

TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT HÀ NỘI

=====



BÁO CÁO HỌC THUẬT

**TÌM HIỂU VỀ NGÔN NGỮ MÔ HÌNH HÓA
THỐNG NHẤT UML**

NGUYỄN THỊ HẢI YẾN

Hà Nội, T12/2024

MỤC LỤC

MỤC LỤC	2
MỞ ĐẦU	3
Chương I: Tổng quan về ngôn ngữ mô hình hóa thống nhất (UML)	4
I.1 Khái niệm	4
I.2 Lịch sử phát triển của UML	4
I.3 Ứng dụng của UML	5
I.4. Đặc điểm của UML	7
I.5 Mục đích của UML	7
Chương II: Sử dụng UML trong phát triển phần mềm.....	9
II.1. Các phần tử cơ bản của UML	9
II.2. Các quy tắc ngữ nghĩa của UML	16
II.3. Một số cơ chế chung trong UML.....	16
II.4. Công cụ hỗ trợ mô hình hóa UML.....	16
KẾT LUẬN	21
TÀI LIỆU THAM KHẢO	22

MỞ ĐẦU

UML (Ngôn ngữ mô hình hóa thống nhất) là một công cụ mô hình hóa chuẩn mực, giúp mô tả và xác định cấu trúc của nhiều hệ thống phần mềm khác nhau thông qua các sơ đồ trực quan. Nó không chỉ cho phép lập trình viên xây dựng tài liệu kỹ thuật mà còn hỗ trợ ánh xạ trực tiếp sang các ngôn ngữ lập trình như Java, C++, và Visual Basic. Với nhiều loại sơ đồ khác nhau, UML cung cấp một phương thức mạnh mẽ để giao tiếp trong quy trình phát triển phần mềm.

- UML bao gồm nhiều loại sơ đồ, mỗi loại phục vụ một mục đích nhất định trong việc cung cấp thông tin về hệ thống.
- Các sơ đồ lớp và sơ đồ đối tượng là hai trong số những loại sơ đồ chính của UML.
- UML giúp cải thiện khả năng giao tiếp giữa các thành viên trong nhóm phát triển phần mềm.
- Ngôn ngữ này không chỉ dừng lại ở việc mô hình hóa mà còn hỗ trợ tạo tài liệu cho hệ thống phần mềm.
- UML có thể được sử dụng cho nhiều giai đoạn khác nhau trong quy trình phát triển phần mềm, từ phân tích yêu cầu đến thiết kế.

Trong khuôn khổ của báo cáo sinh hoạt học thuật này, tôi tìm hiểu về ngôn ngữ mô hình hóa thống nhất UML nhằm cung cấp một phương pháp chuẩn hóa để mô tả, thiết kế và phát triển các hệ thống phần mềm.

CHƯƠNG I: TỔNG QUAN VỀ NGÔN NGỮ MÔ HÌNH HÓA THỐNG NHẤT UML

I.1. Khái niệm

Ngôn ngữ mô hình hóa UML (Unified Modeling Language) là một ngôn ngữ chuẩn dùng để thiết kế, mô phỏng và tài liệu hóa các hệ thống phần mềm và quy trình kinh doanh. UML cung cấp một tập hợp các ký hiệu và quy tắc để mô tả các khía cạnh khác nhau của hệ thống, giúp cho việc giao tiếp và hiểu biết giữa các thành viên trong nhóm phát triển.

UML đại diện cho một tập hợp các phương pháp kỹ thuật tốt nhất đã được chứng minh là thành công trong việc mô hình hóa các hệ thống lớn và phức tạp. UML là một phần rất quan trọng trong việc phát triển phần mềm hướng đối tượng và quy trình phát triển phần mềm. UML chủ yếu sử dụng các ký hiệu đồ họa để thể hiện thiết kế của các dự án phần mềm. Sử dụng UML giúp các nhóm dự án giao tiếp, khám phá các thiết kế tiềm năng và xác thực thiết kế kiến trúc của phần mềm.

I.2. Lịch sử phát triển UML

- Số lượng các phương pháp luận hướng đối tượng gia tăng từ dưới 10 đến 50 trong khoảng những năm 1989 đến 1994, và do đó nảy sinh vấn đề là làm cho người phát triển khó tìm thấy một phương pháp luận duy nhất thoả mãn đầy đủ nhu cầu của họ.

- Vào tháng mười năm 1994, Rumbaugh đã liên kết với công ty Booch (Rational Software Corporation) để kết hợp phương pháp Booch và phương pháp OMT. Và cho ra một bản phác thảo về phương pháp có tên là Unified Process vào tháng mười năm 1995.

- Cũng trong năm 1995, Jacobson đã nỗ lực tích hợp phương pháp này với OOSE. Và những tài liệu đầu tiên về UML đã được trình làng vào trong năm 1996.

- Phiên bản 1.0 của UML đã được công bố vào tháng giêng 1997

- UML phiên bản 1.5 đã được tạo vào tháng ba năm 2003.

- Phiên bản UML 2.0 được tạo vào 2004.

....

- Và phiên bản hiện tại là 2.5.1 được công bố vào tháng 12 năm 2017.

