

# PHÂN TÍCH YÊU CẦU PHẦN MỀM

## MÔ HÌNH HÓA YÊU CẦU

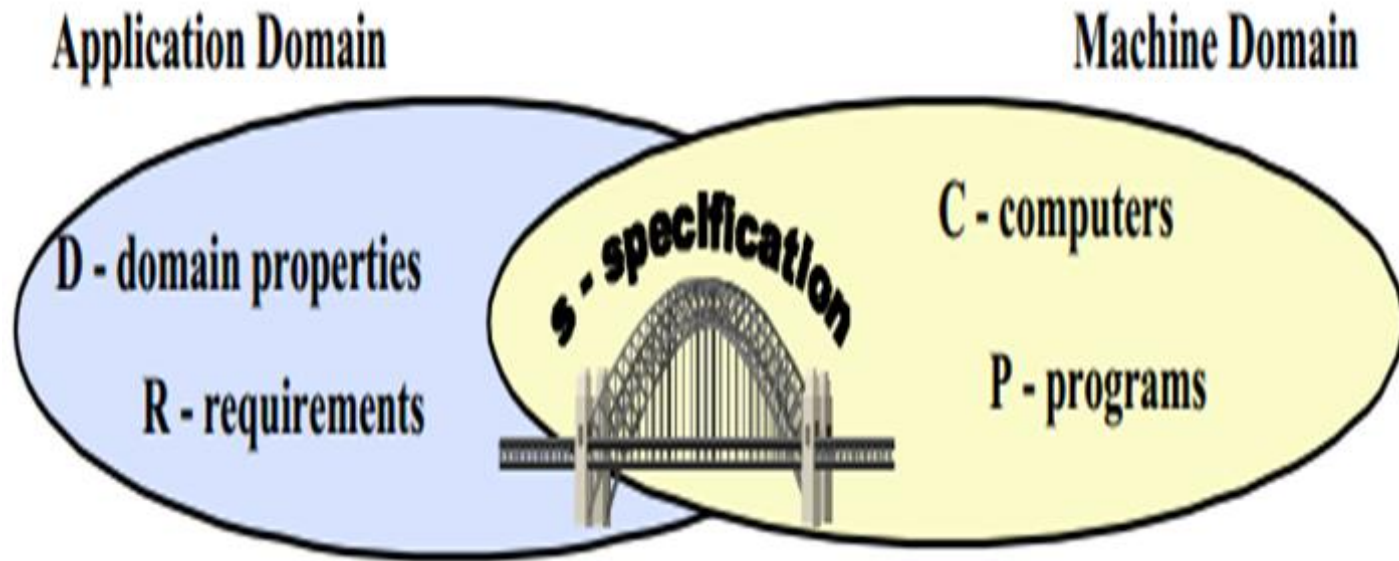
Nguyễn Thị Thu Hương  
BM CNPM - Khoa CNTT  
Email: [huongnt@tlu.edu.vn](mailto:huongnt@tlu.edu.vn)



# NỘI DUNG

- Các khái niệm cơ bản
- Các ngôn ngữ mô hình hóa
- Nguyên tắc mô hình hóa
- Mô hình hoá tương tác
- Mô hình đối tượng

# Các khái niệm cơ bản



# Các khái niệm cơ bản

- D – Đặc tính lĩnh vực: những thứ có thật trong lĩnh vực ứng dụng dù chúng ta có thiết kế chúng trong hệ thống hay không.
- R – Các yêu cầu: những thứ trong lĩnh vực ứng dụng mà chúng ta mong muốn trở thành hiện thực bằng cách đưa vào hệ thống.

