# Ngôn Ngữ UML & Ứng Dụng

mercury@Updatesofts.com

Ebooks Team

# MỤC LỤC

# LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, TpHCM đã tạo điều kiện cho chúng em thực hiện đề tài tốt nghiệp này. Chúng em xin chân thành cảm ơn Thầy Dương Anh Đức và Thầy Lê Đình Duy đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài. Chúng em cũng xin chân thành cảm ơn quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức cần thiết trong suốt quá trình học tập tại trường, và cũng xin gửi lòng biết ơn sâu sắc đến ba, mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt những năm học vừa qua. Mặc dù đã cố gắng hoàn thành luận văn với tất cả sự nổ lực của bản thân, nhưng luận văn chắc chắn không tránh khỏi những thiếu sót nhất định, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô.

*Nhóm thực hiện*
*Trung Nam&Quang Vũ.*

# LỜI MỞ ĐẦU

Ngày nay, công nghệ thông tin đã và đang đóng vai trò quan trọng trong đời sống kinh tế, xã hội của nhiều quốc gia trên thế giới, là một phần không thể thiếu trong một xã hội ngày càng hiện đại hóa. Nói đến công nghệ thông tin, chúng ta không thể không nhắc đến công nghệ phần mềm, phần mềm đóng một vai trò cực kỳ quan trọng trong lĩnh vực công nghệ thông tin. Hiện nay, việc phát triển công nghệ phần mềm thành một lĩnh vực kinh tế mũi nhọn là mục tiêu quan tâm hàng đầu ở nước ta.

Giờ đây, công nghệ phần mềm đã và đang tiến bộ từng ngày, hàng loạt những kỹ thuật, những công nghệ mới ra đời giúp cho việc phát triển các hệ thống phần mềm ngày càng đơn giản hơn. Một trong những lĩnh vực quan trọng và có ảnh hưởng rất lớn đến sự thành công của việc phát triển phần mềm là việc mô hình hóa phần mềm. Có rất nhiều ngôn ngữ mô hình hóa hỗ trợ cho việc mô hình hóa phần mềm, nhưng có lẽ nổi bật nhất là ngôn ngữ UML (Unified Modeling Language) từ hãng phần mềm Rational. UML không ngừng được phát triển và ngày càng được sử dụng rộng rãi trên thế giới, đa số các công cụ hỗ trợ phát triển phần mềm hiện nay đều có hỗ trợ ngôn ngữ UML. Do vậy, chúng em đã đầu tư nghiên cứu đề tài "Nghiên cứu ngôn ngữ UML và ứng dụng" nhằm nắm bắt một ngôn ngữ hiệu quả trong việc mô hình hóa phần mềm, cũng như có thể tìm hiểu và sử dụng một số CASE tool hỗ trợ cho việc phát triển phần mềm.

Với đề tài này, chúng em đã thử nghiệm ứng dụng UML trong việc mô hình hóa phần mềm quản lý giáo vụ đại học đơn giản, triển khai theo qui trình phát triển phần mềm Rational Unified Process của hãng Rational và bước đầu đã đạt được một số kết quả khả quan.

Mục đích của đề tài là nghiên cứu lý thuyết về ngôn ngữ UML (Unified Modeling Language) và ứng dụng của UML trong qui trình phát triển phần mềm RUP (Rational Unified Process). Trong khuôn khổ đề tài này, luận văn chúng em được trình bày trong năm chương chủ yếu tập trung trình bày về hệ thống ngữ nghĩa, ký hiệu của ngôn ngữ UML và ứng dụng của UML trong qui trình phát triển phần mềm. Luận văn có năm chương chính, bao gồm:

**Chương 1** – Tổng quan

Giới thiệu tổng quan về đề tài, mục đích nghiên cứu, phạm vi nghiên cứu, giới thiệu tổng quan về ngôn ngữ UML : khái niệm, các đặc điểm, lợi ích, nguyên nhân phát triển, lịch sử phát triển của ngôn ngữ UML...

**Chương 2** – Hệ thống ngữ nghĩa (UML Semantics)

Trình bày phần kiến trúc của ngôn ngữ UML dựa trên lớp metamodel, chủ yếu là các thành phần nền tảng của UML, các thành tố hành vi (Behavioral Elements), và Model Management.

**Chương 3** – Hệ thống ký hiệu (UML Notation)

Chương này trình bày hệ thống các loại lược đồ và hệ thống các ký hiệu được sử dụng trong UML để mô hình hóa hệ thống.

**Chương 4** – Ứng dụng của UML

Trình bày về ứng dụng của UML trong qui trình phát triển phần mềm RUP (Rational Unified Process), trong chương này cũng trình bày về qui trình phát triển phần mềm RUP và một ví dụ phân tích hệ thống sử dụng ngôn ngữ UML theo qui trình RUP.

**Chương 5** – Tổng kết

Là chương cuối của đề tài, tổng kết lại những kết quả đạt được và hướng phát triển trong tương lai.

# MỘT SỐ TỪ VIẾT TẮT

CASE Tool Computer Aided Software Engineering Tool
OCL Object Constraint Language
OMG Object Management Group
OMT Object Modeling Technique
OOSE Object-Oriented Software Engineering
RUP Rational Unified Process
UML Unified Modeling Language

# Chương 1

# TÔNG QUAN 1

## 1.1. Yêu cầu thực tế

Cùng với xu hướng phát triển ngành công nghệ thông tin, công nghệ phần mềm đã và đang trở thành lĩnh vực mũi nhọn của nhiều quốc gia trên thế giới. Ngày nay, việc phát triển một phần mềm với qui mô và chất lượng cao không còn là công việc đơn lẻ của những nhà lập trình. Đó là sản phẩm của một tập thể, một công ty phần mềm theo một qui trình công nghệ chuẩn được quản lý chặt chẽ và được hỗ trợ tối đa bởi các công cụ và môi trường phát triển phần mềm. Do đó, việc lập trình ngày càng trở nên dễ dàng hơn và nhường lại vai trò mấu chốt cho việc phân tích và thiết kế phần mềm, trong đó quan trọng nhất là đặc tả và mô hình hoá thế giới thực.

Trong tình hình đó các công ty phần mềm lớn trên thế giới đã nhanh chóng đưa ra nhiều công cụ hỗ trợ phân tích thiết kế dựa trên nhiều phương pháp khác nhau.

Các ngôn ngữ mô hình hóa ra đời và ngày càng được cải tiến, trong đó sự ra đời của UML (Unified Modeling Language) dựa trên ba phương pháp hướng đối tượng Booch, OMT, OOSE đã nhanh chóng trở thành một ngôn ngữ chuẩn dùng để trực quan hoá, đặc tả, xây dựng và làm sưu liệu cho các sản phẩm phần mềm và được hỗ trợ bởi các tổ chức,các công ty phát triển phần mềm trên thế giới.

Hiện nay hầu hết các CASE tools đều có hỗ trợ UML do đó việc tìm hiểu ngôn ngữ UML trước hết giúp tiếp cận với ngôn ngữ mô hình hóa tiên tiến, nắm bắt và sử dụng một số CASE tools hiện tại và từng bước xây dựng CASE tools đặc thù hỗ trợ cho việc xây dựng và phát triển phần mềm sau này.

# Chương 1
# TÔNG QUAN 2

## 1.2. Giới thiệu UML

Theo một bài báo của một nhà khoa học nỗi tiếng trong lĩnh vực công nghệ thông tin – Sinan Si Alhir – với tựa đề "Tri thức – nhân tố quyết *định của sự thành công !", bài báo viết* "Tri thức là sức mạnh – đây là câu nói của một nhà triết gia nỗi tiếng – FrancisBacon. Ngày nay, trên thị trường toàn cầu – và nhất là trong lĩnh vực công nghệ thông tin –nơi mà sự cạnh tranh trở nên rất phổ biến và quyết liệt, tri thức và khả năng áp dụng chúng vào trong công việc một cách hiệu quả sẽ mang lại cho chúng ta một lợi thế quan trọng vào loại bậc nhất. Chính điều này đã dẫn tới một câu hỏi –làm thế nào một tổ chức có thể nắm bắt, truyền đạt, trao đổi, và nâng cao tri thức của mình để đạt được lợi thế cạnh tranh trên thị trường ? Có lẽ câu trả lời chính là ngôn ngữ UML từ hãng phần mềm Rational và tổ chức OMG (Object Management Group)." Vậy UML là gì, tại sao nó lại được giới thiệu ấn tượng đến như thế ?

UML –Unified Modeling Language – tạm dịch là ngôn ngữ mô hình hợp *nhất,* nó được hiểu như là một ngôn ngữ thống nhất những xu hướng và hình thái của cuộc cách mạng tri thức trong lĩnh vực thông tin. Nó là một phương tiện giúp cho các tổ chức có thể nhận thức một cách tốt nhất lợi thế cạnh tranh thông qua việc nắm bắt, truyền đạt, trao đổi và nâng cao tri thức trong lĩnh vực công nghệ phần mềm. Chính xác hơn UML là một ngôn ngữ mô hình hóa dùng để đặc tả, trực quan hóa, xây dựng và làm sưu liệu cho các hệ thống phần mềm...

_ Unified (hợp nhất) UML được đưa ra lần đầu tiên bởi hãng Rational và ba chuyên gia về phương pháp luận hàng đầu trong lĩnh vực hệ thống thông tin/ kỹ thuật công nghệ Grady Booch, James.

# Chương 1
# TÔNG QUAN 3

## 1.3 Nguyên nhân ra đời

Rumbaugh, Ivar Jacobson. Nó là sự hợp nhất giữa những phương pháp cũ (Booch, OMT, OOSE...), kết hợp với những kinh nghiệm, những kiến thức thực tế trong lĩnh vực công nghệ thông tin. Hình 1–1 Sự hợp nhất củưa UML

_ Modeling (mô hình hóa) giúp chúng ta hiểu được thế giới thực, mô hình hóa thế giới thực để có thể hiểu được những đặc trưng, tính toán các thông số và dự đoán kết quả sẽ đạt được.

_ Language (ngôn ngữ) chức năng của UML như là một phương tiện để bày tỏ và trao đổi tri thức. Nó có bốn đặc điểm chủ yếu có thể phân biệt với các ngôn ngữ mô hình hóa khác

–General–purpose –đa dụng

–Broadly applicable – có thể ứng dụng rộng rãi

–Tool–supported – được hỗ trợ bởi các công cụ

–Industry standardized – chuẩn công nghiệp

## 1.3. Nguyên nhân ra đời

Sự ra đời của UML là một tất yếu khách quan trước sự bùng nỗ của ngành công nghệ thông tin, nó làm nỗi bật những xu hướng then chốt trong ngành công nghệ phần mềm, đưa ra được những vấn đề do sự phân rã của những phương thức mô hình hóa trước đây gây ra. UML được hình thành trên cơ sở của các vấn đề chính –tại sao chúng ta lại cần mô hình hóa phần mềm, xu hướng phát triển trong ngành công nghệ phần mềm ngày nay, sự hội tụ của các công nghệ...

# Chương 1
# TÔNG QUAN 4

## 1.4. Tầm quan trọng cua việc mô hình hóa

Mô hình là gì ? Đó chính là sự đơn giản hóa của thế giới thực.

Hình 1–2 Kiến trúc một ngôi nhà

_ Việc phát triển một mô hình cho một hệ thống trong công nghệ phần mềm cũng cần thiết như là việc lập một bảng thiết kế cho một tòa nhà lớn.

_ Những mô hình tốt giúp cho việc phối hợp giữa các nhóm phát triển tốt hơn.

_ Chúng ta cần xây dựng mô hình cho những hệ thống phức tạp bởi vì chúng ta không thể hiểu được toàn bộ hệ thống trong một môi trường rộng lớn như thế, khi sự phức tạp của hệ thống càng tăng, thì nó cũng đòi hỏi kỹ thuật mô hình hóa tốt hơn. Việc xây dựng mô hình giúp chúng ta hiểu rõ hơn về hệ thống mà chúng ta đang xây dựng.

_ Mô hình cung cấp cho chúng ta một khuôn mẫu về thế giới thực, giúp chúng ta có thể định hướng trong qúa trình xây dựng, có thể tính toán các chi phí, xác định các rủi ro, làm sưu liệu cho hệ thống...

_ Trong các nhân tố quyết định đến sự thành công của dự án, nhân tố cần thiết là một mô hình chuẩn, đặc tả đầy đủ, chi tiết về thế giới thực.

_ Trong một hệ thống mà độ phức tạp càng tăng, việc trực quan hóa và mô hình hóa càng cần thiết. Ngôn ngữ UML là một sự lựa chọn hoàn hảo và trên thực tế nó cũng đã được sử dụng và được chấp nhận rộng rãi trên thế giới.

# Chương 1

# TÔNG QUAN 5

## 1.5. Xu hướng phát triển trong ngành công nghệ phần mềm

– Phát triển về phạm vi, qui mô, chất lượng và tự động hóa trong qúa trình phát triển phần mềm

_ Vì giá trị của những phần mềm chiến lược tăng lên ở nhiều công ty, ngành công nghiệp này đã và đang tìm kiếm những kỹ thuật để có thể tự động sản xuất ra các sản phẩm phần mềm.

_ Tìm kiếm các kỹ thuật công nghệ mới để có thể nâng cao chất lượng, giảm giá thành và giảm thời gian đưa sản phẩm ra thị trường, những kỹ thuật đó bao gồm component technology, visual programming, patterns, frameworks... ngoài ra còn có những kỹ thuật dùng để quản lý những dự án lớn ngày càng tăng về phạm vi và qui mô.

_ Độ phức tạp của công việc, của các bài toán ngày càng tăng, và tùy thuộc vào lĩnh vực ứng dụng cũng như các công đoạn trong tiến trình phát triển phần mềm.

_ Một trong những động cơ chính của những nhà phát triển UML là tạo ra một bộ các ngữ nghĩa và ký hiệu nhằm phục vụ cho những dự án có kiến trúc phức tạp trên những phạm vi và lĩnh vực ứng dụng khác nhau.

# Chương 1

# TÔNG QUAN 6

,

## 1.3.3. Sự hội tụ cua các công nghệ

_ Trước khi UMỈ ra đời, không có ngôn ngữ mô hình hóa nào trội hơn hẳn các ngôn ngữ khác. Người dùng phải lựa chọn trong những ngôn ngữ khá tương tự nhau với những khác biệt nhỏ và cùng chia sẽ trên một tập khái niệm chung.

_ Chính sự thiếu tương đồng này đã ngăn cản những người mới tiếp cận với các kỹ thuật hướng đối tượng và mô hình hướng đối tượng.

_ Việc thường xuyên phải chi phí cho việc sử dụng và hỗ trợ cho nhiều ngôn ngữ mô hình hóa đã thúc đẩy nhiều công ty đầu tư vào sản xuất hoặc sử dụng kỹ thuật mới, họ tán thành và hỗ trợ cho việc phát triển ngôn ngữ UML.

_ Trong khi UML không hứa hẹn được những thành công thì nó đã làm được nhiều điều, chẳng hạn như:

+ Làm giảm đáng kể những chi phí thường xuyên cho việc huấn luyện và thay đổi công cụ khi thay đổi dự án hoặc tổ chức.

+ Cung cấp cơ hội cho việc tích hợp mới giữa các công cụ, các tiến trình và các domains.

+ Tạo ra một kiểu mẫu chuẩn, thống nhất cho các công việc.

# Chương 1
# TÔNG QUAN 7

## 1.7. UML hợp nhất các ý tương nỗi bật và những vấn đề thực tế trong quá trình phát triển cua công nghệ phần mềm..

Các nhà phẩt triển đã cố gắng duy trì tính đơn giản của UML, loại bỏ các thành phần không được sử dụng trong thực tế từ các phương pháp Booch, OMT, OOSE, thêm các thành phần và ý tưởng hiệu quả hơn từ các phương pháp khác nhau và chỉ xây dựng mới các phần cần thiết. Một số khái niệm mới đã được sử dụng trong UML bao gồm :

_ Cơ chế mở rộng (extension mechanism)
_ Luồng (thread) và tiến trình (process)
_ Sự phân tán (distribution) và đồng thời (concurrency) (dùng để mô hình hóa các ứng dụng ActiveX, DCOM và CORBA)
_ Khuôn mẫu (patterns) và sự cộng tác (collabarations)
_ Những lược đồ hoạt động –activity diagrams (cho business modeling)
_ Sự chọn lọc –refinement (xử lý các mối liên quan giữa các mức trừu tượng)
_ Giao diện (interface) và thành phần (component)
_ Ngôn ngữ mô tả ràng buộc (constraint language)

Một số trong các ý tưởng trên có thể được tìm thấy trong các phương pháp khác nhau, tuy nhiên, UML đã liên kết chúng chặt chẽ với nhau. Thêm vào đó là một số cải tiến mang tính cục bộ trên ba phương pháp Booch, OMT, OOSE bao gồm cả về ngữ nghĩa và lẫn ký hiệu. UML được phát triển dựa trên Booch, OMT, OOSE và các phương pháp hướng đối tượng khác, các nguồn khác nhau này kết hợp nhiều ý tưởng khác nhau từ nhiều tác giả. Có thể nói ý tưởng của UML dựa trên những kỹ thuật hướng đối tượng và cũng chịu ảnh hưởng của một số phương pháp không hướng đối tượng khác. Các nhà phát triển UML không tạo ra hầu hết các ý tưởng này mà vai trò của họ là chọn lọc và tích hợp các ý tưởng nỗi bật trong mô hình hóa hướng đối tượng và các vấn đề thực tế của công nghệ phần mềm.

Hình 1–3 Nền tảng của UML

# Chương 1
# TỔNG QUAN 8

## 1.8. UML độc lập với ngôn ngữ lập trình và qui trình phát triển phần mềm

UML là một ngôn ngữ mô hình hóa chuẩn nhưng không phải là một qui trình phát triển phần mềm chuẩn. Mặc dù UML phải được áp dụng trong phạm vi một qui trình cụ thể, các qui trình phát triển này thường khác nhau ở các tổ chức phát triển phần mềm, ở các vấn đề thuộc các lĩnh vực khác nhau. Do đó, các nhà phát triển UML đã cố gắng tập trung vào định nghĩa một mô hình mức siêu (meta)để thống nhất các khái niệm về ngữ nghĩa và ký hiệu, có thể hỗ trợ cho nhiều ngôn ngữ lập trình và qui trình phát triển phần mềm khác nhau.

# Chương 1
# TỔNG QUAN 9

## 1.9. UML là ngôn ngữ mô hình hóa đa dụng (general purpose)

UML tổng hợp các khái niệm của Booch, OMT và OOSE tạo thành một ngôn ngữ mô hình hóa chung và có thể sử dụng rộng rãi cho những người trước đây đã quen với ba phương pháp trên hay các phương pháp khác. Ngoài ra, UML mở rộng phạm vi mô hình hóa của các phương pháp hiện có và có thể mô hình hóa đầy đủ các hệ thống đồng thời hay phân tán.

UML là ngôn ngữ có thể được sử dụng cho nhiều mục đích khác nhau. UML cung cấp cơ chế cho việc tổ chức và phân loại tri thức theo ngữ cảnh của vấn đề cần giải quyết. Các tri thức này được nắm bắt đầy đủ bởi mô hình bao gồm nhiều thành phần và được thể hiện qua tập các lược đồ khác nhau có liên hệ chặt chẽ với nhau. Hơn nữa, mỗi lược đồ nắm bắt vấn đề ở những khía cạnh khác nhau qua các khái niệm, cấu trúc, các thành phần mô hình thể hiện những ngữ nghĩa và tri thức khác nhau. Các lược đồ này mô tả nội dung giao tiếp giữa các

thành viên trong qui trình phát triển phần mềm và được tích hợp với nhau để tạo nên tri thức mô tả hệ thống, những vấn đề cũng như cách thức thực hiện để giải quyết chúng.

UML được áp dụng rộng rãi, có thể mô hình hóa nhiều loại hệ thống khác nhau UML có thể được áp dụng trên nhiều phạm vi ở nhiều lĩnh vực khác nhau, các hệ thống khác nhau kể cả các hệ thống không phải phần mềm. UML có thể mô hình hóa nhiều loại hệ thống khác nhau như : hệ thống quản lý thông tin, hệ thống thời gian thực, hệ thống xử lý phân tán, các phần mềm hệ thống, hệ điều hành, cơ sở dữ liệu...

# Chương 1
# TÔNG QUAN 10

,

## 1.10. UML được hỗ trợ bơi các công ty, công cụ phát triển phần mềm

Nhiều nhà phát triển công cụ, tham gia hay không tham gia vào UML Partner Consortium hỗ trợ UML nhằm mục đích thúc đẩy việc sử dụng UML trong tổ chức. Nhận thấy được lợi ích của UML, các nhà phát triển đã xây dựng các công cụ hỗ trợ UML để dễ dàng nắm bắt và xử lý các tri thức cho mục đích của mình.

### *1.10.1. UML là một chuẩn công nghiệp*

UML không là một ngôn ngữ độc quyền mang tính chất khép kín mà hoàn toàn có khả năng mở rộng. UML có thể được điều chỉnh nhằm đáp ứng
yêu cầu riêng của một tố chức phát triển phần mềm.

### *1.10.2. Các lợi ích của UML*

Có thể mô hình hóa nhiều loại hệ thống, có thể dùng trong những pha khác nhau của qui trình phát triển phần mềm.

UML là sự thống nhất các khái niệm mô hình hóa nền tảng của những nhà nghiên cứu và phát triển công nghệ hướng đối tượng. UML cung cấp một số tính năng sau

_ Đầy đủ ngữ nghĩa và ký hiệu để giải quyết trực tiếp và kinh tế các vấn để hiện tại trong mô hình hóa.

_ Đầy đủ ngữ nghĩa để giải quyết một số khó khăn tương lai trong mô hình hóa đặc biệt liên quan đến công nghệ component, xử lý phân tán, framework và executability.

_ Cơ chế mở rộng metamodel cho mô hình hóa các ứng dụng đặc biệt. Cơ chế này cũng khiến cho các hướng tiếp cận mô hình hóa tương lai có thể phát triển dựa trên nền tảng UML.

_ Đầy đủ ngữ nghĩa để dễ dàng chuyển đổi mô hình giữa các công cụ hỗ trợ phân tích thiết kế khác nhau cũng như định rõ giao tiếp với các repository để lưu trữ và chia xẻ các thành phần mô hình.

# Chương 1
# TÔNG QUAN 11

## 1.11. Đối với người sư dụng UML

Cung cấp một ngôn ngữ mô hình hóa trực quan mang tính diễn đạt cao để phát triển và trao đổi giữa các mô hình. Một ngôn ngữ mô hình hóa nói chung được cấu trúc dựa trên các thành phần cơ bản nhất ở mức meta–meta. Nếu cấu trúc này thay đổi theo một tập các khái niệm mô hình hóa khác nhau theo các phương pháp khác nhau thì việc chuyển đổi giữa các mô hình sẽ không tránh khỏi mất mát thông tin. Để khắc phục vấn đề này, UML đã tập hợp các khái niệm mô hình hóa cốt lõi (core modeling concepts) được sử dụng trong nhiều phương pháp và công cụ mô hình hóa khác nhau. Các khái niệm này có thể hỗ trợ cho phạm vi lớn các ứng dụng. Ngoài ra, các khái niệm mô hình hóa ở mức thấp hơn và cụ thể hơn cho việc giao tiếp cũng được định nghĩa cho người sử dụng để mô hình hóa một hệ thống cụ thể.

### *1.11.1. UML cung cấp cơ chế mở rộng và đặc biệt hóa để mở rộng các khái niệm cốt lõi.*

Dựa trên những khái niệm đã được định nghĩa này, OMG mong đợi ở UML khả năng biến đổi để đáp ứng các yêu cầu mới của những phạm vi ứng dụng đặc biệt. Các nhà phát triển UML không muốn rằng mỗi khi có thay đổi thì các khái niệm cốt lõi phải được định nghĩa lại. Vì vậy, họ tin rằng việc đưa ra cơ chế mở rộng cho UML sẽ hỗ trợ những xu hướng phát triển mới. Người sử dụng có thể khai thác các tính năng sau của UML

_ Xây dựng mô hình bằng cách sử dụng những thành phần cơ bản đã được định nghĩa không sử dụng cơ chế mở rộng cho hầu hết các ứng dụng thông thường.

_ Thêm các khái niệm và ký hiệu mới cho những vướng mắc không giải quyết được với các khái niệm cơ bản.

_ Đặc biệt hóa các khái niệm, ký hiệu và ràng buộc cho một phạm vi ứng dụng (application domain) cụ thể.

# Chương 1
# TÔNG QUAN 12

## 1.12. UML đẩy mạnh tái sư dụng trong nền công nghệ phần mềm.

Tái sử dụng là một trong những vấn đề được quan tâm hàng đầu trong công nghệ phần mềm. Nguyên tắc của tái sử dụng là dựa trên các thành phần hiện có đã được kiểm chứng về chất lượng và chỉ xây dựng các thành phần mới khi thực sự cần thiết. Điều này không những giúp đương đầu với mức độ phức tạp ngày càng cao của ứng dụng mà còn giảm chi phí, giảm thời gian phát triển và tăng khả năng cạnh tranh của nhà phát triển phần mềm. UML cho phép tái sử dụng hiệu quả các thành phần của một hệ thống vì được xây dựng trên nền tảng hướng đối tượng. Ngoài ra, UML còn hỗ trợ các khái niệm phát triển phần mềm mức cao như collabarations, frameworks, patterns và components. Ngữ nghĩa của chúng được định nghĩa rất rõ ràng và điều này giúp đạt được những giá trị thực sự đầy đủ của hướng đối tượng và tái sử dụng.

# Chương 1
# TÔNG QUAN 13

,

## 1.13. So sánh với các phương pháp khác

UML không hoàn toàn tách biệt khỏi ba phương pháp cơ bản là Booch, OMT(*Object Modeling Technique*), OOSE (*Object-Oriented Software Engineering) mà nó tổng hợp những tinh hoa của cả ba phương pháp trên. Vì* vậy nếu trước đây bạn từng là người sử dụng các phương pháp Booch, OMT, OOSE thì những kiến thức, kinh nghiệm, các công cụ vẫn còn có giá trị sử dụng. UML có thể mô tả hệ thống một cách rõ ràng và thống nhất hơn so với các phương pháp Booch, OMT, OOSE và các ngôn ngữ khác. Điều này có nghĩa rằng việc chuyển qua dùng UML sẽ mang đến cho người sử dụng một gía trị nhất định nào đó, bởi vì nó cho phép bạn lập mô hình mọi công việc trong dự án, điều mà trước đây chưa có ngôn ngữ nào làm được.

Những người trước đây đã từng dùng các phương thức và các ngôn ngữ mô hình hóa khác sẽ có được lợi ích khi chuyển qua sử dụng ngôn ngữ UML, nó giúp cho họ loại bỏ những khác biệt không cần thiết về ngữ nghĩa và kỹ thuật thường xảy ra ở hầu hết những ngôn ngữ, những phương pháp đã đề cập ở trên. UML có hệ thống ký hiệu rất rõ ràng, mang tính thống nhất cao, được hỗ trợ bởi nhiều công cụ phát triển phần mềm. Đồng thời, trên một công cụ có hỗ trợ UML, người dùng có thể chuyển đổi các mô hình hiện tại của họ sang UML mà không sợ mất đi thông tin nào.

Đối với những người đã biết đến một phương pháp hướng đối tượng trước đó, sẽ có thể học UML trong một thời gian khá ngắn để có thể đạt được một trình độ tương ứng so với phương pháp mà họ đã biết trước đây.

# Chương 1

# TÔNG QUAN 14

UML là sự hợp nhất của các phương pháp hướng đối tượng, vì vậy nó cũng kế thừa một số khái niệm từ các phương pháp này, ví dụ như :

_ Lược đồ Use-ease tương tự như trong phương pháp OOSE.

_ Lược đồ lớp được kết hợp từ OMT và Booch và hầu hết những phương pháp hướng đối tượng khác.

_ Cơ chế mở rộng (extension mechanism) được định nghĩa trên nhiều loại lược đồ và hỗ trợ cho nhiều loại mô hình của UML để tạo ra các thành phần đa dạng mang đặc điểm riêng biệt của hệ thống nhằm mục đích hỗ trợ các góc độ mô hình hóa khác nhau. Nhiều khái niệm mới được bổ sung liên quan đến cơ chế mở rộng trước đây chưa từng được mô tả trong các các ngôn ngữ mô hình hóa chủ yếu khác bao gồm các khuôn mẫu (stereotypes), các ràng buộc (constraints) và giá trị đính kèm (tagged Values).

_ Lược đồ State-chart cơ bản dựa trên lược đồ cùng loại của David Harel với một ít thay đổi nhỏ. Lược đồ hoạt động cũng dựa trên phần lớn các ngữ nghĩa cơ bản dùng để định nghĩa State-chart của UML và tương tự như lược đồ luồng công việc (workflow diagram) trong nhiều phương pháp khác.

_ Lược đồ tuần tự được tìm thấy trong nhiều phương pháp hướng đối tượng dưới nhiều tên gọi khác nhau (interaction, message trace hoặc event trace).

