MÔ HÌNH HOÁ PHẦN MỀM

TUẦN 1:
TỔNG QUAN

GVLT: NGUYỄN THỊ MINH TUYỀN

NỘI DUNG

1. Mô hình hoá
2. Ngôn ngữ mô hình hoá
3. Hướng đối tượng
4. UML

NỘI DUNG

1. Mô hình hoá
2. Ngôn ngữ mô hình hoá
3. Hướng đối tượng
4. UML

MỞ ĐỀ [1]

- Giả sử: Bạn cần phát triển một hệ thống phần mềm khách hàng yêu cầu.
- Thử thách đầu tiên:
  - Chỉ rõ khách hàng thật sự cần gì,
  - Hiểu chính xác yêu cầu của khác hàng cho hệ thống sẽ xây dựng chưa?
- Bước đầu tiên đã quan trọng cho sự thành bại của dự án.
- Câu hỏi đặt ra: Làm thế nào để giao tiếp với khách hàng?
  - Ngôn ngữ tự nhiên: không thực sự là một lựa chọn tốt (vì không chính xác và nhập nhằng).

MỞ ĐỀ [2]

- Những gì bạn cần: Tạo một **mô hình** cho phần mềm bạn cần xây dựng.
- Mục tiêu của **mô hình**:
  - Nổi rõ các khía cạnh quan trọng của phần mềm ở một dạng thức rõ ràng về khái niệm,
  - Đơn giản ở mức có thể nhưng đủ trừu tượng, loại bỏ các chi tiết không liên quan.
- Ví dụ: bản vẽ xây dựng.
  - Một bản vẽ xây dựng cho một toà nhà chứa thông tin ví dụ như kế hoạch xây dựng sàn.
  - Các vật liệu xây dựng không được chỉ rõ tại thời điểm này vì chúng không liên quan và làm cho kế hoạch trở nên phức tạp hơn mức cần thiết.
  - Kế hoạch cũng không chứa thông tin về thiết kế điện nước của toà nhà (sẽ sử dụng một bản kế hoạch tách biệt, nhằm tránh biểu diễn quá nhiều thứ trong cùng một bản kế hoạch).

MÔ HÌNH

- Khái niệm của một mô hình là quan trọng không chỉ trong công nghệ thông tin mà trong ngành khoa học khác (toán, lý, triết, kinh tế, ...).
- Xuất phát từ từ latin "modulus", để chỉ một tỉ lệ trong kiến trúc, suốt thời kỳ Phục hưng,
- Từ "modello" được dùng ở Ý cho đối tượng minh hoạ nhằm mục đích trình bày hình thức và thiết kế của một toà nhà đã có kế hoạch đến khách hàng và làm rõ các câu hỏi liên quan đến thiết kế và kiến trúc.
- Những thế kỷ sau đó, khái niệm "model" được sử dụng trong các ngành khoa học khác nhau để mô tả đơn giản các sự kiện phức tạp từ thực tế.

MÔ HÌNH HOÁ [1]

- Con người có khả năng tái hiện lại thực tế bằng việc áp dụng các quá trình nhận diện chủ quan của nó.
  - Trừu tượng hoá (abstraction) là một trong những quá trình nổi bật nhất.
- Trừu tượng hoá gồm khả năng tìm kiếm điểm chung từ nhiều quan sát khác nhau và sau đó tái hiện thực tế cùng lúc có khả năng:
  - tổng quát hoá các tính năng cụ thể của các đối tượng thật (generalization),
  - phân loại các đối tượng thành nhóm đồng nhất (classification) và
  - tổng hợp các đối tượng thành các đối tượng phức tạp hơn (aggregation)

MÔ HÌNH HOÁ [2]

- Tổng quát hoá, phân loại, tổng hợp đại diện cho các hành vi tự nhiên của tâm trí con người và được thực hiện bởi con người trong cuộc sống hàng ngày.
- Trừu tượng hoá cũng được áp dụng rộng rãi trong khoa học và công nghệ ➔ gọi là mô hình hoá (modeling)
- Ta có thể định nghĩa một mô hình bằng cách biểu diễn một phần hoặc đơn giản hoá thực tế, nhằm hoàn thành được một tác vụ hoặc đạt được thoả thuận về một chủ đề.

➔ Theo định nghĩa, mô hình không bao giờ mô tả đầy đủ thực tế.

VAI TRÒ VÀ MỤC ĐÍCH [1]

- Các mô hình đã và đang có tầm quan trọng trung tâm trong nhiều bối cảnh khoa học: vật lý, hoá học, triết học,...
- Mô hình thể hiện ít nhất hai vai trò bằng cách áp dụng trừu tượng hoá:
  - Tính năng giảm nhẹ: các mô hình chỉ phản ánh sự chọn lựa của các thuộc tính gốc, vì vậy tập trung vào các khía cạnh quan tâm
  - Tính năng ánh xạ: mô hình dựa vào các cá thể gốc, được lấy làm mẫu thử nghiệm của một loại cá nhân và được trừu tượng hoá và tổng quát hoá thành mô hình.
- Mục đích của mô hình:
  - descriptive purposes, prescriptive purposes
  - hoặc để định nghĩa cách hệ thống được cài đặt

VAI TRÒ VÀ MỤC ĐÍCH [2]

- **Everything is a model, since nothing can be processed by the human mind without being "modeled" .**
- Các mô hình quan trọng trọng các lĩnh vực kỹ thuật, như cơ học, kỹ thuật dân dụng, khoa học máy tính và kỹ thuật máy tính.
- Trong quy trình sản xuất: mô hình cho phép chúng tôi kiểm tra, xác minh, viết tài liệu và thảo luận các thuộc tính của sản phẩm trước khi chúng được sản xuất thực sự.
- Trong nhiều trường hợp: mô hình còn được sử dụng để trực tiếp tự động hoá việc sản xuất hàng hóa.

