**1.Giới Thiệu**

Ngôn ngữ mô hình hóa thống nhất (tiếng Anh: Unified Modeling Language, viết tắt thành UML) là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.

Cách xây dựng các mô hình trong UML phù hợp mô tả các hệ thống thông tin cả về cấu trúc cũng như hoạt động. Cách tiếp cận theo mô hình của UML giúp ích rất nhiều cho những người thiết kế và thực hiện hệ thống thông tin cũng như những người sử dụng nó; tạo nên một cái nhìn bao quát và đầy đủ về hệ thống thông tin dự định xây dựng. Cách nhìn bao quát này giúp nắm bắt trọn vẹn các yêu cầu của người dùng; phục vụ từ giai đoạn phân tích đến việc thiết kế, thẩm định và kiểm tra sản phẩm ứng dụng công nghệ thông tin. Các mô hình hướng đối tượng được lập cũng là cơ sở cho việc ứng dụng các chương trình tự động sinh mã trong các ngôn ngữ lập trình hướng đối tượng, chẳng hạn như ngôn ngữ C++, Java,... Phương pháp mô hình này rất hữu dụng trong lập trình hướng đối tượng. Các mô hình được sử dụng bao gồm Mô hình đối tượng (mô hình tĩnh) và Mô hình động.

UML sử dụng một hệ thống ký hiệu thống nhất biểu diễn các Phần tử mô hình (model elements). Tập hợp các phần tử mô hình tạo thành các Sơ đồ UML (UML diagrams). Có các loại sơ đồ UML chủ yếu sau:

* Sơ đồ lớp (Class Diagram)
* Sơ đồ đối tượng (Object Diagram)
* Sơ đồ tình huống sử dụng (Use Cases Diagram)
* Sơ đồ trình tự (Sequence Diagram)
* Sơ đồ cộng tác (Collaboration Diagram hay là Composite Structure Diagram)
* Sơ đồ trạng thái (State Machine Diagram)
* Sơ đồ thành phần (Component Diagram)
* Sơ đồ hoạt động (Activity Diagram)
* Sơ đồ triển khai (Deployment Diagram)
* Sơ đồ gói (Package Diagram)
* Sơ đồ liên lạc (Communication Diagram)
* Sơ đồ tương tác (Interaction Overview Diagram - UML 2.0)
* Sơ đồ phối hợp thời gian (Timing Diagram - UML 2.0)

**2.Một số dạng biểu đồ UML phổ biến**

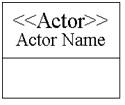
**2.1.Biểu đồ Use case (Use Case Diagram)**

Một biểu đồ Use case chỉ ra một số lượng các tác nhân ngoại cảnh và mối liên kết của chúng đối với Use case mà hệ thống cung cấp. Một Use case là một lời miêu tả của một chức năng mà hệ thống cung cấp. Lời miêu tả Use case thường là một văn bản tài liệu, nhưng kèm theo đó cũng có thể là một biểu đồ hoạt động. Các Use case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao. Các Use case định nghĩa các yêu cầu về mặt chức năng đối với hệ thống.

* Hệ thống: Với vai trò là thành phần của biểu đồ use case, hệ thống biểu diễn ranh giới giữa bên trong và bên ngoài của một chủ thể trong phần mềm chúng ta xây dựng.Một hệ thống ở trong biểu đồ use case không nhất thiết là một hệ phần mềm; nó có thể là một chiếc máy,hoặc là một hệ thống thực như một doanh nghiệp, một trường đại học,…
* Tác nhân(actor):là người dùng của hệ thống, một tác nhân có thể là một người dùng thực hoặc các hệ thống máy tính khác có vai trò nào đó trong hoạt động của hệ thống. Như vậy, tác nhân thực hiện các use case. Một tác nhân có thể thực hiện nhiều use case và ngược lại một use case cũng có thể được thực hiện bởi nhiều tác nhân

Tác nhân được kí hiệu:

hoặc



* Các use case: Đây là thành phần cơ bản của biểu đồ use case. Các use case được biểu diễn bởi các hình elip.Tên các use case thể hiện một chức năng xác định của hệ thống.