Các phiên bản UML, năm xuất bản và link tài liệu của các phiên bản UML được thể hiện trong bảng 1 dưới đây.

Bảng 1.1: Các phiên bản UML

Phiên bản	Ngày bắt đầu	URL
2.5.1	December 2017	https://www.omg.org/spec/UML/2.5.1
2.4.1	July 2011	https://www.omg.org/spec/UML/2.4.1
2.3	May 2010	https://www.omg.org/spec/UML/2.3
2.2	January 2009	https://www.omg.org/spec/UML/2.2
2.1.2	October 2007	https://www.omg.org/spec/UML/2.1.2
2.0	July 2005	https://www.omg.org/spec/UML/2.0
1.5	March 2003	https://www.omg.org/spec/UML/1.5
1.4	September 2001	https://www.omg.org/spec/UML/1.4
1.3	February 2000	https://www.omg.org/spec/UML/1.3
1.2	July 1999	https://www.omg.org/spec/UML/1.2
1.1	December 1997	https://www.omg.org/spec/UML/1.1

I.3. Ứng dụng của UML

UML đóng vai trò quan trọng trong phát triển phần mềm và nhiều lĩnh vực liên quan. Dưới đây là một số điểm nổi bật về vai trò của UML:

1. Hỗ trợ Mô hình hóa

Diễn tả hệ thống: UML cho phép người phát triển mô hình hóa các hệ thống phức tạp từ các khía cạnh khác nhau, giúp hình dung tổng thể và các chi tiết cụ thể.

Tiêu chuẩn hóa hình thức mô hình: Bằng cách cung cấp các biểu đồ và ký hiệu tiêu chuẩn, UML giúp đảm bảo rằng tất cả những người liên quan đều hiểu và diễn giải mô hình theo cách nhất quán.

2. Cải thiện Giao tiếp

Ngôn ngữ chung: UML cung cấp một ngôn ngữ đồ họa dễ hiểu, giúp cải thiện giao tiếp trong nhóm phát triển phần mềm cũng như với các bên liên quan, bao gồm khách hàng và người dùng cuối.

Đóng vai trò cầu nối: UML giúp các nhà phân tích, lập trình viên, và quản lý dự án cùng làm việc hiệu quả hơn bằng cách sử dụng các biểu đồ để diễn tả ý tưởng và thiết kế.

3. Giúp Phân tích yêu cầu

Xác định yêu cầu: Biểu đồ tình huống (Use Case Diagram) giúp xác định và tài liệu hóa các yêu cầu của người dùng và các tác nhân tương tác với hệ thống.

Nắm bắt dự đoán: UML cung cấp một cái nhìn tổng quát về tính năng và cách thức người dùng tương tác, từ đó giúp phát hiện sớm các yêu cầu không rõ ràng hoặc thiếu sót.

4. Thiết kế hệ thống phần mềm

Kiến trúc và cấu trúc: UML cho phép thiết kế cấu trúc hệ thống qua các biểu đồ lớp (Class Diagram), biểu đồ thành phần (Component Diagram), và biểu đồ triển khai (Deployment Diagram).

Mô hình hóa hành vi: Với các biểu đồ hoạt động (Activity Diagram) và biểu đồ tuần tự (Sequence Diagram), UML giúp thiết kế và hiểu được hành vi của hệ thống, các quy trình và tương tác giữa các thành phần.

5. Hỗ trợ phát triển phần mềm hướng đối tượng

Tạo mô hình đối tượng: UML rất thích hợp cho các phương pháp phát triển phần mềm hướng đối tượng, giúp tạo và quản lý các lớp, đối tượng và mối quan hệ giữa chúng một cách hiệu quả.

Hỗ trợ kế thừa và đa hình: UML giúp mô hình hóa các hệ thống phức tạp với các khái niệm kế thừa và đa hình, giúp xây dựng các giải pháp linh hoạt và có thể tái sử dụng.

6. Tài liệu hóa hệ thống

Tạo tài liệu hướng dẫn: Biểu đồ UML có thể được sử dụng để tạo tài liệu cho hệ thống, giúp dễ dàng hiểu, bảo trì và phát triển thêm trong tương lai.

Giúp người mới vào dễ tiếp cận: Tài liệu UML dễ đọc và minh họa rõ ràng giúp người mới vào dễ dàng hiểu được cấu trúc và chức năng của hệ thống mà không cần phải nghiên cứu mã nguồn.

7. Ứng dụng trong nhiều lĩnh vực

Phát triển phần mềm: Được sử dụng trong việc thiết kế và phát triển phần mềm ứng dụng, hệ thống doanh nghiệp, và các giải pháp công nghệ thông tin khác.

Mô hình hóa quy trình kinh doanh: UML cũng có thể được áp dụng để mô hình hóa quy trình nghiệp vụ, giúp tổ chức và cải tiến các quy trình nội bộ.

Hệ thống nhúng và IoT: Sử dụng UML để mô hình hóa và thiết kế các hệ thống nhúng, thiết bị IoT và các ứng dụng phát triển phần mềm khác.

Nhìn chung, UML là một công cụ mạnh mẽ có vai trò tối quan trọng trong việc phát triển phần mềm và cải thiện quy trình làm việc của các nhóm phát triển, đảm bảo rằng các dự án công nghệ thông tin được thực hiện một cách hiệu quả, chính xác và đáp ứng nhu cầu của người dùng.