MÔ HÌNH HOÁ ĐỂ PHÁT TRIỂN PHẦN MỀM

- Sự cần thiết phải dựa vào mô hình để phát triển phần mềm dựa trên bốn sự kiện:
  - Sản phẩm phần mềm ngày càng phức tạp ➔ cần thảo luận ở các mức trừu tượng khác nhau
  - Phần mềm ngày càng phổ biến, và nhu cầu về phần mềm mới hoặc sự phát triển của phần mềm không ngừng tăng.
  - Thị trường việc làm liên tục thiếu hụt các kỹ năng phát triển phần mềm phù hợp với yêu cầu công việc.
  - Việc phát triển phần mềm cần tương tác với tác nhân bên ngoài (không có kiến thức về IT).

SỬ DỤNG MÔ HÌNH

- Bản phát thảo
  - Mô hình được sử dụng cho các mục đích giao tiếp, như một góc nhìn về hệ thống
- Bản thiết kế
  - Mô hình cung cấp đặc tả chi tiết và đầy đủ về hệ thống
- Chương trình
  - Mô hình được sử dụng để phát triển hệ thống thay vì sự dụng mã nguồn.
- Trong quá trình phát triển phần mềm, ta có thể sử dụng những mô hình trên theo nhiều cách khác nhau.

NỘI DUNG

1. Mô hình hoá
2. Ngôn ngữ mô hình hoá
3. Hướng đối tượng
4. UML

NGÔN NGỮ MÔ HÌNH HOÁ

- Modeling language
- Là công cụ cho phép định nghĩa một biểu diễn cụ thể của một mô hình khái niệm.
- Có thể ở dạng text (ví dụ, một ngôn ngữ lập trình như Java) hoặc biểu diễn đồ hoạ (ví dụ, một ngôn ngữ cung cấp các ký hiệu cho bóng bán dẫn, điốt ...) hoặc cả hai.
- Được phân thành hai lớp
  - Domain-Specific Languages (DSLs)
  - General-Purpose Modeling Languages (GPMLs, GMLs, or GPLs)

DOMAIN-SPECIFIC LANGUAGES

- Là ngôn ngữ được thiết kế/phát triển được đưa ra để giải quyết nhu cầu của một miền ứng dụng cụ thể.
- Đặc biệt hữu ích vì chúng được điều chỉnh theo yêu cầu của miền, cả về ngữ nghĩa và khả năng diễn đạt và về ký hiệu và cú pháp.
- Một số ví dụ về DSL:
  - VHDL language (VHSIC Hardware Description Language) là ngôn ngữ đặc tả để mô tả các thành phần phần cứng điện tử.
  - BPMN (Business Process Model and Notation)
  - WebML, HTML để phát triển web, Mathematica và MatLab cho toán học, SQL để truy cập cơ sở dữ liệu, ...

GENERAL-PURPOSE MODELING LANGUAGES

- Sử dụng đa mục đích, có thể áp dụng cho bất kỳ lĩnh vực hay miền nào cho mục đích lập mô hình
- Ví dụ: UML, Petri-nets hoặc máy trạng thái (state machines).

OCL

- Object Constraint Language
- Là một ngôn ngữ hình thức đa mục đích (dạng text) được OMG chấp nhận là một chuẩn.
- Ban đầu OCL (được phát triển bởi IBM) để khắc phục các nhược điểm của UML, được tích hợp vào UML từ năm 1997.
- OCL được dùng để bổ sung cho các metamodel với một tập hợp các quy tắc ở dạng text mà mỗi mô hình tương thích với metamodel đó phải tuân thủ.
- Là thành phần quan trong của MDE (Model-Driven Engineering)

VÍ DỤ VỀ OCL

```
context Meeting
      inv: self.end > self.start
context Team::size:Integer
      derive: self.members->size()
context Meeting
      inv: self.oclIsTypeOf(TeamMeeting)
      implies self.participants->includesAll(self.forTeam.members)
context TeamMember::numConfMeeting():Integer
      post:
      result=meetings->select(isConfirmed)->size()
```

MÔ HÌNH HOÁ HƯỚNG ĐỐI TƯỢNG

- Là một dạng thức mô hình hoá tuân thủ mô hình hướng đối tượng.