# Các khái niệm cơ bản

- S – Đặc tả: mô tả các hành vi chương trình cần thực hiện để đáp ứng với các yêu cầu
- C - Máy tính: Môi trường để hệ thống được vận hành.
- P – Chương trình: sản phẩm (hệ thống) mà dự án phải đáp ứng

# Mô hình hóa

- Giúp đơn giản hóa thế giới thực bằng các mô hình
- Giúp hiểu rõ hơn về hệ thống dưới các góc nhìn khác nhau

# Vai trò của mô hình hóa trong RE

## ➤ Hướng dẫn suy luận yêu cầu

- Nó có thể giúp bạn chỉ ra câu hỏi gì để hỏi
- Nó có thể giúp làm nổi rõ các yêu cầu ẩn chứa

## ➤ Cung cấp sự đo lường cho quy trình

Việc hoàn thiện của mô hình -> hoàn thiện của suy luận

# Vai trò của mô hình hóa trong RE

## ➤ Giúp làm rõ các vấn đề

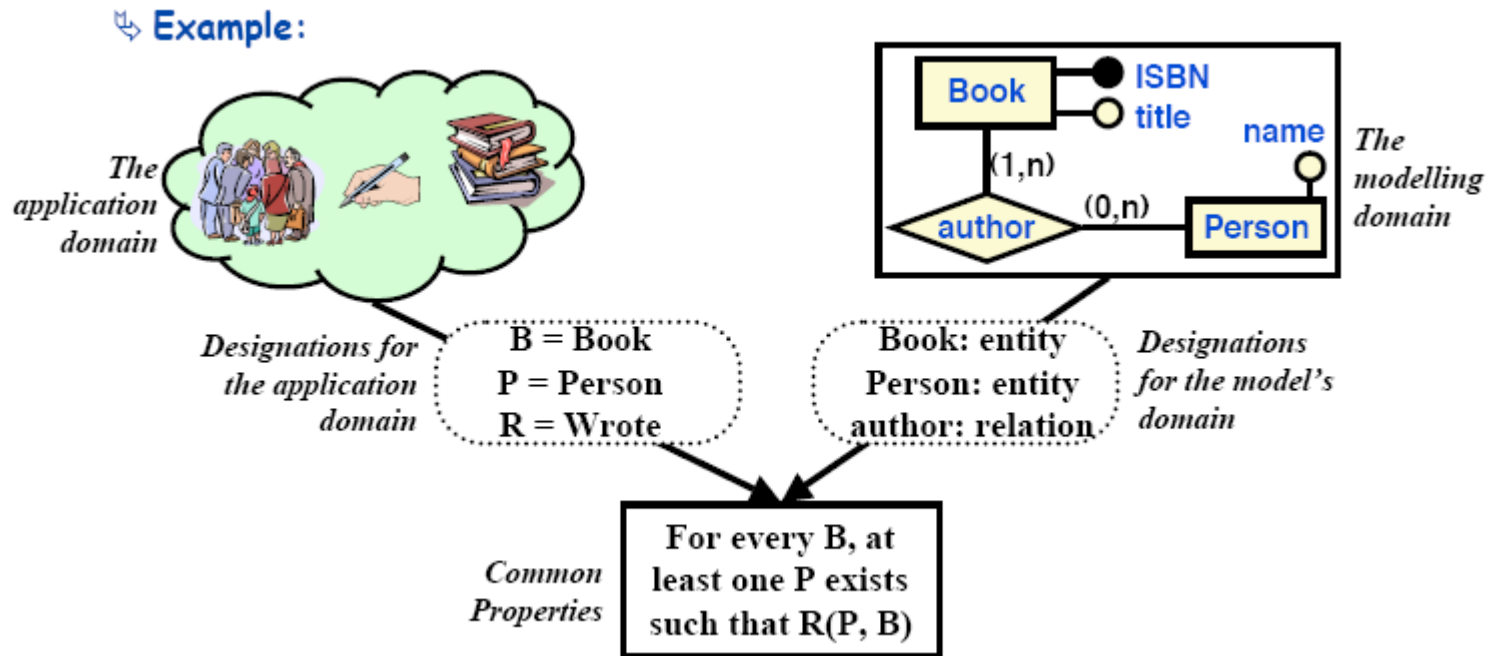
- Việc hoàn thiện của mô hình → hoàn thiện của suy luận (ví dụ: các yêu cầu xung đột/không thể thực hiện, nhầm lẫn, bất đồng giữa các đối tác)

## ➤ Kiểm tra sự thấu hiểu của nhà phân tích yêu cầu về hệ thống

- Lý giải trên các mô hình để hiểu kết quả của nó
- Xây dựng hình ảnh bằng các mô hình giúp quan sát/kiểm chứng các yêu cầu



