều khiển của ATM sẽ xuất hiện. Giao diện này sẽ hiển thị các nút cho phép người dùng chọn để thực hiện các dịch vụ tương ứng như là: gửi tiền, rút tiền, và xem thông tin về tài khoản.

GiaoDịchGD

Giao diện rút tiền

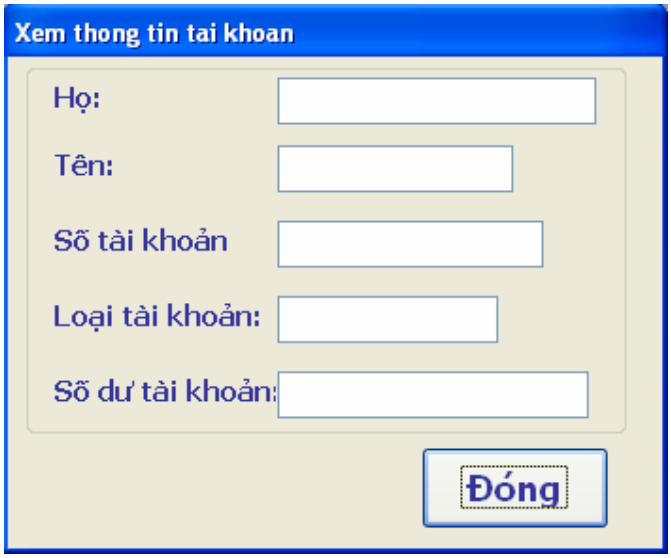


Giao diện gửi tiền



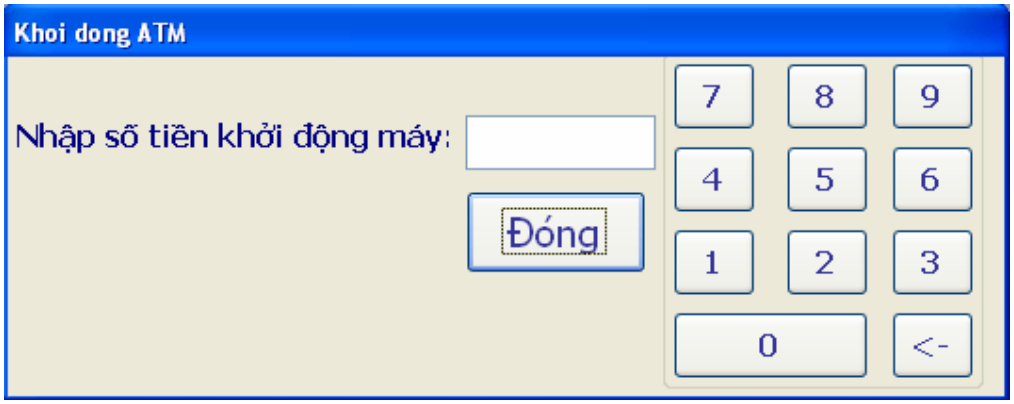
Giao diện thực hiện giao dịch được thiết kế gồm hai thẻ một thẻ cho giao dịch rút và một thẻ cho giao dịch gửi. Tập các số giả lập bàn phím cho phép khách hàng nhập số liệu trực tiếp từ màn hình. Giao dịch sẽ được thực hiện khi khách hàng chọn nút rút tiền hoặc gửi tiền. Khách hàng có thể chọn huỷ bỏ giao dịch bằng cách nhấn nút đóng.

TàiKhoảnGD



Giao diện này cho phép hiển thị thông tin về tài khoản của khách hàng bao gồm: họ, tên, số tài khoản, loại tài khoản, và số dư tài khoản

MáyATMKhởiĐộngGD



Sau khi máy đã được bật lên, giao diện này sẽ hiển thị cho phép nhân viên vận hành nhập vào số tiền khởi động cho máy.

Xác định các method

Mục tiêu của các đối tượng giao diện là cho phép người dùng thực hiện thao tác với hệ thống nhằm trợ giúp xử lý công việc nghiệp vụ như là: nhấn một nút, nhập vào một ký tự,…. Các thao tác này sẽ được chuyển tới các đối tượng tầng nghiệp vụ để xử lý. Khi hoàn thành xử lý, giao diện sẽ cập nhật lại thông tin nhằm hiển thị các thông tin mới hoặc mở một giao diện mới,…. Xác định hành vi cho đối tượng giao diện bằng cách xác định các sự kiện của hệ thống phải đáp ứng tương ứng tới các thao tác người dùng trên giao diện và các hành động khi sự kiện xảy ra. Một hành động được xem như một hành vi và được hình thành bởi sự kết hợp một đối tượng với một thông điệp gởi tới đối tượng đó. Xem xét lại từng đối tượng giao diện theo từng use use case của hệ thống ATM chúng ta lần lượt xác định các method cho các lớp giao diện .[2]

KháchHàngGD - Use case Đăng nhập

Khi khách hàng đưa thẻ ATM vào máy các sự kiện và hành động:

- Đưa thẻ vào máy -> hiển thị giao diện đăng nhập (KháchHàngGD)
- Khách hàng chọn đồng ý -> kiểm tra mật khẩu (KháchHàng)
  -> hiển thị giao diện điều khiển chính (MáyATM_GD)
  -> thông báo nếu đăng nhập không thành công (KháchHàngGD)
  -> đóng giao diện đăng nhập (KháchHàngGD)
- Khách hàng chọn huỷ bỏ -> đóng giao diện đăng nhập (KháchHàngGD)

Chúng ta xác định được các method:

KháchHàngGD::+hiểnThị()

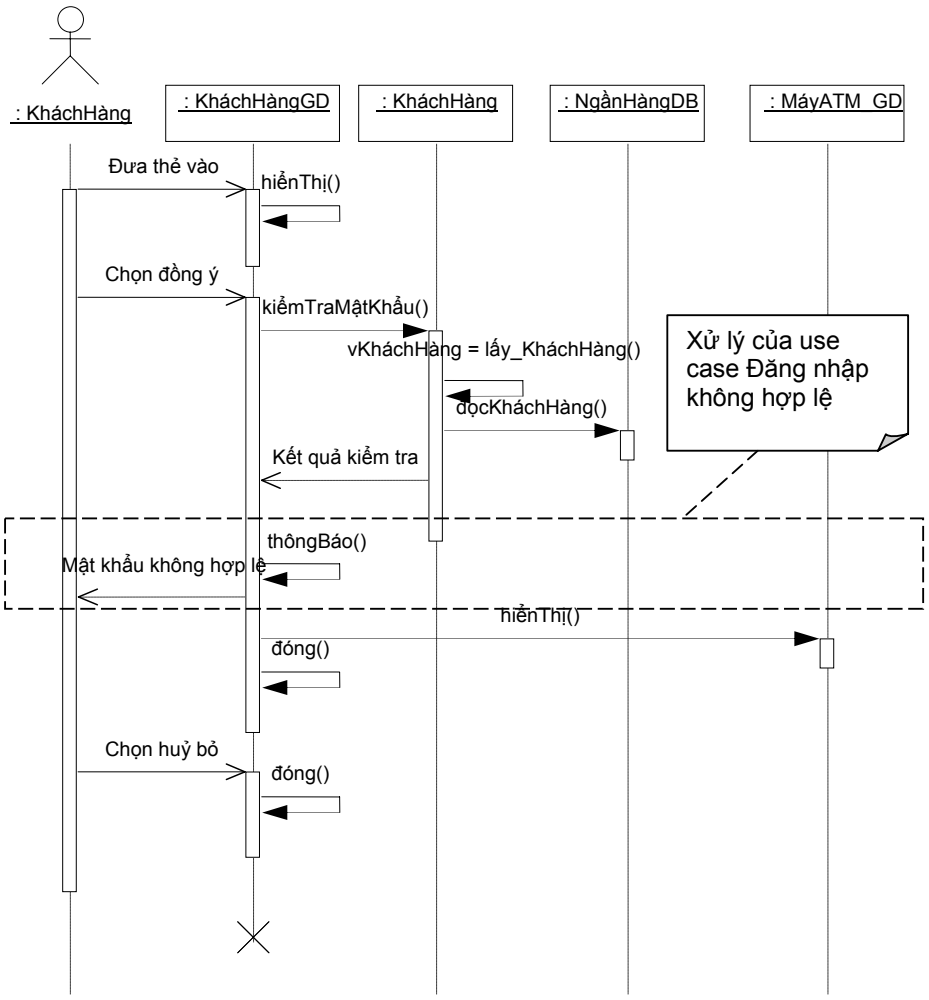
KháchHàngGD::-thôngBáo(thôngBáo:String)

KháchHàngGD::+đóng()

MáyATM_GD::+hiểnThị()

Sơ đồ tuần tự tinh chế tầng giao diện cho use case Đăng nhập

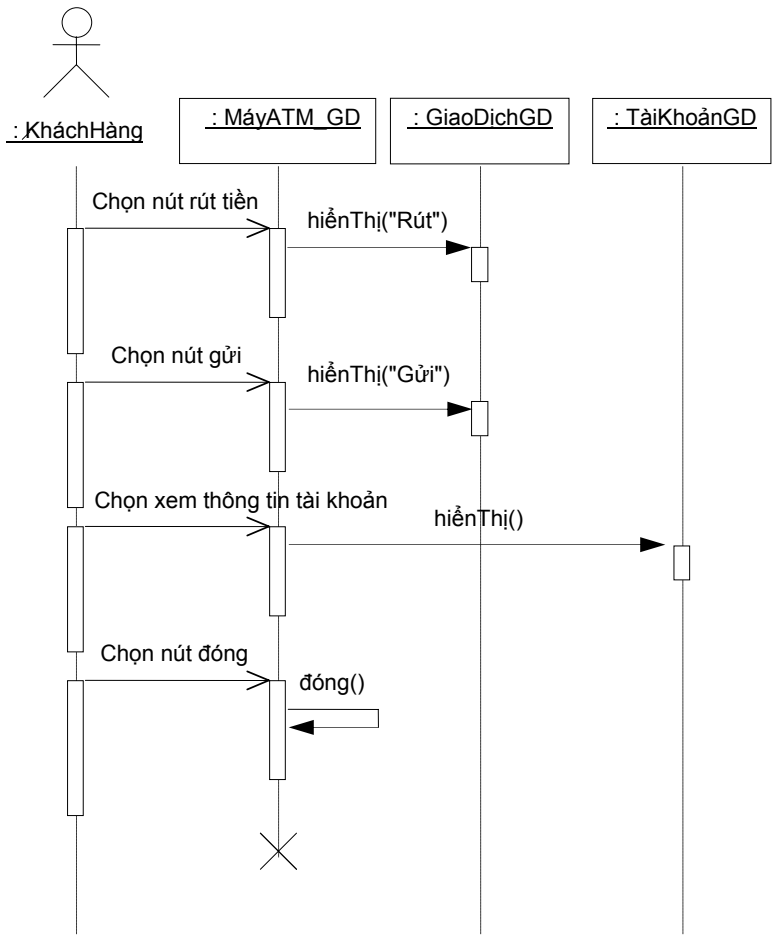
[2] Trong quá trình xác định các method cho tầng giao diện có thể phát sinh thêm các method ở các tầng nghiệp vụ hoặc truy cập dữ liệu và chúng ta phải quay lại để thiết kế thêm các method mới phát sinh này



MáyATM_GD

Khi giao diện chính của máy được hiển thị các tương tác của khách hàng làm phát sinh các sự kiện và các hành động:

- Chọn nút rút tiền -> hiển thị giao diện rút tiền (GiaoDịchGD)
- Chọn nút gửi tiền -> hiển thị giao diện gửi tiền (GiaoDịchGD)
- Chọn nút xem tài khoản -> hiển thị giao diện xem thông tin tài khoản (TàiKhoảnGD)
- Chọn nút thoát -> đóng giao diện chính (MáyATM_GD)



Chúng ta xác định được các method

GiaoDịchGD::+hiểnThị(loạiGD:String)

TàiKhoản::+hiểnThị()

MáyATM_GD::+đóng()

GiaoDịchGD - Use case Rút tiền

Khi khách hàng chọn dịch vụ rút tiền từ giao diện chính các sự kiện và hành động:

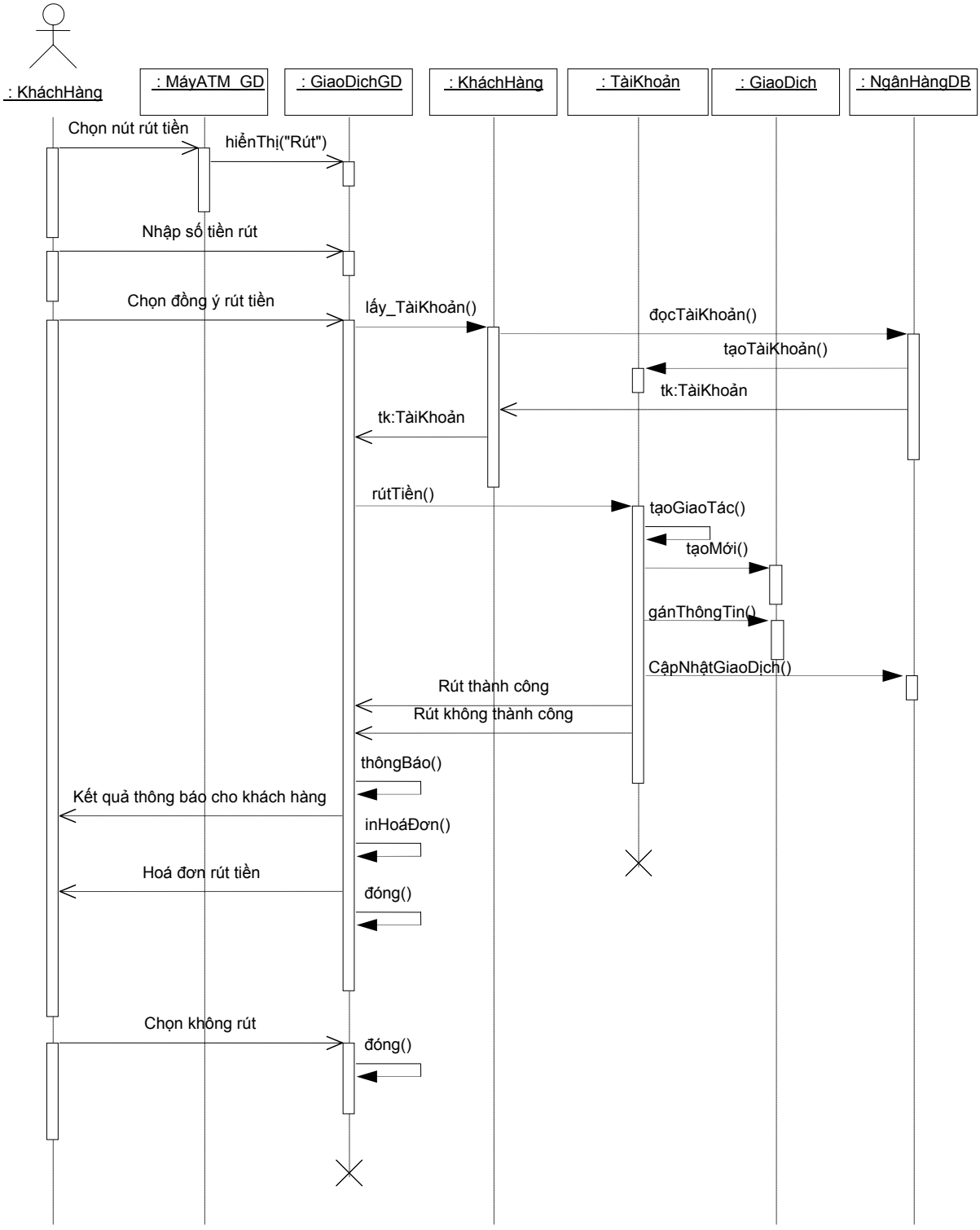
- Chọn rút tiền -> hiển thị giao diện rút tiền (GiaoDịchGD)
- Khách hàng chọn rút tiền -> thực hiện rút tiền (TàiKhoản)
  -> thông báo kết quả (GiaoDịchGD)
  -> in hoá đơn rút (GiaoDịchGD)
  -> đóng giao diện rút tiền (GiaoDịchGD)
- Khách hàng chọn đóng -> đóng giao diện rút tiền (GiaoDịchGD)

Chúng ta xác định được các method:

GiaoDịchGD::-thôngBáo(thôngBáo:String)

GiaoDịchGD::-inHoáĐơn()

GiaoDịchGD::+đóng()



GiaoDịchGD - Use case Gửi tiền

Khi khách hàng chọn dịch vụ gửi tiền từ giao diện chính các sự kiện và hành động:

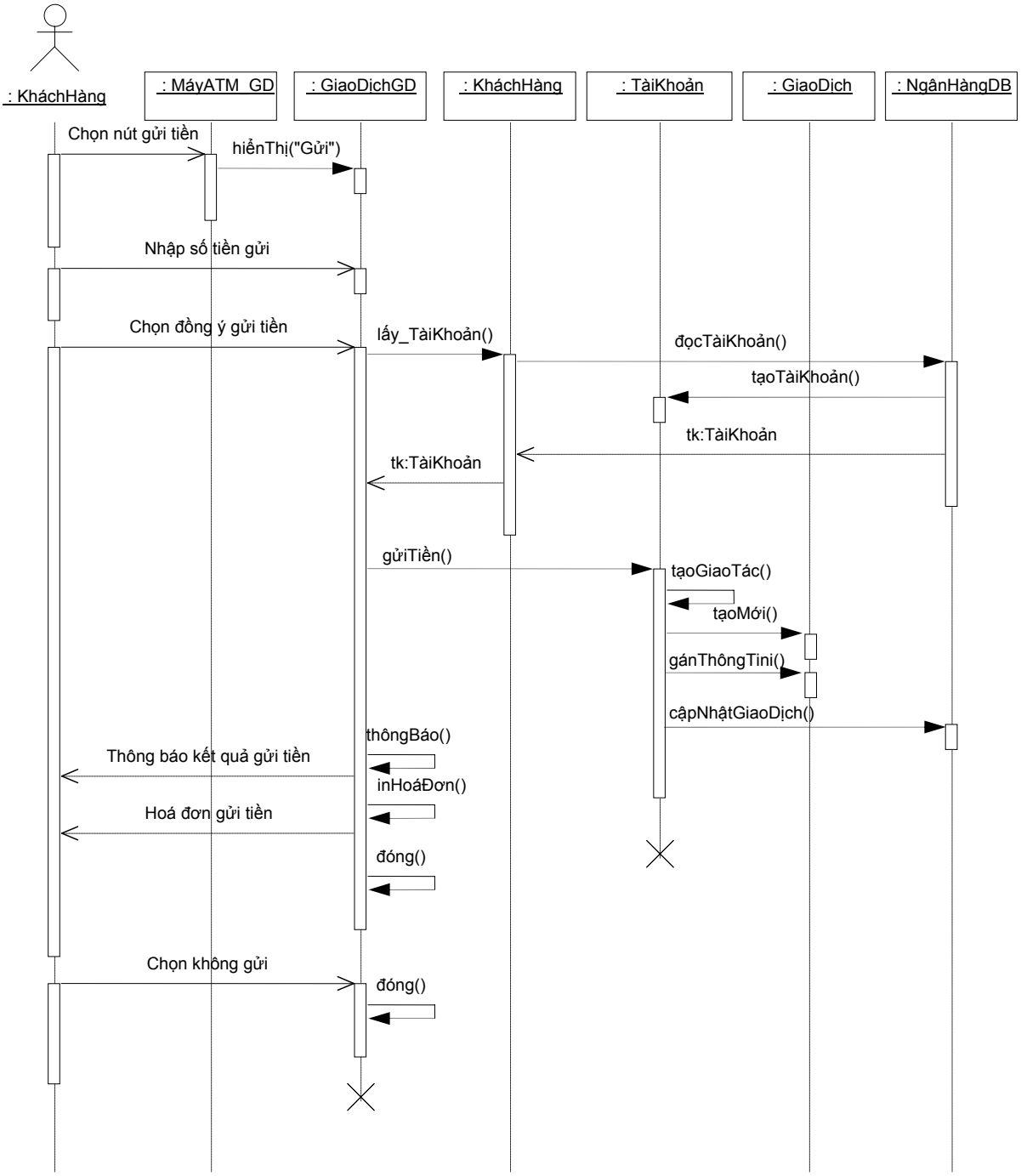
- Chọn gửi tiền -> hiển thị giao diện gửi tiền (GiaoDịchGD)
- Khách hàng chọn gửi tiền -> thực hiện gửi tiền (TàiKhoản)
  -> thông báo kết quả (GiaoDịchGD)

-> in hoá đơn gửi (GiaoDịchGD)

-> đóng giao diện gửi tiền (GiaoDịchGD)

- Khách hàng chọn đóng -> đóng giao diện gửi tiền (GiaoDịchGD)

Các method xác định trong use case này giống như của use case Rút tiền



Trong sơ đồ tuần tự trên cho rút tiền và gửi tiền, chúng ta quan sát ba đối tượng: Kháchhàng (tác nhân), và hai đối tượng giao diện MáyATM_GD, GiaoDịchGD. Bắt đầu các dòng từ tác nhân khách hàng mô tả các thao tác khách hàng trên giao diện và trả lời của hệ thống từ giao diện. Từ các dòng này, các đối tượng giao diện sẽ tương tác với các đối tượng ở tầng nghiệp vụ bằng các thông điệp nhằm thực hiện tác vụ của hệ thống.

TàiKhoảnGD - Use case Truy vấn thông tin tài khoản[3]

Khi khách hàng chọn truy vấn thông tin tài khoản từ giao diện chính các sự kiện và hành động:

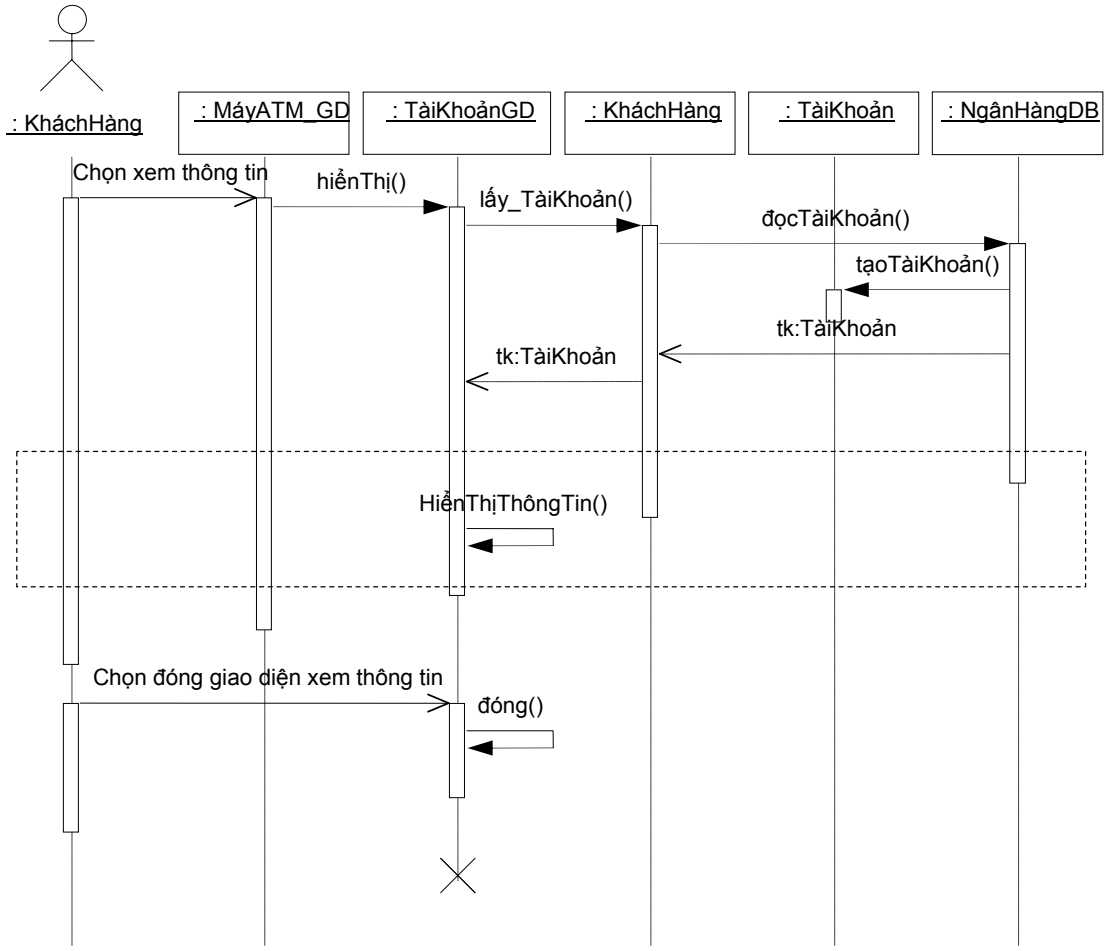
- Chọn xem thông tin tài khoản -> hiển thị giao diện truy vấn (TàiKhoảnGD)
  -> đọc thông tin tài khoản (KháchHàng)
  -> hiển thị thông tin tài khoản (TàiKhoảnGD)
- Khách hàng chọn đóng -> đóng giao diện truy vấn (TàiKhoảnGD)

Chúng ta xác định được các method:

TàiKhoảnGD::+hiểnThị()

TàiKhoảnGD::-hiểnThịThôngTin(tk:TaiKhoan)

TàiKhoảnGD::+đóng()



MáyATMKhởiĐộngGD - Use case Khởi động hệ thống

Khi máy được bật công tắc khởi động các sự kiện và hành động:

- Khởi động máy hoàn thành -> hiện thị giao diện khởi động máy (MáyATMKhởiĐộngGD)

[3] Chú ý rằng: use case "Truy vấn thông tin tài khoản" là một use case mở rộng của use case "Giao dịch", do vậy trong thiết kế use case "Truy vấn thông tin tài khoản" chúng ta thừa kế các hoạt động của use case "Giao dịch".

- Nhân viên chọn đóng -> cập nhật số tiền cho hiện hành cho máy (MáyATM)
  -> thực hiện kết nối tới mạng ngân hàng (NgânHàng)
  -> đóng giao diện khởi động (MáyATMKhởiĐộngGD)

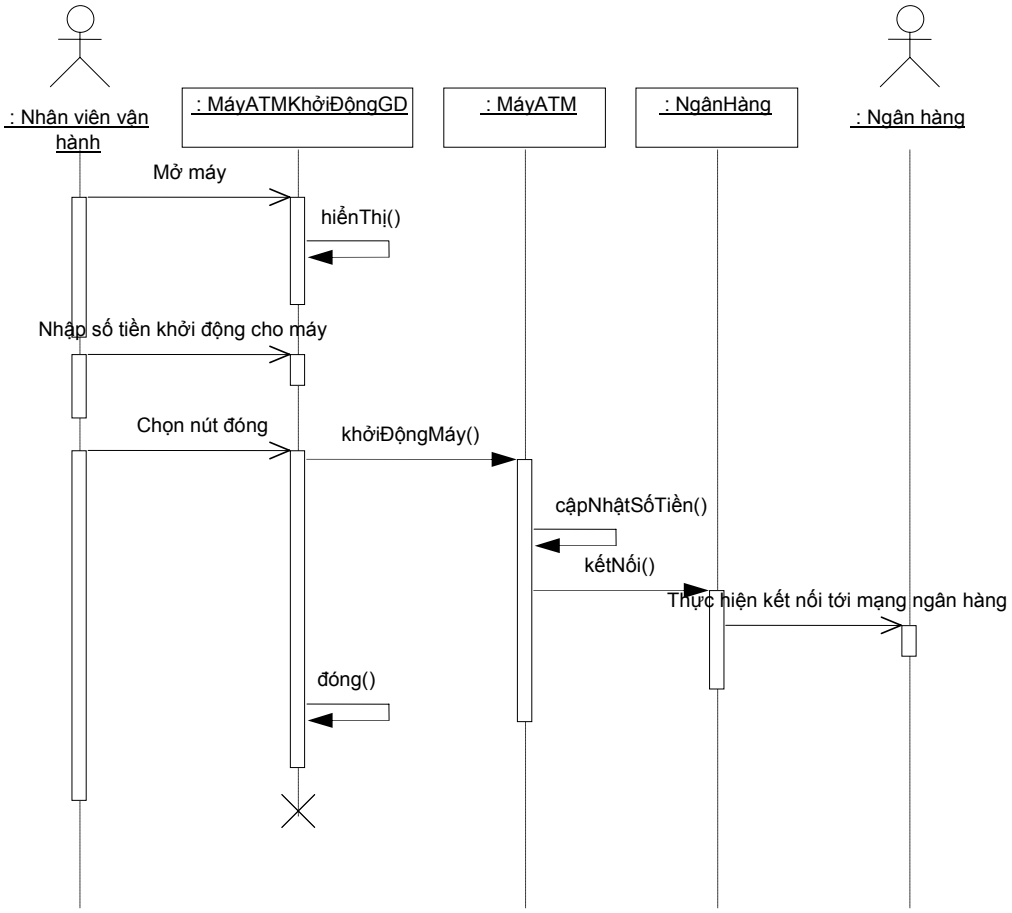
Chúng ta xác định được các method:

MáyATMKhởiĐộngGD::+hiểnThị()

MáyATM::+cậpNhậtSốTiền(sốTiền:float)

NgânHàng::+kếtNối()

MáyATMKhởiĐộngGD::+đóng()



Use case Đóng máy

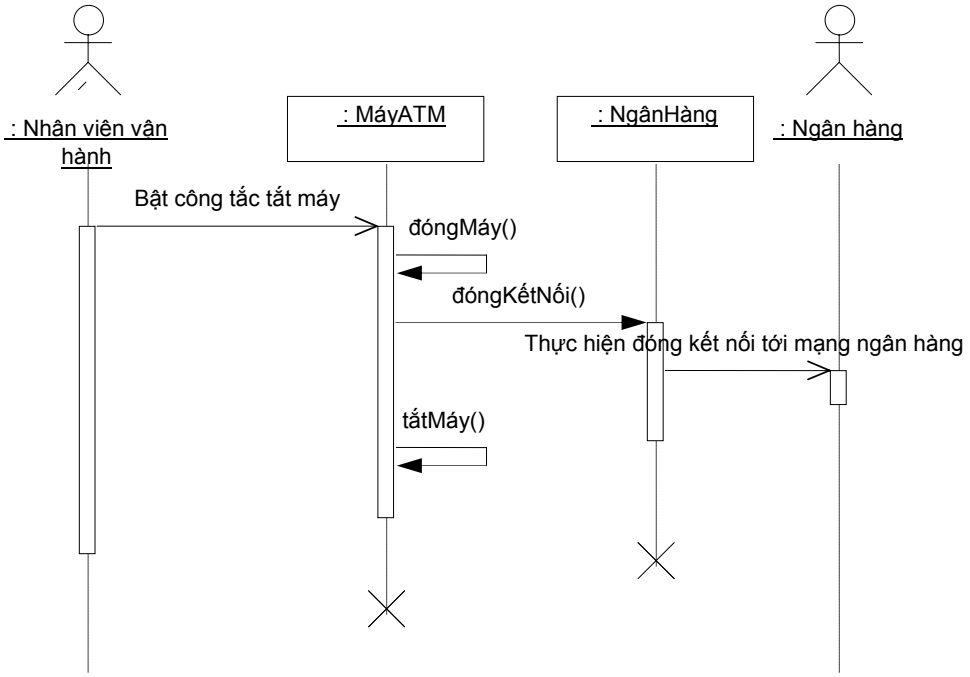
Khi nhân viên bật công tắc tắt máy, các sự kiện và hành động:

- Trước khi tắt máy -> thực hiện đóng kết nối tới mạng ngân hàng (NgânHàng)
  -> tắt máy (MáyATM)

Chúng ta xác định được các method

NgânHàng::+đóngKếtNối()

MáyATM::-tắtMáy()