NỘI DUNG

1. Mô hình hoá
2. Ngôn ngữ mô hình hoá
3. Hướng đối tượng
4. UML

LỚP

- Lớp mô tả các thuộc tính và hành vi của một tập đối tượng ở mức trừu tượng ➔ phản ánh đặc tính chung của nhóm đối tượng.
- Ví dụ:
  - Lớp Nguoi có tên, địa chỉ, số CMND.
  - Lớp KhoaHoc có ID, tên, mô tả.
  - Lớp PhongHoc có tên và vị trí, ...
- Một lớp cũng định nghĩa một tập các thao tác được cho phép có thể áp dụng cho các instance của lớp.
- Ví dụ: Ta có thể đặt một phòng học cho một ngày nào đó, một học sinh có thể đăng ký một kỳ thi ...

ĐỐI TƯỢNG

- Instance của một lớp.
- Một đối tượng có một trạng thái nào đó. Trạng thái được biểu diễn bởi các thuộc tính.
- Ví dụ:
  - E301 là một instance của lớp PhongHoc.

ĐÓNG GÓI

- Encapsulation
- Là một cơ chế bảo vệ chống lại việc truy cập trái phép vào trạng thái bên trong của một đối tượng thông của giao điện được định nghĩa duy nhất.
- Các mức độ visibility của các giao diện giúp ta định nghĩa các quyền truy cập khác nhau.
- Ví dụ: `public`, `private`, `protected` trong Java tương ứng cho phép truy cập bởi tất cả, chỉ trong phạm vi đối tượng và chỉ cho các thành viên của lớp, lớp con, cùng gói.

THÔNG ĐIỆP (MESSAGES)

- Các đối tượng giao tiếp với nhau thông qua thông điệp.
- Một thông điệp gởi đến một đối tượng biểu diễn một yêu cầu thực thi một thao tác.
- Bản thân đối tượng quyết định có thực thi thao tác đó hay không và bằng cách nào. Thao tác chỉ được thực thi nếu người gởi có quyền gọi thao tác đó
- Trong nhiều ngôn ngữ lập trình và mô hình hoá hướng đối tượng, khái niệm overloading được hỗ trợ: Cho phép một thao tác được định nghĩa một cách khác cho các loại tham số khác nhau.
  - Ví dụ, phép toán + được cài đặt tuỳ thuộc vào toán tử là số nguyên (1+1 =2) hay nối chuỗi ("a"+"b" = "ab")

KẾ THỪA [1]

- Inheritance
- Là một cơ chế cho việc phát sinh lớp mới từ các lớp đã tồn tại. Một lớp con được sinh ra từ lớp đã có (lớp cha) kế thừa tất cả những thuộc tính và thao tác có thể nhìn thấy được của lớp cha.
- Một lớp con có thể:
  - Định nghĩa các thuộc tính và/hoặc thao tác mới
  - Ghi đè cài đặt của các thao tác kế thừa
  - Thêm code riêng của nó vào các thao tác được kết thừa

KẾ THỪA [2]

- Việc kế thừa cho phép tạo ra các lớp mở rộng ➔ tạo ra cây kế thừa lớp, là cái cơ bản của việc phát triển hệ thống hướng đối tượng.
- Việc kế thừa có nhiều ưu điểm:
  - tái sử dụng các phần chương trình hay mô hình (vì vậy tránh dư thừa và lỗi),
  - định nghĩa nhất quán các giao diện,
  - sử dụng như một trợ giúp mô hình thông qua việc phân loại tự nhiên của các yếu tố xảy ra, và hỗ trợ cho việc phát triển tăng dần, ví dụ tinh chỉnh từng bước của các khái niệp chung thành khái niệm cụ thể.

ĐA HÌNH

- Polymorphism
- Là khả năng thích nghi theo các dạng thức khác nhau.
- Trong quá trình thực thi một chương trình, thuộc tính đa hình có thể có các tham chiếu đến các đối tượng từ các lớp khác nhau.
  - Khi thuộc tính này được khái báo, một kiểu (ví dụ lớp Person) được gán cố định tại thời gian biên dịch.
  - Tại thời gian thực thi, thuộc tính này có thể bị ràng buộc động thành kiểu con (ví dụ lớp con Employee hoặc lớp con Student).
- Một thao tác đa hình có thể được thực thi trên các đối tượng từ các lớp khác nhau và có ngữ nghĩa khác nhau cho mỗi trường hợp.

NỘI DUNG

1. Mô hình hoá
2. Ngôn ngữ mô hình hoá
3. Hướng đối tượng
4. UML

UML

- Unified Modeling Language
- Là một ngôn ngữ mô hình hoá đồ hoạ để biểu diễn trực quan, xây dựng, tài liệu hoá các phần của phần mềm.
- Cung cấp một cách chuẩn để tạo các bản thiết kế hệ thống.
- Là một tập các khái niệm, không phải là phương pháp cũng không phải là quy trình.
- Phiên bản mới nhất: 2.5.1

LỊCH SỬ [1]

- Khái niệm hướng đối tượng trong Công nghệ thông tin bắt nguồn từ đầu những năm 1960.
  - Các ý tưởng ban đầu được cài đặt trên hệ thống Sketchpad.
- Ngôn ngữ lập trình SIMULA: ngôn ngữ lập trình hướng đối tượng đầu tiên, giới thiệu các khái niệm: classes, objects, inheritance, và dynamic binding
- Các thập kỷ sau đó: Các khái niệm trên mở ra một cuộc cách mạng trong việc phát triển phần mềm, cho ra đời các ngôn ngữ lập trình C++, Eiffel, Smalltalk: chứa nhiều khái niệm quan trọng của ngôn ngữ lập trình hiện đại.