# Ví dụ



Mô hình sẽ hữu ích khi các hiện tượng của mô hình phù hợp một cách có hệ thống với các hiện tượng trong lĩnh vực mà nó cần được mô hình hóa

# Yêu cầu đối với các thành phần

- Đặc tả (**S**) được cho trong thuộc tính của lĩnh vực (**D**) thỏa mãn các yêu cầu (**R**)
- Chương trình (**P**) thực hiện trên một máy tính (**C**) cụ thể đáp ứng với đặc tả (**S**)

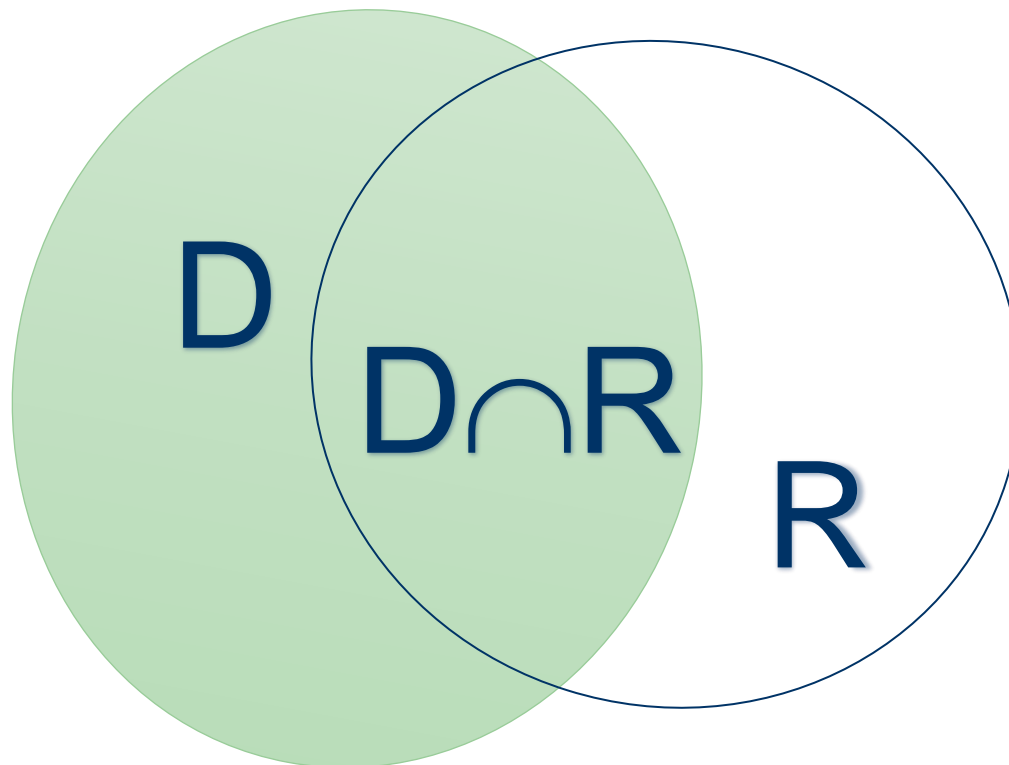
# Yêu cầu đối với các thành phần

- Chúng ta đã xem xét (và hiểu) tất cả các yêu cầu (Requirements) quan trọng?
- Chúng ta đã xem xét (và hiểu) tất cả các thuộc tính lĩnh vực (Domain properties) liên quan?

# Hạn chế của mô hình hóa

- Hiện tượng trong mô hình lại không có trong lĩnh vực ứng dụng.
- Hiện tượng trong lĩnh vực ứng dụng lại không được mô tả trong mô hình.

# Mô tả miền D và miền R



## Ví dụ

Ngăn chặn truy cập trái phép từ các máy tính

**D: Đặc tính lĩnh vực**

- Những người có quyền truy cập thì có passwords
- Passwords không bao giờ được chia sẻ với những người không có quyền truy cập