_ Lược đồ cộng tác (collaboration diagram) được sửa đổi lại từ lược đồ đối tượng (object diagram) của Booch và lược đồ tương tác đối tượng (object interaction) của Fusion.

_ Lược đồ thực thi (Implementation diagram) hay còn gọi là lược đồ thành phần và triển khai bắt nguồn từ phương pháp của Booch. Tuy nhiên, chúng giờ đây mang tính chất hướng thành phần (componentcentered) và liên kết với nhau tốt hơn nhiều so với trước.

_ Stereotypes là một trong ba cơ chế mở rộng ngữ nghĩa của các thành phần UML sẵn có. Các stereotype cho phép biến đổi UML theo hướng mở rộng nghĩa là tạo ra các thành phần mang ngữ nghĩa mới đặc trưng riêng của hệ thống mà vẫn giữ nguyên các thành phần đã định nghĩa.

_ Ngôn ngữ mô tả ràng buộc Object Constraint Language được UML sử dụng để đặc tả ngữ nghĩa và được xem là ngôn ngữ mô tả trong quá trình mô hình hóa. OCL có nguồn gốc từ phương pháp Syntropy và chịu ảnh hưởng bởi một số ngôn ngữ cùng loại trong các pháp khác như Catalysis nhưng được chuẩn hóa và mở rộng hơn.

## Chương 1

## TÔNG QUAN 15

## 1.15. Lịch sư phát triển

UML được phát triển bởi hãng Rational và những đối tác. Được bắt đầu phát triển vào tháng 10 năm 1994, khi Grady Booch và Jim Rumbaugh bắt đầu công việc hợp nhất hai phương pháp Booch và OMT.

Hình 1-4 Lịch sử phát triển của UML

## Chương 1

## TÔNG QUAN 16

Bản phác thảo của phiên bản 0.8 được đưa ra vào tháng 10 năm 1995 với tên ban đầu là Unified Method. Vào mùa thu năm 1995, Ivar Jacobson cùng công ty của ông đã quyết định phối hợp với hãng Rational, bằng nỗ lực kết hợp thêm phương pháp OOSE, để tiếp tục phát triển Unified Method.

Với những nỗ lực của Booch, Rumbaugh, và Jacobson đã đưa ra phiên bản 0.9 và 0.91 vào tháng 6 và tháng 10 năm 1996 với tên là UML. Trong suốt năm 1996, nhóm tác giả của UML đã nhận được rất nhiều sự phản hồi từ phía người sử dụng và các chuyên gia trong lĩnh vực, họ đúc kết và bổ sung từ những

ý kiến này, nhưng rõ ràng cần phải có sự quan tâm nhiều hơn nữa từ phía người sử dụng. Từ đây UML được sự quan tâm nhiều hơn từ các tổ chức, các công ty phần mềm lớn, và với sự cộng tác của những công ty hàng đầu như Digital Equipment, HP, IBM, Microsoft, Oracle... phiên bản UML 1.0 trở thành một ngôn ngữ mô hình hóa được định nghĩa tốt hơn, rõ ràng, dễ hiểu, mạnh hơn và có khả năng ứng dụng rộng rãi.

Phiên bản UML 1.1 là sự phát triển về mặt ngữ nghĩa của phiên bản 1.0 đồng thời cũng tích hợp thêm những đóng góp của những nhà công tác mới. UML lần đầu tiên được đệ trình lên tổ chức OMG vào tháng 1/1997 và lần cuối vào tháng 9/1997 trước khi được đưa vào danh sách những kỹ thuật được thừa nhận của OMG vào tháng 11/1997. Kể từ đây OMG chịu trách nhiệm cho sự phát triển của UML trong tương lai. Sau khi được thừa nhận vào tháng 11/1997, OMG chịu trách nhiệm kiểm tra và phản hồi những kiến nghị từ phía các đối tác sử dụng, đồng thời tổ chức OMG cũng chịu trách nhiệm xử lý các lỗi kỹ thuật, những điểm bất tương đồng, những điểm còn mơ hồ và những thiếu sót nhỏ mà không cần phải sử đổi nhiều so với bản thảo ban đầu. Kể từ đây UML được đưa vào sử dụng rộng rãi và được cải tiến không ngừng, phiên bản UML 1.3 alpha được giới thiệu vào tháng 3/1999 và sau đó phiên bản UML 1.3 chính thức được giới thiệu vào thánh 6/1999. Và phần kiến thức lý thuyết về UML mà chúng em trình bày trong báo cáo này dựa trên phiên bản mới nhất hiện nay là UML 1.3.

# Chương 1
# TÔNG QUAN 17

## 1.17. Kiến trúc tổng quan cua UML

### *1.17.1. Kiến trúc của UML*

UML được định nghĩa trên một cơ cấu quan niệm sử dụng cho việc mô hình hóa, bao gồm bốn mức trừu tượng sau

_ Lớp meta-metamodel bao gồm các thành phần cơ bản nhất.

_ Lớp metamodel gồm tất cả những thành phần tạo nên UML bao gồm cả các khái niệm trong các mô hình hướng đối tượng và thành phần.

_ Lớp model bao gồm các mô hình UML. Đây là lớp dùng cho việc mô hình hóa các bài toán, các hệ thống và giải pháp.

_ Lớp user model bao gồm các thành phần minh họa cho UML model.

Hình 1–5 Kiến trúc tổng quan của UML

# Chương 1
# TÔNG QUAN 18

,

## 1.18. Các mô hình

Các mô hình – xét về tĩnh – nắm bắt một số đặc điểm và hành vi của hệ thống –xét về động – nắm bắt những đặc điểm của hệ thống, về cơ bản chúng lưu trữ các tri thức về mặt ngữ nghĩa.

### *1.18.1. Cấu trúc View*

Chúng ta không thể mô hình hóa một hệ thống phức tạp chỉ bằng một mô hình hay một lược đồ, hệ thống phải được phân tích dưới những góc độ khác nhau : các yêu cầu chức năng, phi chức năng, cách tổ chức hệ thống... Vì vậy để có thể mô hình hóa hệ thống một cách chi tiết, UML đưa ra định nghĩa về cấu trúc View, mỗi View (thể hiện) là một thể hiện của hệ thống được mô hình hóa, mỗi View có thể bao gồm nhiều loại lược đồ khác nhau. UML cung cấp cấu trúc View theo việc mô hình những bài toán và những giải pháp, bao gồm

Hình 1-6 Cấu trúc Viiew trong UML

_ User model View – hay còn gọi là Use Case View hoặc Scenario View – thể hiện các vấn đề và các giải pháp liên quan đến chức năng tổng quát của hệ thống.

_ Structural model View –hay còn gọi là Static hoặc Logical View –thể hiện các vấn đề liên quan đến cấu trúc thiết kế của hệ thống.

_ Behavioral model View – hay còn gọi là Dynamic, Process, Concurrent, hoặc Collaboration View – thể hiện các vấn đề liên quan đến việc xử lý giao tiếp và đồng bộ trong hệ thống.

_ Implementation model View –hay còn gọi là Component View – thể hiện các vấn đề liên quan đến việc tổ chức các thành phần trong hệ thống.

_ Environment model View – hay còn gọi là Deployment View – thể hiện các vấn đề liên quan đến việc triển khai hệ thống.

_ Và một số model View khác có thể được định nghĩa và sử dụng khi cần thiết.

## Chương 1
## TÔNG QUAN 19

## 1.19. Những lược đồ

Hình 1–7 Những lược đồ trong UML

## Chương 1
## TÔNG QUAN 20

Các lược đồ miêu tả các tri thức về mặt cú pháp, được tổ chức xung quanh cấu trúc View :

_ User model View

+ Lược đồ Use Case (Use case diagram) – mô tả các chức năng của hệ thống.

_ Structural model View

+ Lược đồ lớp (Class Diagram) – mô tả cấu trúc tĩnh của hệ thống, thể hiện các phần hệ thống xử lý được.

+ Lược đồ đối tượng (Object Diagram) – mô tả cấu trúc tĩnh của hệ thống tại một thời điểm xác định, nó có thể được xem
như một thể hiện của lược đồ lớp.

_ Behavioral model View

+ Lược đồ tuần tự (Sepuence Diagram) – mô tả sự tương tác giữa các thành phần trong hệ thống tuần tự theo thời gian.

+ Lược đồ cộng tác (Collaboration Diagram) – mô tả sự tương tác của các thành phần theo thời gian và cả không gian.

+ Lược đồ trạng thái (State Diagram) – mô tả trạng thái và sự hồi đáp giữa các thành phần trong hệ thống, bổ sung cho

lược đồ lớp.

+ Lược đồ hoạt động (Activity Diagram) – mô tả hoạt động của các thành phần trong hệ thống.

_ Implementation model View

+ Lược đồ thành phần (Component Diagram) – mô tả tổ chức của các thành phần thực thi trong hệ thống.

_ Environment model View

+ Lược đồ triển khai (Deployment Diagram) – mô tả cấu hình của các thành phần môi trường mà sự xếp đặt của các thành phần hệ thống thực thi trên đó.

# Chương 1
# TÔNG QUAN 21

Và một số lược đồ khác có thể được định nghĩa và sử dụng khi cần thiết.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 22

## 2.1. Giới thiệu

UML bao gồm UML metamodel và UML model. UML metamodel giữ chức năng định nghĩa các thành tố và cú pháp UML. UML model mô tả ký hiệu các thành tố và các lược đồ dựa trên UML metamodel.

UML metamodel bao gồm các thành tố và một số quy tắc về cú pháp. Ngoài việc thành tố UML mang một ý nghĩa xác định, cú pháp UML còn mô tả cách liên kết những thành tố nào với nhau để tạo ra ý nghĩa gì. Ở góc độ mô hình hóa, các thành tố UML có thể phân chia làm ba loại là các thành tố mô hình hóa tĩnh, các thành tố mô hình hóa tương tác và các thành tố quan hệ có chức năng liên kết giữa hai thành tố trên với nhau. UML metamodel giữ vai trò hướng dẫn người sử dụng UML về cú pháp trong mô hình hóa. Ngoài ra, UML metamodel còn được sử dụng bởi các nhà phát triển CASE tool để mô hình hóa dữ liệu cho một CASE tool hỗ trợ UML. Mô hình dữ liệu này sử dụng lại định nghĩa thành tố UML để thiết kế các lớp cơ bản và bổ sung thêm các lớp mới tùy theo chức năng CASE tool cung cấp cho người sử dụng. UML model là biểu diễn ký hiệu của các thành tố UML đồng thời cung cấp cho người sử dụng các lược đồ UML cụ thể để mô hình hóa cũng như làm ngôn ngữ giao tiếp giữa các thành viên của nhóm trong quá trình phát triển phần mềm. Nói cách khác, các lược đồ trong UML model là thể hiện của các cú pháp tương ứng trong UML metamodel. UML metamodel được chia thành nhiều gói thành phần (package) dựa trên ý nghĩa của cú pháp được mô tả. Mỗi gói định nghĩa các thành tố khác nhau và mô tả một nhóm cú pháp dựa trên các thành tố này. Trong mỗi gói lại có thể bao gồm các gói con. Việc phân chia này giúp cho định nghĩa của UML metamodel rõ ràng hơn, chỉ quan tâm đến các thành tố trong gói và loại bỏ các thành tố không cần thiết vượt ra khỏi phạm vi ngữ nghĩa cần mô tả của gói. Gói được biểu diễn như sau

Hình 2–1 Ký hiệu package

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 23

## 2.2 Tổng quan về các loại quan hệ giữa các thành tố

Trong quá trình định nghĩa thành tố cần phải mô tả các mối liên hệ giữa thành tố này với các thành tố khác nên UML sử dụng một tập hợp các quan hệ. Mỗi quan hệ có một ý nghĩa xác định. Các quan hệ này bao gồm quan hệ tổng quát hóa (generalization), quan hệ kết hợp (association), quan hệ phụ thuộc (dependency).

Mỗi thành tố đều có ngữ nghĩa riêng. Để biểu diễn thành tố và quan hệ giữa các thành tố, UML sử dụng các ký hiệu riêng. Một thành tố có ký hiệu như sau

Tên thành tố
Các thuộc tính
Hình 2–2 Ký hiệu thành tố

Phần sau trình bày sơ lược các loại quan hệ. Chi tiết về các loại quan hệ giữa các thành tố được trình bày trong chương 3 phần 3.5.2.4.3.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 24

## 2.3. Quan hệ tổng quát hóa (generalization)

Generalization là quan hệ giữa một thành tố tổng quát hơn và một thành tố đặc biệt hơn. Thành tố đặc biệt hơn chứa đầy đủ các đặc điểm của thành tố tổng quát hơn và ngoài ra còn có những thông tin riêng. Quan hệ tổng quát hóa có ký hiệu như sau:

Thành tố B

Thành tố A

Hình 2–3 Ví dụ về quan hệ tổng quát hóa

### *2.3.1. Quan hệ kết hợp (association)*

Quan hệ kết hợp thể hiện liên hệ về mặt ngữ nghĩa giữa hai thành tố. Nghĩa là thành tố này có sử dụng hay nhận biết các thông tin của thành tố kia. Association có thể bao gồm hai loại con là quan hệ ngữ nghĩa thông thường (association)và quan hệ toàn thể – bộ phận (aggregation). Quan hệ ngữ nghĩa thông thường Học phần mở Sinh viên

0..*

*đăng ký*

0..*

multiplicity association (quan hệ kết hợp) Một sinh viên đăng ký nhiều học phần mở. Một học phần mở có thể được đăng ký bởi nhiều sinh viên.

Hình 2-4 Ví dụ về quan hệ kết hợp (associiatiion)

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 25

## 2.5. Quan hệ phụ thuộc (dependency)

Dpendency thể hiện sự phụ thuộc chức năng của một hay nhiều thành tố nhận vào một hay nhiều thành tố cho. Dependency kém chi tiết về mức độ ngữ nghĩa hơn quan hệ kết hợp và thường sử dụng để mô tả sự phụ thuộc lẫn nhau giữa các gói.

Tên gói nhận

Tên gói cho

Dependency Dependency mô tả phụ thuộc giữa gói nhận vào gói cho.

Hình 2-6 Ví dụ của quan hệ phụ thuộc (Dependency)

Quan hệ toàn thể – bộ phận: thành tố này chứa thành tố kia theo nghĩa vật lý.

Slider

Header

Panel

Window 2 1

+ScrollBar 1 1 +WindowTittle 1 1

+WindowClient

quan hệ toàn thể-bộ phận (aggregation) tên tham chiếu (name)

Hình 2-5 Ví dụ về quan hệ toàn thể-bộ phận (aggregatiion)

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 26

## 2.6. Tổng quan về các thành tố và cấu trúc UML metamodel

### *2.6.1. Phân loại thành tố trong UML metamodel*

Ở góc độ định nghĩa, các thành tố trong UML có thể được chia làm hai loại là thành tố trừu tượng và thành tố cụ thể. Các thành tố trừu tượng có tính tổng quát cao giữ chức năng tham gia vào định nghĩa các thành tố khác. Các thành tố cụ thể thường có quan hệ tổng quát hóa qua nhiều tầng với các thành tố trừu tượng, ngoài ra còn có các quan hệ kết hợp (association) với các thành tố khác. Chỉ các thành tố cụ thể mới có ký hiệu trong UML model và được sử dụng trong mô hình hóa.

### *2.6.2. Cấu trúc UML metamodel*

UML metamodel bao gồm ba gói (package) chính như sau

Hình 2-7 Các package chính của UML

Gói nền tảng (Foundation) là gói bao gồm phần lớn các thành tố trừu tượng và một số thành tố cụ thể mang tính chất cốt lõi. Các thành tố trong gói này được sử dụng bởi hai gói thành tố hành vi (BehavioralElements) và quản trị mô hình (ModelManagement).

BehavioralElements là gói định nghĩa các thành tố sử dụng cho việc mô tả quá trình vận động của một thành tố hay tương tác giữa các thành tố trong thế giới thực.

Model Management là gói định nghĩa các thành tố cho việc quản lý mô hình của người sử dụng.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 27

## 2.7. Package Foundation (gói nền tang)

Foundation định nghĩa những thành tố UML cơ bản. Foundation bao gồm ba gói con là gói cốt lõi (Core), gói các kiểu dữ liệu (Data Types) và gói cơ chế mở rộng (Extension Mechanism).

Hình 2-8 Gói nền tảng của UML metamodell

Core định nghĩa những thành tố cốt lõi bao gồm cả các thành tố quan hệ và đa số là ở mức trừu tượng.

Extension Mechanism định nghĩa cơ chế mở rộng cho các thành tố UML để bổ sung các thành tố mới.

Data Types định nghĩa các kiểu dữ liệu được sử dụng trong UML metamodel. Các thuộc tính của các thành tố trong UML meatamodel có kiểu dữ liệu thuộc về Data Types.

### *2.7.1. Package Core (gói cốt lõi)*

Core bao gồm các thành tố cốt lõi và được mô tả bởi năm mô hình là sườn (Backbone), quan hệ (Relationships), phụ thuộc (Dependencies), Classifiers và bổ sung (Auxiliary Elements).

Core giới thiệu cú pháp cho mô hình hóa tĩnh, không quan tâm đến quá trình vận động và tương tác giữa các đối tượng trong thế giới thực.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 28

## 2.8. Mô hình Backbone (sườn)

Backbone chủ yếu định nghĩa thành tố Classifier. Classifier là thành tố trừu tượng đóng vai trò tổng quát hóa trực tiếp của phần lớn các thành tố cụ thể khác. Ngoài ra, các thành tố cụ thể cơ bản của UML được định nghĩa trong Core bao gồm thuộc tính (attribute), phương thức (operation) và cách thực hiện phương thức (method), tham số (parameter) và ràng buộc (constraint).

Để phục vụ cho quá trình định nghĩa Classifier, UML đưa ra các thành tố trừu tượng có vai trò là tổng quát hóa (trực tiếp hay gián tiếp) của Classifier. Các thành tố này có quan hệ với nhau và có quan hệ với Classifier được mô tả trong mô hình Backbone bao gồm:

_ Element (thành tố) : Element là một thành tố trừu tượng ở mức cao nhất, tổng quát nhất trong các thành tố UML.

_ ModelElement (thành tố mô hình) : ModelElement là thành tố được định danh trong mô hình và là tổng quát hóa cấp cao nhất thứ hai cho các thành tố khác sau Element. ModelElement là thành tố được xác định qua tên (name).

_ Namespace (không gian các thành tố tham chiếu theo tên): Namespace là tập hợp các thành tố ModelElement với điều kiện định danh của một ModelElement trong một Namespace là duy nhất.

_ ElementOwnership: ElementOwnership định nghĩa tầm vực (visibility) của một thành tố chứa trong không gian các thành tố (Namspace). ElementOwnership quy định tầm vực của một thành tố được giới hạn trong Namespace (chỉ có thể được tham chiếu bởi các thành tố trong Namespace) hay vượt khỏi Namespace (có thể được tham chiếu bởi các thành tố ngoài Namespace).

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 29

_ GeneralizableElement (thành tố có thể tổng quát hóa hay đặc biệt hóa): GeneralizableElement là các thành tố có thể tham gia vào quan hệ tổng quát hóa hay đặc biệt hóa. Do đó một GeneralizableElement có thể là tổng quát hóa hay đặc biệt hóa của một GeneralizableElement khác.

_ Feature (đặc tính) : mô tả các đặc tính của một Classifier chủ yếu là tầm vực (visibility) của đặc tính. Tầm vực này xác định một đặc tính của Classifier có thể được tham chiếu bởi các Classifier khác hay chỉ được sử dụng bởi chính Classifier chứa đặc tính đó.

_ StructuralFeature (đặc tính cấu trúc) : được thừa kế từ Fearture, StructuralFeature mô tả đặc tính về mặt cấu trúc của một Classifier, mô tả cấu trúc này có thể thay đổi hay cố định qua thuộc tính *changeability của StructuralFeature. StructualFeature có một đặc biệt* hóa là thuộc tính (Attribute).

_ BehavioralFeature (đặc tính hành vi) Được kế thừa từ Feature và biểu diễn các đặc tính về mặt hành vi của một Classifier đồng thời mô tả đặc tính hành vi này có ảnh hưởng lên trạng thái của Classifier hay không qua thuộc tính isQuery. BehavioralFeature gồm hai đặc biệt hóa là phương thức (Operation) và Method.

## Chương 2
## NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 30

Classifier có khả năng chứa các thành tố khác. Các thành tố này có định danh duy nhất trong Classifier và có tầm vực cho các thành tố bên ngoài tham chiếu đến qua thuộc tính visibility trong Là đặc biệc hóa của thành tố trừu tượng ModelElement, Classifier chịu ảnh hưởng bởi các ràng buộc gắn với Classifier. Là đặc biệt hóa của GeneralizableElement, Classifier có thể tham gia vào quan hệ tổng quát hóa. GeneralizableElement là thành tố có thể tham gia vào quan hệ tổng quát hóa được mô tả rõ ràng hơn trong mô hình Relationships. Classifier bao gồm các đặc điểm cấu trúc và hành vi, cụ thể là thuộc tính (Attribute), phương thức (Operation) và Method. Ngoài ra, Mô hình Backbone còn định nghĩa các thành tố cụ thể đóng vai trò quan trọng bậc nhất là thuộc tính (Attribute), phương thức (Operation), mô tả phương thức (Method), tham số (Parameter) và ràng buộc (Constraint).

## Chương 2
## NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 31

_ Attribute (thuộc tính) : Attribute mô tả các giá trị mà một Classifier có thể sử dụng để thể hiện trạng thái. Attribute có các thuộc tính chính là tên (name), giá trị khởi đầu (initial value).

_ Operation (phương thức) : Operation là phương thức có thể được yêu cầu từ một Classifier chứa Operation để tác động lên Classifier này. Operation có quan hệ kết hợp (association) với tham số (parameter) nghĩa là Operation sử dụng một tập tham số để khởi đầu cho việc thi hành. Một Operation có thể được kế thừa từ các Operation khác.

_ Method (mô tả phương thức) : Method có quan hệ kết hợp với phương thức (Operation) mô tả cụ thể cách thức thực hiện một phương thức bao gồm các quy trình và các thuật toán. Method có tác động đến kết quả của phương thức.

_ Parameter (tham số) : Parameter là tham số có thể thay đổi, gởi và nhận. Một Parameter có thể bao gồm tên, kiểu dữ liệu và quan hệ với các thành tố khác giao tiếp với nó. Parameter được sử dụng trong mô tả phương thức (Operation), mẫu (Templates)...

_ Constraint (ràng buộc) : Constraint là các điều kiện về mặt ngữ nghĩa hay các giới hạn cho một thành tố, có thể diễn tả ở dạng văn bản hay một biểu thức logic của một ngôn ngữ mô tả ràng buộc. Ngoài việc định nghĩa thành tố ràng buộc Constraint, UML còn định nghĩa một ngôn ngữ cho mô tả ràng buộc là ngôn ngữ ràng buộc đối tượng (Object Constraint Language). Giữa các Classifier có quan hệ tổng quát hóa. Do Classifier là thành tố trừu tượng nên tất cả các thành tố thừa kế Classifier đều có tính chất này.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 32

## 2.12. Mô hình Relationships (các quan hệ)

Mô hình Relationships định nghĩa các quan hệ giữa các thành tố UML bao gồm hai loại quan hệ cơ bản là quan hệ tổng quát hóa (generalization), quan hệ kết hợp (association).

Quan hệ tổng quát hóa (generalization) là sự liên hệ giữa hai thành tố đặc biệt hơn và tổng quát hơn. Định nghĩa quan hệ kết hợp (Association) và Classifier tham gia vào mối kết hợp (Ass ociationEnd).

Hình 2–10 Mô hình Rellatiionshiips

### *2.12.1. Quan hệ tổng quát hóa (generalization)*

Một quan hệ tổng quát hóa được định nghĩa là sự liên hệ giữa hai thành tố. Thành tố đặc biệt hơn gọi là thành tố con (child) và thành tố tổng quát hơn là thành tố cha (parent). Thành tố con tham gia vào quan hệ tổng quát hóa. Thành tố cha tham gia vào  quan hệ tổng quát hóa.

Hình 2–11 Định nghĩa quan hệ tổng quát hóa trong mô hình Rellatiionshiips

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 33

## 2.13. Quan hệ kết hợp (Association)

Quan hệ kết hợp mô tả nhiều Classifier tham gia vào nhiều mối kết hợp (AssociationEnd). Association thường gặp là quan hệ kết hợp có hai mối kết hợp (AssociationEnd). Mỗi mối kết hợp gắn với một Classifier. Quan hệ kết hợp mô tả sự liên hệ về ngữ nghĩa giữa các Classifier. Một quan hệ kết hợp có tối thiểu hai mối kết hợp (AssociationEnd). Mỗi mối kết hợp có liên hệ với một thành tố Class ifier.

Hình 2–12 Định nghĩa quan hệ kết hợp trong mồ hình Rellatiionshiips

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 34

## 2.14. Lớp kết hợp (AssociationClass)

Thừa kế từ lớp và quan hệ kết hợp, lớp kết hợp vừa có tính chất của một lớp vừa có tính chất của một quan hệ kết hợp. Lớp hết hợp nối một tập các classifier với nhau và có các thuộc tính riêng đặc trưng cho quan hệ giữa các classifier này. Công việc -tiền lương Nhân sự 1..* Công ty 0..* Công việc là một associationclass. Thuộc tính tiền lương đặc trưng cho mối quan hệ giữa nhân sự và công ty.

Hình 2–13 Ví dụ lớp kết hợp

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 35

## 2.15. Mô hình Classifiers (các đặc biệt hóa cua classifiers)

Mô hình Classifiers mô tả các đặc biệt hóa của Classifier bao gồm các thành tố lớp (Class), giao diện (Interface), kiểu dữ liệu (DataType), nút (Node) và thành phần (Component) Component bao gồm nhiều thành ModelElement. Do thành tố trừu tượng bậc cao do Component cũng chứa các đặc ho a của ModelElement bao gồm thành tố đáng quan tâm là và Node bao nhiều Component. là các nút xử Component là thành đóng gói xử lý và cấp các dịch
Hình 2–14 Các lớp đặc biệt của Cllassiifiiers

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 36

## 2.16. Class (lớp)

Class là tập hợp các đối tượng có cùng các thuộc tính, hành động và ngữ nghĩa. Một Class có thể là trừu tượng (abstract)nghĩa là không có thể hiện (đối tượng)nào được tạo ra trực tiếp từ nó. Class là thành tố cụ thể có biểu diễn ký hiệu trên UML model. Là đặc biệt hóa của Classifier, Class bao gồm các thuộc tính (Attribute), phương thức (Operation) và Method. Giữa các Class có quan hệ tổng quát hóa, quan hệ kết hợp.
Kế toán viên

Thư ký
Phòng ban
Nhân viên
–Tên nhân viên
Lấy thông tin nhân viên()
Công việc thực hiện trực thuộc
Nhân viên bao gồm hai đặc biệt hóa là kiểm toán viên và thư ký. Mỗi nhân viên thực hiện các công việc và thuộc một phòng ban. tổng quát hóa (generalization) quan hệ kết hợp (association) thuộc tính phương thức
Hình 2–15 Ví dụ về lớp (Cllass) và quan hệ giữa các lớp

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 37

## 2.17. Interface (giao diện)

Iterface là tập các phương thức (operation) của một Classifier. ỗi Interface cung cấp một dịch vụ của Classifier bao gồm một nhóm các operation có quan hệ với Interface đó. Mỗi Classifier có thể cung cấp nhiều dịch vụ khác nhau qua các Interface khác nhau.

### *2.17.1. DataType (kiểu dữ liệu)*

DataType mô tả kiểu dữ liệu của người sử dụng. UML không định nghĩa các kiểu dữ liệu cụ thể. Việc định nghĩa các kiểu dữ liệu của người sử dụng tùy thuộc vào môi trường phát triển phần mềm nên thường các CASE tool đảm nhận chức năng định nghĩa các kiểu dữ liệu này.

### *2.17.2. Node (nút)*

Node là thành tố đại diện cho một tài nguyên vật lý có bộ nhớ và khả năng xử lý tính toán. Các node thường đại diện cho các máy tính và mô tả việc phân bố các máy tính trên mạng.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 38

## 2.38. Component (thành phần)

Component là một phần riêng biệt ở mức vật lý của hệ thống.

Component đóng gói các phương thức xử lý và cung cấp tập các dịch vụ xử lý này qua một tập giao diện (interface)khác nhau. Mỗi giao diện bao gồm nhiều phương thức khác nhau để phục vụ cho một mục đích cụ thể. Các phương thức có thể là các đoạn mã thi hành được, các script hay lệnh. Một component thường cung cấp nhiều loại dịch vụ khác nhau liên quan đến một đối tượng cụ thể.

DBinding

DBindingCollection

DBindingCollectionEvents

MSBind là một component nối một control của Window với một recordset. MSBind cung cấp nhiều dịch vụ, trong đó dịch vụ gắn control vào recordset là DBinding.