LỊCH SỬ [2]

- Hướng đối tượng trở thành mô hình phát triển phần mềm quan trọng nhất: phản ánh trong Java, C# và ngôn ngữ mô hình hoá UML.
- Vào những năm 80: Ada hỗ trợ khái niệm package, task
- Vào cuối 80, đầu 90: mô hình hoá có vô số ngôn ngữ khác nhau ➔ các ngôn ngữ này không tương thích với nhau.
- 1996: Để chấm dứt cuộc chiến về khái niệm, công cụ OMG (Object Management Group) đã kêu gọi xây dựng đặc tả cho một chuẩn mô hình hoá thống nhất.

LỊCH SỬ [3]

- 1995: Grady Booch, Ivar Jacobson, và James Rumbaugh đã kết hợp các ý tưởng và phương pháp của họ tại hội nghị OOPSLA (Object-Oriented Programming, Systems, Languages, and Applications) với các mục tiêu:
  - Sử dụng các khái niệm hướng đối tượng để biểu diễn các hệ thống hoàn chỉnh hơn là chỉ một phần của phần mềm.
  - Thiết lập mối quan hệ rõ ràng giữa khái niệm mô hình hoá và mã nguồn chương trình
  - Xem xét các yếu tố mở rộng trong các hệ thống phức tạp và quan trọng.
  - Tạo ra ngôn ngữ mô hình hoá có thể được xử lý bằng máy móc và con người có thể đọc được.

  → UML 1.1 vào năm 1997

CÁC PHIÊN BẢN CỦA UML

Phiên bản	Ngày ra đời	Mô tả
1.1	11/1997	UML 1.1 được chấp nhận bởi OMG
1.3	03/2000	Chứa một số thay dổi về UML metamodel, ngữ nghĩa, khái niệm
1.4	09/2001	Cập nhật : visibility, stereotypes, biểu đồ tương tác
1.5	03/2003	Thêm actions
2.0	08/2005	Thay đổi đáng kể: thêm nhiều biểu đồ mới ...
2.1	04/2006	Thay đổi nhỏ từ UML 2.0
2.2	02/2009	Thay đổi nhỏ từ 2.1
2.3	05/2010	Thay đổi nhỏ từ 2.2
2.4.1	08/2011	Thay đổi nhỏ, cập nhật class, package
2.5	06/2015	"minor revision" của UML 2.4.1
2.5.1	**12/2017**	**Cập nhật từ 2.5**

ƯU ĐIỂM CỦA UML

- Là môt ngôn ngữ hình thức
- Ngắn gọn
- Toàn diện
- Dễ mở rộng
- Được xây dựng dựa trên kinh nghiệm
- Là một chuẩn