I.4. Đặc điểm của UML

- UML là một ngôn ngữ mô hình hóa, trước hết nó bao gồm một tập các ký pháp thống nhất, thể hiện ngữ nghĩa các định nghĩa trực quan tất cả các thành phần của mô hình.

- UML được sử dụng để hiển thị, đặc tả, tổ chức, xây dựng và làm tài liệu các kết quả của quá trình phát triển phần mềm hướng đối tượng, đặc biệt là phân tích, thiết kế dưới dạng các báo cáo, biểu đồ, bản mẫu hay các trang web,...

I.5. Mục đích của UML

- Mô hình được các hệ thống (không chỉ hệ thống phần mềm) và sử dụng được tất cả các khái niệm hướng đối tượng một cách thống nhất.

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

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

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

Tóm lại, UML là ngôn ngữ mô hình hoá, ngôn ngữ đặc tả và ngôn ngữ xây dựng mô hình trong quá trình phát triển phần mềm, đặc biệt là trong phân tích và thiết kế hệ thống hướng đối tượng .

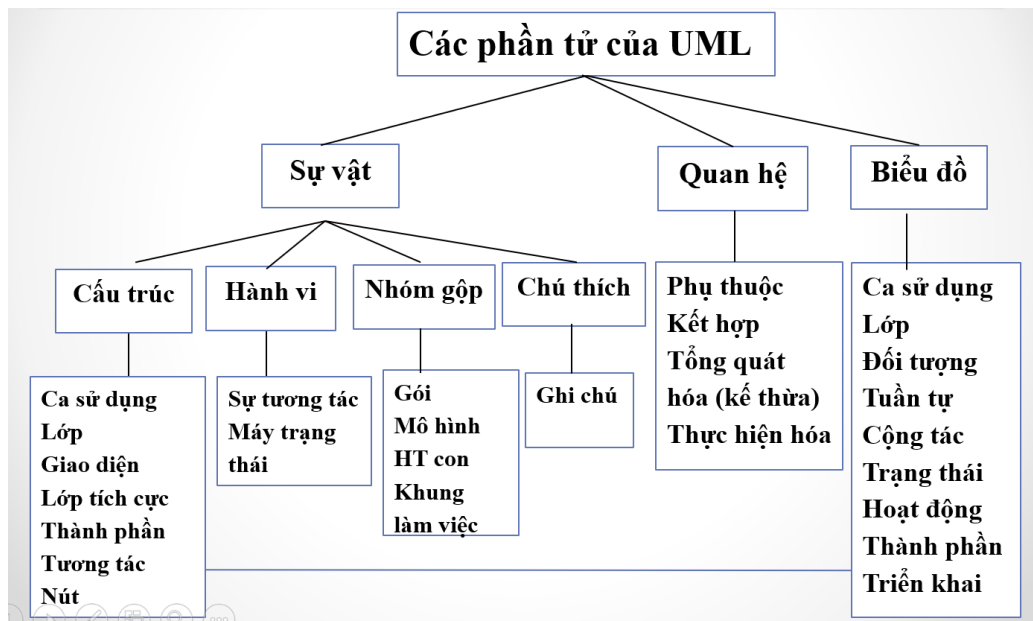
UML là ngôn ngữ hình thức, thống nhất và chuẩn hoá mô hình hệ thống một cách trực quan. Nghĩa là các thành phần trong mô hình được thể hiện bởi các ký hiệu đồ hoạ, biểu đồ và thể hiện đầy đủ mối quan hệ giữa các chúng một cách thống nhất và có logic chặt chẽ.

CHƯƠNG II: SỬ DỤNG UML TRONG PHÁT TRIỂN PHẦN MỀM

Để hiểu và sử dụng tốt UML trong phát triển phần mềm đòi hỏi phải nắm bắt được ba vấn đề sau:

- Các phần tử cơ bản của UML.
- Các quy tắc ngữ nghĩa của UML.
- Một số cơ chế chung được áp dụng cho việc mô hình hóa.

II.1. Các phần tử cơ bản của UML



Hình 2.1: Các phần tử cơ bản của UML

Các phần tử cơ bản của UML bao gồm:

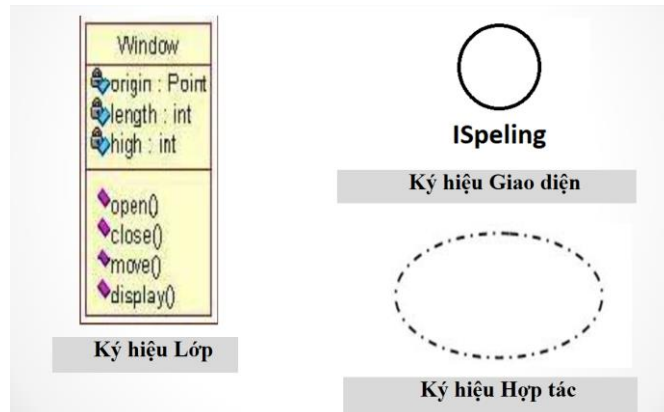
- *Các sự vật (Things)* là các trừu tượng hóa và là những phần tử lớp đầu tiên (những viên gạch) để xây dựng lên các mô hình trong UML.
- *Các quan hệ (Relationships)* gắn kết các sự vật với nhau.
- *Các biểu đồ (Diagrams)* nhóm các sự vật được quan tâm lại tạo nên ngữ nghĩa của nó (cho một mô hình)

II.1.1. Các sự vật trong UML

UML có 4 loại sự vật: *cấu trúc*, *hành vi*, *nhóm gộp* và *chú thích*.