Giao diện (interface) component MSBind <<COM>

Hình 2–16 Ví dụ về thành phần (component) và giao diện (iinterface)

### *2.38.1. Mô hình Dependencies (các quan hệ phụ thuộc)*

Dependency mô tả sự phụ thuộc chức năng giữa hai thành phần cho và thành phần nhận. Thành phần cho đóng vai trò cung cấp dịch vụ cho thành phần nhận. Dependency định nghĩa phụ thuộc giữa hai thành tố ModelElement nên hầu như tất cả các thành tố cụ thể thừa kế ModelElement đều có thể có quan hệ phụ thuộc.

Quan hệ phụ thuộc có các đặc biệt hóa là gắn (Binding), trừu tượng hóa (Abstraction), sử dụng (Usage) và cho phép (Permisson). Quan hệ phụ thuộc là quan hệ giữa thành tố cho và thành tố nhận, cụ thể là giữa hai ModelEleme Các đặc biệt hóa của quan hệ phụ thuộc

Hình 2–17 Mô hình Dependenciies (các quan hệ phụ thuộc)

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 39

## 2.39. Binding (gắn)

Binding định nghĩa quan hệ giữa một mẫu (Template) là thành phần cho của Dependency với một thành phần được tạo từ Template đó là thành phần nhận của Dependency. Binding bao gồm các đối số phù hợp với các tham số của Template.

### *2.39.1. Abstraction (trừu tượng hóa)*

Abstraction mô tả mối liên hệ giữa các thành tố ở các mức trừu tượng hóa khác nhau. Ví dụ như chuyển một khái niệm ở mức phân tích sang mức thiết kế bằng quan hệ Abstraction.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 40

## 2.40. Usage (sư dụng)

Usage lả quan hệ giữa một thành tố ModelElement có sử dụng phương thức của một thành tố ModelElement khác.

### *2.40.1. Permisson (cho phép)*

Permisson cung cấp quyền hạn cho một thành tố ngoài không gian các thành tố (Namespace) tham chiếu các thành tố khác trong Namespace. Thành tố nhận là một ModelElement thành tố cho bắt buộc là một Namespace.

### *2.41.2. Mô hình AuxiliaryElements (các thành tố bổ sung)*

Một ModelElement có thể quan hệ với các ModelElement khác qua việc sử dụng các ModelElement khác làm tham số. Khi đó ModelElement này được go ị là một Template. Một tham số mẫu (TemplateParameter) liên hệ với một ModelElement theo ý nghĩa là Template Parameter này tham chiếu đến giá trị có kiểu là ModelElement. Thành tố biểu diễn chứa thông tin cho biểu diễn trực qua n của một thành tố ModelElement.

Hình 2–18 Mô hình AuxiilliiaryEllements (các thành phần bổ sung)

## Chương 2
## NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 41

### 2.41. TemplateParameter (tham số cho mẫu)

Tham số cho mẫu là tham số cho các thành tố Template. Ví dụ như trong một môi trường ngôn ngữ lập trình hỗ trợ Template, ta có thể xây dựng lớp mới bằng cách cung cấp các lớp tham số cho Template. TemplateParameter định nghĩa quan hệ giữa một thành tố ModelElement với các tham số (các tham số này là các thành tố ModelElement ). ModelElement là một Template khi sử dụng ít nhất một TemplateParameter.

## Chương 2
## NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 42

### 2.42. PresentationElement (thành tố biểu diễn trực quan)

PresentationElement mô tả thông tin cho việc biểu diễn các ModelElement. UML không định nghĩa cụ thể các thông tin này mà để cho các CASE tool tự do định nghĩa.

### *2.42.1. Package Extension Mechanisms (gói cơ chế mở rộng)*

***Khái quát***

Extension Mechanisms định nghĩa cách thức mở rộng ngôn ngữ UML bằng cách đưa ra cơ chế bổ sung các thành tố với ngữ nghĩa mới. Package này định nghĩa Stereotypes, Constraint (ràng buộc) và Tagged Value (thẻ giá trị) là ba cơ chế mở rộng của UML. UML cung cấp cơ chế mở rộng để thêm các thành tố mới cho các lĩnh vực đặc biệt mà UML chuẩn không định nghĩa. Các lĩnh vực cần các khái niệm mới có thể tự định nghĩa các khái niệm này qua cơ chế mở rộng UML. Việc mở rộng này không đơn giản là gắn tên Stereotypes vào thành tố và quy định ngữ nghĩa mới do đôi khi còn có các ràng buộc ngữ nghĩa trong thế giới thực. Do đó các stereotype thường chứa các ràng buộc và các giá trị thẻ. Mỗi StereoType quy định loại thành tố ModelElement mà stereotype này có thể tác động. Thành tố được tác động này là các thành tố trong UML metamodel ví dụ như Class, Association, Component... Khi gắn stereotype vào các thành tố này thì được thành tố mới thừa kế thành tố cũ và có tên của stereotype. Ví dụ như Component có các stereotype là *"document","executable","table". Các stereotype này bản chất cũng là* component nhưng "document" là một thành phần (component) chứa các sưu liệu, "executable" là thành phần chứa các dịch vụ xử lý còn "table" chứa các bảng trong một cơ sở dữ liệu.

Stereotype bao gồm ràng buộc (Constraint), các thẻ giá trị (Tagged Value) tác động lên ModelElement và cho kết quả tương tự như đặc biệt hóa. Loại thành tố mà Stereotype có thể áp dụng được xác định qua thuộc tính *baseClass, ví dụ như Class, Association...* nói chung là tên các thành tố trong UML

Hình 2–19 Mô hình cơ chế mở rộng

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 43

## 2.43. Constraint (ràng buộc)

Là các ràng buộc ngữ nghĩa được gắn với một thành tố cần mở rộng để áp đặt các điều kiện lên thành tố này và có tác dụng thay đổi hay giới hạn ngữ nghĩa. Thành tố mở rộng phải thỏa mãn các ràng buộc này để đảm bảo sự chính

xác về ngữ nghĩa. Constraint cũng có thể được gắn với stereotypes để tác động lên các thành tố có quan hệ với stereotypes này.

Là cơ chế phân loại một thành tố theo quan hệ kết hợp của thành tố này với các stereotype. Mỗi stereotype gắn một thành tố sẽ cho một thành tố mới thừa kế thành tố cũ ngoài ra có thêm các thông riêng. Stereotype chính là sự khác nhau về ngữ nghĩa của hai thành tốă cùng cấu trúc. Ví dụ như trong quy trình phát triển phần mềm Rational Unified Process, các stereotype cho thành tố Class được định nghĩa thêm trong đó có stereotype "boundary". Stereotype này là một Class đóng vai trò giao tiếp với các tương tác bên ngoài hệ thống. Mục đích của mở rộng này là phân loại các Class theo chức năng phục vụ cho quá trình phân tích.

Thiết kế giao diện người sử dụng: Giao diện người sử dụng phân tích Class với stereotype là "boundary". Class chuẩn của UML.

Hiình 2–20 Ví dụ về stereotype

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 44

## 2.44. Tagged Value (the giá trị)

Là các thuộc tính đính kèm cho một thành tố mở rộng. Tagged Value có thể chứa những thông tin bất kỳ cần thiết bổ sung cho một thành tố mới.

### *2.44.1. Các kiểu dữ liệu trong UML metamodel (Data Types)*

***Khái quát***

DataTypes định nghĩa các kiểu dữ kiệu dùng riêng trong UML metamodel nghĩa là thuộc tính của các thành tố trong UML metamodel có các kiểu dữ liệu trong Data Types. Data Types cần thiết cho sự tham khảo sâu hơn về các thuộc tính và ý nghĩa mỗi thuộc tính của một thành tố trong UML metamodel. Data Types không phải là kiểu dữ liệu của người sử dụng. Kiểu dữ liệu của người sử dụng UML được định nghĩa bởi DataType là đặc biệt hóa của Classifiers trong Core. Data Types không định nghĩa cú pháp nào cho người sử dụng.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 45

## 2.45. Các kiểu dữ liệu trong Data Types

–ActionExpression : biểu thức cho kết quả là sự thi hành một Action.
– AggregationKind : kiểu liệt kê bao gồm các giá trị none, *aggregate, composite. Các giá trị này xác định loại* Association.
– ArgListsExpression :biểu thức trả về một danh sách các đối tượng (object).
– Boolean : kiểu liệt kê bao gồm hai giá trị false và true.
– BooleanExpression : biểu thức logic trả về kiểu Boolean.
–CallConcurrencyKind : kiểu liệt kê bao gồm các giá trị *sequential*, *guarded*, *concurrent.*
– ChangeableKind : kiểu liệt kê quy định giá trị một AttributeLink hay một LinkEnd có thể thay đổi bao gồm các *none, frozen và addOnly.*
– Enumeration : định nghĩa kiểu liệt kê.
–EnumerationLiteral : định nghĩa một giá trị thuộc một kiểu liệt kê.
– Expression : biểu thức trả về một kiểu thuộc package DataType.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 46

–Integer : kiểu nguyên.
–IterationExpression : chuỗi trả về cấu trúc kiểm soát lặp.
–LocationReference : vị trí cho việc chèn một use case vào một use case khác.

–Mapping : biểu thức chuyển đổi các ModelElement.
–MappingExpression : biểu thức trả về kiểu Mapping.
–MessageDirectionKind : kiểu liệt kê bao gồm các giá trị *activation và return.*
–Muliplicity : tập các số nguyên không âm.
–MultiplicityRange : miền giá trị số nguyên không âm.
–Name : định danh cho một ModelElement.
–ObjectSetExpression : biểu thức trả về danh sách các đối tượng.
–OperationDirectionKind : kiểu liệt kê quy định một Operation là được yêu cầu hay được cung cấp bởi một Classifier bao gồm các giá trị provide và require.
– ParameterDirectionKind : kiểu liệt kê bao gồm các giá trị in, *inout, out và return.*
–Primitive : định nghĩa kiểu dữ liệu đơn.
–ProcedureExpression : biểu thức trả về một Procedure.
–ProgrammingLanguageType : kiểu dữ liệu trong một ngôn ngữ lập trình.
– PseudostateKind : kiểu liệt kê bao gồm các giá trị initial, *deepHistory, shallowHistory, join, fork, branch, junction và final.*
–ScopeKind : kiểu liệt kê bao gồm các giá trị classifier và *instance.*
–String : chuỗi văn bản.
–Structure : kiểu dữ liệu có cấu trúc.
–Time : kiểu giờ.
–TimeExpression : biểu thức kiểu Time.
–UnlimitedInteger : kiểu nguyên không giới hạn.
–Uninterpreted : kiểu không xác định.
–VisibilityKind : kiểu liệt kê bao gồm các giá trị public, *protected và private.*

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 47

## 2.47. Package Behavioural Elements (gói thành tố hành vi)

Behavioral Elements bao gồm các thành tố cùng với các cú pháp cho mô hình hóa hành vi và tương tác. BehavioralElements bao gồm năm gói là Common Behavior (hành vi tổng quát), Collaborations (mô hình cộng tác), Use

Cases (mô hình chức năng), State Machines (mô hình trạng thái) và Activity Graphs (mô hình hoạt động).

Hình 2–21 Các thành tố hành vi

CommonBehavior định nghĩa các thành tố hành vi cơ bản.

Collaboration định nghĩa các thành tố và cú pháp cho lược đồ

Collaboration và Sequence ở UML model. Collaboration mô tả tương tác giữa các thành tố để thực hiện một tác vụ cụ thể.

Use Cases bao gồm các thành tố mô hình hóa chức năng hệ thống cho từng loại người sử dụng. Use Cases giữ vai trò định nghĩa cho lược đồ Use Case ở UML model.

StateMachine bao gồm các khái niệm cho mô hình hóa quá trình thay đổi trạng thái của một thành tố. StateMachine định nghĩa lược đồ StateChart trong UML model.

Activity Graphs là dạng đặc biệt của StateMachine, được định nghĩa dựa trên StateMachine, mô tả quá trình hoạt động của một hay nhiều thành tố. Activity Graphs định nghĩa lược đồ Activity Graph trong UML model.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 48

## 2.48. Package Common Behavior (gói hành vi tổng quát)

Common Behavior định nghĩa các thành tố cơ bản cho mô hình hóa tương tác. Common Behiavior được mô tả bằng các mô hình Signals(tín hiệu), Actions (tác động), Instances and Links (thể hiện và liên kết).

### *2.48.1. Mô hình Signals (tín hiệu)*

Mô hình này chủ yếu định nghĩa thành tố Signal (tín hiệu). Tín hiệu được tạo ra từ một hành vi (BehavioralFeature) của classifier này và gủi đến một thành tố nhận tín hiệu (Reception) của một classifier khác. Tín hiệu đươc tạo ra tập các hành vi của một Do thừa kế từ (BehavioralElement là thành hành vi của một classifier) Reception cũng là một thành hành vi của classifier. định nghĩa hành vi xử lý tín của một tín hiệu được gửi đến thành tố xử lý tín hiệu của một

Hình 2–22 Mô hình Siignall định nghĩa thành tố Siignall (tín hiệu)

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 49

## 2.49. Reception (thành tố nhận tín hiệu)

Reception là thành tố nhận tín hiệu từ một classifier và giữ chức năng mô tả các tác động (bằng chuỗi văn bản) của tín hiệu đến classifier nhận.

### *2.49.1. Signal (tín hiệu)*

Signal là các tác tương tác không đồng bộ giữa các classifier và là thành tố độc lập với các classifier. Signal được tạo ra do các hành vi (BehvioralFeature) của các classifier này và gửi đến các classifier khác. Do BehavioralFeature là thành tố hành vi trừu tượng nên tất cả các thành tố hành vi thừa kế BehavioralFeature như các phương thức (operation) đều có thể tạo và gửi các signal.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 50

## 2.50. Exception (lỗi biệt lệ)

Thừa kế signal, exception là tín hiệu được gửi đi khi có một lỗi trong quá trình thi hành một hành vi.

### *2.50.1. Mô hình Actions (tác động)*

Action được mô tả bằng mô hình Actions của UML metamodel.

Action là các chỉ thị thi hành có gây ảnh hưởng đến trạng thái của hệ thống các đặc biệt hóa của một tác động (Action) ActionSequence là một tác động thức bao gồm các tác động con theo trình tự xác định. Acti on bao gồm một tập đối số.

Hình 2–23 Mô hình định nghĩa hành động (Actiions)

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 51

## 2.51. Argument (đối số)

Argument là đối số cho một Action, cung cấp giá trị tham số cho một Action.

### *2.51.1. Action (tác động)*

Action là các chỉ thị thi hành trong một quy trình tính toán có ảnh hưởng đến trạng thái của hệ thống. Action bao gồm các đặc biệt hóa sau

- AssignmentAction : gán cho thuộc tính một giá trị mới.
- CallAction : kích hoạt một hành động.
- CreateAction : tạo thể hiện của một Classifier.
- DestroyAction : hủy một đối tượng
- ReturnAction : trả về giá trị cho đối tượng gọi.
- SendAction : gởi tín hiệu (Signal ).
- TerminateAction : hành động tự hủy của một đối tượng.
- UninterpretedAction : hành động không xác định.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 52

## 2.52. ActionSequence (tác động phức)

ActionSequence là một tác động chứa một tập các Action con theo thứ tự xác định.

### *2.52.1. Mô hình Instances and Links (thể hiện và liên kết)*

Instance và Link được định nghĩa trong mô hình Instances and Links của UML metamodel. Instance định nghĩa thể hiện của một classifier, các thể hiện này liên kết với nhau qua các Link.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 53

Liên kết (Link) bao gồm nhiều mối liên kết (LinkEnd) Mỗi liên kết (LinkEnd) là thể hiện của mối kết hợp (Association), được xác định bởi mối kết hợp. Instance là thể hiện của Classifier, được xác định bởi Classifier. Các Instance tương tác với nhau qua các tác nhân (Stimulus). gắn với các Link. Liên kết (Link) là thể hiện của quan hệ kết hợp (Association), được xác định bởi quan hệ kết hợp. Các Instance liên hệ với nhau qua các Link.

Hình 2–24 Mô hình các thực thể và các liên kết của UML

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 54

## 2.54. Stimulus (tác nhân)

Stimulus là một giao tiếp cụ thể giữa hai instance qua một liên kết (link) được gửi đi bởi sự thi hành một Action, có thể là một tín hiệu gửi đến instance hay việc kích hoạt một phương thức. Hay nói cách khác, các thể hiện tương tác với nhau qua stimulus.

### *2.54.2. AttributeLink (thẻ thuộc tính)*

AttributeLink chứa tập các giá trị của Attribute trong một instance do đó thể hiện trạng thái của instance.

### *2.54.2. LinkEnd (mối liên kết)*

LinkEnd là các mối liên kết của một Link. Mỗi LinkEnd tương ứng với một mối kết hợp (AssociationEnd) để xác định LinkEnd. LinkEnd trong UML metamodel là thể hiện của AssociationEnd trong UML metamodel. Các instance liên kết với nhau qua các LinkEnd.

### *2.54.3. Link (liên kết)*

Link là liên kết giữa các instance. Link tương ứng với một Association có vai trò xác định Link. Link trong UML metamodel là thể hiện Association trong UML metamodel. Link chỉ có vai trò liên kết các instance với nhau, các tác động giữa các instance thực hiện qua các stimulus. Một Link có thể tương ứng với nhiều stimulus.

### *2.54.4. Instance (thể hiện)*

Instance là thể hiện của một Classifier. Instance là một thực thể có các thông tin về trạng thái và chịu tác động của các phương thức (Operation) để thay đổi trạng thái. Instance được xác định cấu trúc và hành động qua Classifier có quan hệ kết hợp với nó. Instance có các đặc biệt hóa sau

– DataValue (giá trị dữ liệu): giá trị của một thuộc tính là thể hiện của kiểu dữ liệu của thuộc tính đó.

– ComponentInstance (thể hiện thành phần): thể hiện của một Component.

–NodeInstance (thể hiện nút) : thể hiện của một Node.
–Object (đối tượng) : thể hiện của một Class.

## Chương 2
## NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 55

*2.55. Package Collaborations (gói cộng tác)*

Collaborations định nghĩa thành tố Collaboration và cú pháp cho mô hình hóa tương tác giữa các thành tố để thi hành một tác vụ cụ thể. Mô hình cộng tác mô tả quan hệ giữa instance và link thông qua các thông điệp (message).
Thành tố Collaboration được định nghĩa la ăquá trình trao đổi các thông điệp (message) giữa các vai (role) của các thành tố trong Collaboration. Một thành tố classifier có thể tham gia vào Collaboration thông qua nhiều vai của thành tố đó.

## Chương 2
## NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 56

Classifier tham gia vào Collaboration qua nhiều vai (ClassifierRole). Các ClassifierRole trong Collaboration có liên hệ với nhau qua vai trò các mối liên kết (AssociationRole) tương ứng với các mối kết hợp của Classifier. Các ClassifierRo le trong Collaboration tương tác với nhau qua cơ chế trao đổi thông điệp (message). Một message xác định classifier gửi và calssifier nhận. Một thông điệp trong Colaboration tương ứng với một tác động tạo nên ăthông điệp.
Hình 2–25 Mô hình Collllaboratiions trong UML

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 57

## 2.57. AssociationEndRole (vai cua mối kết hợp)

AssociationEndRole là mối liên kết của một vai Association (AssociationRole) trong Collaboration. Một Association có thể tham gia nhiều vai vào Collaboration nên cũng có nhiều vai của mối liên kết tương ứng.

### *2.57.1. AssociationRole (vai của quan hệ kết hợp)*

Một Association có thể tham gia vào Collaboration qua nhiều AssociationRole. AssociationRole là một vai của Association trong Collaboration.

### *2.57.2. ClassifierRole (vai của Classifier)*

Một Collaboration bao gồm các ClassifierRole và các liên kết giữa các ClassifierRole này. Một Classifier có thể tham gia vào Collaboration qua nhiều ClassifierRole.

### *2.57.3. Collaboration (cộng tác)*

Collaboration mô tả quá trình tương tác giữa các thành tố classifier (qua các ClassifierRole của nó) để thực hiện một phương thức (operation). Các classifier này trao đổi các thông điệp với nhau theo một thứ tự xác định. Mỗi thông điệp gây ra một tác động lên thành tố classifier nhận. Collaboration chính là mô tả cho cơ chế vận hành của một hệ thống

Giao diện sửa đổi thông tin sinh viên : Sinh viên UI
Xử lý sửa đổi thông tin sinh viên : Sinh viên Control
Dữ liệu sinh viên : Sinh viên Data
Xử lý sửa đổi thông tin sinh viên : Sinh viên Control
1: sửa đổi thông tin sinh viên
4: chọn sinh viên và sửa đổi
5: cập nhật thông tin đã sửa đổi
2: hiển thi các sinh viên
6: cập nhật thông tin sinh viên

3: lấy dữ liệu sinh viên

7: cập nhật dữ liệu lên cơ sở dữ liệu

Hai vai của cùng một classifier ” Sinh viên Control” tham gia vào Collaboration. AssociationRole thông điệp với classifier gửi và nhận.

Hình 2–26 Ví dụ về mô hình cộng tác

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 58

## 2.58. Message (thông điệp)

Message là thông điệp giữa các ClassifierRole. Message định nghĩa vai trò của ClassifierRole gửi và ClassifierRole nhận cùng với một tác động (Action) lên Classifier nhận thông điệp.

### *2.58.1. Package Use Cases (gói Use Cases)*

Use Cases định nghĩa các thành tố và cú pháp cho mô hình hóa chức năng của hệ thống cung cấp cho từng loại người sử dụng. Các thành tố trong Use Cases chủ yếu để mô tả các hành vi trong hệ thống trong khi vẫn chưa xác định cấu trúc hệ thống.

Use Cases định nghĩa hai thành tố cơ bản là Actor và UseCase. Actor là các tác nhân bên ngoài hệ thống có tương tác với hệ thống (có thể là người sử dụng hay một hệ thống khác). Trong khi đó, mỗi UseCase mô tả một chức năng của hệ thống và chức năng này được kích hoạt khi có tác động của Actor lên UseCase. Association nối Actor với các UseCase cho biết các chức năng mà hệ thống cung cấp cho Actor UseCase là một classifier chứa một dãy các hành vi được th ực hiện theo thứ tự khi có tác động kích hoạt từ tác nhân (actor). Hai loại quan hệ giữa hai UseCase.

Hình 2–27 Mô hình định nghĩa Use Case trong UML

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 59

## 2.59. Actor (tác nhân)

Actor là tác nhân tác động lên hệ thống thông qua các Use Case của hệ thống. Một Actor có thể tương tác với nhiều Use Case khác nhau. ExtensionPoint là vị trí trong một UseCase có thể mở rộng hành vi bằng cách chèn thêm các hành vi cho Use Case đó.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 60

## 2.60. Extend (mơ rộng)

Extend định nghĩa quan hệ giữa các Use Case với ý nghĩa một Use Case này có thể được mở rộng thêm các hành vi trong Use Case khác bằng cách thêm vào mô tả của Use Case gốc các biểu thức và điều kiện mở rộng tại những vị trí xác định trong một dãy các hành vi của Use Case.

### *2.60.1. Include (bao gồm)*

Include định nghĩa quan hệ giữa các Use Case với ý nghĩa UseCase này sử dụng các hành vi được định nghĩa trong UseCase khác.

### *2.60.2. UseCase*

UseCase là các chức năng của hệ thống dưới góc độ của người sử dụng. UseCase không quan tâm đến cấu trúc bên trong của hệ thống. Mỗi UseCase bao gồm một dãy các hành động nguyên tố có thứ tự (không bị ngắt bởi các hành động nào khác) mà hệ thống sẽ thực hiện khi có tương tác của Actor lên UseCase đó. Mỗi UseCase có thể được mô tả rõ hơn thông qua một

Collaboration chứa các thành phần của hệ thống và tương tác giữa chúng để thực hiện Use Case.

Quản lý sinh viên

Phòng đào tạo

Đăng nhập hệ thống <<include> tác nhân (actor) UseCase UseCase Quản lý sinh viên bao gồm một dãy các hành động mà hệ thống sẽ thực hiện để quản lý sinh viên.

Hình 2–28 Ví dụ Use Case

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 61

## 2.61. Package State Machines (gói mô hình trạng thái)

State Machines cung cấp các thành tố và cú pháp cho mô hình hóa hoạt động và biến đổi trạng thái của một thành tố trong chu kỳ sống dưới các sự kiện tác động lên nó. Một một thành tố (như các Classifiers, Operation, Attribute...) có thể có một quá trình biến đổi trạng thái. Quá trình này được định nghĩa bởi thành tố StateMachine.

Thành tố StateMachine được định nghĩa trong gói này bao gồm thành tố cơ bản là State và Transition. State mô tả trạng thái một thành tố. Transition bao gồm nhiều loại khác nhau mô tả sự chuyển trạng thái sang một trạng thái mới qua tác động của một sự kiện (event). State Machines bao gồm hai mô hình là Main (chủ yếu) và Events (sự kiện). Main mô tả trạng thái (State) và chuyển trạng thái (Transition). Events mô tả các sự kiện tác động lên một thành tố.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 62

Thành tố StateMachine bao gồm trạng thái (State) và chuyển trạng thái (Transition). Trạng thái phức (CompositeState) bao gồm nhiều trạng thái đơn (State). Transition là sự chuyển trạng thái này sang trạng thái khác. Trạng thái có thể chứa các tác động (action) được thực hiện khi vào trạng thái, giữ trạng thái hay kết thúc trạng thái.

Hiình 2–29 Mô hình chính định nghĩa trạng thái và sự chuyển trạng thái

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 63

## 2.63. StateVertex (điểm trạng thái)

StateVertex là điểm biểu diễn trực quan một trạng thái.

## 2.63. State (trạng thái)

State mô tả trạng thái của thành tố (classifier, operation, use case...) ngoài ra còn mô tả một số các Action sẽ được thi hành khi thành tố bắt đầu trạng thái, nắm giữ trạng thái và kết thúc trạng thái cùng một số các sự kiện (Event) được treo không cho tác động lên thành tố. Vẽ hình entry/ thiết lập Device do/ Vẽ trên Device exit/ hủy bỏ Device. Một State cùng với các hành động được thực hiện khi bắt đầu, nắm giữ và kết thúc trạng thái.

Hiình 2–30 Ví dụ trạng thái (State)

State có đặc biệt hóa là FinalState. FinalState là trạng thái kết thúc.

### *2.63.1. PseudoState (trạng thái giả)*

PseudoState định nghĩa các loại State mở rộng cho StateMachine. PseudoState bao gồm các loại sau

–initial : trạng thái khởi đầu để thành tố chuyển sang trạng thái mặc định.

– deepHistory : lấy lại State đã được giữ trước đó ở bất cứ State con nào của CompositeState chứa deepHistory.

– shallowHistory : lấy lại State đã được giữ trước đó ở cùng mức với shalowHistory trong CompositeState.

–join : nhóm các Transition từ nhiều State khác nhau.

–fork : tách Transition thành nhiều Transition đến các State khác nhau.

–junction : tách một Transition thành các Transition có Guard để phân nhánh theo điều kiện.

Trạng thái mặc định

CompositeState1

CompositeState2

C1

C2 H

B1

CompositeState2

C1

C2

B1

Tạm dừng

Một PseudoState loại shalowHistory trong CompositeState1 Nếu sự kiện "Tạm dừng" xảy ra lần đầu và vào lúc trạng thái là B1, cả C1 và C2 đều chưa được kích hoạt thì H đại diện cho trạng thái mặc định là C2. Nếu không H đại diện cho C1 hoặc C2 tuỳ theo trạng thái nào đã được kích hoạt gần nhất.

Tạm dừng

Tiếp tục

Hình 2–31 Ví dụ trạng thái giả

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 64

## 2.64. Transition (chuyển trạng thái)

Transition định nghĩa quan hệ chuyển đổi trạng thái giữa một StateVertex nguồn và một StateVertex đích. Một Transition có thể gắn với các chắn (Guard). Guard là biểu thức logic gắn với mỗi Transition. Khi một sự kiện tác động lên thành tố gắn với StateMachine, Transition sẽ được kích hoạt để thành tố này chuẩn bị chuyển trạng thái. Đối với các Transition có Guard thì biểu thức của Guard sẽ được lượng giá. Sự chuyển trạng thái chỉ xảy ra khi biểu thức này có giá trị là true, ngược lại sẽ bị hủy bỏ.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 65

## 2.65. CompositeState (trạng thái phức)

Composite State đại diện cho trạng thái phức bao gồm nhiều điểm trạng thái (StateVertex ). Do mỗi điểm trạng thái biểu diễn một trạng thái đơn nên CompositeState cũng bao gồm nhiều trạng thái đơn. CompositeState cũng có thể bao gồm các CompositeState khác.
Khởi tạo đăng ký học phần.
Chấm dứt đăng ký học phần
Đăng ký học phần
Còn chỗ
Hết chỗ

Còn chỗ
Hết chỗ
Thêm sinh viên[
Đếm < 10 ]
[ Đếm = 10 ] ^Thông báo.Tạo thông báo Thêm sinh viên / Khởi tạo
Đếm = 0
Phòng đào tạo hủy học phần trạng thái khởi đầu trạng thái kết thúc Một trạng thái phức bao gồm hai trạng thái đơn. Nếu số sinh viên đăng ký cho học phần < 10 thì tiếp tục cho đăng ký. Nếu số sinh viên = 10 thì thông báo hết chỗ và chấm dứt đăng ký học phần. Nếu đang đăng ký mà phòng đào tạo hủy học phần thì cũng chấm dứt việc đăng ký.
Hình 2–32 Ví dụ về trạng thái phức //ghép

### 2.65.1. StateMachine

StateMachine là thành tố mô tả trạng thái và sự biến đổi trạng thái qua các sự kiện cho thành tố gắn với StateMachine. StateMachine bao gồm hai thành tố cơ bản là trạng thái (State) và chuyển trạng thái (Transition).