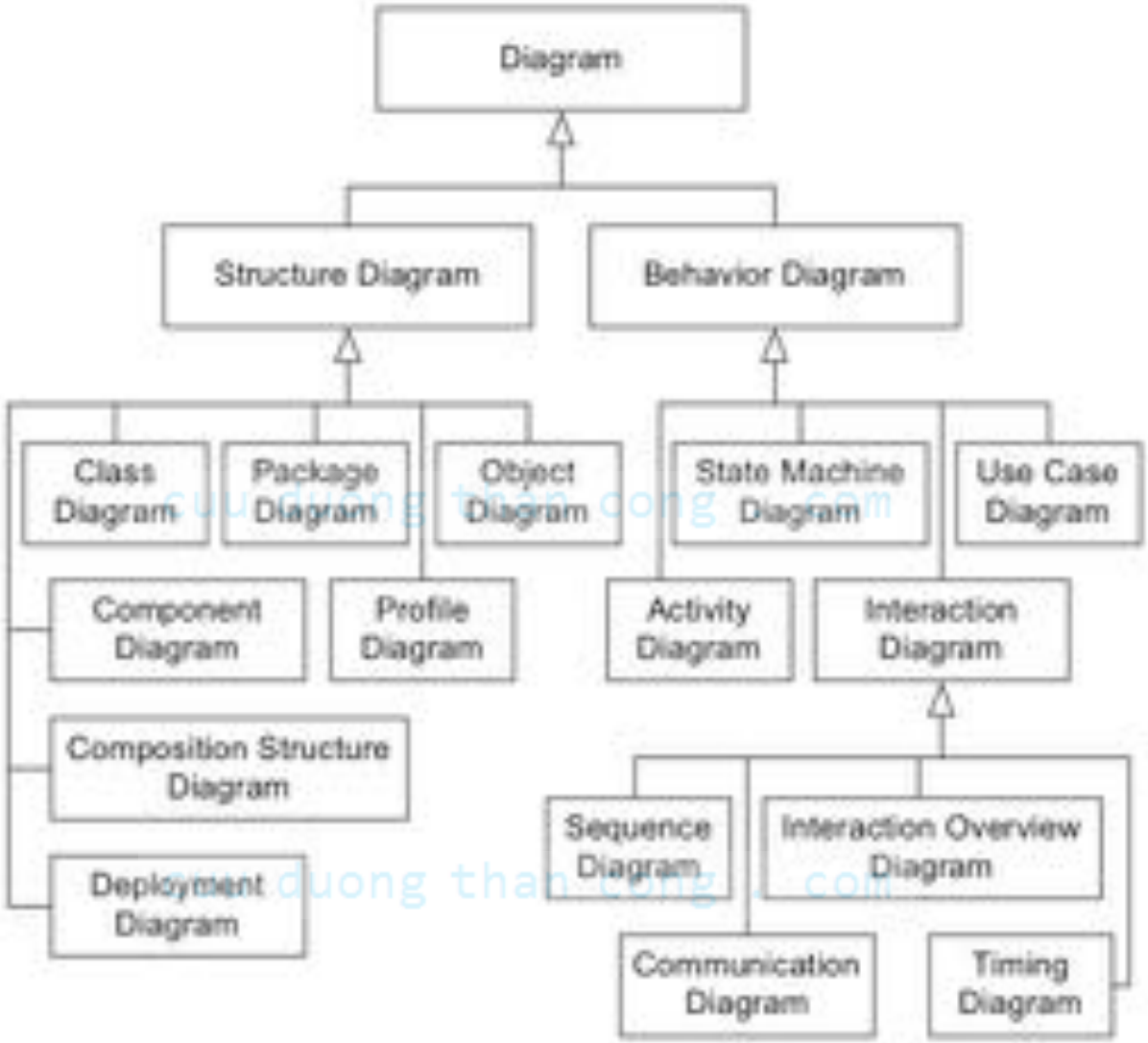
SỬ DỤNG UML

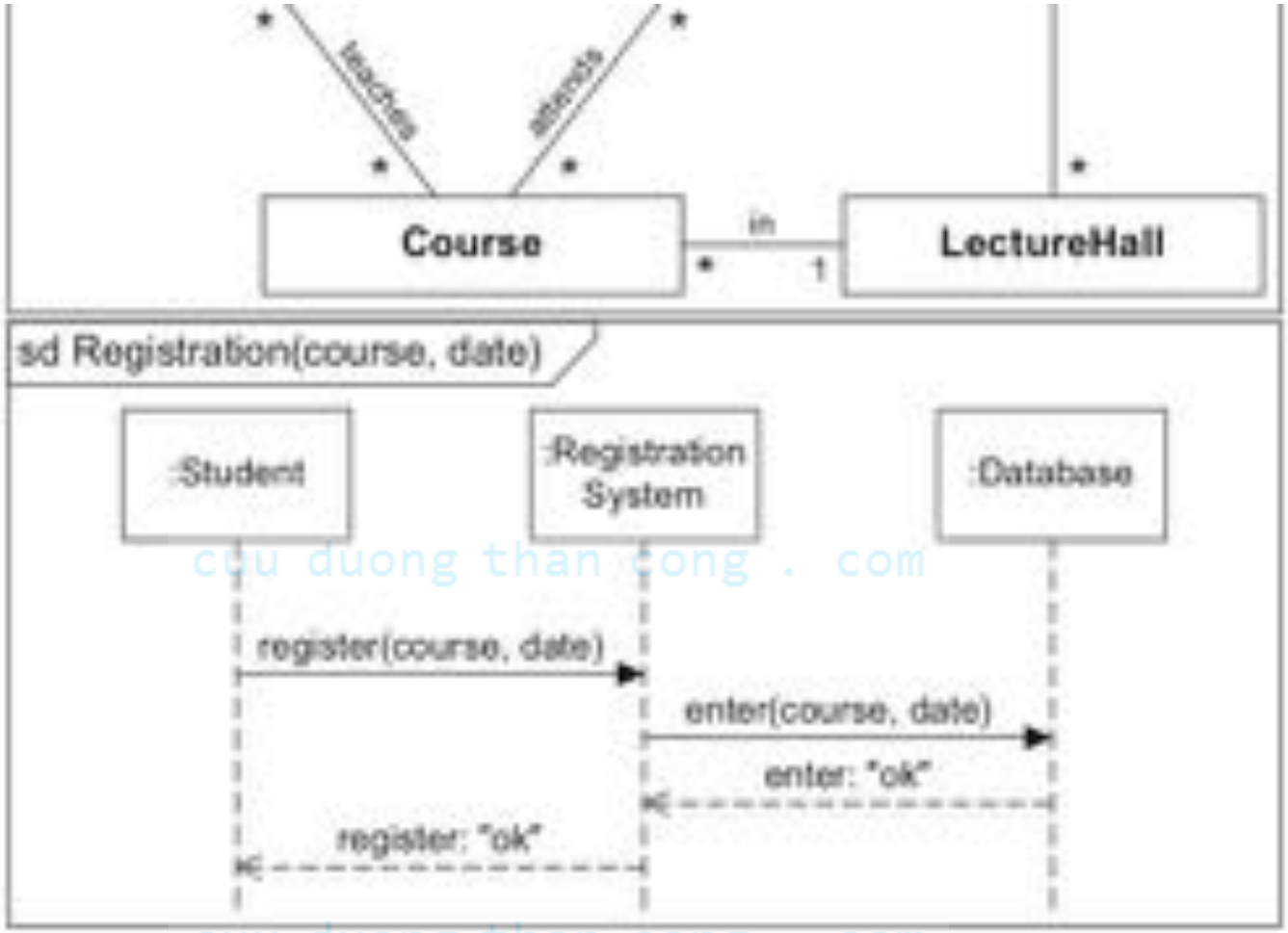
- UML không gắn với một công cụ phát triển, ngôn ngữ lập trình hay nền tảng cụ thể, cũng không cung cấp một quy trình phát triển phần mềm.
- UML tách biệt ngôn ngữ mô hình hoá và phương pháp mô hình hoá.
- UML có thể được sử dụng một cách đồng nhất trong toàn bộ quy trình phát triển phần mềm.
- Tại tất cả các giai đoạn, các khái niệm cùng ngôn ngữ có thể được sử dụng cùng ký hiệu ➔ Mô hình có thể được tinh chỉnh ở các giai đoạn.