1. *Sự vật cấu trúc (Structural things)*: Là các danh từ trong mô hình UML biểu diễn các thành phần khái niệm hay vật lý của hệ thống.
 - *Lớp (Class)*: Là một tập hợp các đối tượng có cùng một tập thuộc tính, các hành vi, các mối quan hệ với những đối tượng khác.

- *Giao diện (Interface)*: Là một tập hợp các phương thức (operation) tạo nên dịch vụ của một lớp hoặc một thành phần (component). Nó chỉ ra một tập các operation ở mức khai báo chứ không phải ở mức thực thi (Implementation).
- *Hợp tác (Collaboration)*: Thể hiện một giải pháp thi hành bên trong hệ thống, bao gồm các lớp/ đối tượng mối quan hệ và sự tương tác giữa chúng để đạt được một chức năng mong đợi của Ca sử dụng (Use case).



Hình 2.2: Một số ký hiệu sự vật cấu trúc của UML

- *Ca sử dụng (Use case)*: Là mô tả một tập hợp của nhiều hành động tuần tự mà hệ thống thực hiện để đạt được một kết quả có thể quan sát được đối với một tác nhân (Actor) cụ thể nào đó.

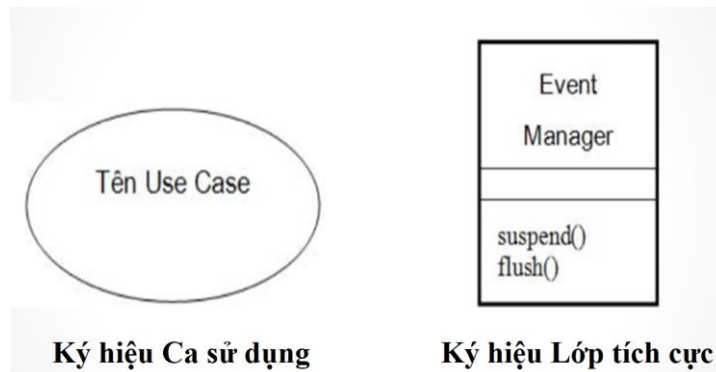
Ca sử dụng mô tả sự tương tác giữa tác nhân và hệ thống. Nó thể hiện chức năng mà hệ thống sẽ cung cấp cho tác nhân.

Tập hợp các ca sử dụng của hệ thống sẽ tạo nên tất cả các trường hợp mà hệ thống có thể được sử dụng.

- *Lớp tích cực (Active class)*:

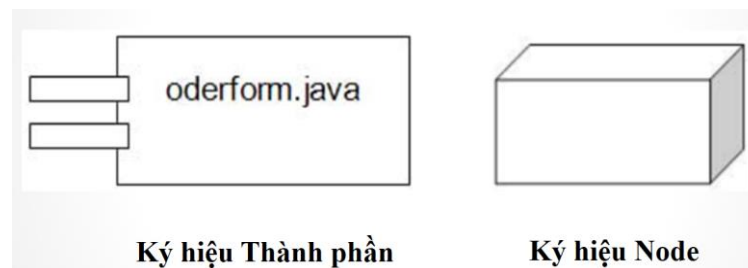
Là một lớp mà các đối tượng của nó thực hiện các hoạt động điều khiển. Lớp tích cực cũng giống như lớp bình thường ngoại trừ việc các đối tượng của nó thể hiện các phần tử mà ứng xử của chúng có thể thực hiện đồng thời với các phần tử khác.

Lớp này thường dùng để biểu diễn tiến trình (process) và luồng (thread).



Hình 2.3: Ký hiệu ca sử dụng và lớp tích cực

- **Thành phần (Component):** Là biểu diễn vật lý của mã nguồn. Trong hệ thống ta sẽ thấy các kiểu khác nhau của component như các thành phần COM+ hay JavaBeans cũng như là các thành phần như các file mã nguồn, các file nhị phân tạo ra trong quá trình phát triển hệ thống.
- **Nodes:** Là thể hiện một thành phần vật lý như là một máy tính hay một thiết bị phần cứng. Một tập các thành phần có thể lưu trên một nút và có thể di trú từ nút này sang nút khác.



Hình 2.3: Ký hiệu ca sử dụng và lớp tích cực

2. Sự vật hành vi (Behavioral things): Biểu diễn hành vi trong tương tác của các thành phần và biến đổi trạng thái của hệ thống.