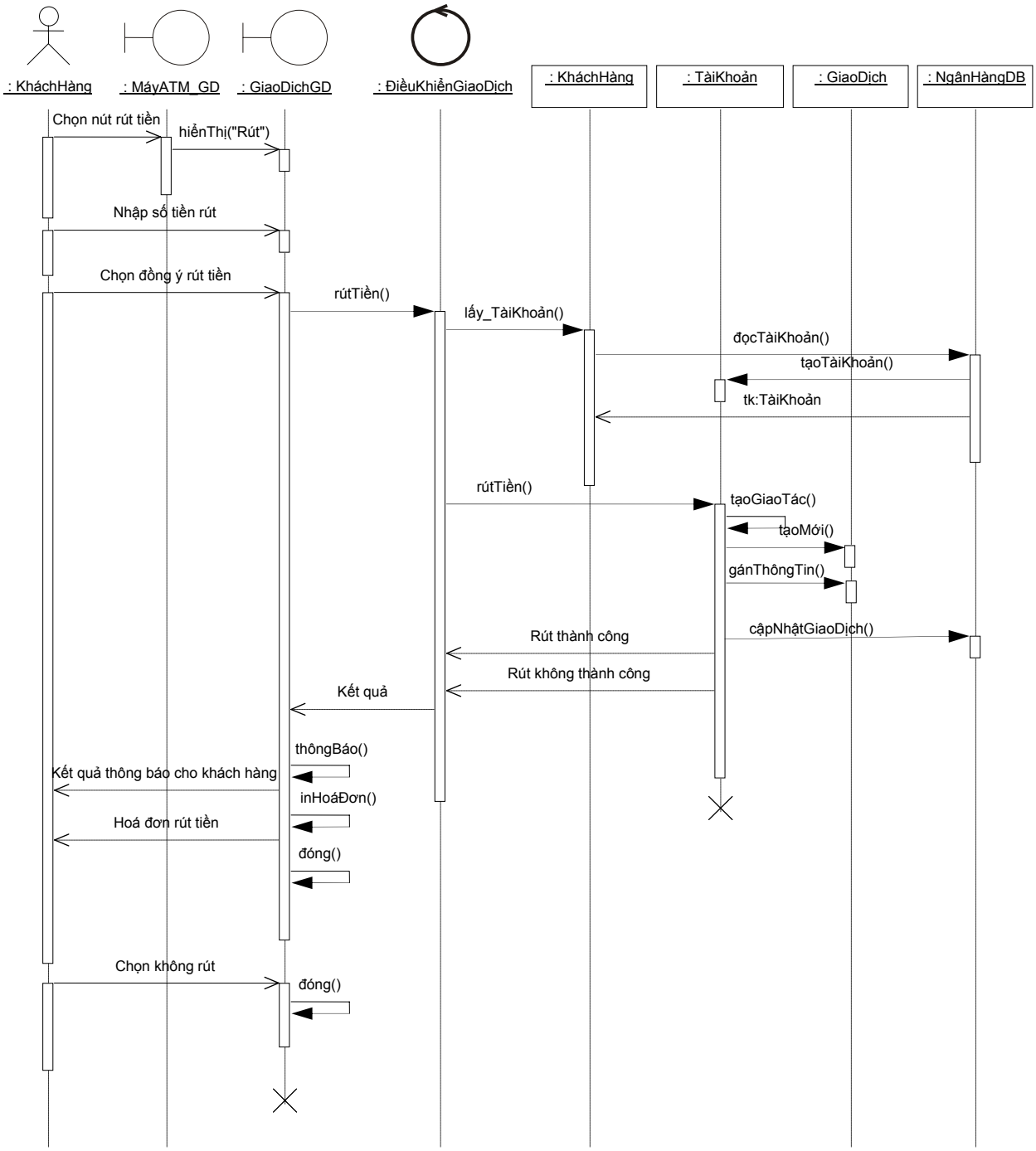


Một giải pháp khác nhằm quản lý việc điều khiển các lớp nghiệp vụ để đáp ứng cho các sự kiện ở tầng giao diện là dùng đối tượng điều khiển. Các thông điệp từ tầng giao diện sẽ được tiếp nhận bởi đối tượng điều khiển và đối tượng này sẽ điều khiển các hoạt động của các đối tượng nghiệp vụ nhằm thực thi cho thông điệp đó. Như vậy, đối tượng điều khiển cũng có thể hiểu như là một đối tượng bao bọc tầng nghiệp vụ từ các đối tượng tầng giao diện.

Một cách tổng quát, một đối tượng điều khiển có thể đảm nhận nhiều use case hoặc một use case có thể có nhiều đối tượng điều khiển. Tuy nhiên, không phải tất cả use case đều phải có đối tượng điều khiển, bởi vì ý nghĩa của đối tượng điều khiển là tạo ra sự phối hợp, do đó, trong trường hợp use case chỉ có một lớp thì ý nghĩa phối hợp không còn.

Trong UML lớp điều khiển là một stereotype và có ký hiệu như sau:

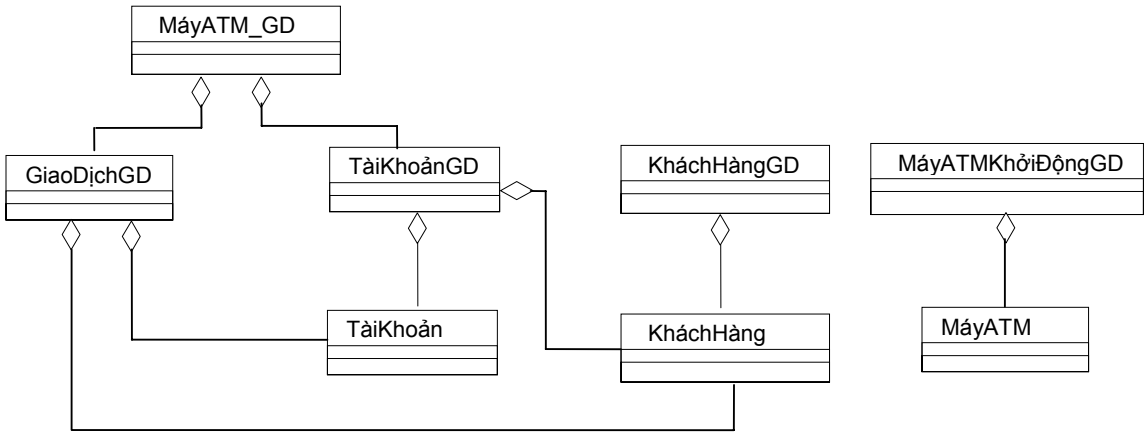
Sơ đồ tuần tự cho use case Rút tiền có đối tượng điều khiển:



Chúng ta có thể chọn một kiểu stereotype để biểu diễn phù hợp với một đối tượng tầng giao diện như là: boundary, form, interface, page, webpage,… Ví dụ, trong sơ đồ use case Rút tiền trên chúng ta chọn stereotype cho các lớp giao diện là <<boundary>>. Quan sát sơ đồ trên chúng ta thấy các đối tượng giao diện sẽ tượng tác (rút tiền) với các đối tượng nghiệp vụ qua một đối tượng điều khiển *ĐiềuKhiểnGiaoDịch*, và đối tượng này sẽ điều phối các hoạt động của các đối tượng nghiệp vụ (KháchHàng, TàiKhoản, GiaoDịch) để thực hiện việc rút tiền thay vì trong các sơ đồ trước đó, việc điều phối này do đối tượng giao diện đảm nhận.

Xác định thuộc tính các lớp tầng giao diện

Các thuộc tính được xác định cho các lớp tầng giao diện chủ yếu là các thuộc tính mô tả tham chiếu. Một lần nữa, dựa vào các sơ đồ tuần tự chúng ta tinh chế lại mối kết hợp giữa các lớp ở tầng giao diện và các lớp tầng nghiệp vụ:



Các thuộc tính tham chiếu của các lớp lần lượt là:

Lớp MáyATM_GD

-giaoDịchGD:GiaoDịchGD

-tàiKhoảnGD:TàiKhoảnGD

Lớp KháchHàngGD

-khácHàng:KháchHàng

Lớp GiaoDịchGD

-tàiKhoản:TàiKhoản

-khácHàng:KháchHàng

Lớp TàiKhoảnGD

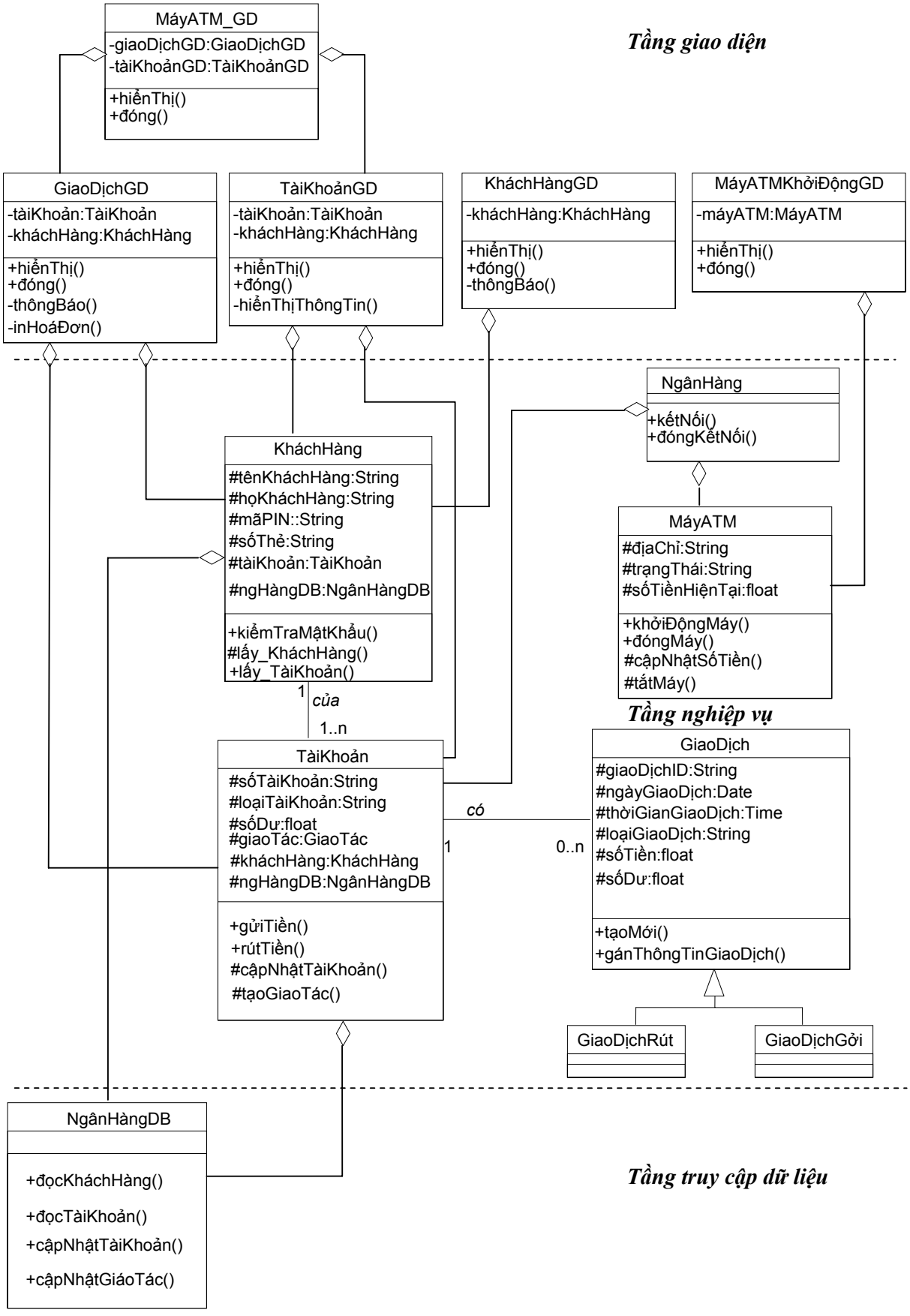
-tàiKhoản:TàiKhoản

-khácHàng:KháchHàng

Lớp MáyATMKhởiĐộngGD

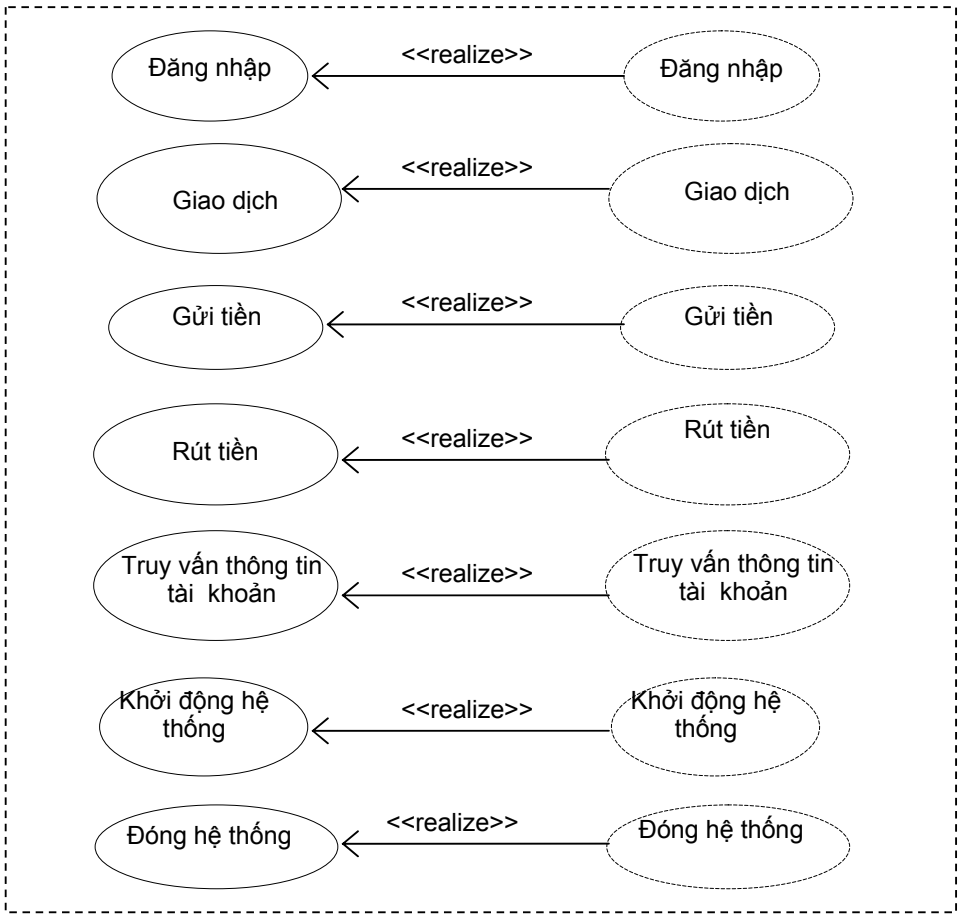
-máyATM:MáyATM

Sơ đồ lớp đầy đủ ba tầng của hệ thống ATM:



Mô tả hiện thực hoá use case

Việc mô tả hiện thực hoá một use case chính là mô hình hoá nội dung hoạt động bên trong của một use case nhằm cung cấp một chức năng hệ thống tới một tác nhân. Việc mô hình hoá này bao gồm: các đối tượng tham gia trong use case và cách thức các đối tượng này hợp tác hoạt động và trao đổi thông điệp với nhau. Chúng ta dùng sơ đồ lớp để mô tả các đối tượng cùng tham dự trong một use case hiện thực hoá và sơ đồ tương tác để biểu diễn sự phối hợp hoạt động và trao đổi thông điệp giữa các đối tượng này.