- Không cần một mô hình để dịch sang ngôn ngữ mô hình hoá khác ➔ cho phép sử dụng trong tiến trình phát triển phần mềm tiến hoá.
- UML cũng tương thích với các lĩnh vực ứng dụng khác.
- Các phần tử mô hình UML và việc sử dụng đúng đắn được đặc tả trong UML metamodel.

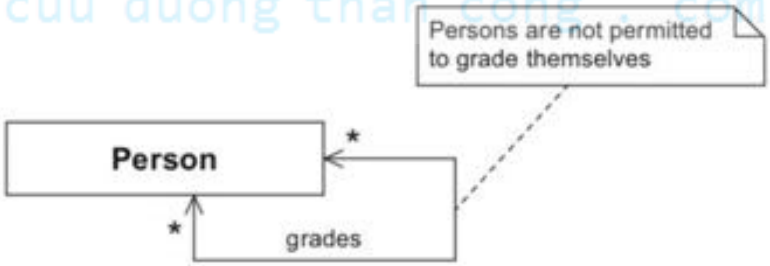
CÁC BIỂU ĐỒ UML

- Trong UML, một mô hình được biểu diễn ở dạng đồ hoạ dưới hình thức các biểu đồ/sơ đồ (diagram)
- Một biểu đồ cung cấp một góc nhìn về một phần thực tế được mô tả bởi mô hình.
- UML cung cấp 14 loại biểu đồ khác nhau mô tả cấu trúc hoặc hành vi của hệ thống.





NOTE



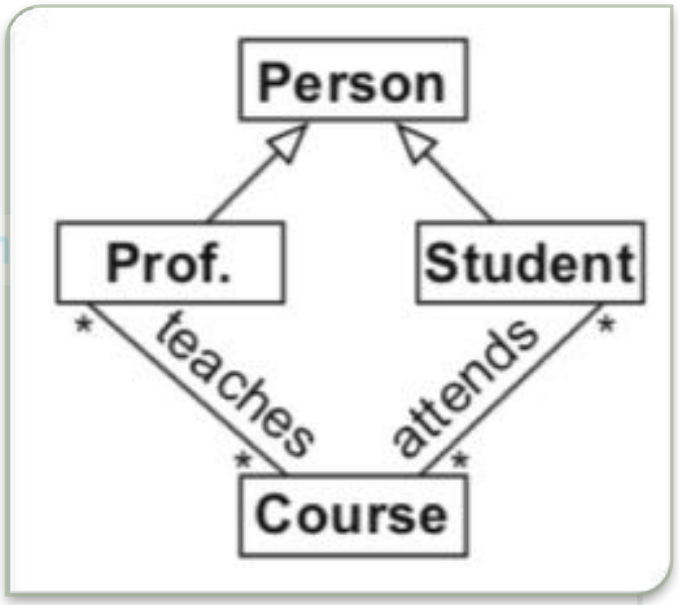
STRUCTURE DIAGRAMS

- 7 loại biểu đồ để mô hình hoá cấu trúc của một hệ thống từ các góc nhìn khác nhau.
  - Class Diagram
  - Object Diagram
  - Package Diagram
  - Component Diagram
  - Composition Structure Diagram
  - Deployment Diagram
  - Profile Diagram

cuu duong than cong . com

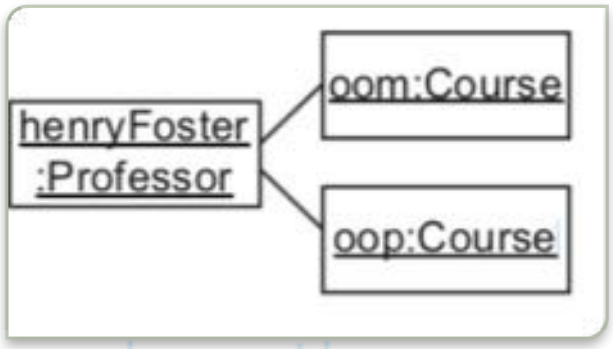
CLASS DIAGRAM

- Khái niệm về biểu đồ lớp bắt nguồn từ việc mô hình hoá dữ liệu khái niệm và phát triển phần mềm hướng đối tượng.
- Các khái niệm này được dùng để đặc tả cấu trúc dữ liệu và cấu trúc đối tượng của một hệ thống.
- Dựa vào các khái niệm: class, generalization, association