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 66

## 2.66. Mô hình Events (sự kiện)

Event là các sự kiện có thể nhận biết, Event bao gồm một tập tham số Parameter). các đặc biệt hóa của Events
Hình 2–33 Mô hình Event định nghĩa sự kiện

Event là các sự kiện tác động lên một ModelElement bao gồm các đặc biệt hóa sau

–Call Event : một Operation được kích hoạt.
–ChangeEvent : một biểu thức logic chuyển sang giá trị true.
–SignalEvent : một Signal được gửi đi.
–TimeEvent : kết thúc một giới hạn thời gian.

### *2.66.1. Package Activity Graphs (gói lược đồ hoạt động)*

Activity Graphs là đặc biệt hoá từ StateMachine. Activity Graphs mô tả các thành tố và cú pháp cho mô hình hóa hành động và sự chuyển đổi hành động giữa các thành tố với nhau để thực hiện một tác vụ. Activity Graphs tập trung vào trình tự và điều kiện xảy ra của các hành động, sự phân chia hành động cho mỗi thành tố liên quan. các loại trạng thái đặc biệt hóa của State sử dụng riêng cho ActivityGraph. ActivityGraph phân loại các trạng thái (ActionState) vào các phần (Parttion). Mỗi Parttion đảm nhận một vai trò riêng trong ActivityGraph.
Hình 2–34 Mô hình Actiiviity Graphs định nghĩa thành tố ActiiviityGraph

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 67

## 2.67. ActionState (trạng thái hoạt động)

ActionState kế thừa SimpleState từ StateMachine đại diện cho việc thi hành một hành động.

### *2.67.1. ActivityGraph (đồ thị hoạt động)*

ActivityGraph mô tả các xử lý của các thành tố liên quan, thành tố nào đảm nhận các xử lý nào và sự chuyển giao xử lý giữa các thành tố với nhau. ActivityGraph được gắn với một classifier (bao gồm UseCase) và có thể được sử dụng để mô hình hóa quy trình nghiệp vụ trong một tổ chức.
Yêu cầu mua hàng
Thanh toán
Nhận hàng
Nhận đặt hàng
Giao hàng
ActionState Cung cấp hàng
Partition Kho hàng
Kho hàng : Bộ phận bán hang : Khách hàng :
Hình 2–35 Actiiviiviity Graph

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 68

## 2.68. ObjectFlowState (trạng thái đối tượng luân chuyển)

ObjectFlowState mô tả sự trao đổi dữ liệu về các đối tượng giữa các trạng thái hoạt động trong ActivityGraph. ObjectFlowState chứa thông tin về một classifier đang ở một trạng thái xác định và đóng vai trò trao đổi thông tin giữa các ActionState.

### *2.68.1. Partition (vùng)*

Parttion phân chia các State trong ActivityGraph thành các nhóm và thường tương ứng với các đơn vị cấu trúc trong một tổ chức. Partition có thể được sử dụng để phân loại trạng thái hay các tài nguyên hệ thống có liên quan đến một nhóm các trạng thái

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 69

## 2.69. Package Model Management (gói quan trị mô hình)

Model Management bao gồm các thành tố có chức năng tập hợp các thành tố khác thành nhóm. Các thành tố được định nghĩa trong Model Management có ý nghĩa logic trong hệ thống được mô hình hóa. Chức năng của chúng là phân chia hệ thống thành các phần con giúp cho việc quản lý mô hình cũng như mô hình hóa thuận lợi hơn. Package có thể tham gia vào quan hệ tổng quát hóa các

đặc biệt hóa của package. Package chứa các thành tố với chỉ danh và tầm vực cho các thành tố ngoài package tham chiếu.

Hình 2–36 Mô hình của các thành tố quản trị mô hình..

*2.69.1. Elementimport*

ElementImport định nghĩa chỉ danh và tầm vực của một ModelElement khi được đưa vào một Package. Một ModelElement trong package sẽ có chỉ danh có thể khác với định danh của ModelElement đó.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 70

## 2.70. Model (mô hình)

Mô hình là trừu tượng hóa của một hệ thống cho một mục đích cụ thể. Mục đích này giúp xác định những thành phần nào của hệ thống sẽ được phản ánh ở mô hình, không phải tất cả các thành phần hệ thống đều tồn tại trong một mô hình. Do đó mô hình sẽ thể hiện hệ thống trên góc độ phù hợp với mục đích này.

*2.70.1. Package (gói)*

Package là tập hợp logic các thành tố do đó Package không đóng vai trò gì trong quá trình mã hóa phần mềm. Tuy nhiên Package giúp quản lý mô hình với chức năng chứa các thành tố và các Package con. Các thành phần trong một Package có tầm vực được xác định qua thuộc tính visibility cho các thành phần ngoài Package tham chiếu đến. Quan hệ "access" (là một stereotype của quan hệ Permission)giữa hai Package mang ý nghĩa các thành tố có visibility là "public" trong Package cho có thể tham chiếu bởi các thành tố trong Package nhận.

Quan hệ "import" (là một stereotype của quan hệ Permission)giữa hai Package mang ý nghĩa các thành tố có visibility là "public" trong Package cho được thêm vào trong Package nhận với các chỉ danh khác (để tránh trùng tên với chỉ danh của các thành tố có sẵn) và có thể được tham chiếu bởi các thành tố có sẵn qua chỉ danh này.

*2.70.2. Subsystem (hệ thống con)*

Subsystem là nhóm các thành tố mô tả hành động của một hệ thống. Subsystem do đó bao gồm các phương thức (Operation) và cung cấp các giao diện (Interface). Các thành phần trong Subsystem có thể chia làm hai loại là các thành phần mô tả hành động và các thành phần thực thi hành động.

# Chương 2
# NGỮ NGHĨA VÀ CÚ PHÁP CÁC THÀNH TỐ TRONG UML (UML Semantic) 71

## 2.71. Tóm tắt

Chương hai trình bày những thành tố cơ bản và cấu trúc UML, cùng với việc phân tích các gói thành tố của ngôn ngữ như: package nền tảng, package hành vi, package quản trị mô hình. Một số khái niệm và cú pháp ngữ pháp khá trừu tượng, trong chương tới sẽ trình bày những ký hiệu trực quan của các thành tố trên trong việc mô hình hoá các hệ thống thông qua các loại lược đồ.

# Chương 3
# HỆ THỐNG KÝ HIỆU (UML Notation) 72

## 3.1. Giới thiệu

Khi mô hình hóa một hệ thống, chúng ta cần xây dựng các loại mô hình khác nhau, trong UML các thông tin trong mô hình thường được biểu diễn bằng các ký hiệu đồ họa cùng với các kết nối thể hiện mối quan hệ giữa các ký hiệu này. Tập các ký hiệu đồ họa và các liên kết giữa chúng dùng để mô tả cho một chức năng, thành phần hoặc một thể hiện của hệ thống được tổ chức thành một lược đồ (diagram), các lược đồ sẽ giúp trực quan hóa các mô hình của hệ thống

cần được xây dựng, mỗi mô hình sẽ được biểu diễn bằng nhiều loại lược đồ và mỗi loại lược đồ có thể được sử dụng trong nhiều loại mô hình khác nhau.

Trong chương này sẽ trình bày hệ thống các ký hiệu của UML, các ký hiệu này được phân loại theo các lược đồ. Mỗi lược đồ sẽ có hệ thống các ký hiệu riêng và cũng có một số ký hiệu chung sử dụng trong nhiều loại lược đồ khác nhau. Trong UML định nghĩa chín loại lược đồ:

_ Lược đồ Use Case (Use case diagram)
_ Lược đồ lớp (Class Diagram)
_ Lược đồ đối tượng (Object Diagram)
_ Lược đồ tuần tự (Sepuence Diagram)
_ Lược đồ cộng tác (Collaboration Diagram)
_ Lược đồ trạng thái (State Diagram)
_ Lược đồ hoạt động (Activity Diagram)
_ Lược đồ thành phần (Component Diagram)
_ Lược đồ triển khai (Deployment Diagram)

# Chương 3
# HỆ THỐNG KÝ HIỆU (UML Notation) 73

## 3.2. Các thành phần cơ ban cua lược đồ

### *3.2.1. Đồ thị và nội dung (Graphs and their Contents)*

Hầu hết các lược đồ trong UML đều ở dạng đồ thị, các thông tin được tổ chức dưới dạng hình học topô, không bị ảnh hưởng bởi sự thay đổi về hình dáng và kích thước. Các ký hiệu trong UML thường ở dạng hai chiều, một số là hình chiếu của các ký hiệu ba chiều (các khối lập phương), nhưng thường được biểu diễn dưới dạng các biểu tượng trên bề mặt hai chiều. Có bốn loại ký hiệu đồ họa cơ bản trong hệ thống ký hiệu của UML :

_ Các biểu tượng (Icons)– là một hình không thể thay đổi về hình dạng cũng như kích thước, Icon không thể mở rộng để chứa nội dung bên trong.
_ Các ký hiệu hai chiều (2-d sumbols)– có kích cỡ khác nhau, có thể mở rộng để chứa các thứ khác như danh sách các chuỗi hoặc ngay cả các ký hiệu khác bên trong, chúng cũng có thể được chia làm nhiều phần. Loại ký hiệu này khi kéo

hoặc xóa sẽ làm ảnh hưởng đến nội dung bên trong cũng như các đường dẫn liên kết với nó.

_ Các đường dẫn (Paths)–là một tập tuần tự các đoạn thẳng, một đường dẫn phải luôn được gắn với các ký hiệu đồ họa khác ở hai đầu. Đường dẫn có thể có các terminator, đó là các biểu tượng ở cuối đường dẫn làm rõ nghĩa cho các đường dẫn này.

_ Các chuỗi (Strings)–có thể tồn tại như là một thành tố độc lập trong các lược đồ hoặc là một phần của các thành tố khác trong lược đồ, hoặc là một nhãn gắn vào các ký hiệu hay các đường dẫn.

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 74

## 3.3. Các đường dẫn (Drawing Paths)

Đường dẫn là một tập các đoạn thẳng có các điểm cuối khớp với nhau, đây là một đơn vị hình học topô đơn. Các đoạn thẳng có thể là các đường trực giao, đường xiên hoặc đường cong. Trong một số quan hệ (ví dụ như kết hợp hoặc tổng quát hóa), vài đường dẫn cùng loại có thể liên kết đến cùng một ký hiệu đơn. Trong một số trường hợp các đường dẫn có thể kết hợp lại hoặc phân nhánh thành nhiều đường như tổ chức của một cây.

### *3.3.1. Các liên kết ẩn và vai trò của công cụ*

Một ký hiệu viết trên giấy thì không thể chứa các thông tin ẩn, ký hiệu trên màn hình có thể chứa các liên kết đến các thông tin ẩn không thể hiển thị trong view tĩnh, muốn thể hiện các thông tin ẩn này phải nhờ đến các công cụ hỗ trợ. Các ký hiệu trình bày dưới đây được định nghĩa như là những ký hiệu tĩnh.

### *3.3.2. Thông tin nền (Background Information)*

Các thể hiện của ký hiệu lớp trong các lược đồ khác nhau có thể khác nhau, ví dụ như ký hiệu lớp chỉ có phần tên lớp hoặc có chứa thêm phần thuộc tính hoặc hàm tuỳ theo yêu cầu của từng lược đồ. Không phải tất cả các thông tin đề ở dạng ký hiệu đồ hoạ, một số thể hiện tốt nhất dưới dạng văn bản, UML không giả thiết tất cả các thông tin trong mô hình sẽ được thể hiện trong lược đồ, một số có thể chỉ được biểu diễn dưới dạng bảng.

*3.3.3. Chuỗi (String), tên (Name),nhãn (Label) và từ khóa*

(Keywords) Chuỗi là một dãy các ký tự dùng để hiển thị một thông tin nào đó trong mô hình. Font và kích cỡ của chuỗi có thể phụ thuộc vào thuộc tính của mô hình, một chuỗi có thể ở dạng dòng đơn hay một phân đoạn.

**Chương 3**

**HỆ THỐNG KÝ HIỆU (UML Notation) 75**

Tên là một chuỗi dùng để định nghĩa cho một thành tố mô hình duy nhất. Một pathname (tên đường dẫn), gồm một tập các tên được nối với nhau bởi một ký hiệu đặc biệt (ví dụ như dấu "::"), được dùng để tìm một thành tố mô hình bắt đầu từ gốc của hệ thống. Tên được thể hiện như là một chuỗi văn bản, thông thường nằm trên một dòng và thường không chứa các ký tự đặc biệt.Thông thường độ dài của tên và tập các ký tự dùng để đặt tên được định nghĩa bởi công cụ và ngôn ngữ hiển thị.

Ví dụ : Tên – Nguyễn Văn A, abstract, interger Pathname – ngân hàngA::ChinhánhB::PhòngC Nhãn (label) cũng là một chuỗi ký tự được gắn vào ký hiệu đồ họa.

Từ then chốt (hay từ khóa – keyword) thường dùng để phân biệt giữa những nhóm, những chủ đề chung. Ký hiệu của một keyword thường là một chuỗi được đặt trong hai dấu¶™ "Ví du ¸ :extend ™™¸include™™ ...

## 3.4. Biểu thức (Expression)

Biểu thức được sử dụng trong một số cấu trúc của UML, đặc biệt là trong ngôn ngữ mô tả ràng buộc dùng để định nghĩa UML. Hàm được thể hiện như là một chuỗi ký tự và cú pháp của chuỗi được định nghĩa tùy thuộc vào công cụ hỗ trợ và công cụ phân tích cho ngôn ngữ.

Ví dụ : Côngty.Giámđốc.Lương > Côngty.Nhânviên.Lương

*3.4.1. Ghi Chú (Note)*

Là một ký hiệu đồ hoạ có chứa thông tin dưới dạng văn bản hoặc chứa cả hình ảnh. Note có thể thể hiện nhiều loại thông tin khác nhau như: ràng buộc, chú thích, thân của các hàm, các giá trị đính kèm...

Hình 3–1 Note dùng để ghi chú cho mô hình

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 76

### 3.5. Sự tương quan giữa các loại thành tố và thể hiện cua nó

Nhiều khái niệm mô hình hóa trong UML có hai thể hiện tương tự nhau đi thành một cặp thành tố và thể hiện, ví dụ như : Lớp và đối tượng, quan hệ kết hợp và liên kết, tham số và giá trị...

Hình 3–2 Lớp Poiint và hai đối tượng Poiint

Role (vai trò) trong một cộng tác (collaboration) cũng là một cặp giữa thành tố và thể hiện. Một role trong một collaboration được thể hiện bằng một tên, dấu hai chấm và kiểu, một thể hiện được biểu diễn bằng tên, dấu "/", danh sách vai trò, dấu hai chấm và danh sách kiểu. Ví dụ : Các vai trò Các đối tượng một collaboration

Hình 3–3 Các vai trò và các đối tượng điểm

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 77

### 3.6. Các thành phần quan trị mô hình (model management)

#### *3.6.1. Gói (Package)*

***Ngữ nghĩa***

Một gói(package) là một nhóm các thành tố mô hình, bản thân package cũng có thể chứa các package con. Tất cả các thành tố mô hình đều có thể tổ chức dưới dạng package.

Lưu ý rằng package có thể sở hữu các thành tố, các mảng mô hình, những thành phần cơ bản để điều khiển cấu hình, để lưu trữ, để điều khiển các truy cập. Mỗi thành tố có thể được sở hữu trực tiếp bởi một package đơn, vì vậy cây kế thừa của package là một cậy kế thừa nghiêm ngặt. Một package có thể tham chiếu đến một package khác bằng cách dùng các khuôn mẫu của quan hệ phụ thuộc như import™™ access™™ hoặc các quan hệ khác như tinh chế hay tổng quát hóa.

***Ký hiệu***

Một package được biểu diễn bằng một hình chữ nhật lớn với một hình chữ nhật nhỏ gắn ở góc trên trái, giống như biểu tượng của một thư mục.

_ Nếu nội dung của package không thể hiện trong hình chữ nhật lớn thì tên của package được đặt bên trong hình chữ nhật lớn. (Cách thể hiện A).

_ Nếu bên trong hình chữ nhật lớn có chứa nội dung thì tên của package được đặt bên trong hình chữ nhật nhỏ. (Cách thể hiện B).

***Cách thể hiện A Cách thể hiện B***

Name Tên package

Hình 3-4 Cách thể hiện tên của Packages

Packages Editor chứa các Package phụ thuộc các quan hệ

Hình 3-5 Package và các quan hệ

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 78

### 3.7. Các thành phần quan trị mô hình (model management)

Hình 3-6 Nội dung của package Ediitor thể hiện dưới dạng Cây

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 79

## 3.8. Subsystem

### *3.8.1. Ngữ nghĩa*

Một subsystem (hệ thống con) miêu tả một đơn vị hành vi của một vấn đề trong hệ thống hay nói các khác là trong một mô hình. Đặc tả của subsystem bao gồm các thao tách hay các hàm của subsystem, các thành tố đặc tả liên quan đến subsystem như use case, state machines...

### *3.8.2. Ký hiệu*

Ký hiệu của một subsystem tương tự như của package, gồm một hình chữ nhật lớn và một hình chữ nhật nhỏ gắn vào góc trên trái của hình chữ nhật lớn, đồng thời có thêm một ký hiệu hình cái nĩa gắn vào góc trên phải của hình chữ nhật chứa tên subsystem.

Hình 3–7 Một thể hiện đơn giản của subsystem

Một subsytem cũng có hai cách thể hiện tên giống như package. Bên cạnh đó ta cũng có thể dùng ký hiệu của package để thay thế cho subsystem bằng cách thêm vào từ khóasubsystem™™ vào phía trên tên của
subsystem. Hình chữ nhật lớn của subsystem co thể được chia làm ba phần :

_ Operation (thao tác)

_ Specification Element (thành tố đặc tả/mô tả)

_ Realization element (thành tố thực thi/ dùng để định rõ các đặc tả tương ứng)

Tuỳ trường hợp sử dụng mà những phần này có thể hiển thị hay không, có thể hiển thị cả ba phần hay từng phần riêng, giữa các phần có thể có các ánh xạ với nhau hoặc với các thành tố bên ngoài thông quan các quan hệ.

Hình 3–8 Một thể hiện đầy đủ của subsystem

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 80

## 3.9 Các thành phần quan trị mô hình (model management)

Hình 3-9 Subsystem và ảnh xạ (mappiing) giữa các thành phần

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 81

## *3.10. Model*

### *3.10.1. Ngữ nghĩa*

Một model (mô hình) là sự trừu tượng hóa của một vấn đề trong hệ thống, nó miêu tả hệ thống dưới một góc độ quan sát đặc biệt với một mức độ trừu tượng nào đó, model có thể chứa tất cả các thành tố mô hình cần thiết cho việc miêu tả vấn đề của hệ thống. Các thành tố trong mô hình được tổ chức dưới dạng cây kế thừa của các package hoặc subsystem.

### *3.10.2. Ký hiệu*

Một model sử dụng ký hiệu của package với một hình tam giác nhỏ được gắn vào trên phải của hình chữ nhật lớn. Trong trường hợp, model có chứa nội dung thì hình tam giác được vẽ trong hình chữ nhật nhỏ cùng với tên của model. Một model cũng có thể sử dụng ký hiệu như là package với một từ khóamodel ™TM được đặt phía trên tên của model.

Ví dụ : một mô hình hệ thống (systemModel) sẽ chứa các mô hình phân tích (analysis models) và thiết kế (design models)

Hình 3–10 Một thể hiện của Modell

Ví dụ về việc tổ chức cây kế thừa trong model. Kế thừa dựa trên Model Kế thừa dựa trên Subsystem

Hiình 3–11 Tổ chức kế thừa trong Modell

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 82

## 3.11. Các cơ chế mơ rộng tổng quát

### *3.11.1. Ràng buộc (Constraint) và chú thích (Comment)*

***Ngữ nghĩa***

*Constraint (ràng buộc) là một quan hệ ràng buộc về ngữ nghĩa được* đính kèm vào các thành tố mô hình nhằm thay đổi và giới hạn về ngữ nghĩa của các thành tố này, đảm bảo hệ thống được mô hình hóa một cách đúng đắn. Một số constraint được định nghĩa ngay trong bản thân UML, một số do người sử dụng tự định nghĩa lấy. Những ràng buộc do người dùng định nghĩa thường được đặc tả bằng các từ trong một ngôn ngữ được qui định (cú pháp và sự thể hiện phụ thuộc vào công cụ hỗ trợ). *Comment (chú thích) là một chuỗi văn bản được gắn trực tiếp vào các* thành tố mô hình có tác dụng ghi chú, giải thích, bổ sung ngữ nghĩa cho thành tố đó.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 83

*3.12. Ký hiệu*

Constraint được biểu diễn bởi một chuỗi văn bản được đặt trong cặp dấu { }. Ngôn ngữ dùng để viết constraint thường được định nghĩa bởi công cụ hỗ trợ (ví dụ như OCL – Object Constraint Language hoặc có thể dùng ngôn ngữ tự nhiên để đặc tả ràng buộc). Đối với các thành tố mà ký hiệu là một chuỗi văn bản (ví dụ như một thuộc tính nào đó...) chuỗi ký tự ràng buộc được ghi ngay sau chuỗi văn bản và được tách bởi một dấu ngăn cách.

Đối với một danh sách các thành tố ở dạng văn bản (ví dụ như danh sách các thuộc tính của một lớp) thì ràng buộc sẽ được áp đặt trên tất cả các thành tố sau chuỗi ràng buộc cho tới khi có một chuỗi ràng buộc mới hoặc kết thúc danh sách.

Ràng buộc 1 áp dụng cho các thuộc tính 1,2 tới n–1

Ràng buộc 2 áp dụng cho các thuộc tính từ n tới cuối .

Hình 3–12 Ràng buộc trên danh sách thành tố

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 84

Đối với các thành tố có ký hiệu đồ họa đơn, chuỗi ràng buộc có thể đặt gần ký hiệu, tốt nhất là đặt gần tên của thành tố nếu có. Đối với các một cặp hai ký hiệu đồ họa, ràng buộc được thể hiện bằng một mũi tên đứt nét từ nhãn của thành tố này tới nhãn của thành tố kia. Chair-of (làm chủ tịch) phải là tập con của Member-of(thành viên) ràng buộc

Hình 3–13 Ràng buộc trên một cặp thành tố

Đối với ràng buộc trên ba ký hiệu đồ thị trở lên, ràng buộc được đặt trong note (để trong cặp dấu {})và có các đường đứt nét liên kết đến các thành tố bị ràng buộc.

Các chú thích là một chuỗi ký tự đặt trong một biểu tương note. Các chú thích cho các hàm hoặc các ràng buộc không được đặc tả trong UML nhưng được đặc tả trong công cụ hỗ trợ.

Hình 3–14 Ràng buộc trên nhiều thành tố

Hình 3–15 Chú thích cho package MFC6..0

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 85

## 3.13. Thuộc tính cua các thành tố (Element Properties)

***Ngữ nghĩa***

Một số thuộc tính của các thành tố không có ký hiệu trực quan. Để thêm vào đó người dùng có thể định nghĩa các thuộc tính mới bằng cách sử dụng cơ chế tagged value (tạm dịch là giá trị đính kèm hay giá trị thẻ). Mỗi tagged value biểu diễn một loại thuộc tính đặc biệt trong ứng dụng tương ứng với một hay nhiều loại phần tử mô hình.

***Ký hiệu***

Một tagged value được thể hiện bên trong một cặp dấu {}, với một thẻ (tag) và một giá trị (value), có dạng : { name = value } name là tên của thẻ (tag), value là giá trị của thẻ, cả hai được biểu diễn bằng chuỗi ký tự. Nếu kiểu của thẻ là Boolean (đúng/sai) thì giá trị mặc định của thẻ là True (đúng), những kiểu khác thì cần phải có giá trị rõ ràng, thông thường thì không có định nghĩa hình thức cho giá trị thẻ, mục đích chủ yếu chỉ để cung cấp thông tin quản lý mô hình ví dụ như version, tác giả của mô hình, không liên quan đến sản phẩm cuối.Ví dụ : {abstract} { Tác giả = " Nguyễn Văn A", Thời hạn cuối = 31–7–2000, Gian đoạn = Phân tích } *(giá trị thẻ )(ràng buộc)*

Hình 3–16 Ví dụ về tagged vallue

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 86

## 3.14. Các mẫu (Stereotypes)

***Ngữ nghĩa***

Stereotype dùng để đặc biệt hóa về mặt ngữ nghĩa của các phần tử đã được định nghĩa trong UML, là một lớp các phần tử mô hình mới được tổng quát từ những phần tử đã được định nghĩa, được giới thiệu tại một thời điểm cần thiết nào đó trong qúa trình mô hình hóa hệ thống.

***Ký hiệu***

Cách thể hiện tổng quát của Stereotype là sử dụng ký hiệu của phần tử cơ sở với một từ khóa nằm bên trên tên của thành tố, chuỗi từ khóa được đặt bên trong cặp dấu ™™ cũng là tên của stereoype. Một cách thể hiện khác là đặt biểu tượng (icon) bên trong hình chữ nhật chứa tên của khuôn mẫu, hoặc chỉ có biểu tượng và bên dưới là tên của khuôn mẫu.

Hình 3–17 Một số stereotype (khuôn mẫu)

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 87

## 3.15. Các lược đồ

### *3.15.1. Giới thiệu*

Các lược đồ là nền tảng cơ bản để mô hình hóa hệ thống trong UML. UML cung cấp chín loại lược đồ khác nhau nhằm giúp mô hình hóa chi tiết hệ thống, chúng được chia thành ba loại chính

_ Các lược đồ tĩnh : Use Case (Use Case Diagram), lớp (Class diagram), đối tượng (Component diagram).

_ Các lược đồ động (hay các lược đồ hành vi) : tuần tự (Sequence Diagram), cộng tác (Collaboration diagram), trạng thái (Statechart diagram), hoạt động (Activity diagram).

_ Các đồ thực thi : thành phần (Component diagram), triển khai (Deployment diagram).

Các lược đồ lần lược được trình bày theo cấu trúc :

_ Ví dụ : gồm hình vẽ và các chú thích trên lược đồ

_ Ngữ nghĩa : ý nghĩa của các lược đồ

_ Chức năng : chức năng sử dụng

_ Các thành phần chính : giới thiệu các thành phần chính trong lược đồ.

Hình 3–18 Tổng quan về các lược đồ trong UML

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 88

## 3.16. Lược đồ lớp (Class Diagram)

*(quan hệ kết hợp hay thu nạp) (tên lớp) (quan hệ tổng quát hóa) (quan hệ kết hợp) (ràng buộc) (thuộc tính) (hàm) (giao tiếp) (quan hệ phụ thuộc) (vai trò) (biểu tượng của lớp) (đa xạ)*

Hình 3–19 Lược đồ lớp (Cllass Diiagram)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 89

***Ngữ nghĩa***

_ Một lược đồ lớp là một lược đồ dùng để mô tả các lớp (class), các giao tiếp (interface), sự cộng tác (collaboration) và các mối quan hệ giữa các thành phần trong mô hình.

_ Class Diagram là một thể hiện dưới dạng đồ thị cấu trúc tĩnh của mô hình.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 90

### 3.17. Chức năng

_ Đặt tên và lập mô hình các khái niệm trong hệ thống.

_ Đặc tả sự cộng tác, các mối quan hệ quan hệ giữa các lớp

_ Đặc tả sơ đồ cơ sở dữ liệu

_ Người ta thường dùng các lược đồ lớp để thể hiện sự kết nối của các thành phần cấu trúc trong design view và process view.

_ Class Diagram được sử dụng trong mô hình phân tích và thiết kế, hỗ trợ cho giai đoạn phân tích và thiết kế hệ thống.

*3.17.1. Các thành phần chính*

***Lớp (Class)***

***Ngữ nghĩa***

Một lớp (class) là một đại diện cho một tập các các đối tượng có những đặc tính tương tư nhau. Đây là thành phần cơ bản nhất của một lược đồ lớp. Lớp cũng có cấu trúc dữ liệu, hành vi và các quan hệ với các lớp khác.

***Ký hiệu***

Biểu tượng của một lớp đơn giản chỉ là một hình chữ nhật gồm có ba phần:

– Phần trên cùng : chứa tên của lớp, tên của lớp phải là duy nhất không được trùng với tên các lớp khác trong cùng một package. Ngoài ra trong một số trường hợp ngăn này còn có thể chứa stereotype (khuôn mẫu) hoặc tagged value (giá trị thẻ).