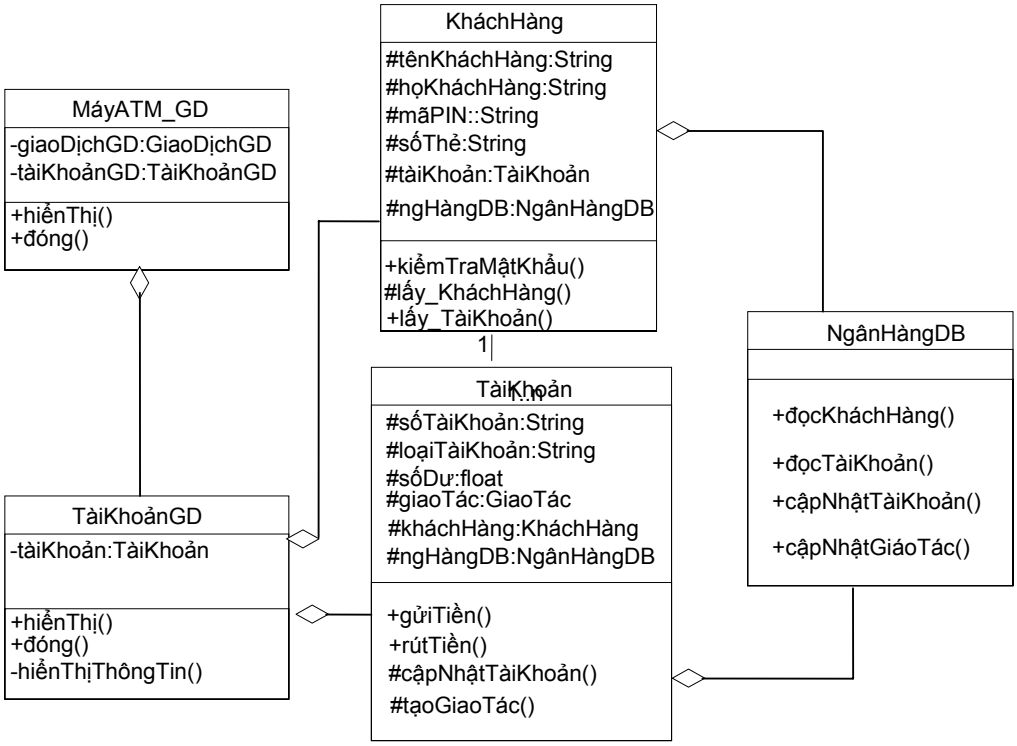


Sơ đồ lớp cho use case hiện thực hoá

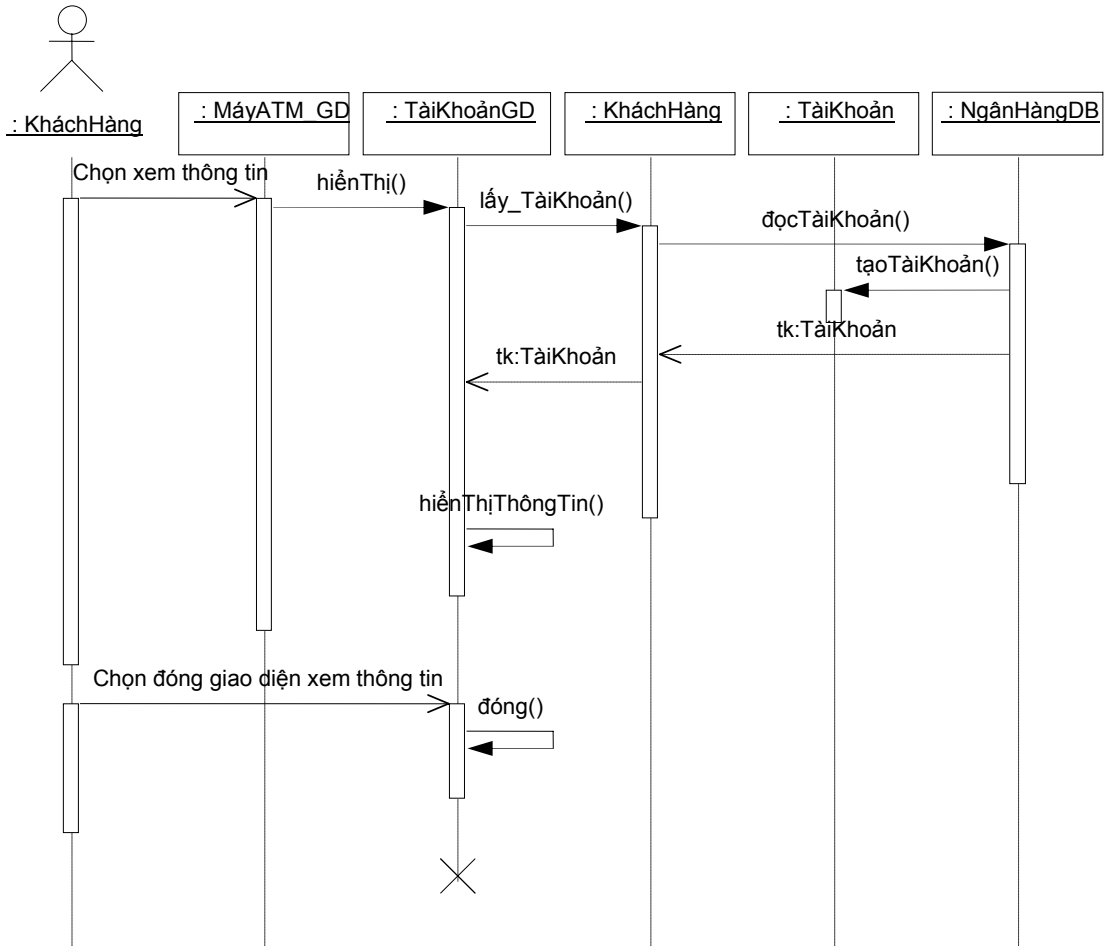
Mỗi use case hiện thực hoá có thể có một hoặc nhiều sơ đồ lớp mô tả các lớp tham dự. Một lớp và đối tượng của nó thường tham dự vào một hoặc nhiều use case hiện thực hoá.

Ví dụ, mô tả hiện thực hoá của use case *Truy vấn thông tin tài khoản*

Sơ đồ lớp tham dự



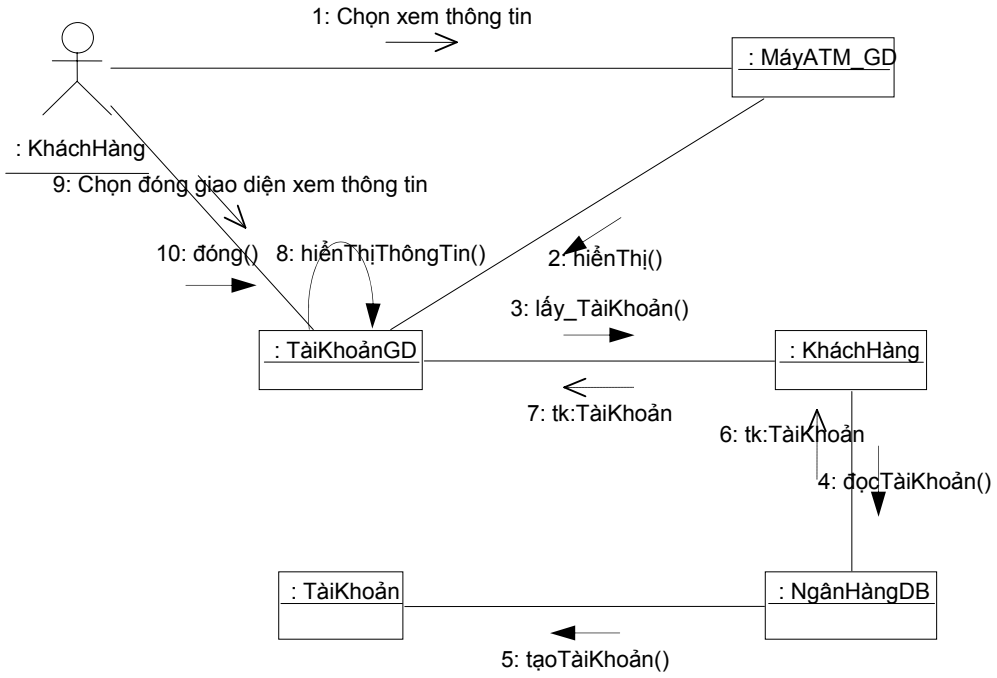
Sơ đồ tương tác trong use case hiện thực hoá



Sơ đồ tuần tự của hiện thực hoá use case *Truy vấn thông tin tài khoản*

Mỗi use case hiện thực hoá sẽ có một một hoặc nhiều sơ đồ biểu diễn sự tương tác giữa các đối tượng của use case. Trong UML chúng ta diễn đạt sự tương tác này qua hai loại sơ đồ: sơ đồ tuần tự và sơ đồ hợp tác.

Sơ đồ hợp tác của hiện thực hoá use case *Truy vấn thông tin tài khoản*



Câu hỏi và bài tập

1. Như thế nào là một kiến trúc ba tầng?
2. Ưu điểm của kiến trúc ba tầng so với kiến trúc client – server?
3. Các lớp ở tầng nghiệp vụ được xác định từ đâu?
4. Nhiệm vụ của tầng truy cập dữ liệu và các xác định các lớp này?
5. Phân biệt dữ liệu tạm thời (transient) với dữ liệu persistent ?
6. Có gì khác nhau giữa một đối tượng persistent và dữ liệu trong một cơ sở dữ liệu?
7. Các phép toán liên quan đến đối tượng persistent cần quản lý gồm những gì?
8. Cách xác định các lớp ở tầng giao diện?
9. Ý nghĩa của đối tượng điều khiển trong việc thiết kế?
10. Ý nghĩa của việc đưa vào đối tượng điều khiển trong thiết kế use case?
11. Ý nghĩa và nội dung mô tả một use case hiện thực hoá gồm những gì?

Chương 10

THIẾT KẾ GÓI VÀ HỆ THỐNG CON

Mục tiêu

Cung cấp cho người học các kiến thức về:

- Hiểu về việc áp dụng các khái niệm về gói (package) và hệ thống con (subsystem) trong việc phân chia hệ thống
- Gom nhóm các thành phần thiết kế thành các gói, hệ thống con nhằm mô tả về kiến trúc tổ chức nâng cao của hệ thống
- Cách thức xác định liên kết giữa các gói, hệ thống con

Thiết kế gói (package)

Mô hình thiết kế tổng thể của một hệ thống có thể được hình thành bởi những thành phần nhỏ hơn nhằm giúp cho người tiếp cận dễ hiểu hơn về hệ thống bằng việc gom nhóm các thành phần của hệ thống thành những gói (package) hoặc những hệ thống con (subsystem), sau đó chỉ ra sự liên kết giữa những nhóm này. Thiết kế gói dùng để kết hợp các thành phần thiết kế lại với nhau cho các mục tiêu về tổ chức. Không giống như thiết kế hệ thống con, thiết kế gói không đề xuất một giao diện hình thức mà nó cho phép trình bày ra các nội dung của nó (được xem như là public). Thiết kế gói nên được dùng như là một công cụ tổ chức mô hình để nhóm các thành phần lại với nhau. Nếu chúng thể hiện về các ngữ nghĩa liên quan đến các thành phần thì chúng ta dùng thiết kế hệ thống con. Một cách nào đó, hệ thống con được xem là một loại gói.

Một lớp nằm trong một gói có thể có phạm vi là toàn cục (public) hoặc nội bộ (private). Nếu là toàn cục thì nó có thể có liên kết đến bất kỳ lớp nào kể cả bên ngoài gói đó. Nếu là cục bộ thì nó chỉ có liên kết với các lớp chứa trong lớp đó.

Các điều kiện để hình thành gói: chúng ta có thể phân chia mô hình thiết kế thành các gói và các hệ thống con dựa trên các lý do sau:

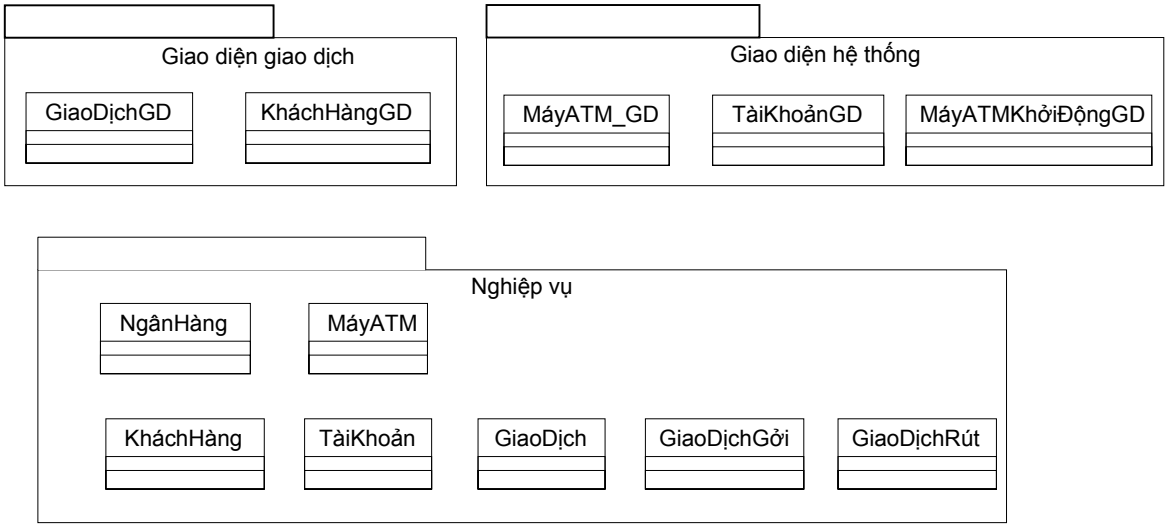
- Tạo ra các đơn vị hình thức và chuyển giao khác nhau của hệ thống.
- Khả năng tài nguyên và năng lực của các nhóm phát triển khác nhau yêu cầu phân chia hệ thống thành những nhóm khác nhau phù hợp với tài nguyên và năng lực của nhóm.
- Các hệ thống con có thể được dùng để cấu trúc hóa mô hình thiết kế nhằm phản ánh tới các loại người dùng. Bởi vì quá trình triển khai hệ thống có thể có nhiều thay đổi từ yêu cầu người dùng. Các gói và hệ thống con được thiết kế để đảm bảo rằng các thay đổi yêu cầu của một loại người dùng chỉ ảnh hưởng các phần của hệ thống liên quan đến loại người dùng đó.

Xây dựng gói cho các lớp tầng giao diện

Khi cá lớp tầng giao diện được gom nhóm thành các gói, có hai chiến lược sau đây có thể áp dụng. Việc chọn lựa chiến lược phụ thuộc vào việc đánh giá xem các giao diện của hệ thống thay đổi như thế nào trong tương lai:

- Nếu giao diện của hệ thống sẽ bị thay thế, chịu thay đổi lớn trong tương lai thì giao diện nên được thiết kế tách biệt với các thành phần còn lại của mô hình thiết kế. Do đó, khi giao diện có sự thay đổi, chỉ có các gói của giao diện bị ảnh hưởng.

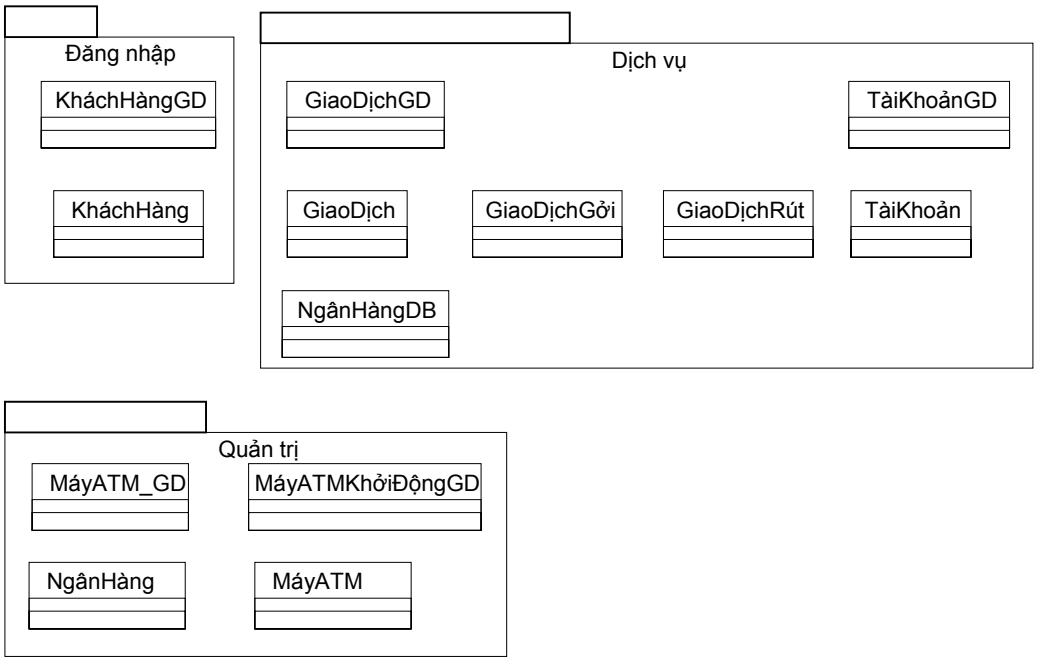
Ví dụ, nếu giao diện của hệ thống ATM được xác định là có khả năng thay đổi nhiều trong tương lai, thì chúng ta có thể tạo các gói cho tầng giao diện như sau:



Trong đó, gói "Giao diện giao dịch" gom nhóm tất cả các lớp giao diện liên quan đến cung cấp các chức năng giao dịch của hệ thống với khách hàng. Gói "Giao diện hệ thống" gom nhóm tất cả các giao diện còn lại liên quan đến đăng nhập và quản trị hệ thống. Gói "Nghiệp vụ" gom nhóm các lớp tầng nghiệp vụ của hệ thống.

- Nếu các giao diện được xác định là sẽ không có sự thay đổi và sẽ ổn định trong tương lai. Thì một thay đổi tới hệ thống nên được hiểu là một thay đổi bên trong thay vì thay đổi chỉ giao diện. Do đó, các lớp ở tầng giao diện sẽ được gom nhóm chung với các lớp tầng nghiệp vụ thành một gói.

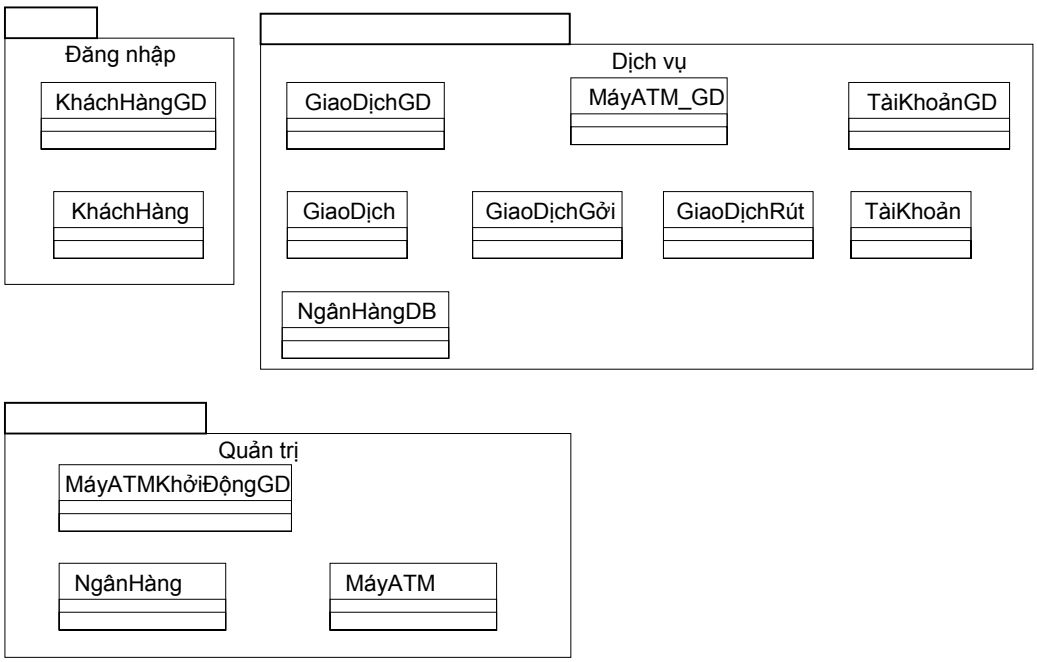
Nếu giao diện của của hệ thống ATM được xác định là ổn định và không có thay đổi trong tương lai. Chúng ta có thể phân chia hệ thống thành các gói như sau:



Theo cách phân chia gói như trên, trong mỗi gói có tất cả các lớp của tầng giao diện lẫn tầng nghiệp vụ và tầng cơ sở dữ liệu.