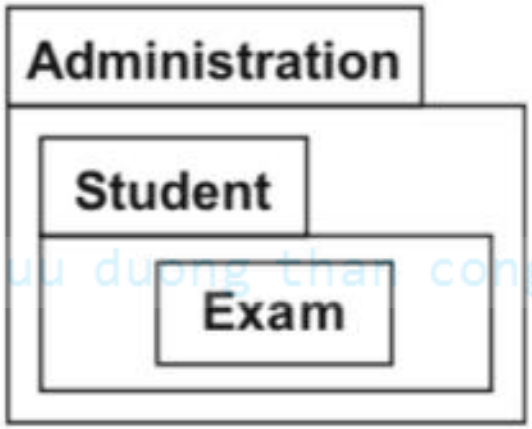
OBJECT DIAGRAM

- Dựa vào các định nghĩa của biểu đồ lớp có liên quan, biểu đồ đối tượng chỉ ra "snapshot" cụ thể của trạng thái hệ thống tại thời gian thực thi cụ thể.



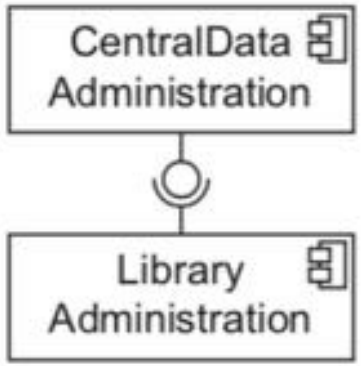
PACKAGE DIAGRAM

- Gom nhóm các biểu đồ hoặc các phần tử mô hình theo thuộc tính chung, ví dụ như sự gắn kết về chức năng.
- Thường được tích hợp trong các biểu đồ khác hơn là thể hiện ở một biểu đồ riêng lẻ



COMPONENT DIAGRAM

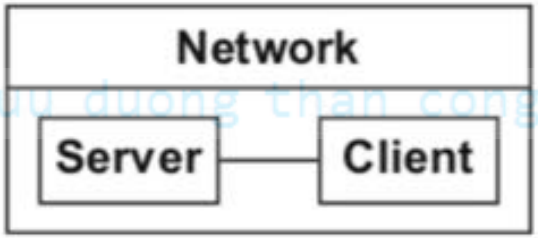
- Một component là một đơn vị độc lập, thực thi được, cung cấp cho các component khác các dịch vụ hoặc sử dụng dịch vụ từ các component khác.
- UML không có quy định chặt chẽ nào về sự tách biệt giữa khái niệm hướng đối tượng và hướng component.
- Khi đặc tả một component, ta có thể mô hình hoá hai góc nhìn một cách rõ ràng:
  - góc nhìn từ bên ngoài (black box view) : biểu diễn đặc tả của component
  - Góc nhìn từ bên trong (white box view): định nghĩa việc cài đặt của component



COMPOSITION STRUCTURE DIAGRAM

- Cho phép phân rã có phân cấp các thành phần của hệ thống.
- Sử dụng biểu đồ này để mô tả cấu trúc bên trong của các lớp và các thành phần ở mức chi tiết.

→ Cho phép ta đạt được mức độ chi tiết cao hơn biểu đồ lớp vì việc mô hình hoá nằm trong bối cảnh cụ thể.

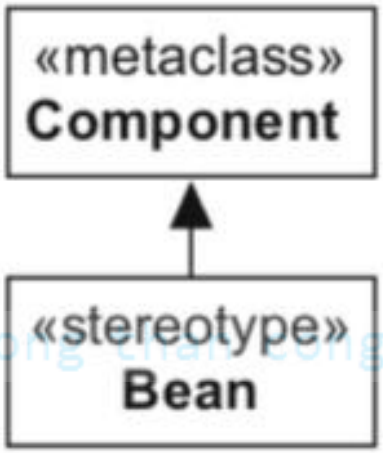


DEPLOYMENT DIAGRAM

- Cấu trúc liên kết phần cứng được sử dụng và hệ thống thời gian thực thi được biểu diễn bằng biểu đồ triển khai.
- Phần cứng bao gồm các đơn vị xử lý dưới dạng các node và mối quan hệ giao tiếp giữa các node. Hệ thống thời gian thực thi chứa các "artifact" được triển khai tới các node.

PROFILE DIAGRAM

- Sử dụng *profiles*, ta có thể mở rộng UML để giới thiệu các khái niệm miền cụ thể.