Các Use case được kí hiệu bằng hình elips.

* Mối quan hệ giữa các use case:
  + Association: thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau

Association.jpg

Ví dụ quan hệ association:

* + Include: là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.



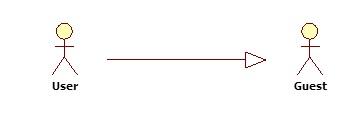
Ví dụ quan hệ include:

* + Extent: Extend dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.

Ví dụ quan hệ extent:

* + Generalization: được sử dụng để thể hiện quan hệ thừa kế giữa các Actor hoặc giữa các Use Case với nhau.

Ví dụ quan hệ Generalization:



**2.2.Biểu đồ lớp (Class Diagram)**

Một biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống. Các lớp là đại diện cho các “đối tượng” được xử lý trong hệ thống. Các lớp có thể quan hệ với nhau trong nhiều dạng thức:

* liên kết (associated - được nối kết với nhau),
* phụ thuộc (dependent - một lớp này phụ thuộc vào lớp khác),
* chuyên biệt hóa (specialized - một lớp này là một kết quả chuyên biệt hóa của lớp khác),
* hay đóng gói ( packaged - hợp với nhau thành một đơn vị).

Tất cả các mối quan hệ đó đều được thể hiện trong biểu đồ lớp, đi kèm với cấu trúc bên trong của các lớp theo khái niệm thuộc tính (attribute) và thủ tục (operation). Biểu đồ được coi là biểu đồ tĩnh theo phương diện cấu trúc được miêu tả ở đây có hiệu lực tại bất kỳ thời điểm nào trong toàn bộ vòng đời hệ thống.

Một hệ thống thường sẽ có một loạt các biểu đồ lớp – không phải bao giờ tất cả các biểu đồ lớp này cũng được nhập vào một biểu đồ lớp tổng thể duy nhất – và một lớp có thể tham gia vào nhiều biểu đồ lớp.

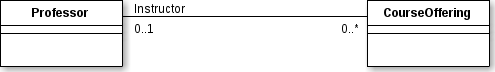
**2.2.1.Một lớp có các thành phần sau**

* Tên lớp
* Các thuộc tính
* Các phương thức

**2.2.2.Liên kết giữa các lớp**

* Liên kết (Association)
  + Mối liên hệ ngữ nghĩa giữa hai hay nhiều lớp chỉ ra sự liên kết giữa các thể hiện của chúng
  + Mối quan hệ về mặt cấu trúc chỉ ra các đối tượng của lớp này có kết nối với các đối tượng của lớp khác.

Bội số quan hệ: là số lượng thể hiện của một lớp liên quan tới một thể hiện của lớp khác. Với mỗi liên kết, có hai bội số quan hệ cho hai đầu của liên kết.

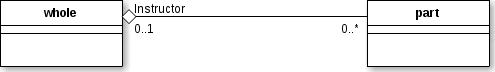
Ví dụ:

Với mỗi đối tượng của Professor, có nhiều Course Offerings có thể được dạy. Với mỗi đối tượng của Course Offering, có thể có 1 hoặc 0 Professor giảng dạy.

* Biểu diễn bội số quan hệ:

| **Biểu diễn** | **Ý nghĩa** |
| --- | --- |
|  | Unspecified |
| 1 | chính xác 1 |
| 0..\* | 0 hoặc nhiều |
| \* | 0 hoặc nhiều |
| 1..\* | 1 hoặc nhiều |
| 0..1 | 0 hoặc 1 |
| 2..4 | Specified Range |
| 2, 4..6 | Multiple, Disjoint Ranges |

* Kết tập (Aggregation)
  + Là một dạng đặc biệt của liên kết mô hình hóa mối quan hệ toàn thể-bộ phận (whole-part) giữa đối tượng toàn thể và các bộ phận của nó.
  + Kết tập là mối quan hệ “là một phần” (“is a part-of”).
  + Bội số quan hệ được biểu diễn giống như các liên kết khác



* + Cấu thành (Composition) là :Một dạng của kết tập với quyền sở hữu mạnh và các vòng đời trùng khớp giữa hai lớp

▫ Whole sở hữu Part, tạo và hủy Part.