– Phần giữa : chứa danh sách các thuộc tính(hay còn gọi là các biến thành phần).

– Phần cuối : chứa danh sách các hành vi (hay còn gọi là các hàm thành phần)

Trong một số lược đồ hai phần dưới cùng thường được bỏ qua, ngay cả khi chúng thực sự hiện diện, thì chúng cũng không hiển thị tất cả danh sách các thuộc tính cũng như các hành vi. Mục tiêu ở đây chỉ thể hiện những thuộc tính và những hành vi có ích trong một lược đồ cụ thể, vì vậy biểu tượng của lớp có thể được rút gọn khi cần thiết.

Hình 3–20 Biểu tượng của một lớp

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 91

*(biểu tượng đơn giản nhất) (thể hiện của lớp ở mức độ phân tích hệ thống) (thể hiện của lớp ở mức độ triển khai – xây dựng hệ thống)*

Hình 3–21 Thể hiện của lớp ở các giai đoạn khác nhau

Các thuộc tính và các hành vi trong một lớp có thể có các tầm vực (visibility) khác nhau, các ký hiệu của các tầm vực được đặt trước các thuộc tính hay hành vi, trong UML định nghĩa ba mức chuẩn :

_ Public – ký hiệu bằng dấu (+)

_ Private – ký hiệu bằng dấu (–)

_ Protected – ký hiệu bằng dấu (#)

Cú pháp cho các khai báo của thuộc tính như sau :

*Tầm vực – Tên thuộc tính – Bản số : Kiểu = Giá trị khởi tạo*

Ví dụ : −color : Color = Red
+ size: Area = (100,100)
# points [2..*] : Point
Cú pháp cho các khai báo của hàm như sau :
*Tầm vực – Tên thuộc tính (danh sách tham số) : Kiểu trả về*
Ví dụ : + display (): Location
+ hide()
−attachXwindow(zwin: Xwindow*)

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 92

## 3.18. Interface

***Ngữ nghĩa***

Có những lớp không có gì ngoài các hàm ảo thuần túy, trong ngôn ngữ Java các lớp như vậy không là đối tượng, chúng được gọi là interface (giao tiếp). Muốn sử dụng interface, lớp phải giao tiếp với nó bằng một quan hệ phụ thuộc, lớp sử dụng chỉ phụ thuộc vào các hành vi trong giao tiếp, chứ không phụ thuộc vào các thứ khác trong interface.

***Ký hiệu***

Ký hiệu đơn giản của một interface là một hình tròn nhỏ có tên ngay phía dưới. Muốn chỉ ra các hành vi trong interface, có thể biểu diễn interface bằng ký hiệu lớp với một stereotype bên trên của tên interface.

Hình 3–22 Thể hiện của một iinterface
Hình 3–23 Một các sử dụng iinterface

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 93

### 3.19. Các loại quan hệ

#### *3.19.1. Quan hệ kết hợp (association)*

Một quan hệ kết hợp là quan hệ về mặt cấu trúc giữa hai, hoặc giữa một và nhiều thành tố (lớp hoặc đối tượng).

–Trên quan hệ kết hợp có thể có hai tên cho hai hướng ngược chiều nhau.

*(2 quan hệ kết hợp – công nhân làm việc cho công ty và công ty quản lý công nhân )*

manage

Hình 3–24 Tền trong quan hệ kết hợp

– Mỗi mối hệ kết hợp cũng có bản số (multiplicity) và tính định hướng (navigate). Ví dụ như: (0..1), (1), (0..*), (1..*), (1..6), (10..*)...

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 94

Hình 3–25 Bản số (mulltiiplliiciity) trong quan hệ kết hợp

–Trong quan hệ kết hợp cũng có sự đệ quy (trong trường hợp một lớp (hay đối tượng) kết hợp với chính nó. *( đệ quy )*

Hình 3–26 Quan hệ kết hợp đệ quy

–Trong quan hệ kết hợp có thể có các vai trò (hoặc thể hiện – *role) khác nhau. Tên của vai trò (hoặc thể hiện) được đặt* trên quan hệ và ngay sát cạnh lớp có thể hiện đó. (xem hình vẽ 3–27 bên dưới).

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 95

*(vai trò) (Person đóng vai trò là người lao động) (Company đóng vai trò là người chủ)*

Hình 3–27 Vai trò (rolles) trong quan hệ kết hợp

–Ngoài ra còn có Or–Association (quan hệ kết hợp ràng buộc or), đây có thể coi là một ràng buộc trên hai hay nhiều quan hệ kết hợp. Nó xác định các đối tượng của một lớp chỉ có thể tham gia vào một trong những quan hệ kết hợp này tại một thời điểm. Xét ví dụ trong hình 3–28 tại một thời điểm nhất định thì lớp Account (tài khoản) chỉ có quan hệ kết hợp với một trong hai lớp Person (người) hoặc Corporation (hãng). *(cách thể hiện của quan hệ Or–Association) (tại một thời điểm thì chỉ có một trong hai quan hệ có tác dụng)*

Hình 3–28 Một của Or–Associiatiion

– Trong quan hệ kết hợp cũng có quan hệ kết hợp hạn chế (qualified association). Xét ví dụ hình 3–29 lớp Person (người) chỉ quan hệ với lớp Bank (ngân hàng) thông qua thuộc tính hạn chế là account (tài khoản).

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 96

*(thuộc tính hạn chế )*

Hình 3–29 Một ví dụ của Qualliifiier Associiatiion

–Trong quan hệ kết hợp ta cũng có một loại thể hiện khác đó là quan hệ thu nạp (Aggregation). Quan hệ này chỉ ra rằng lớp kết hợp được xem là "toàn thể", và lớp kia được xem như là một bộ phận của "toàn thể" đó. Xét ví dụ hình 3–30 lớp Window được xem là lớp "toàn thể" bởi vì nó có thể chứa nhiều hình (lớp

shape). *(lớp "toàn thể ") ( lớp "bộ phận") (cách thể hiện của quan hệ thu nạp)*
Hình 3–30 Quan hệ thu nạp (aggregatiion)

–Ngoài ra còn có quan hệ cấu thành (composition). Đây là một dạng mạnh hơn của quan hệ thu nạp (aggregation), trong trường hợp này khi lớp toàn thể bị hủy thì lớp bộ phận (hay lớp cấu thành) cũng bị hủy theo. Xét ví dụ trong hình bên dưới 3–31 ta có thể thấy lớp Window được cấu thành từ hai thanh cuộn (scrollbar), một header đóng vai trò là tiêu đề và một panel đóng vai trò là body của Window.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 97

*(lớp window được cấu thành từ ba lớp Slider , Header và Panel) (cách thể hiện của quan hệ cấu thành)*
Hình 3–31 Quan hệ cấu thành (Composiitiion)

–Ngoài ra còn có quan hệ n-ary association (quan hệ kết hợp bậc n) giữa nhiều lớp với nhau.Sau đây là một ví dụ về quan hệ giữa ba lớp Year (năm), Team (đội) và Player (cầu thủ). Quan hệ này chỉ ra số bàn để lọt lưới, số bàn đỡ được, số trận thắng, số trận thua, số trận hòa của thủ môn của một đội bóng trong một mùa bóng. *(cách thể hiện của quan hệ kết hợp bậc 3)*
Hình 3–32 Quan hệ kết hợp bậc 3

–Lưu ý trong quan hệ kết hợp bậc n tất cả các nhánh quan hệ đều là quan hệ kết hợp thông thường, không phải là thu nạp hay cấu thành.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 98

### 3.20. Quan hệ tổng quát hóa

Quan hệ tổng quát hóa còn được gọi là quan hệ kế thừa. Quan hệ tổng quát hóa là một quan hệ giữa một thành tố tổng quát (thường được xem là thành tố cha – superclass) và một hay nhiều thành tố chuyên biệt (được xem là thành tố con – subclass). Quan hệ này thường được sử dụng cho các thành tố mô hình như lớp (class), gói (package), use case... Trong quan hệ này thành tố con được kế thừa các đặc điểm của thành tố cha (ví dụ lớp con kế thừa các thuộc tính và hàm của lớp cha). Sau đây là ví dụ về quan hệ tổng quát hóa :

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 99

*(lớp cha) (các lớp cha Shape được tổng quát hóa từ các lớp con Polygon, Ellipse, Spline... ) (cách thể hiện kiểu riêng lẻ) (cách thể hiện kiểu cây chia sẽ) (cách thể hiện của quan hệ kế thừa)*

Hình 3–33 Một số cách thể hiện của quan hệ tổng quát hóa (Generalliizatiion)

Trong quan hệ tổng quát hóa có một số kiểu ràng buộc trên quan hệ :

– Phủ lấp (Overlapping) – quan hệ này cho phép một lớp con ở cấp sâu hơn có thể kế thừa từ nhiều lớp con (cùng một lúc) ở cấp cao hơn của một quan hệ kế thừa trên một lớp cha chung nào đó.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 100

*(lớp con có kế thừa phủ lấp) (lớp con được kế thừa phủ lấp) (lớp cha) (cách thể hiện của quan hệ kế thừa phủ lấp)*

Hình 3–34 Quan hệ kế thừa phủ lấp (Overllappiing)

–Tách rời (Disjoint) – quan hệ này trái ngược với quan hệ kế thừa phủ lấp – lớp con ở cấp sâu hơn không được kế thừa *cùng một lúc nhiều lớp con ở cấp cao hơn.*

–Hoàn toàn (Complete) –có nghĩa là quan hệ này đã đầy đủ các lớp con, không thể thêm mới một lớp con nào nữa. Ví dụ: lớp cha là lớp Gà (con gà)và hai lớp con là Gà trống và Gà mái, hiển nhiên ta không thể thêm một lớp vào trong quan hệ kế thừa giữa ba lớp này. Cách thể hiện cũng tương tự như phần phủ lấp, ta chỉ thay ràng buộc Overlapping bằng *Complete.*

–Không hoàn toàn (Incomplete) – ngược lại với quan hệ hoàn toàn, đây là quan hệ mặc định trong quan hệ kế thừa, có nghĩa là có thể thêm vào các lớp con trong quan hệ này.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 101

*(thể hiện của quan hệ kế thừa không hoàn toàn, tách rời)*

Hình 3–35 Quan hệ kế thừa không hoàn toàn

### 3.21. Quan hệ phụ thuộc (Dependency)

Quan hệ phụ thuộc là một quan hệ ngữ nghĩa giữa hai thành tố mô hình, trong hai thành tố này có một thành tố độc lập và một thành tố phục thuộc vào

thành tố độc lập này. Thường đây là quan hệ giữa hai lớp hai use case hoặc hai package...Ví dụ như một lớp sử dụng tham số là một đối tượng của một lớp khác.

*(lớp Shape phụ thuộc vào lớp DrawingContext vì hàm Draw trong lớp Shape sử dụng tham số là đối tượng của lớp DrawingContext) (cách thể hiện của quan hệ phụ thuộc )*

Hình 3–36 Một ví dụ của quan hệ phục thuộc

Thông thường quan hệ phụ thuộc được biểu diễn bằng một đường đứt nét có một mũi tên ở đầu và thường có các keyword đi kèm bên trên mũi tên để chỉ kiểu phụ thuộc. *(các kiểu quan hệ phụ thuộc)*

Hình 3–37 Các keyword trên quan hệ phụ thuộc

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 101

## 3.22. Các thành tố được tính toán (hay được dẫn xuất – derived Element)

Đây là các thành tố được tính từ các thành tố khác trong mô hình hay trong lược đồ, chúng được thêm vào trong bản phân tích giúp làm rõ thêm các lược đồ, chúng vẫn được thêm vào ngay cả khi việc này không cung cấp thêm thông tin về ngữ nghĩa cho lược đồ. Ký hiệu của các thành tố này là một dấu sổ (/)ngay trước tên của thành tố. *(thuộc tính age được tính từ hai thuộc tính currentDate và birthday) (quan hệ kết hợp dẫn xuất)*

Hình 3–38 Thành tố tính toán – dẫn xuất (Deriived Ellement)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 103

### 3.23. Lược đồ đối tượng (Object Diagram)

Ví dụ**:** *(các đối tượng trong lược đồ (giá trị thuộc tính) (liên kết giữa các đối tượng) (đối tượng ẩn danh)*
Hình 3–39 Lược đồ đối tượng(Objject Diiagram)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 104

*Ngữ nghĩa*
_ Là một đồ thị của các thể hiện, bao gồm các đối tượng và các giá trị cụ thể.
_ Là một thể hiện của lược đồ lớp, đưa ra một cái nhìn khá chi tiết về các đối tượng trong hệ thống tại một thời điểm nhất định.
*Chức năng*
_ Minh họa cấu trúc của dữ liệu
_ Minh họa các đối tượng tại những thời điểm cụ thể.
_ Đặc tả các snapshot (thể hiện của hệ thống)

### 3.24. Các thành phần chính

#### *3.24.1. Các đối tượng (Objects)*

*Ngữ nghĩa*

Một object (đối tượng) là một thể hiện cụ thể của một lớp trong hệ thống, đối tượng có thể lấy tên thật trong thế giơi thực. Một đối tượng có thể được định danh và định trị.

*Ký hiệu*

Ký hiệu của đối tượng trong lược đồ là một hình chữ nhật được dẫn xuất từ ký hiệu của lớp, thường có hai phần chính

–Phần trên của hình chữ nhật chứa tên của đối tượng và tên lớp của đối tượng (cả hai đều được gạch dưới). Nếu đối tượng không có tên, chỉ có lớp phụ thuộc, ta gọi đây là đối tượng ẩn danh (anonymous). Cú pháp đặt tên đối tượng :

*Tên đối tượng : Tên lớp*

–Phần bên dưới chứa các giá trị của đối tượng với các giá trị cụ thể. Cú pháp :

*Tên thuộc tính : Kiểu = Giá trị*

*(một thể hiện đơn giản chỉ với tên của đối tượng) (tên đối tượng) (tên lớp) (đối tượng có các giá trị thuộc tính cụ thể ) ( đối tượng ẩn danh, chỉ có tên lớp ) ( đối tượng với biểu tượng lớp phụ thuộc và tên được gạch dưới)*

Hình 3–40 Một số thể hiện của đối tượng (Objject)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 105

### 3.25. Đối tượng ghép (Composite Object)

*Ngữ nghĩa*

Đối tượng ghép biểu diễn đối tượng ở mức độ cao hơn, đối tượng ghép được cấu thành từ các đối tượng thành phần khác. Đây là một thể hiện của lớp trong quan hệ cấu thành (xem hình 3–34).

*Ký hiệu*

Ký hiệu của đối tượng ghép tượng tự như của đối tượng thường, nhưng phần hình chữ nhật bên dưới có chứa các đối tượng thành phần cấu thành và các liên kết giữa các đối tượng này. *(tên đối tượng và tên lớp)(các đối tượng thành phần)*

Hình 3–41 Thể hiện của đối tượng ghép (Composiite Objject)

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 106

## 3.26. Các liên kết (Link)

*Ngữ nghĩa*

ột liên kết (link) là một tham chiếu giữa các đối tượng. Liên ết là một thể hiện của quan hệ kết hợp.

*Ký kiệu*

Ký hiệu của liên kết là một đường dẫn nối hai đối tượng với nau, cuối đường dẫn có thể có tên vai trò của đối tượng trong liên kết đó. Lưu ý, liên kết không có bản số. Liên kết có ràng buộc hạn chế (qualifier) thể hiện bằnh một hộp nhỏ chứa giá trị hạn chế ngay đầu liên kết. Ngoài ra liên kết còn có một số stereotype khi thực hiện liên kết.

– association™™ – mặc định không cần thể hiện trên liên kết, ngoại trừ trường hợp cần nhấn mạnh.

parameter™™ – đối tượng là tham số hàm

local™™ – đối tượng là biến cục bộ

global™™ – đối tượng là biến toàn cục

self™™ – tự liên kết *(các liên kết giữa các đối tượng) (thể hiện của liên kết hạn chế)*

Hình 3-42 Các liên kết (links)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 107

### 3.27. Lược đồ Use Case (Use Case Diagram)

*( các tác nhân trong hệ thống) (quan hệ kết hợp giữa tác nhân và use case) (ranh giới xác định hệ thống) (các use case trong hệ thống (quan hệ "extend" (mở rộng) giữa hai use case)*
Hình 3-43 Lược đồ Use Case (Use Case Diiagram)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 108

*Ngữ nghĩa*
_ Là một loại lược đồ bao gồm các actor, các use case,và các mối quan hệ giữa chúng.
_ Lược đồ Use Case ghi nhận chức năng của hệ thống dưới góc nhìn của người sử dụng.
*Chức năng*
_ Đặc tả ngữ cảnh của một hệ thống và cách thức hệ thống tương tác với thế giới bên ngoài.
_ Nắm bắt các yêu cầu của hệ thống.
_ Xác nhận tính hợp lệ của kiến trúc hệ thống.
_ Xác định phạm vi ứng dụng và các chức năng của hệ thống
_ Định hướng quá trình cài đặt và phát sinh các trường hợp test.

_ Đóng vai trò trung tâm trong việc hoạch định tiến trình tiến hóa phần mềm.

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 109

## 3.29. Các thành phần chính

### *3.29.1. Các Use case*

*Ngữ nghĩa*

Use case (tạm dịch trường hợp sử dụng) là một chuỗi các hành động hoặc một đơn vị chức năng được cung cấp bởi hệ thống nhằm đáp ứng nhu cầu của các tác nhân bên ngoài hay các hệ thống khác. Use case phải là một đơn vị chức năng toàn diện, luôn trả về kết quả cuối cùng cho tác nhân có quan hệ với nó và không thể chia nhỏ use case. Điểm mở rộng (extension point) là một tham chiếu tới các hành động được mở rộng từ một use case khác.

*Ký hiệu*

Ký hiệu của một use case là một ellipse với tên bên trong hoặc đặt ngay bên dưới ellipse. Đăng ký học phần ĐKHP *(tên use case)*

Hình 3-44 Thể hiện của Use case

Thể hiện của điểm mở rộng là một danh sách nằm bên trong use case được bắt đầu với tiêu đề là Extension Points (xem hình vẽ 3-46 trong phần 3.5.4.4.3 Các quan hệ).

### *3.29.2. Các tác nhân (Actors)*

*Ngữ nghĩa*

Actor (tác nhân) là một thực thể đóng vai trò tương tác với hệ thống, tác nhân có thể là người sử dụng hệ thống hoặc một hệ thống khác. Mỗi tác nhân có một vai trò nhất định đối với Use case mà nó tương tác.

*Ký hiệu*

Ký hiệu của một tác nhân là một hình chữ nhật biểu tượng của lớp với stereotype là actor™™ được đặt ngay trên tên của tác nh ân. Biểu tượng chuẩn của tác nhân trong lược đồ là một "người hình que" (stickman) với tên được đặt ngay bên dưới.

Hình 3-45 Thể hiện của một tác nhân (Actor)

# Chương 3

# Hệ thống ký hiệu (UML Notation) 110

## *3.30. Các quan hệ trên Use case*

*Ngữ nghĩa*

Có một số quan hệ chuẩn được định nghĩa giữa Use case và Actor hoặc Use case :

–Quan hệ kết hợp (association) – đây là quan hệ giữa một actor và một use case, quan hệ xảy ra khi những thực thể của use case và actor có giao tiếp, liên lạc với nhau.

–Quan hệ mở rộng (extend) – quan hệ mở rộng từ use case A tới use case B chỉ ra rằng use case A sẽ mở rộng thêm một số hành vi lấy từ use case A, các hành vi được mở rộng từ use case B tùy thuộc vào điều kiện mở rộng được đặc tả tại điểm mở rộng (extension point) trên use case B.

–Quan hệ tổng quát hóa (generalization) – quan hệ tổng quát hóa từ use case A tới use case B chỉ ra rằng A là một phần đặc biệt hóa của B.

–Quan hệ bao hàm (include) – quan hệ bao hàm giữa use case A và use case B có nghĩa là A sẽ bao hàm những hành vi được đặc tả bởi B.

***Ký hiệu***

–Ký hiệu của các quan hệ :

–Kết hợp – đường dẫn liền nét nối actor và use case

– Mở rộng – đường dẫn đứt nét có hướng nối hai use case trên đó có từ khóa extend™™.

–Tổng quát hóa – đường dẫn liền nét có tam giác rỗng cuối đường dẫn.

– Bao hàm – đường dẫn đứt nét có hướng nối hai use case trên đó có từ khóa include™™.

*(actor) (quan hệ mở rộng) (điểm mở rộng) (quan hệ bao hàm)*

Hình 3-46 Những quan hệ trên use case

## Chương 3
## HỆ THỐNG KÝ HIỆU (UML Notation) 111

### 3.31. Các quan hệ trên actor

***Ngữ nghĩa***

Có hai loại quan hệ trên actor o Quan hệ kết hợp – tương tự như phần use case. –Quan hệ tổng quát hóa (generalization) – quan hệ tổng quát hóa từ tác nhân A tới tác nhân B chỉ ra rằng A cũng có thể giao tiếp, liên lạc với các use case mà tác nhân B giao tiếp được.

***Ký hiệu***

*(quan hệ tổng quát hóa) (quan hệ kết hợp)*

Hình 3-47 Quan hệ trên các actor

## Chương 3
## HỆ THỐNG KÝ HIỆU (UML Notation) 112

### 3.32. Lược đồ tuần tự (Sequence Diagram)

*đối tượng) (tên) (lớp) (thông điệp truyền đi) (nhãn) (kích hoạt) (đệ qui) (đường giới hạn chu kỳ sống của đối tượng) (trả về) (hủy) (tạo)*

Hình 3-48 Lược đồ tuần tự (Sequence Diiagram)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 113

***Ngữ nghĩa***

_ Là một lược đồ tương tác tập trung vào các hành vi động hướng thời gian.

_ Cách thể hiện tuần tự đặc biệt hữu dụng trong các hệ thống với các chức năng phụ thuộc vào thời gian như là các ứng dụng thời gian thực, hoặc cho các kịch bản phức tạp khi mà sự phụ thuộc vào thời gian đóng vai trò quan trọng.

_ Lược đồ tuần tự có hai phương

_ Theo phương thẳng đứng – biểu diễn trục thời gian theo hướng từ trên xuống dưới.

_ Theo phương ngang – biểu diễn các đối tượng khác nhau trong chuỗi tuần tự các sự kiện dùng để thực hiện một chức năng nào đó của hệ thống.

_ Lược đồ tuần tự có hai đặc điểm mà lược đồ cộng tác không có là đường giới hạn chu kỳ sống (lifeline) và kích hoạt (focus of control).

***Chức năng***

_ Mô hình hóa luồng xử lý

_ Minh họa các kịch bản đặc trưng

_ Mô tả một cách rõ ràng sự tuần tự của các sự kiện, thể hiện khi nào một đối tượng được tạo và huỷ, mô tả các hành động đồng thời.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 114

### 3.33. Các thành phần chính

Lược đồ tuần tự bao gồm bốn thành tố chính :

–Các đối tượng (object) liên quan đến lược đồ.

–Các đường giới hạn chu kỳ sống (lifeline) thể hiện sự tồn tại của đối tượng trên trục thời gian.

–Các kích hoạt (activation –focus of control) được thể hiện bằng hình chữ nhật nằm trên đường sống. Độ dài của focus of control cho biết thời gian mà đối tượng tồn tại để thực hiện một số hành động nào đó.

– Các thông điệp (message) thể hiện sự liên lạc giữa các đối tượng, được biểu diễn bằng các cạnh nối giữa những hình chữ nhật của các focus of control có liên quan. Có một số dạng thông điệp : thông điệp đồng bộ, thông điệp không đồng bộ, thông điệp đệ qui...

*3.33.1. Việc tạo và hủy một đối tượng*

Trong lược đồ tuần tự ta có thể đặc tả khi nào một đối tượng được tạo ra và bị hủy đi. Xét ví dụ sau :

Hình 3-49 là một lược đồ tuần tự diễn tả việc khởi tạo và huỷ các đối tượng trong lược đồ tuần tự. Ở đây, ta thấy đối tượng CellularRadio tạo một đối tượng Connection để đáp ứng lại thông điệp Connect. Việc tạo một đối tượng được ký hiệu bằng một mũi tên thông điệp chỉ đến hình chữ nhật mô tả đối tượng cần tạo. Tương tự như vậy, việc huỷ đối tượng ký hiệu bằng một mũi tên thông điệp chỉ đến dấu X nằm ở cuối đường lifeline của đối tượng. Thời gian sống của đối tượng Connection được thể hiện rất rõ trên hình vẽ.

*(tạo đối tượng) (hủy đối tượng) (chu kỳ sống của đối tượng được tạo ra) (thông điệp bất động bộ)*

Hình 3-49 Tạo và hủy đối tượng trên lược đồ tuần tư

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 115

### 3.34. Thông điệp không đồng bộ và đệ qui

Thông điệp không đồng bộ là một thông điệp trở về ngay lập tức sau khi sinh ra một tiểu trình trong đối tượng nhận thông điệp. Trong lược đồ tuần tự, nó được thể hiện bằng một mũi tên không đầy đủ (chỉ có 1 nét). Xem ví dụ hình 3–49, thông điệp Connect(pno) được trả về đối tượng Cellularradio ngay lập tức

sau khi được gởi tới đối tượng Connection. Khi một đối tượng tự kích hoạt có nghĩa là nó sinh ra một thông điệp tự gửi cho chính nó, đây phải là một thông điệp đồng bộ và được biểu diễn bằng một mũi tên tự móc vào thanh kích hoạt (xem hình vẽ 3-48).

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 116

### 3.35. Thời gian chuyển thông điệp trong lược đồ tuần tự

Một thông điệp trong lược đồ tuần tự có thể được đặc tả thời gian gửi đi hoặc thời gian nhận về. Chúng có các tên hình thức có thể được dùng trong biểu thức ràng buộc (xem ví dụ hình 3-50). Một thông điệp thông điệp có thể tồn tại trong một khoảng thời gian đáng kể từ lúc được gửi đi đến lúc được tiếp nhận và xử lý, trong lược đồ tuần tự các thông điệp dạng này được biểu diễn bằng một mũi tên xéo từ hình chữ nhật kích hoạt này tới hình chữ nhật kích hoạt kia.

*(ràng buộc về thời gian nhận và gửi thông điệp) (thời gian chuyển thông điệp là đáng kể)*

Hình 3-50 Ràng buộc về thời gian chuyển thông điệp

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 117

### 3.36. Lược đồ cộng tác (Collaboration Diagram)

***Ví dụ***

*(liên kết giữa 2 đối tượng có tương tác với nhau) (đối tượng) ( message gửi đi giữa các đối tượng) (số thứ tự của các message)*

Hình 3–51 Lược đồ cộng tác(Collllaboratiion Diiagram)

***Ngữ nghĩa***

_ Là một lược đồ tương tác tập trung vào cấu trúc tổ chức, mối quan hệ tác động qua lại giữa các đối tượng.

–Lược đồ cộng tác và tuần tự tương tự nhau về mặt ngữ nghĩa, chúng thể hiện những thông tin tương tự nhưng theo hai cách khác nhau.

–Lược đồ tuần tự chú trọng đến thứ tự các thông điệp được chuyển tải theo thời gian.

–Lược đồ cộng tác chú trọng đến mối quan hệ giữa các đối tượng.

***Chức năng***

_ Ghi nhận các hành vi động của hệ thống (hướng message)

_ Mô hình hóa luồng xử lý

_ Minh họa sự phối hợp giữa cấu trúc đối tượng và các xử lý

_ Thể hiện sự ảnh hưởng lẫn nhau giữa các đối tượng

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 118

## 3.37. Các thành phần chính

### *3.37.1. Các đối tượng*

Các đối tượng được thể hiện tương tự như trong lược đồ tuần tự, đó là một hình chữ nhật với tên đối tượng và tên lớp được gạch dưới, trong một số trường hợp có thể không để tên đối tượng.

### *3.37.2. Các liên kết*

Liên kết giữa các đối tượng được thể hiện bằng một đường nối hai đối tượng có quan hệ với nhau. Các vai trò (role) và hạn chế (qualifier) trong liên kết được thể hiện ở đầu cuối của liên kết (tương tự như trong quan hệ kết hợp giữa hai lớp). Trong lược đồ tuần tự đặc tả sẵn một số khuôn mẫu sẵn cho vai trò của các liên kết :

local™™ –chỉ ra rằng thực thể tương ứng là một biến cục bộ trong một hành vi hay một hàm.

global™™ –chỉ ra rằng thực thể tương ứng là một biến toàn cục trong một hành vi hay một hàm.

parameter™™ –chỉ ra rằng thực thể tương ứng là một tham số trong một hành vi hay một hàm.

self™™ –chỉ ra rằng đối tượng có thể gửi thông điệp cho chính nó...