Tương tác (Interaction):

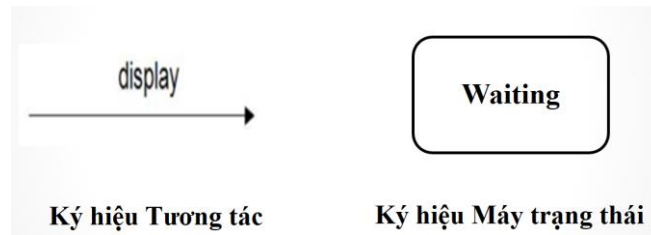
Bao gồm một tập các thông báo (message) trao đổi giữa các đối tượng trong một ngữ cảnh cụ thể nào đó để thực hiện một chức năng nào đó.

Hành vi của nhóm đối tượng hoặc của một thao tác đối tượng có thể được diễn đạt bằng một tương tác.

- **Máy trạng thái (States machine):**

Máy trạng thái là loại hành vi đặc tả chuỗi các trạng thái của một đối tượng hay của một tương tác diễn ra trong vòng đời của chúng để đáp ứng các sự kiện.

Hành vi của một lớp hay sự cộng tác giữa các lớp có thể đặc tả bằng một máy trạng thái. Máy trạng thái thường liên quan đến một vài phần tử khác như các trạng thái, các chuyển đổi.



Hình 2.4: Ký hiệu tương tác và máy trạng thái

3. *Sự vật nhóm gộp (Grouping things)*: là bộ phận tổ chức của mô hình UML. Nó là công cụ để tổ chức các thành phần của một mô hình thành các nhóm.

- *Gói (Package)*: Dùng để nhóm các phần tử có một ý nghĩa chung nào đó vào thành nhóm. Gói dùng để nhìn hệ thống ở một mức độ tổng quát hơn so với việc xem xét từng phần tử trong gói.

- *Mô hình*:

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

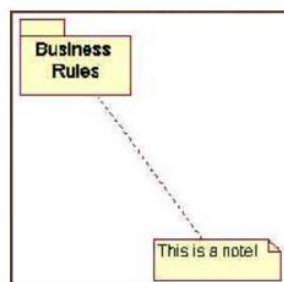
- *Khung công việc*:

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

4. *Sự vật chú thích*: là bộ phận chú giải của mô hình, giải thích về phần tử khái niệm và cách sử dụng chúng.

- *Chú thích (Annotational)*:

Là các chú thích dùng để mô tả, làm sáng tỏ và ghi chú về bất cứ phần tử nào trong mô hình. Thường dùng nhất là Note gồm các ràng buộc hoặc ghi chú, được gắn với một phần tử hoặc một tập hợp các phần tử.



Hình 2.5: Ký hiệu chú thích

II.1.2. Các mối quan hệ trong UML

Có bốn loại quan hệ trong UML, bao gồm:

- ❖ Quan hệ phụ thuộc,
- ❖ Quan hệ kết hợp,
- ❖ Quan hệ khái quát hóa
- ❖ Quan hệ hiện thực hóa.

1. Quan hệ phụ thuộc (Dependency)

Phụ thuộc là quan hệ ngữ nghĩa giữa hai phần tử trong đó một phần tử (phần tử độc lập) thay đổi có thể ảnh hưởng đến ngữ nghĩa của phần tử kia (phần tử phụ thuộc).

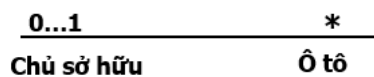
Thường thì bên phụ thuộc cần dùng bên độc lập để đặc tả hay cài đặt cho mình.

Ký hiệu Phụ thuộc:



2. Quan hệ kết hợp (Association)

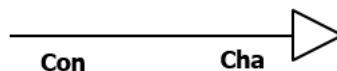
Kết hợp là một quan hệ cấu trúc xác định mối liên kết giữa các lớp đối tượng. Khi một đối tượng của lớp này gửi/nhận thông điệp đến/từ chỗ đối tượng của lớp kia.



Hình 2.6: Một ví dụ về quan hệ kết hợp

3. Quan hệ tổng quát hóa (Generalization)

Đây là quan hệ mô tả sự khái quát hóa mà trong đó một số đối tượng cụ thể (con) sẽ được kế thừa các thuộc tính, các thao tác của các đối tượng tổng quát (cha).



Hình 2.7: Ký hiệu quan hệ tổng quát hóa

4. Quan hệ hiện thực hóa (Realization)

Hiện thực hóa là quan hệ ngữ nghĩa giữa các phân lớp (classifier), trong đó một phân lớp sẽ đặc tả cam kết mà phân lớp kia phải đảm bảo đáp ứng và thực thi.

Quan hệ hiện thực hóa mà ta thường gặp là quan hệ giữa các giao diện và lớp hay thành phần thực thi hoặc quan hệ giữa ca sử dụng và các cộng tác hiện thực hóa chung.



Hình 2.8: Ký hiệu quan hệ hiện thực hóa

II.1.3. Các loại biểu đồ trong UML

Biểu đồ là sự biểu diễn đồ họa của một tập các phần tử ở dạng đồ thị, trong đó các đỉnh ứng với các phần tử và các ứng với các quan hệ.

Các biểu đồ được dùng để trực quan hóa hệ thống từ một khía cạnh nào đó, chúng chính là một phép chiếu lên hệ thống.

Trong hầu hết các hệ thống, biểu đồ tái hiện một khung nhìn về các phần tử làm nên hệ thống. Một phần tử thường xuất hiện trong một vài biểu đồ và đôi khi trong tất cả các biểu đồ.