- Nếu các lớp tầng giao diện mà không có liên hệ về mặt chức năng với bất kỳ lớp nào ở tầng nghiệp vụ thì nên gom chung vào một gói với các lớp tầng giao diện khác cùng phụ thuộc vào một giao diện.

Gói "Quản trị" có các lớp không có liên quan đến bất kỳ một lớp nghiệp vụ nào, theo tiêu chuẩn này thì chúng ta gom chung lớp MáyATM_GD với các lớp GiaoDịchGD, TàiKhoảnGD vì có sự phụ thuộc. Ví dụ: chúng ta chuyển lớp MáyATM_GD qua gói "Dịch vụ".



- Nếu một lớp giao diện có liên quan tới một dịch vụ tùy chọn, thì gom nhóm chung các lớp cùng hợp tác với lớp đó vào một gói.

Xây dựng gói cho các lớp liên quan về chức năng

Một nhóm nên gom chung các lớp có quan hệ với nhau về mặt chức năng. Sau đây là một vài tiêu chuẩn được áp dụng khi nhận thấy rằng có hai lớp có liên hệ về mặt chức năng:

- Nếu các thay đổi trên hành vi hoặc cấu trúc của một lớp mà có ảnh hưởng đến một lớp khác thì hai lớp này có quan hệ về mặt chức năng.

Hai lớp TàiKhoản và GiaoDịch có quan hệ về mặt chức năng. Bởi vì nếu chúng ta thay đổi nội dung của một method của lớp GiaoDịch (ví dụ: hành vi gánThôngTinGiaoDịch()) thì phải cập nhật lại lớp TàiKhoản.

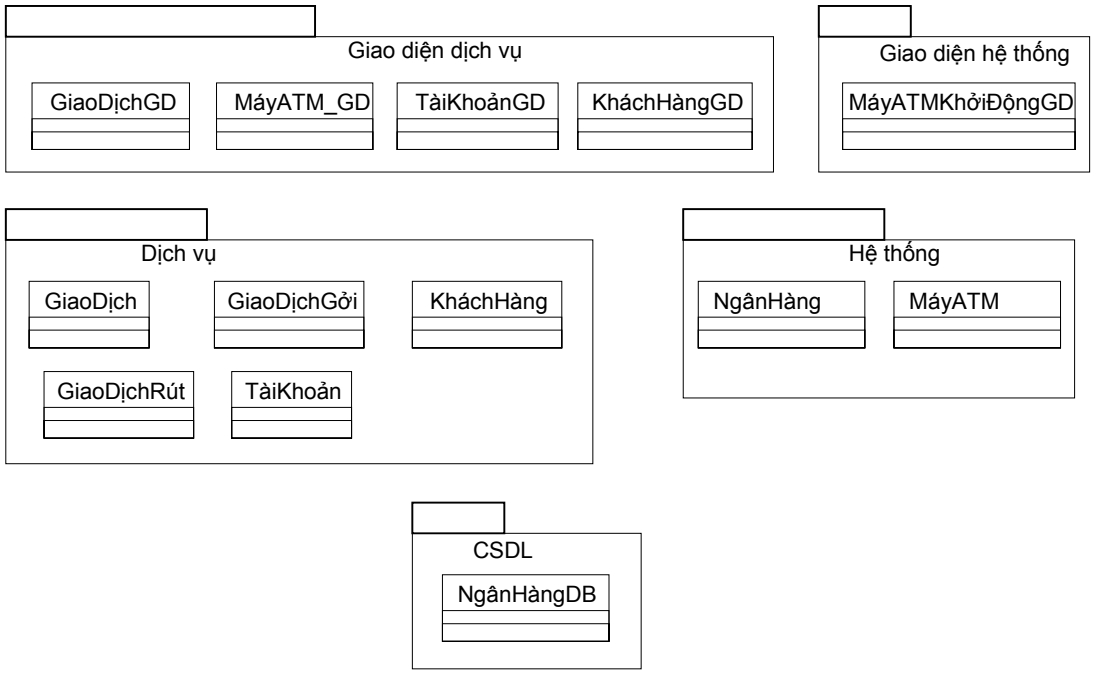
- Nếu chúng ta tìm thấy rằng một lớp có quan hệ chức năng với một lớp khác bằng cách bắt đầu bởi một lớp chúng ta kiểm tra sự ảnh hưởng của nó bằng việc xoá nó khỏi hệ thống. Nếu bất kỳ lớp nào còn lại được xác định là dư thừa (không tham gia vào hoạt động của hệ thống) thì chúng ta nói rằng các lớp này có quan hệ phụ thuộc vào lớp bị xoá.
- Hai lớp có quan hệ chức năng với nhau nếu cùng tương tác với cùng tác nhân. Nếu hai lớp không liên quan đến cùng tác nhân, thì chúng không cùng nằm trong một gói.
- Hai lớp có quan hệ chức năng với nhau nếu giữa chúng có các mối kết hợp (association, aggregation, …).

Hai tiêu chuẩn không cho phép hai lớp thuộc cùng một gói:

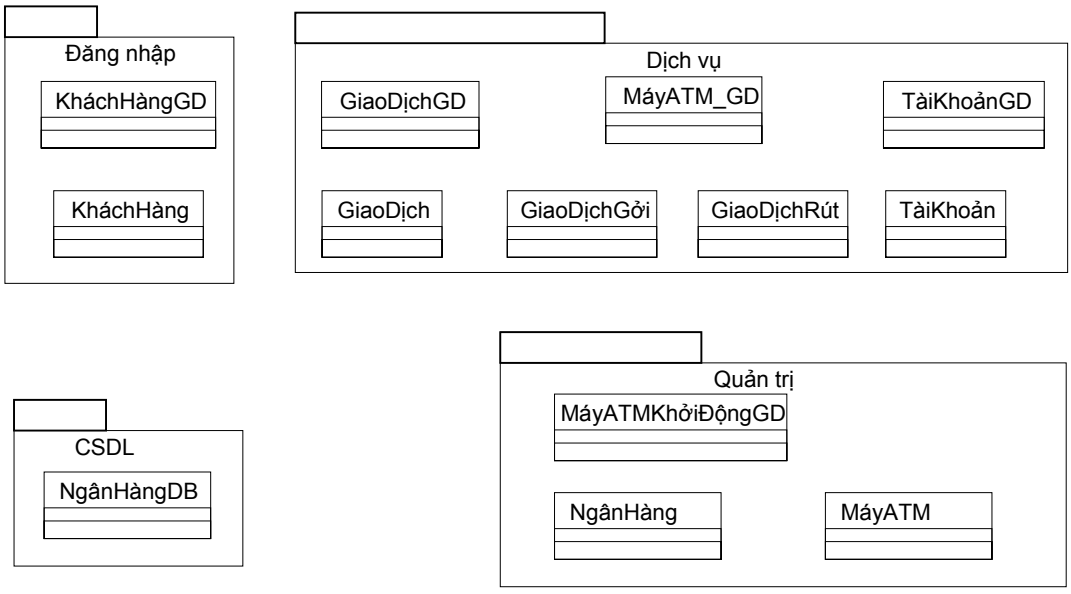
- Hai lớp liên quan đến các tác nhân khác nhau thì không nên cùng thuộc một gói
- Lớp bắt buộc và lớp tùy chọn không nên thuộc cùng một gói

Kết hợp với các tieu chuẩn phân chia gói cho các lớp tầng giao diện các lớp liên quan về chức, có những giải pháp phân chia hệ thống ATM thành các gói như sau:

Giải pháp 1: tách biệt giữa các tầng, một gói chỉ chứa các lớp thuộc một tầng.

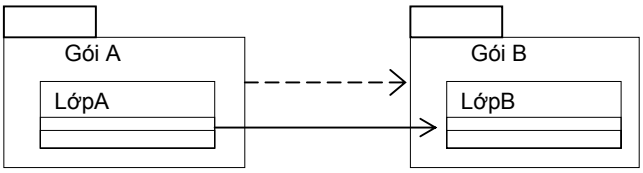


Giải pháp 2: trộn lẫn giữa các tầng, đặc biệt là tầng giao diện và tầng nghiệp vụ



Mô tả sự phụ thuộc giữa các gói

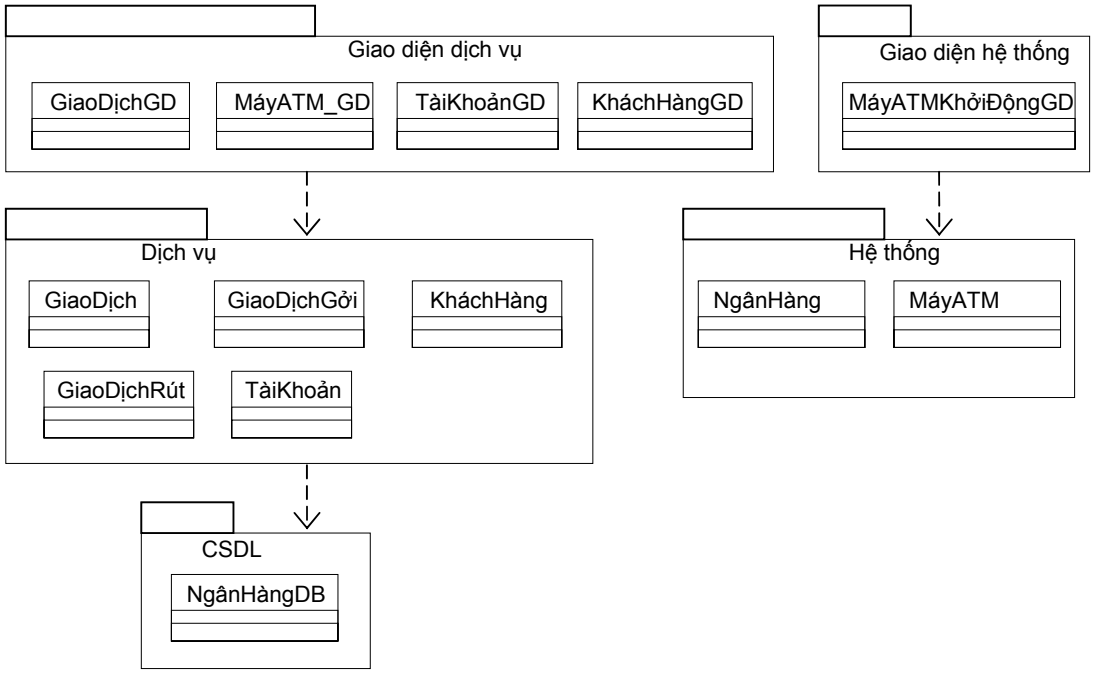
Nếu một lớp trong một gói có mối kết hợp với một lớp trong một gói khác thì những gói này có sự phụ thuộc lẫn nhau. Sự phụ thuộc giữa các gói được mô hình hoá dùng mối kết hợp phụ thuộc (dependency). Mối kế hợp này giúp chúng ta đánh giá hệ quả của sự thay đổi: một gói có các gói khác phụ thuộc vào thì sẽ khó thay đổi hơn một gói không có gói khác phụ thuộc.



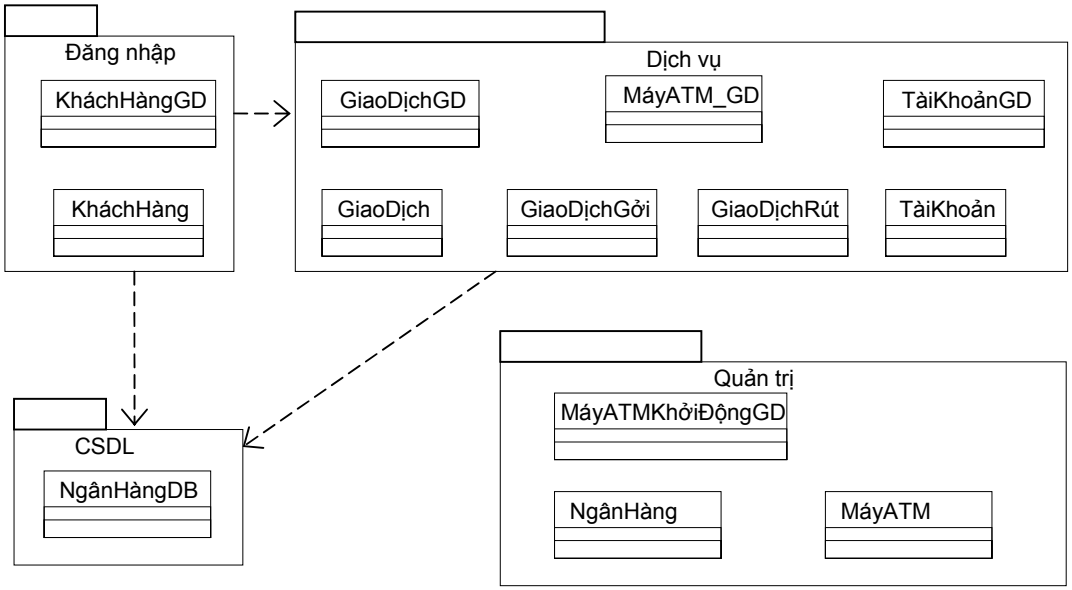
Trong sơ đồ trên, chúng ta nói rằng Gói B có sự phụ thuộc vào Gói A bởi vì có sự phụ thuộc của lớp B trong Gói B vào lớp A trong Gói A.

Quan hệ phụ thuộc các gói trong ATM

Đối với giải pháp 1



Đối với giải pháp 2



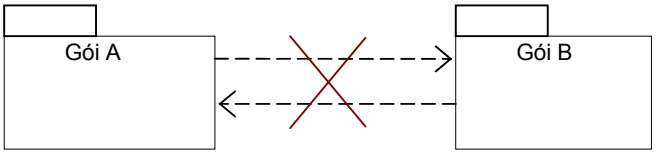
Với giải pháp 1, dễ thấy rằng các lớp giao diện của dịch vụ sẽ phụ thuộc vào các lớp tương ứng tầng nghiệp vụ.

Với giải pháp 2, gói "Đăng nhập" có quan hệ phụ thuộc tới gói "Dịch vụ" vì lớp KháchHàng có quan hệ với lớp TàiKhoản và lớp KháchHàngGD phụ thuộc vào lớp MáyATM_GD; tất cả các gói "Đăng nhập" và "Dịchvụ" đều phụ thuộc tới gói "CSDL" vì lớp KháchHàng và lớp TàiKhoản có quan hệ và tham chiếu đến lớp NgânHàngDB.

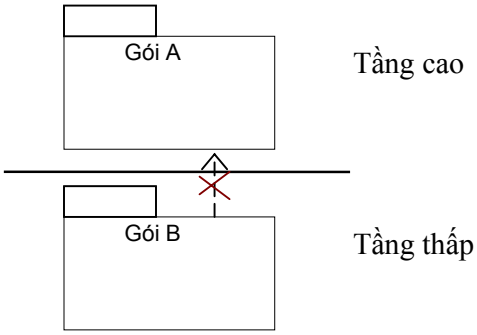
Đánh giá sự liên kết giữa các gói

Liên kết giữa các gói có thể là tốt và không tốt; tốt, bởi vì phản ánh việc tái sử dụng; không tốt, bởi vì phản ánh sự phụ thuộc làm cho hệ thống khó để thay đổi, tiến hoá và bảo trì. Một vài nguyên lý đánh giá như sau:

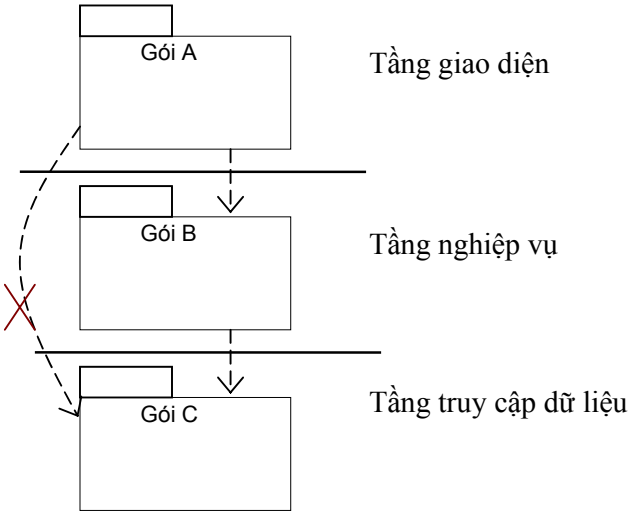
- Hai gói không nên có quan hệ phụ thuộc qua lại



- Các gói biểu diễn ở mức thấp không nên phụ thuộc vào các gói ở mức cao hơn



- Các phụ thuộc của các gói không nên nhảy mức



- Một gói không nên phụ thuộc vào một hệ thống con mà chi nên phụ thuộc vào một gói khác hoặc một giao diện.