*(các thông điệp) (các vai trò trong liên kết) (tự liên kết với khuôn mẫu self)*

Hình 3–52 Ví dụ về các khuôn mẫu trong vai trò liên kết

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 119

## 3.38. Thông điệp và các kích thích

***Ngữ nghĩa***

Trong lược đồ công tác một sự kích thích (stimulus) là một liên lạc giữa hai đối tượng nhằm mục đích truyền đạt thông tin cho nhau. Một stimulus sẽ gây ra một hàm, hoặc đánh thức một sự kiện hoặc tạo ra hoặc hủy một đối tượng. Một thông điệp chính là một đặc tả của một stimulus, nó định rõ vai trò của đối tượng nhận và đối tượng gởi.

***Ký hiệu***

Thông điệp và stimulus được thể hiện bằng một mũi tên có nhãn đặt ngay trên liên kết giữa hai đối tượng. Ơư đây các liên kết (links)đóng vai trò chuyển, thi hành, hay phân phối các kích thích tới các đối tượng nhận. Mũi tên của thông điệp chỉ hướng đến đối tượng nhận. Một số dạng liên lạc giữa các đối tượng Ký hiệu Ý nghĩa (tượng trưng cho) Các thủ tục gọi hoặc các luồng điều khiển con. Luồng điều khiển nền. Mỗi mũi tên biểu diễn việc xúc tiến cho thông điệp kế tiếp trong chuỗi các thông điệp Một kích thích không đồng bộ. Thông điệp trả về của một thủ tục gọi nào đó.

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 120

## 3.39. Cú pháp đặt tên nhãn

Đây là một phần rất quan trọng trong việc đặc tả các thông điệp, cú pháp chuẩn của các thông điệp bao gồm *Predecessor guard-condition sequence–expression return–value := message–name argument-list*

– Predecessor – là một danh sách các chỉ số thứ tự được phân cách nhau bằng dấu phẩy (,)và kết thúc bằng dấu (/). Cú pháp của predecessor như sau

*Chỉ số thứ tự luồng ',' • '/'*

Ý nghĩa : các thông điệp có chỉ số thứ tự trong danh sách trên phải được xử lý xong thì thông điệp hiện tại mới được gởi đi.

– Guard condition – được viết bằng mã giả hoặc bằng ngôn ngữ lập trình tùy thuộc công cụ hỗ trợ thiết kế. Cú pháp *'[' Mệnh đề điều kiện ']'*

– Sequence number – biểu diễn chỉ số của các thông điệp, các thông điệp lồng, các thông điệp đồng thời... Cú pháp

*[ số / tên ] [ recurrence] ': '*

Ý nghĩa : Số chỉ thứ tự của thông điệp. Tên chỉ các thông điệp đồng thời. Ví dụ :

1 – chỉ thông điệp đầu tiên 2.1, 2.2 ... chỉ các thông điệp lồng sau khi xử lý thông điệp 2 3.1a, 3.1b là hai thông điệp đồng thời. *Recurrence – biểu diễn các hoạt động lặp hoặc điều kiện.*

Có hai dạng cú pháp :

'*' '[' Điều kiện lặp ']' – biểu diễn một điều kiện lặp. Ví dụ : * [ i=1..n].

'['điều kiện rẽ nhánh ']' – xác định một điều kiện rẽ nhánh. Ví dụ : [ x > y ].

– Return value – giá trị trả về của thông điệp

– Message name – tên thông điệp

– Argument list – danh sách các đối số

–Một số ví dụ về cách đặt tên nhãn 2: display (x,y) –thông điệp đơn giản
1.3.1 : p:= find(name) –thông điệp lồng [x<0] 4 : invert (x, color) –có điều kiện
A3,A4/ C3.1 * : update() –message đồng thời

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 121

### 3.40. Lược đồ trạng thái (Statechart Diagram)

*(trạng thái kết thúc) (trạng thái bắt đầu) (sự chuyển đổi trạng thái nội tại) (sự kiện) (hành động) (sự chuyển đổi trạng thái) (trạng thái con) (trạng thái)*
Hình 3–53 Lược đồ trạng thái(Statechart Diiagram)

***Ngữ nghĩa***

_ Là một lược đồ thể hiện máy trạng thái

_ Là một lược đồ thể hiện hành vi của các thực thể có thể mang những hành vi động bằng cách đặc tả sự hồi đáp của chúng trước những sự kiện từ các thực thể khác gửi tới.

***Chức năng***

_ Mô hình hóa chu trình sống của đối tượng

_ Mô hình hóa các đối tượng phản hồi (user interfaces, devices...)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 122

### 3.41. Các thành phần chính

#### *3.41.1. Trạng thái (state)*

***Ngữ nghĩa***

Một trạng thái (state) là một hoàn cảnh hay một tình huống trong qúa trình sống của đối tượng thỏa mãn một vài điều kiện nào đó, trạng thái có thể ở tình trạng chủ động tiến hành một số hoạt động hoặc thụ động chờ đợi một vài sự kiện khác xảy ra. Trong lược đồ trạng thái luôn có một trạng thái ban đầu (initial state) và một hoặc nhiều trạng thái kết thúc (final state).

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 123

***Ký hiệu***

Trạng thái ban đầu (hay trạng thái khởi động) được ký hiệu bằng một hình tròn nhỏ được tô đen bên trong, trạng thái kết thúc được ký hiệu bằng một hình tròn nhỏ được tô đen được bao bởi một đường tròn lớn hơn ở phía ngoài (xem hình vẽ3-53). Trạng thái thông thường được biểu diễn bằng một hình chữ nhật với các góc tròn. Hình chữ nhật biểu diễn một trạng thái có thể được chia thành nhiều phần

- Phần chứa tên – chứa tên của trạng thái.
- Phần chứa các sự chuyển đổi nội tại (Internal transitions) – phần này chứa danh sách các hành động nội tại (internal actions) bên trong một trạng thái. Cú pháp : *action-label '/' action-expression* Có một số nhãn hành động (actiona-label) chuẩn được định nghĩa trong UML:
- entry – xác định một hành động đầu vào của một trạng thái. Ví dụ trước trạng thái đánh password (Typing Password – hình 3-54) ta phải có một hành động là định dạng các ký tự đánh vào dưới dạng ẩn (ví dụ như dùng ký tự * để thay thế).
- exit – xác định một hành động khi thoát ra khỏi trạng thái. Ví dụ sau khi thoát khỏi trạng thái đánh password ta phải đặt lại chuẩn thông thường cho việc nhận một ký tự từ bàn phím (không hiển thị dưới dạng ẩn nữa).
- do – xác định một hành động nội tại xảy ra trong trạng thái.
- include – xác định một trạng thái con hoặc một máy trạng thái con có liên quan.
- Trong những trường hợp khác, người dùng có thể định nghĩa các nhãn sự kiện theo cú pháp sau:

*Tên sự kiện '(' danh sách đối số được cách nhau bằng dấu phẩy ')' '[' điều kiện bảo vệ ']' '/' biểu thức hành động*

Hình 3–54 Trạng thái – đánh password

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 124

### 3.42. Trạng thái ghép (Composite state)

***Ngữ nghĩa***

Một trạng thái ghép bao gồm hai hay nhiều trạng thái con có thể được chia thành các nhóm tuần tự hoặc đồng thời (còn gọi là miền – region) với nhau. Một trạng thái con cũng có thể là một trạng thái ghép chứa các trạng thái con khác. Mỗi miền (region) con có thể có trạng thái bắt đầu giả và trạng thái kết thúc.

***Ký hiệu***

Hình 3–55 Trạng thái con tuần tự *(miền) (trạng thái bắt đầu giả)*

Hình 3–56 Trạng thái con đồng thời

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 125

### 3.42. Sự kiện (event)

Đó là một sự việc đáng chú ý, một sự kiện có thể gây ra một sự chuyển đổi trạng thái. Có thể có một số loại sau:

–Một điều kiện được chỉ định trở thành đúng (miêu tả bằng biểu thức có giá trị đúng/sai). Sự kiện xảy ra khi giá trị biểu thức chuyển từ sai sang đúng. Còn được gọi là sự kiện chuyển đổi (change event). Ví dụ (xem hình 3–57): *Connected, busy*

–Nhận một tính hiệu (signal) bên ngoài từ một đối tượng khác. Còn được gọi là một sự kiện tín hiệu (signal event)

–Nhận một lời gọi hành vi từ một đối tượng khác. Còn được gọi là sự kiện gọi (call event).

–Khoảng thời gian trôi qua sau một sự kiện được chỉ định (thường là thời điểm vào của trạng thái hiện tại). Ví dụ: *after (5 seconds)*

Một tín hiệu hoặc một sự kiện gọi có cú pháp như sau:

*Tên sự kiện '(' danh sách các tham số ')'* Ví dụ : xem hình 3–57.

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 126

## 3.43. Các chuyển đổi trạng thái đơn gian (simple transitions)

Đây là quan hệ giữa hai trạng thải, được biểu diễn bằng một cung hoặc một đường dẫn có hướng từ trạng thái đầu đến trạng thái cuối. Nhãn của một simple transition có dạng như sau:

*Event-signature '[guard-condition ']' '/' action-expression* Trong đó event–signature có cú pháp: *Tên sự kiện '(' danh sách tham số ')'* Guard-condition (điều kiện bảo vệ) là một biểu thức đúng/sai. Action-expression (biểu thức hành động) được thực thi khi có một sự chuyển đổi xảy ra.

Ví dụ: Right-mouse-down(location) [location in window] / object:=pick–object(location); Giải thích: khi phím chuột phải được bấm tại vị trí location, nếu vị trí location nằm trong cửa sổ màn hình thì đối tượng sẽ được gán bằng đối tượng được lấy tại vị trí location (thông qua hàm pickobject). *(các sự kiện) (sự chuyển đổi trạng thái)*

Hình 3–57 Một ví dụ về lược đồ trạng thái..

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 127

## 3.44. Các chuyển đổi trạng thái phức tạp (complex transitions)

Một complex transition có thể có nhiều trạng thái đầu và nhiều trạng thái cuối. Nó biểu diễn sự đồng bộ hoặc phân chia các tiến trình đồng thời cho các trạng thái con xảy ra đồng thời.

*(complex transition)*

Hình 3–58 Compllex Transiitiion

Giải thích: qúa trình cài đặt được chia thành nhiều tiến trình con xử lý đồng thời với nhau sau đó tất cả các tiến trình này sẽ bị hủy.

### *3.43.1. History Indicator*

Một history indicator dùng để ghi nhớ các trạng thái, nó được xác định bằng một hình chữ nhật chứa các trạng thái và một hình tròn có chữ H (history) bên trong. Khi một sự kiện xảy ra (ví dụ như hoạt động bị ngắt giữa chừng) sẽ có một tín hiệu resume được gởi đến History indicator, nó sẽ khôi phục lại các trạng thái ở lần sau cùng trong hình chữ nhật.

*(tín hiệu ngắt) (tín hiệu khôi phục) (history indicator) (vùng chịu ảnh hường của history indicator)*

Hình 3–59 History Indiicator

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 128

### 3.45. Các trạng thái đồng bộ (synch states)

Ký hiệu của các trạng thái đồng bộ là một hình tròn có chứa dấu (*) bên trong. Xét ví dụ *(đồng bộ)*

Hình 3-60 Trạng thái đồng bộ

Giải thích: trong ví dụ trên ta có thể thấy trạng thái build frame (xây khung nhà) phải đồng bộ với việc gắn các thiết bị điện nền tảng. Kế đó, sau khi đi dây (theo các khung) cho các thiết bị điện trong nhà sẽ được tiến hành đồng bộ với việc lợp mái và sau đó mới tới việc đổ tường và gắn các thiết bị điện bên ngoài.

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 129

### 3.46. Lược đồ hoạt động (Activity Diagram)

*(trạng thái bắt đầu) (trạng thái hoạt động) (xét điều kiện để tiếp tục thực hiện) (phân thành các nhánh hoạt động đồng thời) (kết hợp các nhánh hoạt động đồng thời lại) (trạng thái kết thúc)*

Hình 3-61 Lược đồ hoạt động (Actiiviitiies Diiagram)

***Ngữ nghĩa***

_ Là một lược đồ thể hiện các luồng của các hành động

_ Là một biến thể của máy trạng thái trong đó các trạng thái thể hiện sự hoạt động của các hành động và sự chuyển đổi được khởi động khi các hành động hoàn thành

_ Sử dụng lược đồ hoạt động trong trường hợp tất cả hoặc hầu hết các sự kiện thể hiện sự hoàn tất của các hành động được sản sinh bên trong tiến trình.

***Chức năng***

_ Mô hình hóa các hoạt động thế giới thực

_ Mô hình hóa các thao tác

_ Nắm bắt các công việc sẽ được thực hiện khi một hành vi được thực hi trong hệ thống.

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 130

## 3.47. Các thành phần chính

### *3.46.1. Các trạng thái hành động (action state)*

Là một dạng đơn giản của trạng thái, nó sẽ trực tiếp biến đổi sang trạng thái khác sau khi thực hiện một hành động nào đó. Một action state không thể có các chuyển đổi nội tại trong trạng thái. Ký hiệu của một action state là một hình tương tự hình chữ nhật nhưng hai cạnh bên được thay bằng hai cung tròn, bên trong chứa biểu thức hành động

Hình 3-62 Các actiion state..

### *3.46.2. Các quyết định (decisions)*

Lược đồ hoạt động sử dụng các quyết định khi cần có những chuyển đổi hợp lý tới các trạng thái hành động khác với những điều kiện do người dùng đặc tả, hướng chuyển đổi phụ thuộc vào giá trị đúng/sai của biểu thức quyết định.

*(decision)*

Hình 3-63 Decisions

Giải thích: trong ví dụ trên khi tính toán tổng chi phí cho một việc nào đó, nếu số tiền nhỏ hơn 50 USD thì có thể chuyển ngay sang trạng thái hành động charge customer's account, nếu lớn hơn 50 USD phải chuyển sang trạng thái hành động là xin thẩm quyền (getauthorization) rồi mới chuyển sang trạng thái hành động chargecustomer's account.

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 131

## 3.48. Swimlanes và đối tượng trong lược đồ hoạt động.

Những hành động hoặc những hoạt động con có thể tổ chức vào các swimlane (các đường phân luồng công việc). Swimlane có tác dụng nhóm các hành động lại theo một mục đích cụ thể (ví dụ như các hành động cùng xảy ra tại một nơi, các hành động của cùng một tác nhân...).

Biểu diễn của swimlane là một hình chữ nhật có tên ngay phía trên bên trong hình chữ nhật. Xem ví dụ hình 3-64, ta thấy các actionstate được chia vào trong ba swimlane là Customer, sales, và stockroom. Ta thấy các action state như Reuest Service (yêu cầu phục vụ), Pay (trả tiền), collect order (chọn) đều là những hành động của khách hàng (customer) nên chúng được tổ chức vào cùng một swimlane. Trong lược đồ hoạt động cũng có thể xuất hiện các đối tượng, nó đóng vai trò là đầu vào hoặc đầu ra cho các action state, hoặc chỉ dùng để minh họa đối tượng bị ảnh hưởng bởi một hoạt động nào đó trong lược đồ. Nếu đối tượng là đầu ra hoặc bị ảnh hưởng bởi một hành động thì sẽ có một mũi tên đứt nét đi từ hành động đến đối tượng, ngược lại nếu là đầu ra thì mũi tên có hướng từ đối tượng tới hành động. *( tên của các swimlane) (đối tượng trong lược đồ hoạt động) (ba swimlanes trong lược đồ là Customer, sales, storckroom)*
Hình 3-64 Các Swimlane và các đối tượng

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 132

## 3.49. Các biểu tượng điều khiển

Các biểu tượng điều khiển cung cấp những ký hiệu dùng để diễn tả cho một loại thông tin nào đó cần được đặc tả trong qúa trình chuyển đổi giữa các hành động. Chúng được đưa vào lược đồ chỉ nhằm mục đích nhấn mạnh những thông tin mà người dùng thêm vào. Chúng có hai loại:

–Tín hiệu nhận : biểu diễn bằng một hình chữ nhật có một đầu lõm, một mũi tên đứt nét được vẽ từ đối tượng tới hình chữ nhật cho biết đây là đối tượng gửi tín hiệu.

–Tín hiệu gửi : biểu diễn bằng một hình chữ nhật có một đầu lồi, và cũng có thể có một mũi tên vẽ từ hình chữ nhật tới đối tượng nhận tín hiệu. *(tín hiệu gửi và đối tượng nhận tín hiệu) (tín hiệu nhận và đối tượng gửi tín hiệu)*

Hình 3-65 Biểu tượng gửi và nhận tín hiệu

Giải thích: khi hành động bật máy pha cà phê được thực hiện (turn on machine) nó sẽ gửi tín hiệu là máy bật (turn on) đến đối tượng là bình pha cà phê (coffeepot) sau đó sẽ chuyển qua trạng thái pha cà phê (Brew coffee), kế đó khi đối tượng bình pha cà phê (coffeepot) báo đèn đã tắt thì sẽ chuyển qua hành động chế cà phê ra ly (Pour coffee).

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 133

### 3.50. Lược đồ thành phần (Component Diagram)

***Ví dụ***

*(các trang web liên kết với các thành phần khác trong hệ thống) (thành phần thực thi) (các thư viện) (các component (thành phần) trong hệ thống)*

Hình 3-66 Lược đồ thành phần (Component Diagram)

***Ngữ nghĩa***

_ Là lược đồ thể hiện sự phụ thuộc giữa các thành phần phần mềm với nhau.

_ Là lược đồ ghi nhận cấu trúc vật lý của phiên bản cài đặt.

***Chức năng***

_ Tổ chức source code

_ Xây dựng một phiên bản thực thi được

_ Đặc tả cấu trúc cơ sở dữ liệu vật lý

_ Được xây dựng như một phần đặc tả cấu trúc hệ thống

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 134

### 3.51. Các thành phần chính

Thành tố chính trong lược đồ thành phần là các thành phần (component), chúng biểu diễn các thành phần trong việc triển khai hệ thống bao gồm chương trình (mã nguồn của phần mềm, mã nhị phân, chương trình thực thi, các trang

web...), và các sưu liệu cho hệ thống... Ký hiệu của component trong lược đồ là một hình chữ nhật có hai hình chữ nhật nhỏ gắn ở cạnh bên của hình, bên trong chứa tên của component với cú pháp: *Component –type*

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 135

Một thực thể component có tên và kiểu được gạch dưới và cách nhau bởi dấu hai chấm. Component cũng có thể chứa các đối tượng (tại thời điểm thực thi) bên trong bản thân nó.

Hình 3-67 Một component chứa các object bền trong..

Ngoài ra trong lược đồ thành phần còn có các quan hệ phụ thuộc giữa các thành phần, quan hệ phục thuộc chỉ ra sự phụ thuộc lẫn nhau giữa các thành phần khi triển khai hệ thống. Một thành phần có thể định nghĩa các giao tiếp cho các thành phần khác sử dụng, tên giao tiếp đặt ngay bên cạnh biểu tượng của giao tiếp (xem phần 3.5.2.4.2)

*(component) (interface)*

Hình 3-68 Component và các giao tiếp (interface)

## Chương 3

## HỆ THỐNG KÝ HIỆU (UML Notation) 136

### 3.52. Lược đồ triển khai (Deployment Diagram)

***Ví dụ***

*(kết nối giữa các node khi triển khai hệ thống) (các node triển khai trong hệ thống)*

Hình 3-69 Lược đồ triển khai (Deplloyment Diiagram)

***Ngữ nghĩa***

_ Là một đồ thị các node được liên kết bởi các kết nối

_ Là lược đồ thể hiện cấu hình của các thành phần đang xử lý tại thời điểm hệ thống đang chạy, các thành phần phần mềm, các tiến trình và các đối tượng đang sống trong chúng

***Chức năng***

_ Đặc tả sự phân bố các thành phần trong hệ thống_

_ Bổ sung cho lược đồ thành phần bằng cách chỉ ra cấu hình của các thành phần xử lý khi hệ thống đang chạy và các thành phần phần mềm.

_ Thể hiện sự chuyển đổi của các thành phần từ node ngày sang node khác hoặc các đối tượng từ thành phần này sang thành phần khác.

***Các thành phần chính***

Thành phần chính trong lược đồ triển khai là các node (nút), đây là các thành tố vật lý, tồn tại tại thời điểm hệ thống đang thực thi, chúng biểu diễn các tài nguyên máy tính. Các component sống (live) trên các node và các node biểu diễn việc triển khai vật lý của các thành phần. Ký hiệu của các node trong lược đồ có thể là các biểu tượng trong thế giới thực hoặc là một hình khối với tên node bên trong.

Hình 3–70 Một số thể hiện của Node

Một node có thể chứa các component bên trong và những component này có thể liên hệ với nhau bằng các quan hệ phụ thuộc, trong khi đó nếu hai node có quan hệ với nhau, chúng sẽ được liên kết bằng một connection (kết nối), và các đối tượng hay các thành phần bên trong node sẽ liên lạc, quan hệ với nhau thông qua kết nối này.

Hình 3–71 Node chứa các component có quan hệ với nhau

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 137

## 3.53. Tóm tắt

# Chương 3

# HỆ THỐNG KÝ HIỆU (UML Notation) 138

## 3.54. Tóm tắt

Chương ba trình bày về các loại lược đồ cùng với hệ thống các ký hiệu được định nghĩa trong UML, bên cạnh đó là một số cú pháp và khuôn mẫu dùng để định nghĩa ký hiệu cho các khái niệm mới mà người dùng muốn định nghĩa. Như ta thấy, các lược đồ trong UML có những đặc điểm riêng biệt, mỗi lược đồ thể hiện một khía cạnh riêng của hệ thống cần xây dựng. Thông qua chín lược đồ ta có thể thấy UML có thể mô hình hóa nhiều hệ thống từ đơn giản đến phức tạp. Trong chương kế tiếp sẽ trình bày về ứng dụng của UML trong qui trình phát triển phần mềm RUP, đặc biệt là cách sử dụng các lược đồ để đặc tả các mô hình trong các giai đoạn khác nhau của qui trình.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 139

## 4.1. Giới thiệu

Ngày nay, chất lượng phần mềm phụ thuộc vào rất nhiều yếu tố khác nhau, trong đó quy trình công nghệ đóng một vai trò quan trọng vào loại bậc nhất. Trước tình hình đó các công ty phần mềm lớn trên thế giới đã nhanh chóng đưa ra nhiều quy trình công nghệ khác nhau, trước tiên là phục vụ cho nhu cầu phát triển phần mềm trong công ty mình, sau đó là công bố rộng rãi để thu thập ý kiến từ những đối tác nhằm tìm kiếm lời giải cho bài toán tối ưu hóa quy trình công nghệ của mình. Hiện nay có rất nhiều quy trình phát triển phần mềm khác nhau được sử dụng trong các công ty phát triển phần mềm như : Water fall Process, OPEN Process, Object-Oriented Software Process,Unified Process... mỗi quy trình đều có những ưu/khuyết điểm riêng của mình nhưng nổi bật nhất và ngày càng được sử dụng rộng rãi nhất là Unified Process của hãng Rational. RUP (Rational Unified Process) ngày càng được sử dụng và được hỗ trợ rộng rãi từ những đối tác sử dụng, mặc dù RUP mới được phát triển trong những năm gần đây, nhưng sự xuất hiện của RUP đánh dấu một xu hướng phát triển mới trong giai đoạn bùng nổ của ngành công nghệ phần mềm.

## 4.2. Giới thiệu Rational Unified Process (RUP)

### *4.2.1. Khái quát về RUP*

RUP – Rational Unified Process – là quy trình công nghệ phần mềm được phát triển bởi hãng Rational, RUP hỗ trợ các hoạt động phát triển phần mềm theo nhóm, phân chia công việc theo thứ tự cho từng thành viên của nhóm trong từng giai đoạn khác nhau của quy trình phát triển phần mềm. RUP sử dụng ngôn ngữ UML để mô hình hóa và cung cấp những hướng dẫn để sử dụng UML một cách hiệu quả nhất.

Hình 4-1 Tóm tắt lịch sử phát triển của RUP

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 140

## 4.3. Giới thiệu Rational Unified Process (RUP)

Mục đích chính của RUP là giúp sản xuất những phần mềm có chất lượng cao thỏa mãn yêu cầu của người dùng cuối, trong khuôn khổ thời gian và ngân sách.

RUP được phát triển và duy trì bởi hãng Rational, đảm bảo qui trình luôn được cải tiến hoàn chỉnh hơn trên cơ sở những kinh nghiệm phản hồi từ đối tác sử dụng, sự tiến hóa và những cách vận dụng tốt nhất trong thực tế. RUP nâng cao năng suất làm việc của nhóm. Các thành viên trong nhóm có hướng dẫn, khuôn mẫu, công cụ hỗ trợ và đều sử dụng một ngôn ngữ chung, một qui trình chung do đó có sự thống nhất trong cách nhìn và phương hướng phát triển một phần mềm. Sử dụng UML để hỗ trợ tất cả các giai đoạn (phase) trong quy trình phát triển phần mềm, hoạt động chính của RUP là tạo, cải tiến và quản lý các loại mô hình. Ngoài ra, RUP còn hướng dẫn làm một lượng lớn các sưu liệu cho phần mềm, RUP nhấn mạnh việc phát triển những mô hình giàu ngữ nghĩa biểu diễn cho hệ thống dưới góc độ của người phát triển. Ngày nay, RUP được hỗ trợ bởi các công cụ, giúp tự động hóa phần lớn quy trình phát triển phần mềm. Các công cụ hỗ trợ RUP có thể kể đến là quản lý đề án, phân công nhân sự, tạo lập và quản lý mô hình, kiểm chứng...

Không một quy trình nào có thể phù hợp cho tất cả các tổ chức phát triển phần mềm. RUP được phát triển trên cơ sở cấu trúc đơn giản và rõ ràng, có thể cấu hình lại cho phù hợp với nhu cầu của tổ chức sử dụng. RUP phù hợp cho những nhóm phát triển nhỏ cũng như những tổ chức lớn. RUP là tập hợp những công việc và kinh nghiệm đã được vận dụng hiệu quả nhất trong thực tế. Việc phát triển theo RUP cho phép những nhóm phát triển có được một số thuận lợi so với những nhóm khác.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 141

## 4.4 Giới thiệu Rational Unified Process (RUP)

### *4.4.1. Kiến trúc của RUP*

***Các giai đoạn (phase) cua quy trình***

RUP là quy trình bao gồm nhiều bước lặp để xây dựng hệ thống gọi là các chu kỳ (cycle). Mỗi chu kỳ cho kết quả là một phiên bản release của phần mềm bao gồm mã nguồn trong các thành phần (componennt) có thể biên dịch và thực thi. Một chu kỳ được chia làm bốn phase là khởi đầu (inception), triển khai (elaboration), xây dựng (contruction), chuyển giao (transition).

_ Inception : xác định phạm vi dự án, các tài nguyên cần thiết và phác thảo chức năng cho người sử dụng (business case).

_ Elaboration : phân tích vấn đề, lập kế hoạch dự án, đánh giá rủi ro và xác định kiến trúc hệ thống.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 142

## 4.5 Giới thiệu Rational Unified Process (RUP)

_ Construction : phát triển các thành phần (component), tích hợp vào sản phẩm và kiểm chứng các chức năng

_ Transition : chuyển giao sản phẩm đến khách hàng, huấn luyện khách hàng cách sử dụng, bảo trì đồng thời điều chỉnh một số chức năng cần thiết.

*Các giai đoạn (phase)*

Inception Elaboration Construction Transition

Khởi đầu Triển khai Xây dựng Chuyển giao

Trực quan hóa

Kiến trúc cơ bản

Các tính năng khởi đầu

Sản phẩm

release

Hình 4–2 Các phase của RUP

Ngoài ra, trong một cycle (chu kỳ) còn bao gồm nhiều bước lặp con (iteration). Mỗi iteration cũng cho kết quả là một phiên bản release và được hoàn thành qua quá trình thực hiện một dãy các công việc cụ thể gọi là luồng công việc (workflow). Các công việc được phân chia vào các workflow để xây dựng phiên bản release cho một iteration. Các luồng công việc cơ bản là business modeling (mô hình hóa nghiệp vụ), requirements (xác định yêu cầu), analysis (phân tích), design (thiết kế), cài đặt (implementation) và deployment. Ngoài ra còn có các luồng công việc hỗ trợ quản lý dự án (project management), quản lý cấu hình và thay đổi (configuration and change management) và quản lý môi trường (enviroment).

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 143

## 4.6. Giới thiệu Rational Unified Process (RUP)

Hình 4–3 Các wokflow và iteration

Hình trên mô tả kiến trúc của RUP theo hai trục :

– Trục hoành tổ chức theo thời gian diễn tả một chu kỳ (cycle) bao gồm các phase và các bước lặp (iteration).

– Trục tung tổ chức theo các luồng công việc (workflow) bao gồm các hoạt động, các thành phần và người thực hiện.

*4.6.1. Cấu trúc tĩnh của quy trình*

RUP mô tả ai (Who) đang làm gì (What), làm như thế nào (How) và khi nào (When). RUP định nghĩa bốn thành phần sau _ Worker –Who – định nghĩa công việc và trách nhiệm của mỗi cá nhân, hoặc một số cá nhân làm việc với nhau trong nhóm. Ví dụ:

–Project Manager : trưởng dự án.