▫ Part bị bỏ đi khi Whole bị bỏ, Part không thể tồn tại nếu Whole không tồn tại.

Sự khác nhau giữa Association, Aggregation và Composition

* Tổng quát hóa (Generalization)
  + Mối quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi với một hoặc nhiều lớp khác
  + Xác định sự phân cấp về mức độ trừu tượng hóa trong đó lớp con kế thừa từ một hoặc nhiều lớp cha

▫ Đơn kế thừa (Single inheritance)

▫ Đa kế thừa (Multiple inheritance)

* + Là mối liên hệ “là một loại” (“is a kind of”)

Lớp trừu tượng và lớp cụ thể (Abstract and Concrete Class)

**2.2.3.Gói**

Nếu bạn đang mô hình hóa một hệ thống lớn hoặc một lĩnh vực nghiệp vụ lớn, thì không thể tránh khỏi, sẽ có nhiều phân loại khác nhau trong mô hình của bạn. Việc quản lý tất cả các lớp có thể là một nhiệm vụ khó khăn, do vậy UML cung cấp một phần tử tổ chức được gọi là gói. Các gói cho phép các nhà tạo mô hình tổ chức các phân loại của mô hình thành các vùng tên, là một kiểu giống như các thư mục trong một hệ thống tệp. Việc phân chia một hệ thống thành nhiều gói làm cho hệ thống trở nên dễ hiểu, đặc biệt là nếu từng gói đại diện cho một phần cụ thể của hệ thống

Có hai cách để vẽ các gói trên sơ đồ. Không có quy tắc để xác định xem ký pháp nào sẽ được sử dụng, ngoại trừ việc tuân theo phán xét riêng của bạn về việc ký pháp nào là dễ đọc các sơ đồ lớp mà bạn đang vẽ nhất. Cả hai cách sẽ bắt đầu bằng một hình chữ nhật lớn với một hình chữ nhật nhỏ hơn (phiếu) nằm ở phía trên cùng bên trái nó, như trong . Nhưng nhà tạo mô hình phải quyết định cách thể hiện các thành viên của gói như thế nào, ví dụ như sau:

Nếu nhà tạo mô hình quyết định hiển thị các thành viên của gói bên trong hình chữ nhật lớn, thì tất cả các thành viên4 sẽ phải được đặt trong hình chữ nhật đó. Cũng vậy, tên của gói cần được đặt trong hình chữ nhật nhỏ hơn của gói Nếu nhà tạo mô hình quyết định hiển thị các thành viên của gói bên ngoài hình chữ nhật lớn, thì tất cả các thành viên sẽ được hiển thị trên sơ đồ cần phải được đặt ở bên ngoài hình chữ nhật ấy. Để cho thấy phân loại nào thuộc về gói, thì một đường thẳng sẽ được vẽ từ từng phân loại đến một vòng tròn có dấu cộng (+) bên trong vòng tròn gắn liền với gói.

Ví dụ:

====> Như vậy trong phần này tôi đã giới thiệu với các bạn về biểu đồ UML và 2 dạng biểu đồ cơ bản hay được sử dụng trong các tài liệu thiết kế hệ thống.Ở phần tiếp theo tôi sẽ tiếp tục giới thiệu tới các bạn 3 dạng biểu đồ tiếp theo là:

* Biểu đồ trạng thái (State Diagram)
* Biểu đồ tuần tự (Sequence Diagram)
* Biểu đồ hoạt động (Active Diagram)

**3.Công cụ vẽ biểu đồ UML**

* Công cụ offline
  + Diagram download tại đây: <http://diagram-designer.en.softonic.com/>
* Công cụ online
  + <https://cacoo.com/diagrams/>
  + <http://creately.com/Draw-UML-and-Class-Diagrams-Online>

**4.Tài liệu tham khảo**

Các bạn có thể xem các qui định chuẩn trong thiết kế biểu đồ UML tại đây:

* <http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm>
* <https://www.smartdraw.com/uml-diagram/>
* <http://www.holub.com/goodies/uml/>