UML chia các biểu đồ thành hai nhóm:

- **Biểu đồ cấu trúc (Structure Diagram):** Nhóm các biểu đồ này biểu diễn các cấu trúc tĩnh của hệ phần mềm cần được mô hình hóa. Các biểu đồ trong mô hình tĩnh tập trung biểu diễn khía cạnh tĩnh liên quan đến cấu trúc cơ bản cũng như các phần tử chính của hệ thống.
- **Biểu đồ hành vi (Behavior Diagram):** Nhóm biểu đồ này nhằm nắm bắt các hoạt động và hành vi của hệ thống, cũng như tương tác giữa các phần tử bên trong và bên ngoài hệ thống.

1. Biểu đồ cấu trúc

UML đề xuất bảy dạng biểu đồ trong mô hình tĩnh bao gồm:

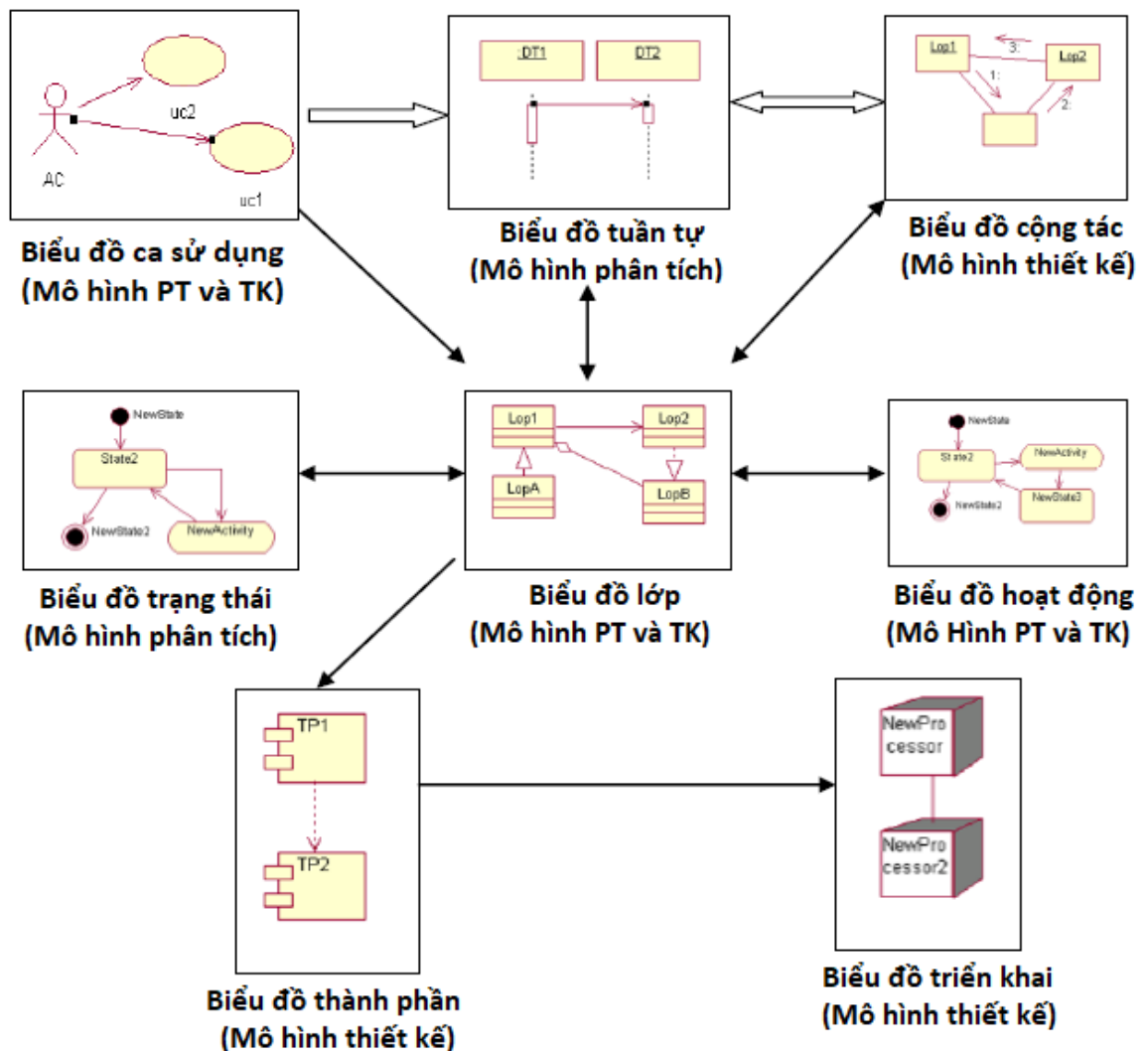
- Biểu đồ lớp (Class Diagram)
- Biểu đồ thành phần (Component Diagram)
- Biểu đồ triển khai (Deployment Diagram)
- Biểu đồ đối tượng (Object Diagram)
- Biểu đồ gói (Package Diagram)
- Biểu đồ cấu trúc phức hợp (Composite Structure Diagram)

- Biểu đồ gói mở rộng (Profile Diagram)