–System Analyst : phân tích viên hệ thống.

–Tester : kiểm tra viên.

–Activities – How – hoạt động của một worker là một tập công việc có một mục đích rõ ràng. Ví dụ: Lập kế hoạch của trưởng dự án

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 144

## 4.7. Giới thiệu Rational Unified Process (RUP)

–Tìm các use case và các actor cho hệ thống của phân tích viên hệ thống.

–Artifacts – What – là những thông tin được phát sinh, thay đổi hoặc sử dụng bởi quy trình. Ví dụ: Mô hình Use Case (Use Case model), Mô hình thiết kế (Design model). Các sưu liệu (document). Các thành phần thực thi.

–Workflows – When – mô tả cách thức tiến hành các hoạt động theo trình tự và vai trò của mỗi worker.

Hình 4-4 Ví dụ về một luồng công việc và vai trò của các worker

*4.7.1. Các đặc điểm phân biệt của RUP so với các quy trình phát triển phần mềm khác*

*RUP là quy trình hướng chức năng hệ thống (use case)*

Thay cho cách mô tả chức năng truyền thống, RUP sử dụng mô hình Use Case để mô hình hóa chức năng cho từng loại người sử dụng. Ngoài ra, các chức năng (use case) còn đóng vai trò dẫn dắt quy trình phát triển đến các bước phân

tích, thiết kế và kiểm chứng. Dựa trên use case, người phát triển tạo một loạt các mô hình phân tích (analysis model), thiết kết (design model) cài đặt (implementation model) và xem xét các mô hình đó có bao gồm các thành phần đáp ứng đầy đủ cho việc thực hiện chức năng hệ thống hay không. Quy trình phát triển theo đó sẽ thực hiện dãy các công việc dựa trên use case.
Thiết kế Cài đặt Kiểm chứng
Phân tích Xác định yêu cầu
Các use case liên kết các workflow này với nhau.
Hình 4–5 RUP hướng chức năng hệ thống

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 145

## 4.8. RUP tập trung vào kiến trúc phần mềm

Kiến trúc là cái nhìn tổng thể về thiết kế của hệ thống, loại bỏ chi tiết và tập trung vào những tính chất quan trọng. RUP cung cấp phương hướng để từng bước xác định kiến trúc của hệ thống, đáp ứng các yêu cầu cho việc thay đổi và tái sử dụng của phần mềm. RUP xác định một mối liên hệ giữa kiến trúc với use case. Kiến trúc phải được xây dựng sao cho đáp ứng tất cả chức năng trong hiện tại và tương lai. Việc xác định kiến trúc đòi hỏi phải xác định những chức năng nào là quan trọng bậc nhất và chủ yếu của hệ thống. Kiến trúc phần mềm được xác định và cải tiến từng bước qua các phase. Kiến trúc phần mềm *Các giai đoạn (phase)* Inception Elaboration Construction Transition Khởi đầu Triển khai Xây dựng Chuyển giao
Hình 4-6 RUP tập trung vào kiến trúc phần mềm

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 146

## 4.9. RUP là quy trình lặp và tăng trương từng bước

Phát triển một phần mềm phức tạp đòi hỏi không chỉ thời gian mà còn kỹ thuật phân chia hệ thống thành những phần nhỏ. Quy trình gồm nhiều iteration (bước lặp) để xây dựng phần mềm. Mỗi tập chức năng của hệ thống sẽ được phát triển trong một iteration và kết quả là sự hoàn chỉnh về tổng thể ngày càng gia tăng. Các iteration phải được thực hiện theo kế hoạch và có kiểm soát. Một iteration là một trình tự các hoạt động được lên kế hoạch theo một tiêu chuẩn xác định và cho kết quả là một phiên bản release của phần mềm. Trong mỗi bước, người phát triển chọn một nhóm các chức năng và tiến hành phân tích (analysis), thiết kế (design), cài đặt (implementation) và kiểm chứng (test) các chức năng này. Nếu iteration đáp ứng được mục đích đề ra thì chuyển sang một iteration mới với một nhóm các chức năng kế tiếp.

Thiết kế
Cài đặt
Kiểm chứng
Phân tích
Xác định
yêu cầu
Inception Elaboration Construction T r a n s i t i o n
Bước lặp mở đầu
Lặp1
Lặp2
Lặp n
Lặp n+1
Lặp n+2
Lặp m

Lặp m+1
Một bước lặp trong phase
Elaboration.
Release Release Release Release ReleaseRelease Release
Hình 4–7 RUP là quy trình lắp và tăng trưởng từng bước

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 147

## 4.10. Ứng dụng UML trong RUP

UML cung cấp một ngôn ngữ cho mô hình hóa, trực quan hóa và làm sưu liệu phần mềm. Nhưng nếu không có một quy trình phát triển phần mềm đặc thù ứng dụng UML thì sẽ không tận dụng khả năng mạnh mẽ cũng như những khái niệm mô hình hóa đa dạng của UML. RUP là quy trình phát triển phần mềm được xây dựng trên nền tảng UML và những yêu cầu thực tế trong công nghệ phần mềm. RUP hướng dẫn cách sử dụng UML hiệu quả nhất hiện nay. UML được sử dụng trong các luồng công việc (workflow) định nghĩa bởi RUP. Mỗi luồng công việc sử dụng những mô hình khác nhau liên quan đến các khái niệm và lược đồ UML khác nhau.
Hình 4–8 Các luồng cồng việc và các mồ hình RUP

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 148

## 4.11. Mô hình hóa nghiệp vụ (business modeling)

Mô hình hóa nghiệp vụ nhằm mục đích nắm bắt quy trình hoạt động của tổ chức nơi cần xây dựng hệ thống phần mềm bao gồm quy trình nghiệp vụ và cách thức thực hiện, đối tượng thực hiện nghiệp vụ và đối tượng thao tác của nghiệp vụ, các tác nhân bên ngoài có giao tiếp và ảnh hưởng đến hoạt động của tổ chức. Mô hình hóa nghiệp vụ là tùy chọn cho từng dự án. Business modeling định nghĩa hai mô hình Business Use Case (mô hình nghiệp vụ) và Business Object (mô hình đối tượng nghiệp vụ).

### *4.11.1. Mô hình nghiệp vụ (Business Use Case)*

Mô hình nghiệp vụ chủ yếu bao gồm lược đồ Use Case của UML với một số mở rộng cho Unified Process cụ thể là định nghĩa thêm business actor, business worker và business use case từ cơ chế mở rộng của UML.

_ Business actor : các tác nhân bên ngoài có liên quan hay tác động đến hoạt động của tổ chức ví dụ như khách hàng và có ký hiệu sau

Tên business actor

Hình 4-9 Ký hiệu busines actor

_ Business worker : nhân viên thực hiện nghiệp vụ, thường đóng một vai trò cụ thể trong tổ chức ví dụ như kế toán viên, thủ kho, quản đốc và có ký hiệu sau

Tên business worker

Hình 4-10 Ký hiệu busines actor

_ Business use case : nghiệp vụ được thực hiện bởi một business worker ví dụ như mở tài khoản cho khách hàng hay nhận các đơn đặt hàng và có ký hiệu sau:

Tên nghiệp vụ

Hình 4-11 Ký hiệu busines use case

Business use case model tập trung vào mô hình hóa các quy trình nghiệp vụ, người thực hiện và mối liên hệ giữa các tác nhân bên ngoài với tổ chức.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 149

## 4.12. Ứng dụng UML trong RUP

Mở tài khoản
Nhân viên giao tiếp
khách hàng
Khách hàng *mở tài khoản cho (Nhân viên giao tiếp khách hàng thực hiện nghiệp vụ mở tài khoản cho khách hàng)*
Hình 4–12 Ví dụ một lược đồ Use Case trong mô hình Busines Use Case

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 150

## 4.13. Mô hình đối tượng nghiệp vụ (Business Object)

Mô hình Business Object sử dụng chủ yếu lược đồ lớp (Class diagram) mở rộng cho RUP dựa trên cơ chế mở rộng của UML bao gồm organization unit (đơn vị tổ chức), business entity (thực thể nghiệp vụ).
–Organization unit : các đơn vị cấu trúc của tổ chức thường tương ứng với các phòng, ban hay bộ phận trong tổ chức và có ký hiệu Tên organization unit
Hình 4–13 Ký hiệu organization unit

–Business entity : đối tượng thao tác của nghiệp vụ thường là dữ liệu, các loại hồ sơ. Các nghiệp vụ thường lấy dữ liệu, thay đổi, lưu trữ các business entity ví dụ như hồ sơ khách hàng hay các loại mặt hàng. Business entity có ký hiệu sau

Tên busines entity

Hình 4–14 Ký hiệu busines entiity

Mô hình đối tượng nghiệp vụ mô tả những business worker nào sử dụng những tài nguyên, tài liệu gì của tổ chức và sử dụng như thế nào để thực hiện nghiệp vụ cụ thể.

Nhân viên giao tiếp khách hàng

Tài khoản *tạo mới*

*(Nhân viên thêm tài khoản khách hàng vào hồ sơ các tài khoản). (Nghiệp vụ mở một tài khoản)*

Hình 4–15 Ví dụ về lược đồ lớp (Cllass diiagram) trong mồ hình Busines Objject

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 151

## 4.13. Xác định yêu cầu (requirements)

Mục đích của giai đoạn xác định yêu cầu là có được một sự thống nhất giữa khách hàng với các nhà phát triển về những gì mà hệ thống sẽ thực hiện. Một sưu liệu trực quan sẽ được xây dựng qua mô hình hóa các chức năng mà hệ thống hỗ trợ cho từng loại người sử dụng. Giai đoạn này sử dụng mô hình chức năng (Use Case model) bao gồm chủ yếu lược đồ Use Case của

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 152

UML. Actor là người sử dụng hệ thống. Use case là chức năng hệ thống được thực hiện khi có tác động từ actor.

Hình 4–16 Mô hình Use Case và các lược đồ UML

Các chức năng hệ thống có thể được lựa chọn từ các nghiệp vụ (business use case) trong mô hình business use case để xác định những nghiệp vụ nào sẽ được hỗ trợ và cho người sử dụng nào.

Mở tài khoản

Nhân viên giao tiếp khách hàng

*(Xác định chức năng hệ thống cho mỗi loại người sử dụng qua các nghiệp vụ)*

*(Use Case)* (*Actor)*

Hình 4–17 Ví dụ về lược đồ Use Case trong mô hình Use Case

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 153

## 4.14. Phân tích (analysis)

Trong mỗi iteration, để phát triển từng nhóm các use case, các nhà phát triển tiến hành mô tả rõ ràng hơn các yêu cầu chức năng này bằng ngôn ngữ của những người phát triển phần mềm. Workflow phân tích bắt đầu mô hình hóa hoạt động bên trong cũng như xây dựng cơ bản kiến trúc của hệ thống nhưng chỉ dừng lại ở mức quan niệm, chưa xem xét đến các khía cạnh chi tiết cũng như các

yêu cầu phi chức năng. Trong workflow này, RUP định nghĩa mô hình Analysis sử dụng lược đồ lớp (Class diagram) với một số mở rộng và các lược đồ mô hình hóa tương tác như Sequence hay Collaboration của UML.

Hình 4–18 Mô hình phần tích và các lược đồ UML

RUP mở rộng lược đồ Class cho workflow analysis bằng cách định nghĩa thêm boundary, control và entity từ cơ chế mở rộng của UML.

_ Boundary : lớp trong hệ thống đảm nhận vai trò giao tiếp giữa hệ thống với các tác nhân bên ngoài ví dụ như giao diện người sử dụng. Boundary có ký hiệu sau:

Tên boundary

Hình 4–19 Ký hiệu boundary

## Chương 4
## ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 154

–Control : lớp mang chức năng xử lý, điều khiển các hoạt động xử lý, tính toán. Control có ký hiệu sau tên entity

_ Entity : lớp đại diện cho thực thể nghĩa là các đối tượng dữ liệu có thể lưu trữ, tham chiếu hay sửa đổi. Entity có ký hiệu sau tên entity

Mô hình Analysis có chức năng làm rõ các use case, mô tả các quan hệ giữa ba thành phần boundary, control, entity trong hệ thống để thực hiện use case.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 155

Nhân viên giao tiếp khách hàng
Giao diện m ở tài khoản
Xử lý m ở tài khoản
Tài khoản
*(Chức năng mở tài khoản của hệ thống) (Nhân viên sử dụng giao diện mở tài khoản để thao tác trên đối tượng tài khoản thông qua xử lý mở tài khoản) (boundary)* control entity
Hình 4–22 Ví dụ về lược đồ Class trong mô hình Analysis
Lược đồ tương tác sử dụng chủ yếu trong mô hình Analysis là Collaboration của UML. Lược đồ này mô tả cơ chế vận hành của hệ thống, các hoạt động của các thành phần trong hệ thống theo một trình tự xác định cũng như quá trình tương tác giữa các thành phần thông qua các thông điệp (message) để thực hiện một chức năng cụ thể.
: Nhân viên giao tiếp khách hàng
: Giao diện mở tài khoản
: Xử lý mở tài khoản
: Tài khoản
1: yêu cầu chức năng mở tài khoản
2: hiển thi giao diện nhập dữ liệu
3: nhập dữ liệu tài khoản
4: yêu cầu mở tài khoản
5: mở tài khoản
6: tạo mới tài khoản
7: đóng chức năng mở tài khoản
*Lược đồ Collaboration mô tả cơ chế tương tác giữa các thành phần boundary, control và entity để thực hiện chức năng mở tài khoản. thông điệp (message)*

*thứ tự thi hành thông điệp vào chính đối tượng gửi.*

Hình 4-23 Ví dụ về một lược đồ Collaboration trong mô hình Analysis

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 156

## 4.15. Thiết kế (design)

Design là workflow sử dụng các kết quả ở mức phân tích (analysis) cho một nhóm use case để tiếp tục phát triển hệ thống về kiến trúc, chi tiết các chức năng cũng như lưu ý đến các yêu cầu phi chức năng, các ràng buộc có thể có. Các vấn đề quan tâm trong workflow này có thể kể đến là

_ Môi trường phát triển phần mềm, ngôn ngữ lập trình, hệ điều hành, các yêu cầu xử lý đồng thời hay phân tán và môi trường cơ sở dữ liệu.

_ Định dạng kiến trúc hệ thống trên cơ sở bảo toàn kiến trúc ở mức phân tích nhưng có thể tiếp tục phân rã các hệ thống con thành các đơn vị nhỏ hơn để có thể quản lý và cài đặt một cách hiệu quả.

_ Thiết kế các giao diện (interface) chủ yếu để có thể giao tiếp với các hệ thống con và phát triển tương đối độc lập các hệ thống con này. Workflow Design định nghĩa mô hình Design bao gồm các lược đồ Class, lược đồ Deployment và các lược đồ tương tác như Colaboration và Sequence từ UML. Design sử dụng chủ yếu lược đồ Sequence để mô hình hóa tương tác. Ngoài ra còn sử dụng các lược đồ mô tả trạng thái và hoạt động là State Chart và Activity Graph.

Hình 4-24 Mô hình thiết kế và các lược đồ UML

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 155

Lược đồ Class được xây dựng cho một môi trường cụ thể. Các kiểu dữ liệu hay các lớp đều mang tính chất đặc trưng của môi trường cài đặt và ngôn ngữ lập trình.

Nhân viên giao tiếp khách hàng

CTaikhoanUI

CtaikhoanCtrl 1

CTaikhoanRs +m_lSotien : long

+m_TaikhoanCtrl111

*Lược đồ lớp cho use case mở tài khoản với các bổ sung so với mức phân tích. tham chiếu multiplicity*

+m_TaikhoanRs

*thuộc tính có kiểu dữ liệu của ngôn ngữ lập trình cụ thể. navigation*

Hình 4–25 Ví dụ về lược đồ lớp trong mô hình Design

Lược đồ Sequence mô tả tương tác giữa các lớp để thực hiện chức ăng trong đó quan tâm đến trình tự thời gian là điểm khác biệt chủ yếu so ới lược đồ Collaboration.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 155

: Nhân viên giao tiếp khách hàng

: CTaikhoanUI : CTaikhoanCtrl : CTaikhoanRs

1: OnClick()
2: DisplayUI()
3: GetData()
4: UpdateData()
5: NewAccount()
6: InsertAccount()
7: Close()

*thời gian tồn tại thông điệp (focus of control) hàm tương ứng với lớp CTaikhoanCtrl. Lược đồ Sequence mô tả tương tác ở mức thiết kế.*

Hình 4–26 Ví dụ về lược đồ Sequence trong mô hình Design

## 4.16. Cài đặt (implementation)

Implementation sử dụng các kết quả từ analysis và design để tiến hành mã hóa và cài đặt các thành phần (component) dưới dạng mã nguồn, các script, các file nhị phân hay các file thi hành. Mục đích của workflow này là

_ Tổ chức cài đặt các lớp vào các thành phần (component) và phân bố các thành phần vào các node trên mô hình Deployment.

_ Kiểm chứng các thành phần.

_ Tích hợp các phần cài đặt của những người hay những nhóm phát triển vào hệ thống.

Mô hình Implementation bao gồm lược đồ Component của UML. Mỗi component có thể bao gồm nhiều lớp và cung cấp dịch vụ của nó qua các giao diện (interface). Component là các thành phần độc lập và các component của hệ thống có thể được cài đặt trên các môi trường khác nhau.

*Implementation Design*

Interface tài khoản

+Motaikhoan() <<Interface>> Tài khoản

+Motaikhoan() *Đưa lớp tài khoản vào component tài khoản để thêm dịch vụ mở tài khoản cho component. realize (cung cấp)*

Component tài khoản

Interface tài khoản

Hình 4–27 Ví dụ tổ chức một lớp vào một component *sử dụng (dependency)*

Chương trình chính

Component tài khoản

Interface tài khoản

*Component tài khoản đóng gói các dịch vụ liên quan đến tài khoản và cung cấp các dịch vụ đó qua interface tài khoản.*

*Interface tài khoản bao gồm dịch vụ mở tài khoản và các dịch vụ khác về tài khoản.*

MFC 6.0

Hình 4-28 Ví dụ về lược đồ Component trong mô hình Component

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 159

## 4.16. Kiểm chứng (test)

RUP là một quy trình lặp. Mỗi bước như vậy đều trải qua kiểm chứng nên cho phép người phát triển phần mềm có thể phát hiện lỗi rất sớm. Test có thể bao gồm

_ Kiểm tra quá trình tương tác giữa các đối tượng.

_ Kiểm tra tính đúng đắn của việc tích hợp các component trong phần mềm.

_ Kiểm tra các chức năng có được cài đặt chính xác hay không. Mô hình Test liên quan đến tất cả các mô hình trong iteration và tham chiếu đến các lược đồ thích hợp.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 160

## 4.17. Phát triển một ứng dụng quan lý giáo vụ theo RUP

### *4.17.1. Giới thiệu ứng dụng*

Đây là ứng dụng mà hiện trạng đã được nắm bắt bởi nhiều người và là lĩnh vực khá quen thuộc trong mô hình hóa và phát triển phần mềm. Giáo vụ đại học là bài toán tương đối đặc thù trong công nghệ phần mềm và có ứng dụng thực tế. Việc phân tích và thiết kế không quá phức tạp nhưng liên quan đến một tập khái niệm mô hình hóa tương đối đầy đủ của UML và điều này giúp người phát triển nắm bắt khả năng cũng như cách ứng dụng UML trong quy trình phát triển phần mềm RUP.

Mục đích của ứng dụng là minh họa chi tiết các luồng công việc trong RUP sử dụng UML trong bước lặp (iteration) thứ nhất.

*4.17.2. Sơ lược yêu cầu và đặc điểm*

Ứng dụng quản lý giáo vụ bao gồm quản lý giảng viên và quản lý sinh viên. Việc quản lý giảng viên chỉ đơn giản là lưu trữ thông tin giảng viên nhằm da dạng hóa quy trình nghiệp vụ phục vụ cho quá trình minh họa. Việc quản lý sinh viên tuân theo một số yêu cầu sau

_ Quản lý thông tin sinh viên : bao gồm việc lưu trữ, tra cứu, tìm kiếm dữ liệu một sinh viên như họ tên, ngày sinh.

_ Quản lý việc đăng ký học phần của sinh viên : lưu trữ và tra cứu cũng như thay đổi những học phần mà sinh viên đăng ký trong học kỳ.

_ Quản lý quá trình học tập và kết quả của sinh viên : lưu trữ kết quả thi các học phần của học kỳ, tra cứu điểm số và in bảng điểm.

Ứng dụng được mô hình hóa sử dụng Rational Rose 2000, cài đặt trên VC++ 6.0, hệ quản trị cơ sở dữ liệu là SQL Server 7.0. Ứng dụng cũng sử dụng các khái niệm phát triển phần mềm theo component, kiến trúc phần mềm ba lớp (three-tiered application) và kiến trúc client-server nhằm mục đích minh họa trên một tập khái niệm đa dạng UML và RUP.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 161

## 4.18. Phát triển ứng dụng theo các workflow cua RUP

*4.18.1. Mô hình hóa nghiệp vụ (business modeling)*

Xác định các business actor và nghiệp vụ (business use case) cho mỗi business actor. Lập sưu liệu mô tả quy trình thực hiện nghiệp vụ.

Phong dao tao (from Business Actors)

Tổ chức thi cử, mở học phần

Ghi nhận kết quả

Dựa trên khả năng của trường và yêu cầu của sinh viên, phòng đào tạo tiến hành mở các học phần vào đầu mỗi học kỳ.

*sưu liệu một nghiệp vụ (business use case). Phòng đào tạo đảm nhận các công việc mở học phần, tổ chức thi và ghi nhận kết quả.*

Hình 4–29 Một lược đồ Use Case của ứng dụng trong mồ hình Busines

Use Case

Tìm các business entity được sử dụng trong mỗi nghiệp vụ để mô tả rõ hơn về nghiệp vụ này qua liên hệ giữa các busisness entity.

Học phần (from Business Object Model)

Phòng đào tạo (from Business Actors)

*Mở học phần*

*Nghiệp vụ mở học phần của phòng đào tạo đòi hỏi thao tác trên một danh sách các học phần. businness entity businness worker*

Hình 4–30 Một lược đồ Clas của ứng dụng trong mồ hình Busines Objject

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 162

## 4.19. Xác định yêu cầu (requirements)

### *4.19.1. Phân loại người sử dụng (actor) và tìm các chức năng của hệ thống (use case) cho mỗi loại người sử dụng này.*

Xác định những người sử dụng hệ thống từ việc chọn lựa các business actor và business worker trong quá trình mô hình hóa nghiệp vụ. Mỗi use case nên được thiết kế không quá rộng và cũng không quá nhỏ để thuận lợi trong quá trình phân tích. Các use case đóng vai trò quan trọng trong khởi đầu phát triển ứng dụng và mô hình Use Case không đơn giản là một danh sách các use case.

Mục đích của mô hình này là tạo các use case sao cho thuận lợi trong sửa đổi, kiểm tra và quản lý vì vậy một số use case không đứng riêng lẻ mà tham gia vào các quan hệ với use case khác. Các quan hệ giữa các use case bao gồm nhiều loại như Association (với các stereotype như "include",
*"extend"...) hay tổng quát hóa (Generalization).*
Quản lý học phần (from Hoc phan)
Xếp thời khoá biểu (from Giang day)
Phòng đào tạo (from Actors)
Đăng nhập hệ thống (from He thong)
Phân công giảng viên(from Giang day)
Mở học phần (from Hoc phan)
Tra cứu học phần (from Hoc phan)
*Thiết kế use case và quan hệ giữa các use case. use case tổng quát hóa (generalization) Use case quản lý học phần thừa kế tất cả các thuộc tính và hành vi của use case mở học phần.*
Hình 4-31 Một Lược đồ Use Case của ứng dụng trong mô hình Use Case

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 163

## 4.20. Phân loại các use case theo độ ưu tiên

Xác định thứ tự phân tích, thiết kế, cài đặt, thử nghiệm của mỗi use case. Các use case có độ ưu tiên cao sẽ được phát triển trước. Độ ưu tiên có thể được đánh giá dựa trên tính chất quan trọng, chủ yếu của chức năng trong hệ thống hay trình tự thực hiện chức năng trong quy trình nghiệp vụ.

### *4.20.1. Lập sưu liệu mô tả chi tiết cho từng chức năng*

Mô tả chi tiết từng sự kiện cho mỗi chức năng bao gồm cách thức chức năng được kích hoạt, kết thúc và quá trình tương tác với người sử dụng. Đồng thời có thể sử dụng các lược đồ State Chart hay Activity để mô hình hóa chu kỳ sống và hoạt động của chức năng.
*Mô tả chi tiết chức năng*
1. Quản lý học phần (from Hoc phan)

Phòng đào tạo (from Actors)
Đăng nhập hệ thống (from He thong)
Mở học phần (from Hoc phan)
Tra cứu học phần (from Hoc phan)
Chức năng mở học phần cho một khoá trong một học kỳ
Phòng đào tạo kích hoạt chức năng cho việc duyệt một danh sách các học phần của học kỳ cho một khoá
2. Tuỳ theo khả năng của nhà trường và yêu cầu của sinh viên, phòng đào tạo quyết định mở một học phần hay không
3. Khi quyết định mở học phần một transaction sẽ được thực hiện và chuyển phần đó sang danh sách các học phần mở.
4. Chức năng mở học phần kết thúc
Hình 4–32 Số liệu mồ tả chi tiết chức năng mở học phần
entry/ Lấy dữ liệu
entry/ Khởi tạo danh sách học phần
Đăng duyệt học phần
do/ Duyệt học phần
Chọn học phần
do/ Đánh dấu học phần được chọn
Duyệt học phần kích hoạt chức năng
Quyết định mở danh sách học phần
Tiếp tục duyệt
Chức năng mở học phần kết thúc
khởi đầu kết thúc trạng thái (state) Lược đồ State Chart mô tả chuyển đổi giữa các trạng thái xử lý trong quá trình thi hành chức năng mở học phần. chuyển trạng thái (transition) sự kiện tác động (event) các hành động được thực hiện khi nắm giữ trạng thái
Mở các học phần
do/ Chuyển các học phần được chọn sang danh sách học phần mở
do/ Danh sách các học phần đã mở trên danh sách học phần
Một học phần được chọn
Hình 4–33 Lược đồ State Chart mồ tả trạng thái hoạt động của chức năng mở học phần

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 164

## 4.21. Cấu trúc các use case bằng cách xác định các quan hệ giữa

Các use case: Tìm các use case mang chức năng tổng quát và kế thừa cho các use case khác (quan hệ tổng quát hoá). Tìm các use case mang chức năng mở rộng cho các use case khác (quan hệ "extend"). Xác định các quan hệ khác (như "include") giữa các use case.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 165

## 4.22. Phân tích (analysis)

### *4.22.1. Phân tích kiến trúc hệ thống*

Xác định các gói (package) cho hệ thống thông qua việc phân loại thành nhóm các chức năng cho một quy trình nghiệp vụ tương đối rộng hay nhóm các chức năng cho một actor cụ thể.

Xác định quan hệ phụ thuộc (dependency) giữa các gói

Sinh viên

Hệ thống

Phòng đào tạo

Quản lý sinh viên

Quản lý học phần

Phân chia phân hệ phòng đào tạo thành các package hướng chức năng. Phân tích kiến trúc hệ thống qua việc phân chia thành các package.

Hình 4–34 Phân chia hệ thống thành các package

Trong mỗi gói, xác định các entity dễ dàng nhận thấy cho mỗi use case. Các entity này có thể được chọn lựa từ mô hình Busines Object và thường mang đặc điểm đặc trưng cho phạm vi của ứng dụng.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIỂN PHẦN MỀM 166

## 4.23. Phân tích một use case

Xác định các analysis class (bao gồm boundary, control và entity) cần thiết cho việc thi hành use case và thiết lập mô hình analysis.

Học phần

Khoá

Phòng đào tạo (from Actors)

Duyệt học phần

Mở học phần UI

Học phần mở

Mở học phần Control

Học kỳ

Lược đồ Class trong mô hình analysis mô tả mối liên hệ giữa các lớp analysis để thi hành use case mở học phần. Boundary control entity association

Hình 4–35 Sử dụng các analysis cllas để phần tích use case mở học phần

Mô tả tương tác giữa các đối tượng qua mô hình Collaboration để thực hiện một số cải tiến trên analysis model nếu có.

: Mở học phần Control

: Mở học phần UI

: Học phần

: Học phần mở

: Sinh viên
: Học kỳ
: Khoá
: Duyệt học phần
1: Yêu cầu mở học phần
2: Hiển thị học kỳ
3: Lấy dữ liệu học kỳ
4: Chọn học kỳ
5: Hiển thị học phần trong học kỳ của khoá
6: Lấy dữ liệu khoá
7: Lấy dữ liệu học phần
8: Duyệt học phần
9: Chọn học phần
10: Mở học phần
11: Mở học phần
12: Tạo học phần mở
Lược đồ Collaboration mô tả tương tác giữa các đối tượng để thi hành use case mở học phần. đối tượng (object)
Hình 4-36 Lược đồ Collaboration mô tả cách thức thi hành use case mở học phần

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 167

## 4.24. Phân tích một analysis class

Xác định các analysis class có vai trò tương đồng và tham gia vào nhiều use case để lập danh sách các liên hệ với các analysis class khác.