Chương 11 MÔ HÌNH HOÁ CÀI ĐẶT HỆ THỐNG

Mục tiêu

Cung cấp cho người học các kiến thức về:

- Vận dụng các khái niệm thành phần của UML để thiết kế các đối tượng tập tin vật lý cài đặt các lớp trong thiết kế
- Mô hình hoá tài nguyên thiết bị sử dụng trong hệ thống sử dụng sơ đồ triển khai
- Mô tả việc cài đặt của các thành phần của hệ thống theo các tài nguyên sử dụng bằng cách tích hợp sơ đồ thành phần với sơ đồ triển khai hệ thống

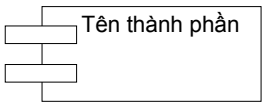
Giới thiệu

Kết quả của quá trình thiết kế trước đó đã cung cấp cho chúng ta nội dung chi tiết về hệ thống phần mềm theo cách nhìn luận lý. Nghĩa là tập hợp các lớp và đối tượng đầy đủ về hệ thống (đối tượng nghiệp vụ, đối tượng giao diện, đối tượng truy cập cơ sở dữ liệu,…). Vấn đề tiếp theo là làm sao chuyển hoá các lớp và đối tượng luận lý này thành các yếu tố tổ chức vật lý (các tập tin nguồn, tực thi,…) để phục vụ cho việc cài đặt và thực thi hệ thống trong một môi trường tài nguyên xác định.

Có hai sơ đồ trong UML dùng để mô hình hoá việc cài đặt hệ thống máy tính là sơ đồ thành phần (component diagram) và sơ đồ triển khai (deployment diagram). Sơ đồ thành phần dùng để mô hình hoá việc thiết kế tổ chức các thành phần của phần mềm và mối quan hệ giữa chúng trong hệ thống. Mối quan hệ này thường là mối quan hệ giữa các chương trình nguồn, giữa các phần mềm đang chạy hoặc giữa tập tin nguồn với tập tin thi hành tương ứng. Sơ đồ triển khai dùng để mô hình hoá các tài nguyên máy tính liên kết với nhau và liên kết với các thành phần để thi hành hệ thống. Vì các sơ đồ này mô tả mức vật lý của hệ thống, do đó, chúng ta phải xác định môi trường (ngôn ngữ lập trình, công cụ phát triển, hệ điều hành,…) và đơn vị tổ chức cụ thể để thực hiện thiết kế.

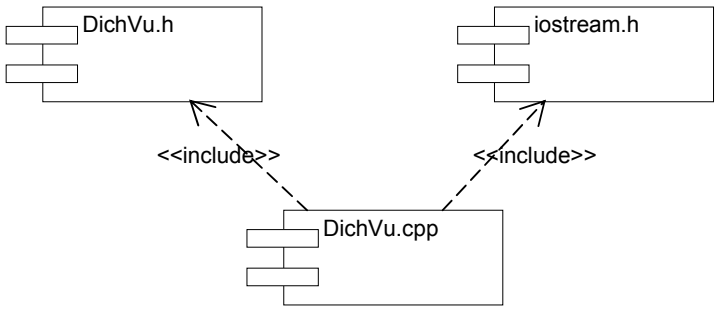
Xây dựng sơ đồ thành phần

Mô hình thành phần cung cấp một cách nhìn vật lý về mô hình hệ thống. Một mô hình thành phần trình bày việc tổ chức và sự phụ thuộc giữa các thành phần mềm, bao gồm mã nguồn (source code) thành phần mã nhị phân và thành phần thực thi. Các sơ đồ này cũng cho thấy các hành vi từ bên ngoài của các thành phần thông qua giao diện của chúng.

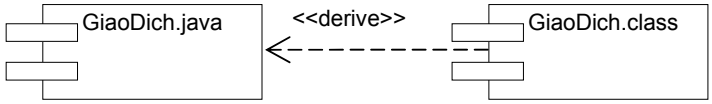


Tên của thành phần là tên của tập tin vật lý hoặc tên của một nhóm hoặc hệ thống con được thiết kế thành thành phần. Quan hệ giữa các thành phần đa phần là quan hệ phụ thuộc. Trong UML có nhiều loại quan hệ phụ thuộc được xác định: <<include>>, <<friend>>, <<derive>>, <<import>>,…

Ví dụ: biểu đồ biểu diễn sự phụ thuộc giữa các tập tin nguồn trong C++

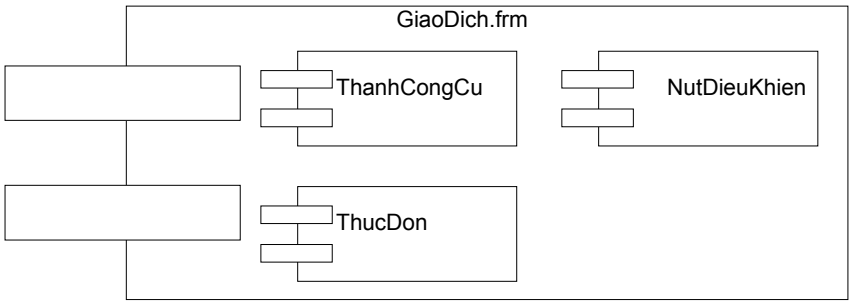


Sự phụ thuộc giữa tập tin nguồn và thi hành trong java

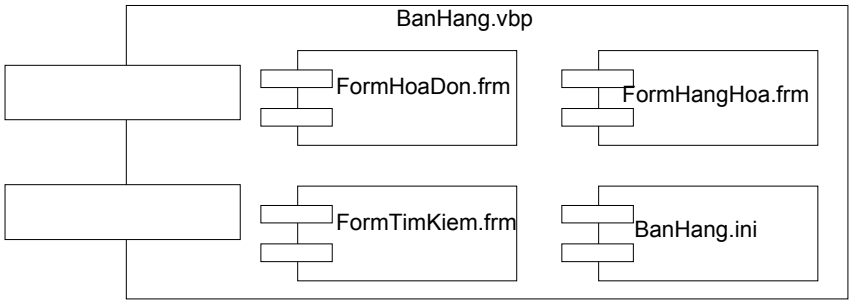


Một thành phần có thể là một thành phần chứa đựng. Nó được biểu diễn bằng cách đặt các thành phần khác vào trong. Thành phần này được dùng để một hệ thống con, một nhóm hoặc một đơn vị cài đặt có chứa các đơn vị cài đặt khác. Các thành phần chứa đựng có thể xem như các gói (package) và theo một cách nào đó, chúng ta cũng có thể xem một thành phần này là một cài đặt cho một gói trong mô hình các gói.

Ví dụ: mô tả thành phần của môt form trong VB



Mô tả một project trong VB



Xác định thành phần

Vì các thành phần được xây dựng phụ thuộc vào môi trường phát triển hệ thống (ví dụ: có thể là package trong Java hoặc project trong Visual Basic). Trong quá trình cài đặt, biểu đồ thành phần được đưa ra để mô tả các thành phần ở mức thấp hơn như các tập tin java, class của java, hoặc form hay module của visual basic. Tiến trình bao gồm các bước sau:

- Xác định mục tiêu của sơ đồ
- Xây dựng các thành phần của sơ đồ

- Xác định các phần tử chi tiết của sơ đồ (như: lớp, đối tượng, giao diện,…)
    - Xác định các đối tượng nội dung của thành phần
    - Xây dựng giao diện cho thành phần
- Xác định liên kết giữa các thành phần

Xác định mục tiêu của sơ đồ

Mục tiêu thông qua việc xác định mục đích để mô hình hoá quan hệ giữa các lớp và các thành phần, giữa các thành phần mã nguồn với nhau, giữa các tập tin mã nguồn với các tập tin thi hành hoặc giữa các tập tin thi hành với các thành phần hỗ trợ.

Trong java, các lớp được cài đặt thành các tập tin .class và tập tin .jar đóng gói nhiều lớp thành một tập tin lưu trữ

Xây dựng các thành phần của sơ đồ

Một thành phần được xác định gồm các lớp sẽ được cài đặt chung với nhau để đảm bảo việc thực thi dịch vụ được thực hiện.

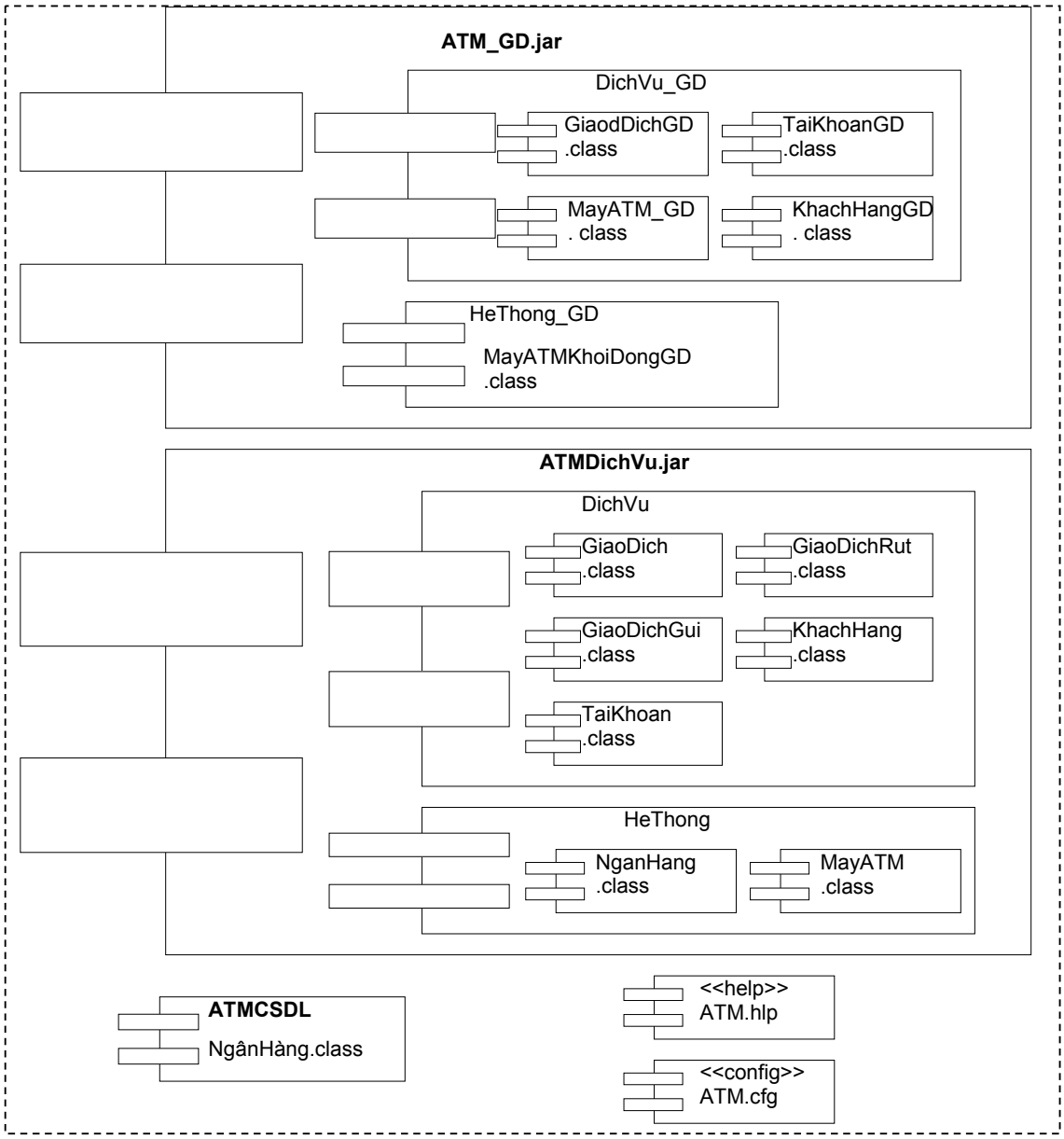
Ví dụ: trong hệ thống ATM, có nhiều giải pháp để xây dựng các thành phần. Một giải pháp có thể là xây dựng ATM thành ba thành phần gồm: thành phần giao diện của ATM (lưu trữ thành ATM_GD.jar), thành phần nghiệp vụ của ATM (lưu trữ thành ATMDichVu.jar), thành phần cơ sở dữ liệu (ATMCSDL.class).

Xác định các lớp đối tượng

Xác định đối tượng nội dung thành phần

Thêm các thành phần hoặc các đối tượng thiết kế vào các thành phần để hoàn thành nội dung của thành phần.

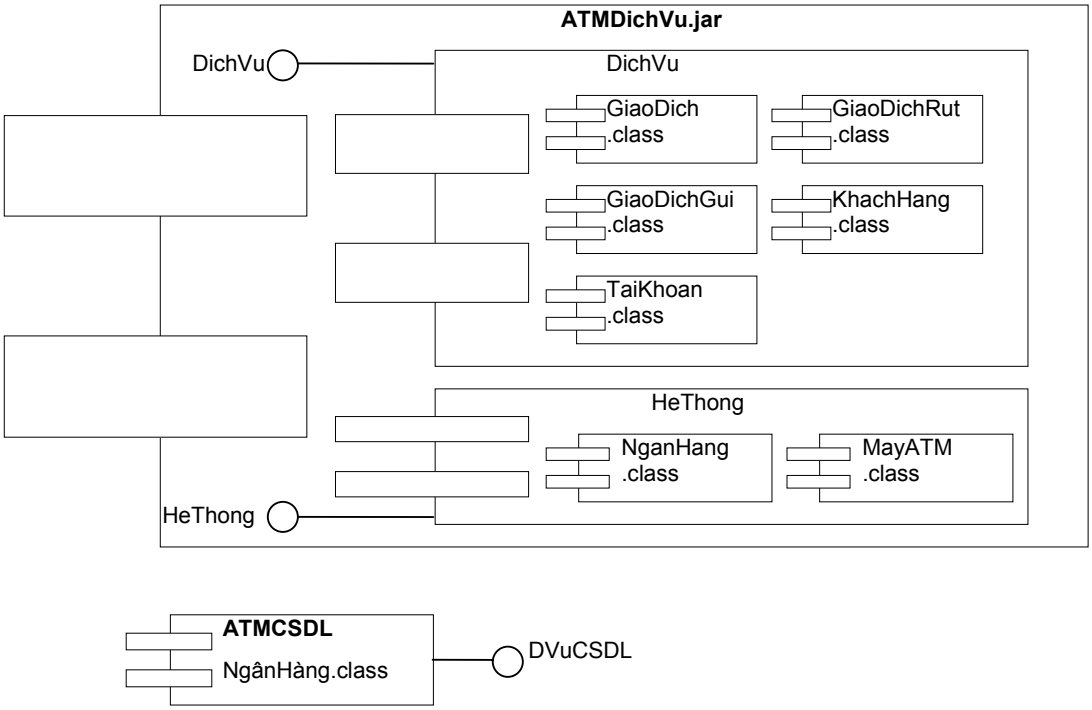
Ví dụ: xác định nội dung của ba thành phần trong hệ thống ATM. Ngoài ra, chúng ta phát triển thêm các tập tin khác như là: các tập tin cấu hình, hướng dẫn sử dụng,… Hệ thống sẽ cài đặt bằng java, vì vậy các thành phần sẽ là các tập tin .class, .java, .jar và các tập tin khác như là .hlp, .cfg,…



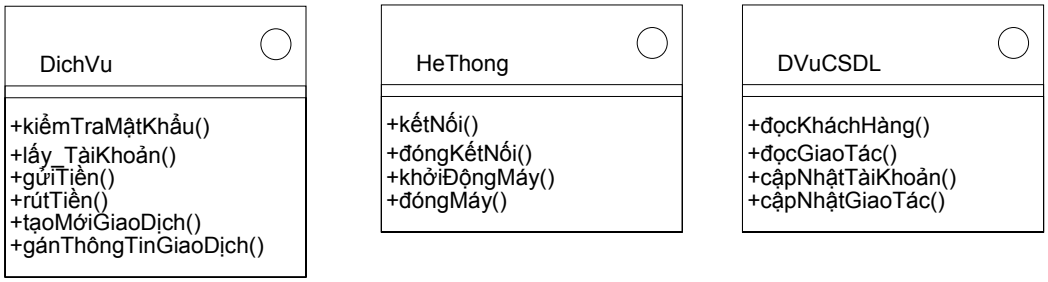
Trong sơ đồ trên, thành phần ATM_GD.jar sẽ bao gồm một thành phần chứa đựng DichVu. Trong đó, có bốn thành phần được lồng vào trong nó. Tương tự, trong thành phần ATMDichVu.jar bao gồm hai thành phần chứa đựng. Trong đó, có các thành phần của các lớp được lồng vào.