2. Biểu đồ hành vi

UML đề xuất trình bày dạng biểu đồ trong mô hình hành vi bao gồm :

- Biểu đồ ca sử dụng (Use case Diagram)
- Biểu đồ tuần tự (Sequence Diagram)
- Biểu đồ cộng tác (Collaboration Diagram)
- Biểu đồ trạng thái (State Diagram)
- Biểu đồ hoạt động (Activity Diagram)
- Biểu đồ bao quát tương tác (Interactive Overview Diagram)
- Biểu đồ thời khắc (Timing Diagram)



Hình 2.9: Quy trình xây dựng các biểu đồ UML

II.2. Các quy tắc ngữ nghĩa của UML

UML đưa ra các quy tắc ngữ nghĩa:

- *Tên gọi*: là cái mà ta gọi là các sự vật, các mối quan hệ và các sơ đồ.
- *Phạm vi*: là khuôn khổ cho ý nghĩa cụ thể đối với một tên gọi.
- *Tính trực quan*: các tên gọi có thể nhìn thấy và được các phần tử khác sử dụng.
- *Tính tích hợp*: các sự vật có thể liên kết với các sự vật khác như thế nào.
- *Tính thực hiện được*: nó có ý nghĩa gì để vận hành và mô phỏng một mô hình động.

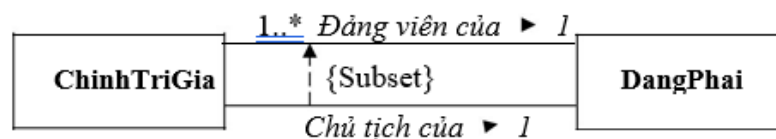
Trong UML có hai quy tắc chính:

- Quy tắc ràng buộc được sử dụng để giới hạn phạm vi của mô hình, ví dụ các quy tắc hạn chế, quy định rõ phạm trù của các mối quan hệ như kết hợp, kế thừa hay khả năng nạp chồng trong các lớp.
- Quy tắc suy diễn chỉ ra cách các sự vật có thể suy diễn được, ví dụ tuổi của một người có thể suy ra được từ *ngày/tháng/năm* hiện thời trừ đi *ngày/tháng/năm sinh*.

Lưu ý: Các qui tắc ràng buộc và suy dẫn thường được đặt trong cặp dấu ngoặc ‘{’ và ‘}’ ở bên cạnh những phần tử của mô hình, thường là các thuộc tính, hay các mối quan hệ cần phải tuân theo.

Ví dụ:

- ✓ Khi mô tả mối quan hệ giữa hai lớp DangPhai và ChinhTriGia, ta có thể sử dụng qui tắc ràng buộc để không chế các đối tượng tham gia vào các quan hệ đó.
- ✓ Mỗi ràng buộc giữa hai quan hệ được mô tả trong UML như sau:



Hình 2.10: Ví dụ về ràng buộc giữa hai quan hệ Chính trị gia và Đảng phái

- ✓ Các thuộc tính có thể bị không chế, bị giới hạn trong phạm vi xác định, ví dụ: $\{0 \leq \text{mau} \leq 255\}$ chỉ ra rằng thuộc tính mau (màu) có giá trị trong phạm vi các số nguyên từ 0 đến 255.
- ✓ Một số thuộc tính có thể được suy dẫn từ những thuộc tính khác.

Ví dụ khi thiết kế lớp *SanPham* có thuộc tính *giaBan* và *giaSanXuat*. Trong kinh doanh ta có thể xác định được ngay cách tính lợi nhuận $loiNhuan = giaBan - giaSanXuat$.

SanPham	
giaBan	{loiNhuan = giaBan - giaSanXuat}
giaSanXuat	
/ loiNhuan	

II.3. Một số cơ chế chung trong UML

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

- Các đặc tả (Specification)
- Các bài trí (Adornments)
- Sự phân hoạch chung (Common divisions)
- Các cơ chế mở rộng (Extensibility mechanisms)

II.3.1. Các đặc tả sử dụng trong UML

UML mạnh hơn một ngôn ngữ đồ họa: vì rằng bên cạnh các ký hiệu đồ họa nó còn cung cấp một cách diễn tả vượt trội bằng văn bản theo cú pháp và ngữ nghĩa của khối đồ họa sử dụng.

Ví dụ như một tập đầy đủ các thuộc tính, các tác vụ và các hành vi mà lớp đó chứa.

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

II.3.2. Các bài trí

Hầu hết các phần tử trong UML đều có ký hiệu đồ họa duy nhất, trực tiếp để cung cấp một sự thể hiện trực quan về các khía cạnh quan trọng nhất của phần tử đó.

Transaction
+execute() +rollback() #priority() -timestamp()

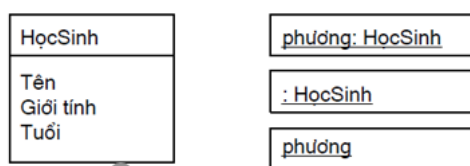
Hình 3.1: Các bài trí của một lớp

Hình 3.1 chỉ ra một lớp được bài trí cho biết lớp trừu tượng này có 2 tác vụ chung (public: +), một tác vụ được bảo vệ (protected: #) và một tác vụ riêng (private: -).