Xác định các thuộc tính cho mỗi analysis class bằng cách sử dụng các liên hệ trên để tìm các thuộc tính đầy đủ cho mỗi analysis class. Điều này có nghĩa là tập thuộc tính tìm được phải có khả năng đáp ứng cho tất cả các use case chứa analysis class này nhưng mỗi use case chứa nó chỉ sử dụng tập con thuộc tính.

Xác định các quan hệ quan hệ ngữ nghĩa (Association) và tổng quát hóa (Generalization) giữa các analysis class thông qua các quan hệ Association trên lược đồ lớp hay các liên kết trên lược đồ Collaboration do các quan hệ và liên kết này phản ánh sự tham chiếu lẫn nhau giữa các lớp. Số các quan hệ cần phải được tối thiểu hóa.

Học phần

–Tên học phần

–Số tín chỉ LT

–Số tín chỉ TH

–Khoá

–Tên khoa

–Duyệt học phần

–Mở học phần UI

–Học phần mở

–Mở học phần Control

–Học kỳ

–Học kỳ

–Năm học

Tinh chế các quan hệ và xác định thuộc tính hiển nhiên của các analysis class. thuộc tính (attribute)

Hình 4–37 Các thuộc tính của analysis cllas

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 168

## 4.25. Thiết kế (design)

### *4.25.1. Thiết kế kiến trúc ứng dụng*

Xác định cấu hình mạng và các node được triển khai vì có ảnh hưởng trực tiếp đến kiến trúc ứng dụng và việc tổ chức các chức năng cho mỗi node trên mạng. Đồng thời, mô tả việc tổ chức này bằng lược đồ Deployment.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 169

Client Phòng đào tạo intranet <<network>>University Server Client Sinh vien
Lược đồ Deployment client-server cho ứng dụng. node xử lý thiết bị (device)
Hình 4–38 Lược đồ Deplloyment của ứng dụng
Xác định các hệ thống con (subsystem) và interface là cách phân chia hệ thống thành các thành phần nhỏ để có thể quản lý và bao gồm các bước sau
_ Xác định các hệ thống con đảm nhận chức năng cung cấp dịch vụ cho các hệ thống khác.
_ Xác định các phần mềm cơ sở và các phần mềm hệ thống cần thiết cho việc phát triển ứng dụng. Có thể kể đến hệ điều hành, hệ quản trị cơ sở dữ liệu, các phần mềm dịch vụ truyền thông, các công nghệ phân tán, các phần mềm phát triển giao diện cũng như công nghệ quản lý giao tác.

Sinh viên
Phòng đào tạo
COM / ActiveX MFC 6.0
Hệ thống cho phòng đào tạo sử dụng MFC 6.0 để phát triển giao diện và các component cho các thao tác trên cơ sở dữ liệu. Package phụ thuộc (dependency)
Hình 4–39 Xác định các thành phần hỗ trợ
_ Xác định các quan hệ phụ thuộc giữa các hệ thống con.
Phòng đào tạo
Client UI Control Objects Data Services
Hệ thống cho phòng đào tạo được phân chia thành ba hệ thống con là giao diện, xử lý và dịch vụ dữ liệu.
Hình 4–40 Phụ thuộc giữa các hệ thống con

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 170

## 4.27. Phát triển một ứng dụng quan lý giáo vụ theo RUP

_ Lưu trữ dữ liệu : có thể sử dụng một hệ quản trị cơ sở dữ liệu hay tổ chức theo tập tin. Hệ quản trị cơ sở dữ liệu phổ biến và đầy đủ tính năng hiện nay là hệ quản trị cơ sở dữ liệu quan hệ. Tuy nhiên, cần phải lưu ý đến việc nâng cấp hệ thống lên hệ quản trị cơ sở dữ liệu hướng đối tượng khi hệ thống cũ lỗi thời.
_ Phân tán dữ liệu và xử lý : xác định các node để phân tán và nhóm các lớp vào các node.
_ Bảo mật
_ Xử lý lỗi và khôi phục hệ thống
_ Kiểm soát các giao tác : sử dụng một phần mềm dịch vụ giao tác (Microsoft Transaction Server).

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 171

## 4.28. Thiết kế một use case

Xác định các lớp tham gia vào use case, cần thiết cho quá trình thi hành use case và không dư thừa bằng cách sử dụng các lớp ở mức analysis cho use case đồng thời xem xét các yêu cầu đặc biệt để bổ sung thêm lớp hay thêm đặc

tính. Mô tả tương tác giữa các đối tượng bằng lược đồ Sequence và Collaboration.

Xác định các hệ thống con và interface tham gia trong quá trình thi hành use case. Mô tả tương tác trong Use case qua lược đồ Sequence với sự tham gia của các hệ thống con này.

Xác định các yêu cầu cho quá trình cài đặt thường là các yêu cầu phi chức năng.Học phần Fields (from DS)

Học phần

–Tên học phần

–Số tín chỉ LT

–Số tín chỉ TH (from Quản lý học phần) <<realize>>

Xác định các lớp thiết kế từ các lớp phân tích cho một use case. chuyển đổi giữa hai mức trừu tượng

Hình 4-41 Chuyển mỗt anallysis clas sang mức thiết kế

: Học phần MoRs

: Học phần

UI : Phòng đào tạo : Học phần

KhóaRs

: Học kỳ Rs

: Học phần Ctrl

1: Yêu cầu mở học phần

2: Hiển thị học kỳ

3: Lấy dữ liệu học kỳ

4: Chọn học kỳ

5: Hiển thị các học phần của khoá

6: Lấy dữ liệu học phần của khoá

7: Chọn học phần mở

8: Mở học phần

9: Mở học phần

10: Cập nhật dữ liệu cho học phần mở

11: Kết thúc chức năng

lớp thiết kế (design class) Lược đồ Sequence mô tả trình tự quá trình thi hành use case mở học phần. thời gian tồn tại thông điệp (focus of control) thông điệp (message)

Hình 4-42 Lược đồ Sequence ở mức thiết kế mô tả cho use case mở học phần

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 172

## 4.28. Thiết kế một lớp

Xây dựng lớp và các interface dựa trên các lớp ở mức analysis. Trong đó bao gồm các vấn đề liên quan đến môi trường cài đặt.

_ Các lớp boundary : phụ thuộc môi trường phát triển giao diện như Visual Basic hay Visual C++.

_ Các lớp entity : thường sử dụng công nghệ cơ sở dữ liệu và trải qua bước ánh xạ từ các lớp thiết kế sang các bảng trên một mô hình cơ sở dữ liệu quan hệ. Bước này khá tinh vi có thể tự động hóa một phần nhờ các CASE tool và sử dụng các nguyên tắc thiết kế cơ sở dữ liệu cùng với các mô hình dữ liệu.

_ Các lớp control : có chức năng kết hợp boundary và entity. Khi thiết kế cần và phải quan tâm đến các yêu cầu phân tán trên mạng, tốc độ hay các xử lý giao tác. Xác định các thuộc tính, phương thức, các quan hệ Association, tổng quát hóa (generalization) giữa các lớp và sử dụng cú pháp của một ngôn ngữ lập trình cụ thể để mô tả. Đồng thời thêm các tinh chế cho các quan hệ như multiplicity, navigation và xác định thuật toán cũng như quy trình cho mỗi phương thức (có thể mô tả bằng ngôn ngữ tự nhiên).

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 173

## 4.28. Phát triển một ứng dụng quan lý giáo vụ theo RUP

Học phần Fields
–m_str Tên học phần : CString
–m_iSoTCLT : int
–m_iSoTCTH : int
–m_iGiai đoạn : int
–m_lTên học phần Status : ULONG
–m_lSoTCLTStatus : ULONG
–m_lSoTCTHStatus : ULONG
–m_lGiaidoanStatus : ULONG
–Học phần
–Tên học phần
–Số tín chỉ LT
–Số tín chỉ TH <<realize>>
Các thuộc tính lấy ở mức analysis Các thuộc tính bổ sung khi xem xét môi trường cài đặt là VC + 6.0 Analysis Design
Hình 4-43 Thiết kế một lớp trong use case mở học phần
–Học phần UI (from User Interfaces)
–Học kỳ Rs (from DS)
–Rs (from DS)
–Học phần khóa Rs (from DS)
–Học phần Ctrl (from Controls)
–1+m_HocphanCtrl+m_HocKyRs+m_HocPhanMoRs+m_HocPhanKhoaRs1111
tên tham chiếu (rolename) navigation multiplicity
Hình 4-44 Bổ sung ngữ nghĩa cho lược đồ Cllas
của use case mở học phần

Mô tả các trạng thái qua lược đồ State Chart với một số tinh chế so với bước analysis. Kiểm soát các yêu cầu đặc biệt và thêm chức năng cho lớp theo yêu cầu này.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 174

## 4.29. Thiết kế một hệ thống con

Kiểm soát các các quan hệ phụ thuộc giữa các hệ thống con và cố gắng tối thiểu hóa các quan hệ phụ thuộc này. Thiết kế interface của hệ thống con ở mức chi tiết các phương thức (operation) của interface. Thiết kế các thành phần trong hệ thống con để cung cấp các interface này.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 175

## 4.30. Cài đặt (implementation)

### *4.30.1. Kiến trúc cài đặt (architectural implementation)*

Phác thảo mô hình Component bằng cách xác định các component mang tính chất chủ yếu, cấu tạo nên kiến trúc hệ thống như các component thi hành và loại bỏ các component ít quan trọng khỏi mô hình như các component chứa tài liệu hay đóng gói các file mã nguồn và gắn các component chính này vào các node.MFC6.0TSManagement
<<EXE>>ADO<<ActiveX>>Controls<<ActiveX>>

Chương trình chính và các component.

Hình 4-45 Lược đồ Component của ứng dụng

*4.30.2. Cài đặt và tích hợp hệ thống*

Lập kế hoạch cho việc cài đặt và tích hợp từng bước các thành phần của hệ thống khởi đầu từ các use case trong iteration. Điều này khiến việc kiểm chứng một use case hoàn chỉnh dễ dàng hơn.Quá trình cài đặt các thành phần cần phải dựa vào các phụ thuộc giữa các thành phần. Các thành phần nào đóng vai trò cung cấp dịch vụ cần phải được cài đặt và tích hợp trước đồng thời được kiểm chứng chức năng. Mỗi lần tích hợp một thành phần nên dựa vào kết quả lần tích hợp trước.

Việc xây dựng một use case đôi khi đòi hỏi phải xây dựng một loạt các component mới. Vì vậy thường phải có sự thỏa hiệp. Chỉ xây dựng nhiều component mới khi use case đóng vai trò quan trọng, nếu không thì tạm thời hoãn lại đến lần cài đặt và tích hợp kế tiếp. Khi đã có kế hoạch có thể tiến hành chọn phiên bản cài đặt thích hợp của một hệ thống con và các component để tiến hành biên dịch, liên kết và thi hành.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 176

## 4.31. Cài đặt các hệ thống con (subsystem)

Cài đặt từng hệ thống con theo vai trò của nó trong toàn bộ hệ thống nghĩa là các dịch vụ mà hệ thống con sử dụng đã được cài đặt và tích hợp trước đó. Mỗi lớp và giao diện trong trong một package được sử dụng phải được cài đặt trong component tương ứng.

*4.31.1. Cài đặt các lớp*

Cài đặt một số các lớp cần thiết để tích hợp. Mã nguồn được chứa trong các tập tin cài đặt của một lớp và được lưu trong một component tham chiếu các file này. Chi tiết của lớp và các quan hệ giữa các lớp đã được mô tả trong quá trình thiết kế theo một ngôn ngữ lập trình cụ thể. Có thể phát sinh thẳng mã

nguồn cho các lớp này. Lựa chọn các thuật toán và cấu trúc dữ liệu thích hợp hay sử dụng các sưu liệu mô tả phương thức trong phần thiết kế để cài đặt phương thức của lớp.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 177

## 4.42. Kiểm chứng (test)

### *4.42.1. Lập kế hoạch kiểm chứng*

Xác định chiến lược kiểm chứng, nhân sự và các tài nguyên liên quan, các kết quả đạt được trong mỗi quy trình kiểm chứng.

### *4.42.2. Thiết kế các quy trình kiểm chứng (test case)*

Test case cho một use case mô tả cách kiểm chứng cho một use case bao gồm cả kết quả tương tác giữa người sử dụng với hệ thống, trình tự thi hành theo kịch bản (script) của use case. Test case cho một use case ở mức design (use case realization) mô tả cách kiểm chứng quá trình tương tác giữa use case với các component liên quan đến việc thi hành use case.

Test case cho hệ thống mô tả quy trình kiểm chứng việc thi hành chính xác các chức năng của hệ thống xét về tổng thể chủ yếu là sự thi hành phối hợp lẫn nhau giữa các use case dưới một số điều kiện cụ thể bao gồm cấu hình phần cứng, người sử dụng, kích thước cơ sở dữ liệu...

### *4.42.3. Thực hiện kiểm chứng*

Tự động hóa các quy trình kiểm chứng một cách tối đa bằng cách tạo các component kiểm chứng. Mỗi component chứa các kịch bản (script) cho quy trình kiểm chứng. Kiểm chứng kết quả tích hợp. Kiểm chứng hệ thống qua thực hiện test case cho hệ thống. Đánh giá kết quả kiểm chứng các độ đo về mức độ hoàn chỉnh và tin cậy.

# Chương 4
# ỨNG DỤNG UML TRONG QUY TRÌNH PHÁT TRIÊN PHẦN MỀM 178

*4.43. Tóm tắt*

Chương bốn đã trình bày sơ lược về qui trình phát triển phần mềm Rational Unified Process, qua đó cũng giới thiệu khả năng ứng dụng của UML trong một qui trình phát triển phần mềm cụ thể. RUP và UML được phát triển đồng hành với nhau và có nguồn gốc từ hãng phần mềm Rational, vì vậy việc kết hợp giữa UML và RUP mang lại một công cụ rất hiệu quả trong việc phár triển phần mềm. Để minh họa thực tế cho ứng dụng này chúng em đã sử dụng phần mềm Rational Rose 2000 (có hỗ trợ hoàn toàn ngôn ngữ UML và RUP) để thiết kế ứng dụng, việc tiếp cận và sử dụng một công cụ mới trên thực tế có không ít khó khăn, nhưng những kết quả đạt được trong ứng dụng có thể cho thấy khả năng ứng dụng của UML trong thực tế là rất khả quan.

# Chương 5
# TÔNG KẾT 179

## 5.1. Kết luận

Qua tìm hiểu và nghiên cứu đề tài, luận văn chúng em đã trình bày về hệ thống thống ngữ nghĩa cốt lõi, hệ thống các loại lược đồ cùng với các ký hiệu mô tả của các thành tố được định nghĩa trong ngôn ngữ UML, bên cạnh đó cũng giới thiệu tổng quan về qui trình phát triển phần mềm RUP và khả năng ứng dụng của UML trong qui trình này.

Trên cơ sở phân tích và sử dụng ngôn ngữ UML,nó cho thấy nó không chỉ là một ngôn ngữ hợp nhất đơn thuần, UML còn bao gồm cả những khái niệm mới cùng với cách mô tả, định nghĩa và sử dụng các khái niệm này. Việc nghiên cứu ngôn ngữ UML không chỉ dừng ở việc tìm hiểu, sử dụng các khái niệm, các ký hiệu trong ngôn ngữ, mà bên cạnh đó chúng ta cần phải tìm hiểu cách thức để mô hình hóa một hệ thống phần mềm, cũng như việc tích hợp UML với một qui trình phát triển phần mềm cụ thể. Hiện nay, hầu hết các công cụ hỗ trợ phân tích thiết kế đều có hỗ trợ ngôn ngữ UML, việc nghiên cứu UML giúp chúng ta có thể tìm hiểu và sử dụng các CASE tool hỗ trợ cho việc phát triển phần mềm, đồng thời có thể tiếp cận với những qui trình sản xuất phần mềm tiên tiến trên thế giới. Trong điều kiện nền công nghệ phần mềm nước ta còn khá non trẻ, việc đưa vào sử dụng một qui trình công nghệ và một ngôn ngữ mô hình hóa mới sẽ gặp không ít khó khăn, nhưng trong giai đoạn bùng nổ của ngành công nghệ phần mềm hiện nay việc tìm hiểu và sử dụng một ngôn ngữ mô hình hóa hiệu quả như UML là rất cần thiết.

# Chương 5

# TÔNG KẾT 180

## 5.2. Hướng phát triển

UML là một ngôn ngữ mô hình hóa rất rộng, bên cạnh những phần đã được trình bày trong luận văn, UML còn có nhiều ứng dụng rất đa dạng như các ứng dụng mô hình hóa hệ thống thời gian thực, các hệ thống phân tán, các ứng dụng trên web...

Trên cơ sở những phần đã thực hiện, đề tài của chúng em có một số hướng phát triển sau:

_ Xây dựng tài liệu nghiên cứu giảng dạy UML

_ Dựa trên qui trình công nghệ RUP, tìm hiểu và xây dựng một phương pháp phát triển phần mềm phù hợp với điều kiện của nền công nghệ phần mềm nước ta hiện nay.

_ Dựa trên cấu trúc ngữ nghĩa của UML, từng bước xây dựng công cụ phân tích thiết kế hỗ trợ ngôn ngữ UML.

Một lần nữa chúng em xin chân thành cảm ơn Thầy Dương Anh Đức, Thầy Lê Đình Duy cùng toàn thể quý Thầy Cô trong khoa đã tận tình giảng dạy chúng em trong suốt những năm học vừa qua. Mặc dù chúng em đã hết sức cố gắng để hoàn thiện đề tài nhưng chắc không tránh khỏi những thiết sót nhất định, kính mong quý thầy cô tận tình chỉ bảo để đề tài của chúng em được hoàn thiện hơn.

# Phụ lục A

# CÁC KHÁI NIỆM

A

abstract class Lớp trừu tượng không có thực thể đại diện.

action Hành động, hành động có kết quả làm thay đổi một trạng thái nào đó của hệ thống.

action state Một trạng thái hoạt động gây ra một hành động nào đó.

activation Sự kích thích một hành động.

active class Một lớp có những thực thể là một active object.

active object Một đối tượng sở hữu một tiến trình có thể điều khiển
được.

actor Tác nhân –là một thực thể đóng vai trò tương tác với hệ thống, tác nhân có thể là người sử dụng hệ thống hoặc một hệ thống khác.

aggregation Là một quan hệ thu nạp giữa một lớp đóng vai trò toàn thể và một lớp đóng vai trò là bộ phận.

artifact Một phần thông tin được dùng hoặc được phát sinh từ hệ thống.

association Quan hệ kết hợp giữa hai thành tố trong hệ thống.

association end Điểm cuối của quan hệ kết hợp liên kết với một classifier.

attribute Thuộc tính của một thành tố trong lược đồ.

B

behavior Hành vi của thành tố trong hệ thống.

binary
association Quan hệ kết hợp nhị phân/ quan hệ kết hợp giữa hai lớp.

C

call Một trạng thái hành động dẫn đến một hành động trên
classifier.

class Class là tập hợp các đối tượng có cùng các thuộc tính

classifier Là một thành tố trừu tượng miêu tả các đặc điểm về hành vi và cấu trúc.

class diagram Là một lược đồ dùng để mô tả các lớp (class), các giao tiếp (interface), sự cộng tác (collaboration) và các mối quan hệ giữa các thành phần trong mô hình.

collaboration Một collaboration mô tả qúa trình thực hiện của một thao tác hay một classifier trong một tập các classifier có tương tác với nhau.

collaboration
diagram Là một lược đồ tương tác tập trung vào cấu trúc tổ chức, mối quan hệ tác động qua lại giữa các đối tượng.

comment Là một chú thích được gắn vào các thành tố trong mô hình nhằm làm rõ nghĩa cho các thành tố này.

component Là một phần của hệ thống được triển khai.

component
diagram Là một lược đồ ghi nhận các tổ chức và sự phụ thuộc giữa các thành phần trong hệ thống.

composite state Là một trạng thái chứa các trạng thái con luân phiên hoặc tuần tự (tách tời).

composition Quan hệ cấu thành, là một dạng mạnh hơn của quan hệ thu nạp.

Concurrent substate Là một trạng thái con được tiến hành đồng thời với một trạng thái con khác bên trong một trạng thái ghép.

D

datatype Kiểu dữ liệu, mô tả kiểu dữ liệu của người sử dụng.

dependency Quan hệ phụ thuộc giữa hai thành tố mô hình.

deployment
diagram Là một lược đồ thể hiện cấu hình lúc chạy của các thành phần, các thiết bị, bộ xử lý.

derived element Là một thành tố được dẫn xuất từ các thành tố khác trong hệ thống.

disjoint substate Một trạng thái con không thể tiến hành đồng thời với các tiến trình con khác trong một trạng thái ghép.

E

entry action Là một hành động đầu vào của một trạng thái.

event Một sự kiện, có thể gây ra sự chuyển đổi trạng thái trong lược đồ trạng thái.

exit action Là một hành động đầu ra của một trạng thái.

extend Quan hệ mở rộng giữa hai Use case, hành vi của use case này được mở rộng từ những hành vi của một use case khác.

F

final state Trạng thái kết thúc của một máy trạng thái.

focus of control Là một ký hiệu trên lược đồ thể hiện khoảng thời gian khi một đối tượng thi hành một hành động.

G

generalization Mối quan hệ giữa một thành tố tổng quát và một thành tố phụ đặc biệt.

guard condition Là một điều kiện cần được thỏa mãn để có thể thi hành một một sự chuyển đổi trạng thái.

I

interface Tên của một tập các thao tác đặc trưng cho hành vi của một thành tố mô hình.

internal

transition Là một sự chuyển đổi tín hiệu hồi đáp cho một sự kiện mà không cần thay đổi trạng thái của đối tượng.

L

link Là một tham chiếu giữa các đối tượng..

M

message Là một sự chuyển đổi thông tin giữa các thực thể.

metaclass Là một lớp trừu tượng mà thể hiện của nó là những lớp.

model element Thành tố mô hình.

multiplicity Bản số, đặc tả số lượng cho phép của các thực thể trong một mối quan hệ.

N

n-ary association Quan hệ kết hợp bậc n, là một quan hệ kết hợp giữa ba hay nhiều lớp với nhau.

name Là một chuỗi định nghĩa cho thành tố mô hình.

node Là một thành phần biểu diễn các tài nguyên máy tính.

O

object Là một thể hiện cụ thể của một lớp trong hệ thống.

object diagram Là một đồ thị của các thể hiện, bao gồm các đối tượng và các giá trị cụ thể.

object lifeline Là một đường trong lược đồ tuần tự thể hiện sự tồn tại của đối tượng trong một khoảng thời gian nào đó.

P

package Là một cơ chế tổng quát cho việc tổ chức các thành tố thành các nhóm.

Q

qualifier Là một thuộc tính của quan hệ kết hợp nhằm hạn chế tập đối tượng quan hệ với một đối tượng khác thông qua quan hệ kết hợp.

R

role Là tên đặt biệt của một thực thể tham gia trong một ngữ cảnh đặc biệt nào đó.

S

sequence

diagram Là một lược đồ tương tác tập trung vào các hành vi động hướng thời gian.

signal Đặc tả một mối liên hệ kích thích không đồng bộ giữa các thực thể. Tín hiệu có thể có tham số.

state Là một hoàn cảnh hoặc một tình huống trong qúa trình sống của đối tượng thỏa mãn một điều kiện nào đó, biểu diễn một số hoạt động hoặc chờ một vài sự kiện.

Statechartdi Là một lược đồ thể hiện máy trạng thái.

Diagram stereotype Là một loại phần tử mô hình dùng để mở rộng ngữ nghĩa của UML. Khuôn mẫu phải dựa trên các thành tố đã được định nghĩa trong UML.Stereo chỉ mở rộng về ngữ nghĩa không mở rộng về cấu trúc..

substate Trạng thái là một phần của trạng thái ghép.

subpackage Một package chứa các package khác.

subsystem Nhóm các thành tố mô hình biểu diễn các đơn vị hành vi trong hệ thống vật lý.

swimlane Dùng để nhóm các hành động có cùng một mục đích nào đó trong lược đồ đối tượng.

T

tagged value Gía trị thẻ định nghĩa một thuộc tính theo dạng tên–giá trị. Dùng để bổ sung thông tin cho các thành tố mô hình.

transition Quan hệ giữa hai trạng thái, chỉ ra rằng một đối tượng chuyển từ trạng thái này sang một trạng thái khác khi có một sự kiện hoặc một tín hiệu nào đó xảy ra.

U

use case Là một chuỗi các hành động hoặc một đơn vị chức năng được cung cấp bởi hệ thống nhằm đáp ứng nhu cầu của các tác nhân bên ngoài hay các hệ thống khác.

use case diagram Lược đồ Use Case ghi nhận chức năng của hệ thống dưới góc nhìn của người sử dụng.

V

visibility Tầm vực, phạm vi tham chiếu của của thành tố mô hình bao gồm các giá trị (public, protected, or private).

# Phụ lục B

# CÁC KÝ HIỆU

*Tên Ký hiệu*
Actor (tác nhân)
Name
Association (Quan hệ kết hợp)
Aggregation (Quan hệ thu nạp/kết hợp)
Composition (Quan hệ cấu thành)
Class (lớp)
Collaboration (sự cộng tác)
Component (thành phần)
Constraint (ràng buộc)
Dependency (quan hệ phụ thuộc)
Generalization (quan hệ tổng quát hóa)
Interface (giao tiếp)
Node (Nút)
Note (ghi chú)
Object (đối tượng)
Package (gói)
Stereotype (khuôn mẫu) *(stereotype)*
Tagged Value (giá trị thẻ)
Use case Name

# TÀI LIỆU THAM KHẢO

(1) Bruce Powel Douglass – Ph.D.Chief Evangelist, *The Unified Modeling Language for Systems Engineering*, I-Logix 1/1999.
(2) CRaG System Report, *An Introduction to the UML*, 1998.
(3) Engineering Notebook C++ Report, *UML Use Case Diagrams*, 10/1998.
(4) James Rumbaugh, *UML – The View from the front*, Rational Software Corporation, 3/1999.
(5) Grady Booch, *Software Architecture and the UML*, Rational Software, 4/2000.
(6) Grady Booch, James Rumbaugh, Ivar Jacobson, *The Importance of Modeling*, The UML User's Guide,1998.
(7) Gunnar Overgaard, Bran Selic và Conrad Bock, Object Modeling with *UML Behavioral Modeling*, 1/2000.
(8) Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified Software Development Process Book*, 4/1999.
(9) OMG & Rational Corporation, *OMG Unified Modeling Language Specification V1.3*, 8/1999.
(10) Popkin Software, *Modeling Systems with UML*, A Popkin Software White Paper, 1998.
(11) Philippe Kruchten, *A Rational Development Process*, 4/2000.
(12) Rational Software Corporation, *Analysis and Design with UML*, 1997.
(13) Rational Software Corporation, *Rational Unified Process – Best Practices for Software Development Teams*,A Rational Software Corporation White Paper, 12/1999.
(14) Robert C.Martin, *UML Tutorial – Class Diagrams*, 9/1997.
(15) Robert C.Martin, *UML Tutorial – Collaboration Diagrams*, 10/1997.
(16) Robert C.Martin, *UML Tutorial – Finite State Machines*, 6/1998.
(17) Robert C.Martin, *UML Tutorial – Sequence Diagrams*, 4/1998.

(18) Scott W.Amber, *Enhancing the Unified Modeling Language*, A Ronin International White Paper, 3/2000.

(19) Scott W.Amber, *The Unified Modeling Language and Beyond: The Techniques of Object-Oriented Modeling*, An AmbySoft Inc.White Paper 2/2000.

(20) Sinan Si Alhir, *Applying the Unified Modeling Language*, 8/1998.

(21) Sinan Si Alhir, *Description of the Public Model for Unified Modeling Language metamodel abstract syntax V1.3*, OMG Revision Task Force 11/1998.

(22) Sinan Si Alhir, *Extending the UML*, 1/1998

(23) Sinan Si Alhir, *The Foundation of the UML*, Updated 8/1998

(24) Sinan Si Alhir, *The UML – One year sfter Adoption of the Standard*, 1/1999.

(25) Sinan Si Alhir, *The UML – One year sfter Adoption of the Standard*, 12/1999.

(26) Sinan Si Alhir, *Reuse and the UML*, Updated 1/1999.

(27) Sinan Si Alhir, *Succeeding with UML*, 8/1998.

(28) Sinan Si Alhir, *What is the UML*, 8/1998.

(29) Sinan Si Alhir, *The True Value of the Unified Modeling Language*, 9/1998.

(30) Sinan Si Alhir, *Unified Modeling Language – Extension Mechanisms*, 10/1998.

(31) Tony Clark & Andy Evans, *Foundation of Unified Modeling Language*, University of Bradford, UK 8/1999.

(32) Xiaobing Qiu, *Object-Oriented Software Development using UML*, 1998.