Xây dựng giao diện cho thành phần

Chúng ta có thể xây dựng các đối tượng giao diện cho các thành phần,. Mục đích của các giao diện này nhằm cung cấp ra bên ngoài các dịch vụ được thực hiện bởi thành phần cũng như nhằm bảo đảm tính bao bọc cho thành phần. Các thành phần hoặc đối tượng khác truy cập đến một thành phần đều thông qua giao diện của thành phần.



Ví dụ: chúng ta phát triển hai lớp giao diện cho thành phần ATMDichVu là DichVu và HeThong. Một lớp giao diện DVuCSDLcho thành phần ATMCSDL. Các thuộc tính chi tiết cho hai lớp giao diện này như sau:

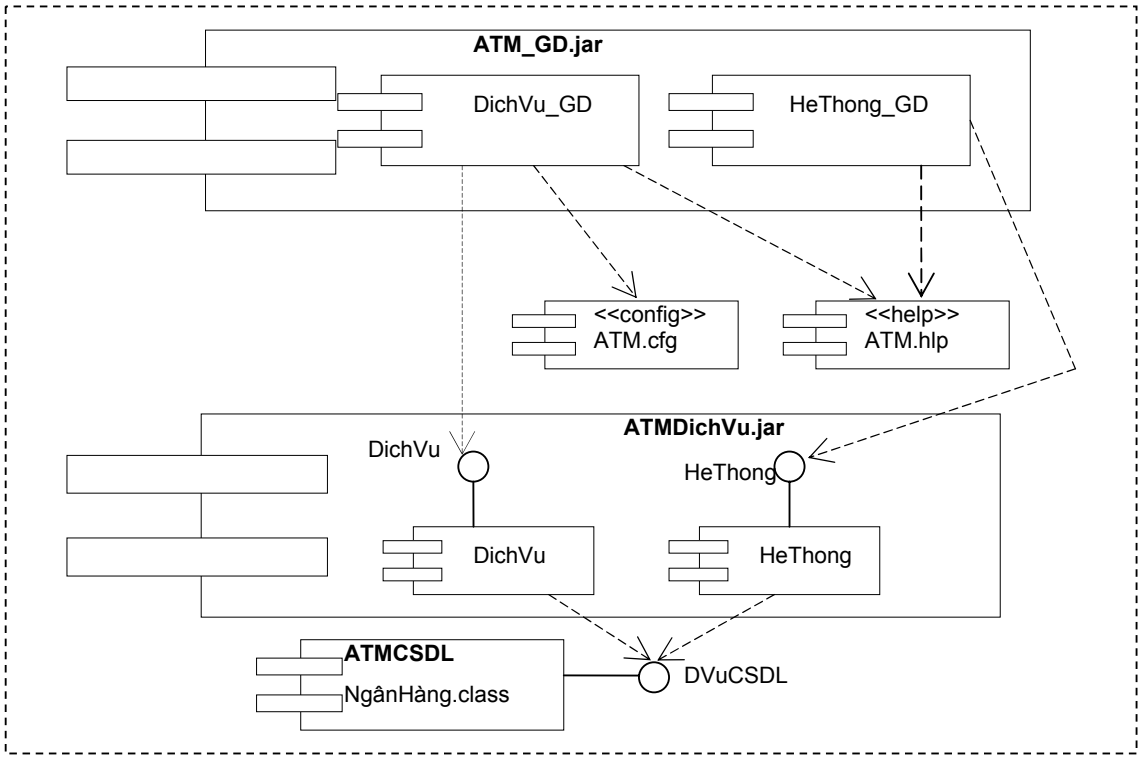


Xác định mối liên kết các thành phần

Các loại mối kết hợp này có thể là: mối quan hệ phụ thuộc, tổng quát hoá, hiện thực hoá và cácmối kết hợp khác

Ví dụ: xây dựng mối liên kết phụ thuộc giữa các thành phần trong hệ thống ATM

Sơ đồ cho thấy, có các phụ thuộc giữa thành phần DichVu_GD và giao diện DichVu của thành phần thành DichVu cũng như thành phần HeThong_GD với giao diện HeThong của thành phần HeThong để mô tả việc sử dụng các dịch vụ của các thành phần ở tầng giao diện hệ thống; sự phụ thuộc giữa thành phần DichVu và HeThong với giao diện DVuCSDL của thành phần ATMCSDL mô tả sử dụng các dịch vụ truy cập CSDL của các thành phần DichVu và HeThong. Ngoài ra, các phụ thuộc giữa các thành phần DichVu_GD và HeThong_GD với các thành phần ATM.cfg và ATM.hlp mô tả việc truy cập đến các thành phần này.

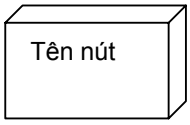


Xây dựng sơ đồ triển khai

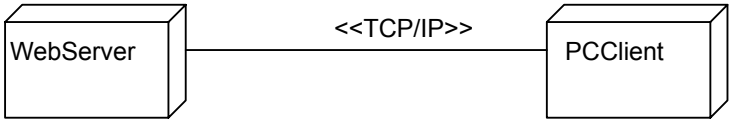
Sơ đồ cài đặt mô hình hoá cấu hình các yếu tố xử lý run-time và các thành phần phần mềm, các tiến trình, và các đối tượng thực thi trên các yếu tố này. Các thể hiện thành phần phần mềm mô tả sự xuất hiện run-time của các đơn vị mã phần mềm. Các thành phần không tồn tại như là các thực thể thực thi không được xuất hiện trong sơ đồ này.

Một sơ đồ thành phần có thể xem như là một đồ thị trong đó, các nút (node) liên kết với nhau qua các liên kết truyền thông (communication). Các nút đại diện cho các tài nguyên xử lý hệ thống, thường là các máy tinh có bộ xử lý và bộ nhớ. Tuy nhiên, cũng có thể là các thiết bị ngoại vi, cảm ứng và các hệ thống nhúng. Các nút có thể chứa các thể hiện của thành phần, khi đó nó chỉ ra rằng các thành phần chạy và thực thi trên nút này. Sơ đồ thành phần cũng cho biết thành phần nào của sẽ được cài đặt trên nút nào bằng việc sử dụng liên kết phụ thuộc với loại liên kết <<deploy>> từ thành phần đến nút hoặc bằng việc nhúng biểu tượng thành phần vào trong biểu tượng nút.

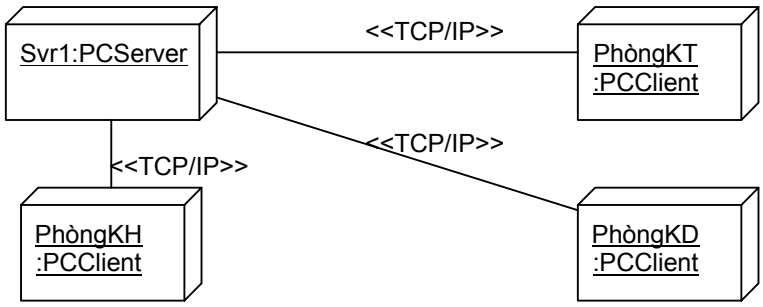
Ký hiệu nút



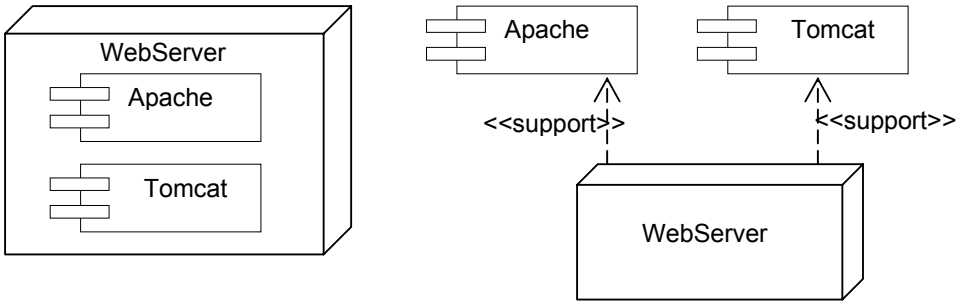
Sơ đồ triển khai biểu diễn kiểu các nút



Sơ đồ triển khai biểu diễn thể hiện các nút



Biểu diễn thành phần trên nút



Xây dựng một biểu đồ triển khai theo các bước như sau:

- Xác định các nút của biểu đồ
- Xác định các liên kết (truyền thông) giữa các nút
- Thêm các phần tử vào biểu đồ: các thành phần hoặc các đối tượng hoạt động
- Thêm liên kết phụ thuộc giữa các thành phần và đối tượng (nếu cần thiết)

Xác định các nút

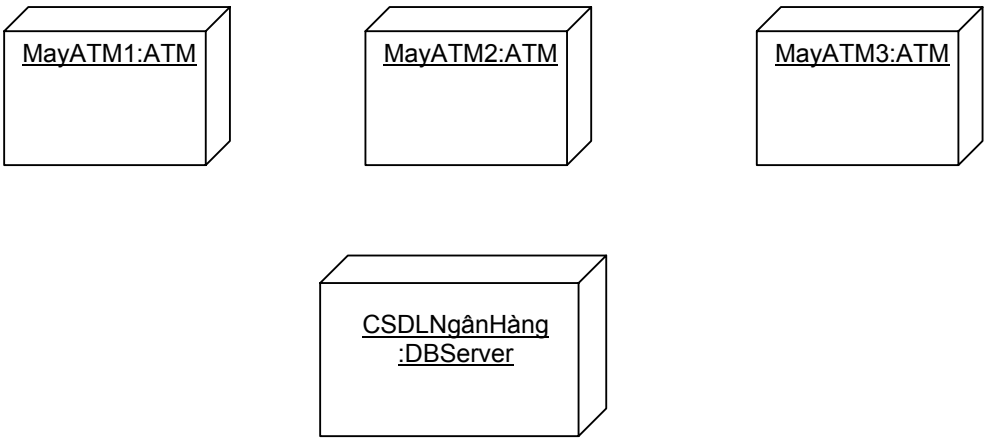
- Khảo sát về mặt không gian triển khai: đầu tiên chúng ta xem xét lại giải pháp kiến trúc của hệ thống về mặt không gian: các địa điểm triển khai hệ thống; hệ thống phân tán? cấu trúc mạng.

Ví dụ: trong hệ thống ATM, các địa điểm triển khai bao gồm: một ngân hàng, ba vị trí đặt máy ATM. Trong mô hình này, chúng ta chỉ cần biểu diễn một vị trí cho máy ATM, các vị trí ATM khác chỉ là một bản sao và giống nhau cho tất cả các máy ATM.

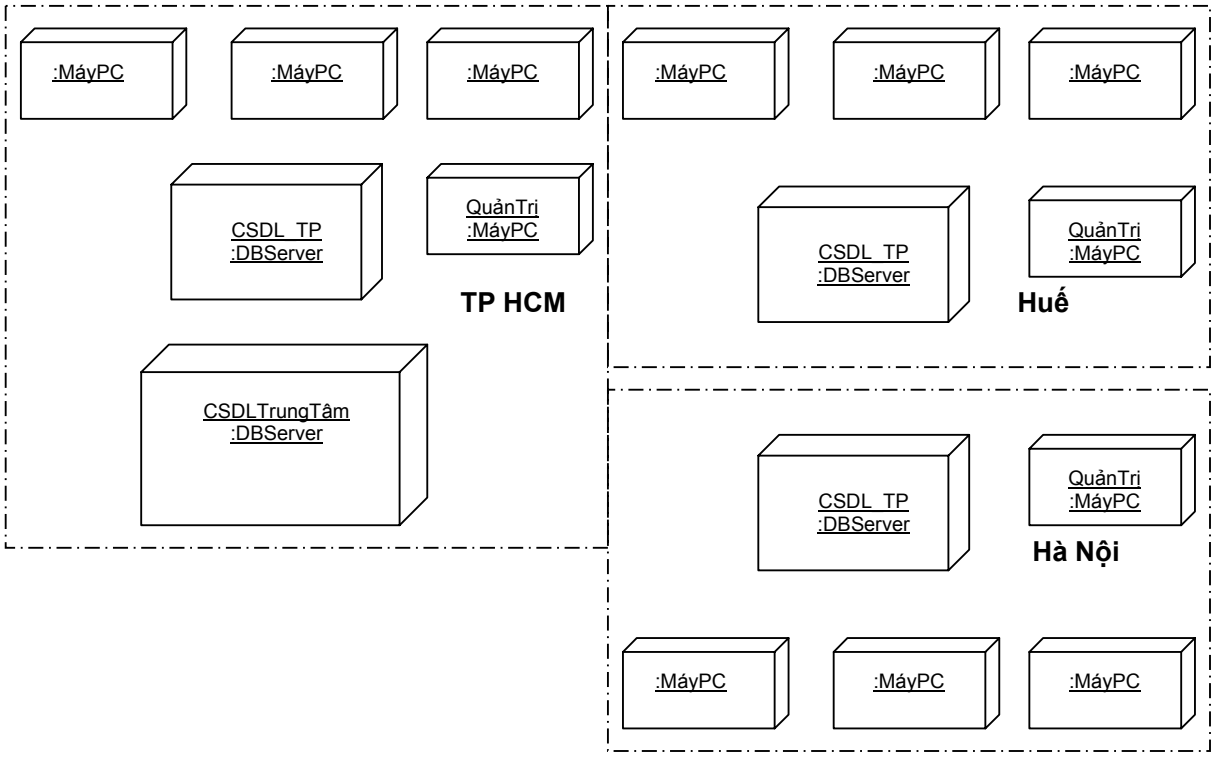
Ví dụ khác: giả sử một ngân hàng thương mại ABC có các địa điểm triển khai: một trung tâm ở TPHCM và cũng được xem như là một chi nhánh, một chi nhánh ở Huế và một chi nhánh ở Hà Nội.

- Xác định các nút trong các địa điểm: với mỗi địa điểm được xác định, chúng ta xem xét có bao nhiêu bộ xử lý được bố trí (hoặc sẽ bố trí) bao gồm các máy tính cá nhân (máy cho nhân viên sử dụng), các máy chủ mà hệ thống sẽ thi hành trên đó. Ứng với mỗi máy chúng ta dùng một nút để mô tả.