BEHAVIOR DIAGRAMS

- Cung cấp cơ sở hạ tầng cho phép ta định nghĩa hành vi ở mức chi tiết.
- Đề cập đến kết quả trực tiếp của một hành động của ít nhất một đối tượng.
- Ảnh hưởng đến cách mà các trạng thái của đối tượng thay đổi theo thời gian.
- Hành vi có thể được đặc tả thông qua các hành động của một đối tượng đơn lẻ hoặc kết quả từ việc tương tác giữa các đối tượng.

USE CASE DIAGRAM

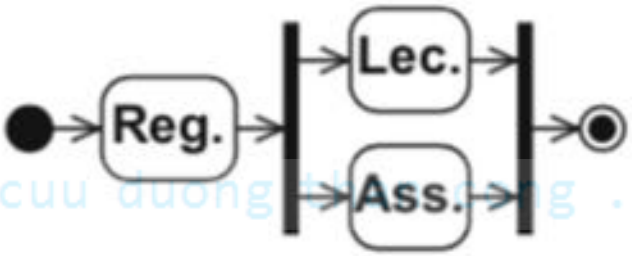
- Cho phép định nghĩa các yêu cầu của một hệ thống cần phải thực hiện.
- Biểu đồ mô tả người sử dụng nào sử dụng chức năng nào của hệ thống mà không cần đặc tả chi tiết việc cài đặt.
- Các đơn vị của chức năng mà hệ thống cung cấp cho người dùng được gọi là các use case

STATE MACHINE DIAGRAM

- Trong vòng đời của mình, đối tượng đi qua một tập các trạng thái.
- Mô tả hành vi được phép của một đối tượng ở dạng thức các trạng thái có thể có và việc chuyển tiếp giữa các trạng thái được kích hoạt bởi các sự kiện khác nhau.

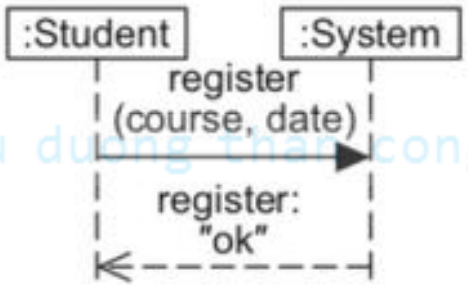
ACTIVITY DIAGRAM

- Được sử dụng để mô hình hoá các tiến trình: cả tiến trình nghiệp vụ và các tiến trình phần mềm.
- Cung cấp các cơ chế điều khiển luồng cũng như cơ chế dòng dữ liệu để điều hướng các hành động sinh ra một hoạt động, đó là một tiến trình.



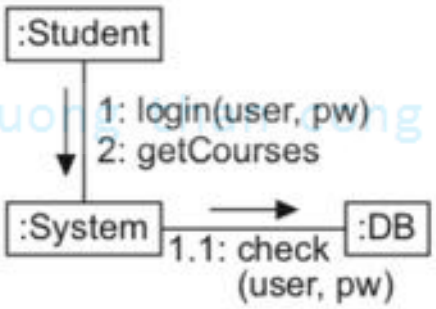
SEQUENCE DIAGRAM

- Mô tả các tương tác giữa các đối tượng để thực hiện một tác vụ cụ thể.
- Tập trung vào trình tự theo thời gian của các thông điệp được trao đổi giữa các đối tác tương tác với nhau.
- Các cấu trúc khác nhau để điều khiển thứ tự thời gian của thông điệp cũng như khái niệm cho việc mô-đun hoá cho phép ta mô hình hoá các tương tác phức tạp.



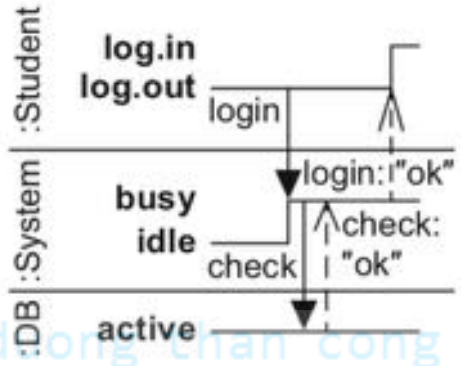
COMMUNICATION DIAGRAM

- Tương tự như biểu đồ tuần tự, mô tả việc giao tiếp giữa các đối tượng khác nhau.
- Trọng tâm là quan hệ giao tiếp giữa các đối tác tương tác với nhau hơn là dựa vào thứ tự thời gian của việc trao đổi thông điệp.
- Biểu đồ này chỉ rõ ai tương tác với ai.



TIMING DIAGRAM

- Chỉ rõ những thay đổi trạng thái của các đối tác tương tác có thể xảy ra do các sự kiện về thời gian hoặc kết quả của việc trao đổi thông điệp.



INTERACTION OVERVIEW DIAGRAM

- Mô hình hoá việc kết nối giữa các tiến trình tương tác khác nhau bằng việc thiết lập các biểu đồ tương tác riêng lẻ (biểu đồ tuần tự, biểu đồ giao tiếp, biểu đồ thời gian, ...) trong chuỗi thời gian và nguyên nhân.
- Nó cũng đặc tả các điều kiện trong đó các tiến trình tương tác được phép thực hiện.
- Để mô hình hoá dòng điều khiển, các khái niệm từ biểu đồ hoạt động được sử dụng.