II.3.3. Sự phân hoạch chung trong UML

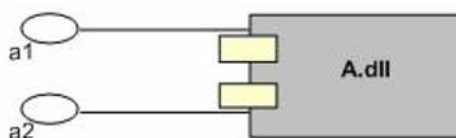
Các khái niệm mô hình hóa trong UML thường được phân thành cặp theo 2 cách:

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



Hình 3.2 : Ví dụ về lớp và đối tượng

- *Phân hoạch giao diện và triển khai một giao diện,* như khai báo một hợp đồng và triển khai một hợp đồng. Trong UML có thể mô hình hóa cả giao diện và triển khai. Ví dụ thành phần A thực hiện giao diện a1 và a2.



Hình 3.3 : Giao diện a1 và a2 của thành phần A

II.3.4. Các cơ chế mở rộng trong UML

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

Các cơ chế dùng để mở rộng gồm có:

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

II.4. Công cụ hỗ trợ mô hình hóa UML

Có rất nhiều phần mềm công cụ được sử dụng cho phát triển các hệ phần mềm hướng đối tượng theo ngôn ngữ mô hình hóa UML, bao gồm cả phần mềm bản quyền và phần mềm mã nguồn mở.

1. StarUML

Là công cụ mã nguồn mở, miễn phí và có đầy đủ chức năng để tạo ra 11 sơ đồ UML khác nhau.

Hoạt động trên đa nền tảng.

Liên kết tải xuống: <http://staruml.io/>

2. Visual Paradigm

Cung cấp một công cụ hoàn chỉnh như phân tích quy trình, thiết kế hệ thống, thiết kế cơ sở dữ liệu, v.v.

Cung cấp tính năng Lịch sử của người dùng để nắm bắt và duy trì nhu cầu của người dùng.

Liên kết tải xuống:

<https://www.visual-paradigm.com/>

3. Draw.IO

Là công cụ trực tuyến miễn phí.

Cho phép người dùng tạo và quản lý bản vẽ dễ dàng.

Không giới hạn vẽ số biểu đồ.

Liên kết tải xuống: <https://www.draw.io/>

4. Edraw

Dễ sử dụng để tạo các sơ đồ như bản đồ và các bản vẽ định hướng kinh doanh...

Cho phép tạo và quản lý bản vẽ dễ dàng.

Xuất và chia sẻ bản vẽ ở nhiều định dạng tệp quen thuộc, như PDF, Word, JPEG, PPT, v.v

Liên kết tải xuống:

<https://www.edrawsoft.com/edraw-max.php>

5. Umbrello

Là công cụ mã nguồn mở miễn phí tạo sơ đồ của phần mềm và hệ thống khác ở định dạng chuẩn.

Chạy trên đa nền tảng.

Hỗ trợ tạo mã cũng như kỹ thuật đảo ngược cho C++ và Java.

Liên kết tải xuống:

<https://umbrello.kde.org/>

6. Rational Rose

Do tập đoàn IBM sản xuất và có bản quyền.

Là phần mềm công cụ mạnh hỗ trợ cho quá trình phân tích, thiết kế hệ thống hướng đối tượng..

Hỗ trợ phát sinh mã khung chương trình trong nhiều ngôn ngữ lập trình khác nhau như: C++, Java, Visual Basic, Oracle 8,...

KẾT LUẬN

UML thường được sử dụng trong các giai đoạn khác nhau của quy trình phát triển phần mềm, bao gồm:

- **Phân tích yêu cầu:** Sử dụng biểu đồ tình huống để xác định yêu cầu của người dùng.
- **Thiết kế hệ thống:** Sử dụng các biểu đồ lớp và biểu đồ tuần tự để thiết kế kiến trúc phần mềm.
- **Tài liệu và bảo trì:** Đảm bảo rằng tài liệu hệ thống luôn được cập nhật và dễ hiểu.

UML bao gồm nhiều loại sơ đồ, mỗi loại phục vụ một mục đích nhất định trong việc cung cấp thông tin về hệ thống.

Các sơ đồ lớp và sơ đồ đối tượng là hai trong số những loại sơ đồ chính của UML.

UML giúp cải thiện khả năng giao tiếp giữa các thành viên trong nhóm phát triển phần mềm.

Ngôn ngữ này không chỉ dừng lại ở việc mô hình hóa mà còn hỗ trợ tạo tài liệu cho hệ thống phần mềm.

TÀI LIỆU THAM KHẢO

- [1] Đặng Văn Đức, Phân tích thiết kế hướng đối tượng bằng UML (Thực hành với Rational Rose), NXB Khoa học và Kỹ thuật, Hà Nội 2002.
- [2] Đặng Văn Hưng, Đoàn Văn Ban, Nguyễn Ngọc Thuận, Maintaining the amount of global information in local states of processes of distributed systems, Proceeding of the National Centre for Science and Technology of Vietnam, Volume 9, No 2, 1997.
- [3] Đoàn Văn Ban, Phân tích, thiết kế và lập trình hướng đối tượng, NXB Thống Kê 1997.
- [4] Nông Thị Oanh. Bài giảng Phân tích và thiết kế hướng đối tượng +BTL