Ví dụ: trong hệ thống ATM, với vị trí ngân hàng chúng ta tạo một nút cho một server xử lý và lưu trữ về cơ sở dữ liệu về khách hàng, tài khoản, giao dịch. Mỗi máy ATM chính là một nút và ta có mô hình sau:



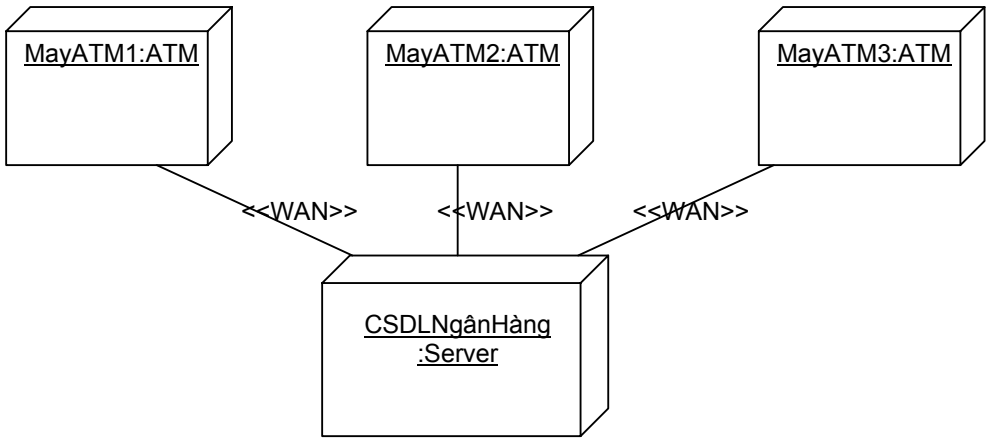
Với hệ thống ngân hàng thương mại ABC, ở trung tâm cài đặt một server CSDL quản lý toàn bộ dữ liệu hệ thống. Ở mỗi chi nhánh sẽ có một server CSDL quản lý dữ liệu của từng chi nhánh và các máy tính làm việc cho các nhân viên. Đây là một hệ thống phân tán và dữ liệu tại các server CSDL chi nhánh sẽ được tổng hợp với server CSDL trung tâm. Các nút được xác định cho hệ thống triển khai của ngân hàng như sau:



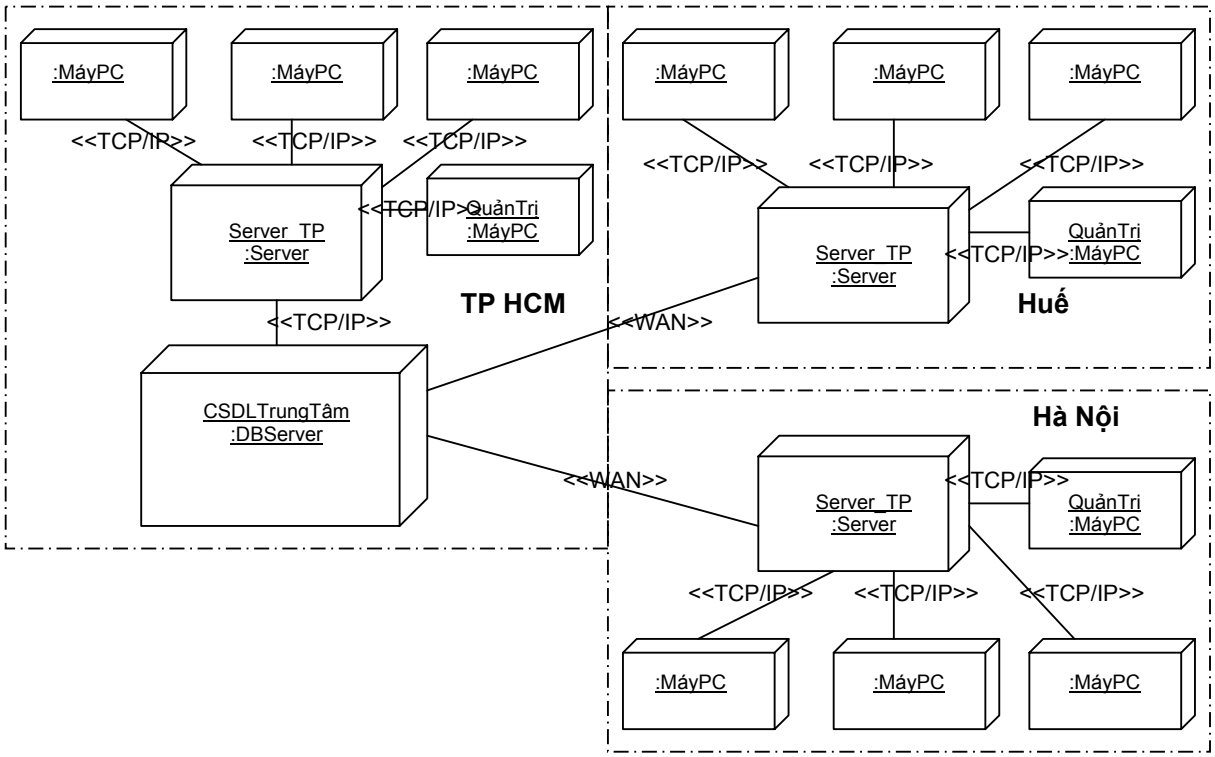
Xác định liên kết truyền thông giữa các nút

Các giao thức liên kết truyền thông giữa các nút. Các giao thức này sẽ có nhiều loại khác nhau: <<TCP/IP>>, <<HTTP>>, <<FTP>>, <<LAN>>, <<WAN>>, <<ADSL>>,…

Ví dụ: các liên kết các nút của hệ thống ATM



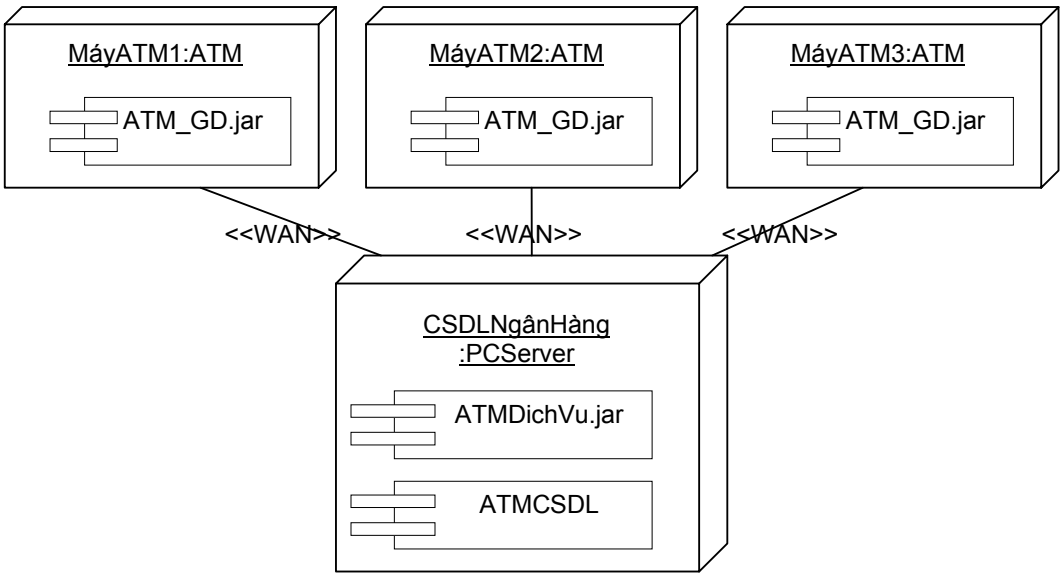
Liên kết các nút trong hệ thống ngân hàng thương mại ABC. Các nút trong cùng một địa điểm kết nối với nhau theo mạng cục bộ trên phương thức TCP/IP, các chi nhánh từ xa liên kết với server trung tâm theo mạng diện rộng (WAN).



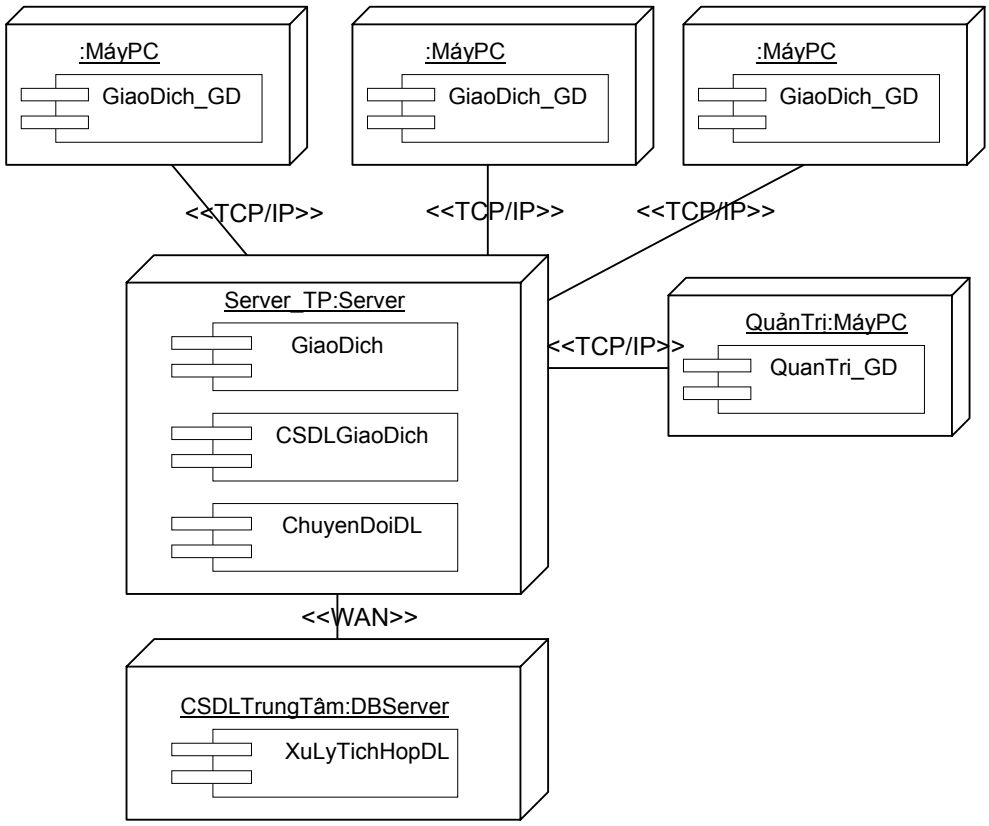
Xác định các thành phần của nút

Xác định các thành phần hoặc các đối tượng cho các nút nhằm biểu diễn vị trí triển khai của chúng trong hệ thống

Ví dụ: các thành phần được thêm vào các nút trong hệ thống ATM



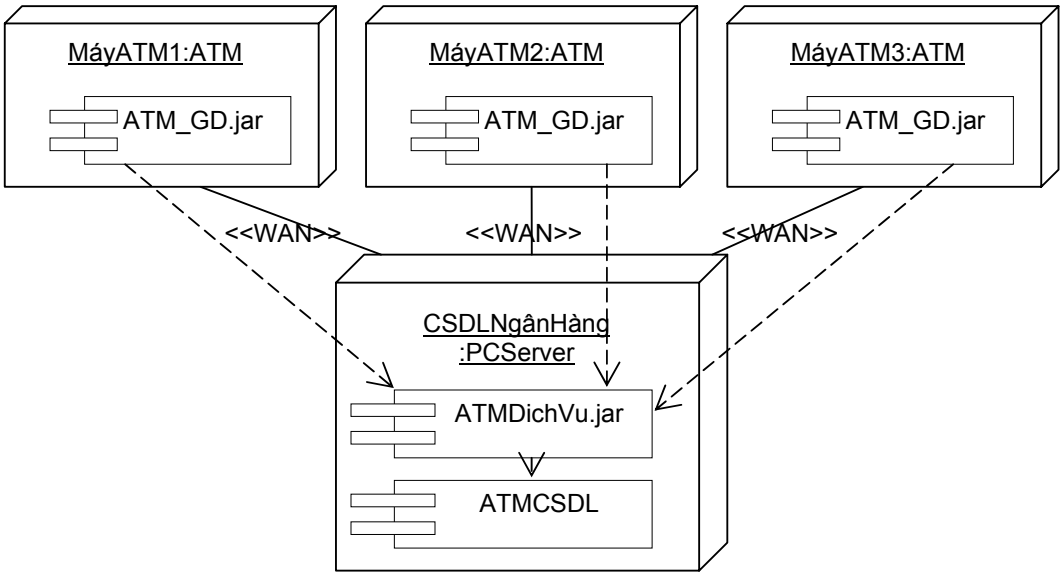
Các thành phần thêm vào nút của hệ thống ngân hàng thương mại ABC ở một chi nhánh. Trong đó, các máy PC cho nhân viên xử lý nghiệp vụ sẽ được cài một thành phần các đối tượng giao diện (GiaoDich_GD) (các form trong VB) nhằm phục vụ cho nhân viên làm việc tại chổ thông qua các giao diện này. Một máy PC của nhân viên quản trị sẽ được cài đặt các giao diện cho phép quản trị hệ thống (QuanTri_GD). Server của chi nhánh đón vai trò như một AppServer và được cài đặt các thành phần GiaoDich, CSDLGiaoDich gồm các lớp cung cấp các dịch vụ xử lý giao dịch và truy cập CSDL tại chi nhánh; thành phần ChuyenDoiDL được cài đặt tài server chi nhánh gồm các lớp nhằm xứ lý các dịch vụ chuyển đổi dữ liệu với server trung tâm. Tại nút trung tâm, thành phần xử lý tích hợp dữ liệu (XuLyTichHopDL) được cài đặt gồm các lớp quản lý việc tích hợp và đồng bộ hoá dữ liệu với các server chi nhánh và server trung tâm.



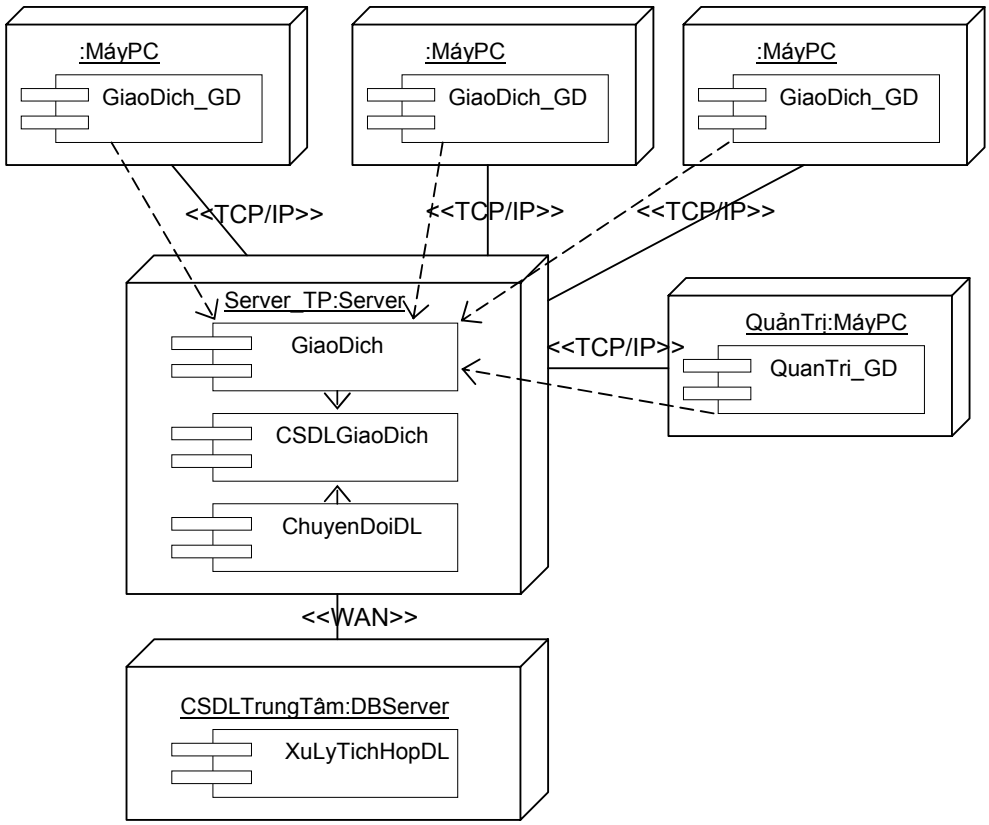
Thêm liên kết phụ thuộc

Nếu cần thiết chúng đưa vào các liên kết pụ thuộc giữa các thành phần với nhau hoặc giữa thành phần với các đối tượng trong sơ đồ nhằm mô tả rõ hơn sự ảnh hưởng lẫn nhau giữa các yếu tố của sơ đồ.

Ví dụ: với hệ thống ATM



Với hệ thống ngân hàng thường mại ABC



Chú ý rằng khi chúng ta tiếp cận với một hệ thống lớn lúc đó mô hình triển khai sẽ trở nên phức tạp với nhiều nút và liên kết. Khi đó, chúng ta nên chia thành nhiều biểu đồ triển khai với mỗi biểu đồ minh hoạ một phần của hệ thống.

Câu hỏi và bài tập

1. Mục tiêu của thiết kế cài đặt ?
2. Có bao nhiêu loại thành phần có thể sử dụng trong mô hình thành phần? ý nghĩa của mỗi loại?
3. Liên kết phụ thuộc giữa các thành phần trong sơ đồ mô tả điều gì?
4. Tại sao và khi nào chúng ta xây dựng giao diện cho một thành phần?
5. Mục tiêu sử dụng sơ đồ triển khai? Có bao nhiêu loại sơ đồ triển khai?
6. Một nút trong sơ đồ triển khai dùng để mô hình hoá nội dung gì?
7. Có bao nhiêu bước xây dựng sơ đồ triển khai?
8. Có bao nhiêu bước để xác định các nút trong hệ thống?
9. Liên kết giữa các nút trong sơ đồ triển khai mô tả liên kết gì?
10. Nội dung của một nút bao gồm những thành phần nào?

Tài liệu tham khảo

Bahrami Ali . *Object Oriented Systems Development*. McGraw-Hill, Singapore 1999.

Bruce E. Wampler. *The Essence of Object Oriented Programming with Java and UML*. Addison –Wesley 2001.

Philippe Kruchten. *The Rational Unified Process an Introduction Second Edition*. Addison – Wesley 2000.

OMG. *OMG Unified Modeling Language Specification*. An Adopted Formal Specification of the Object Management Group, Inc. 2002.

Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Software Development Process*, Addison-Wesley, 1999

Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

Ivar Jacobson, Maria Ericsson, Agneta Jacobson, *The Object Advantage:Business Process Reengineering with Object Technology*, Addison-Wesley, 1994

H. V. Đức, Đ. T. Ngân. *Giáo Trình nhập môn UML*. NXB Lao Động Xã Hội, 2003