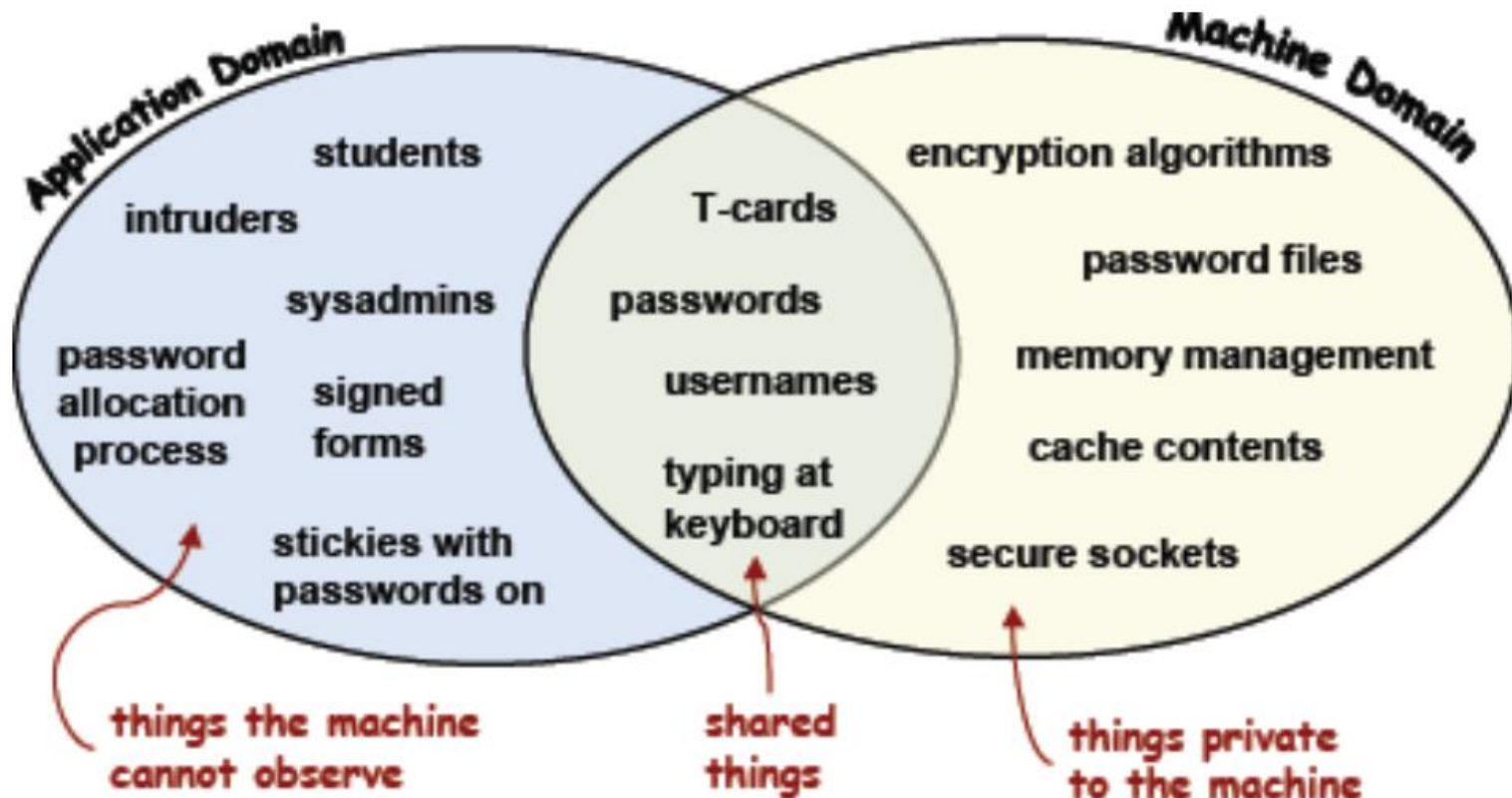
**R: Yêu cầu:**

Cơ sở dữ liệu chỉ có thể được truy cập bởi những người có quyền

**S – Đặc tả:**

- Truy cập vào CSDL chỉ được chấp nhận sau khi người dùng gõ vào một password được cấp

Ví dụ: Ngăn chặn truy cập trái phép từ các máy tính



# Nguyên tắc mô hình hóa

- Trừu tượng hóa (Abstraction)
- Phân tách (Decomposition)
- Quy chiếu (Projection)
- Mô-đun hóa (Modularity)
- Mẫu (Patterns)



# Nguyên tắc mô hình hóa

- Trừu tượng hóa: Không đi vào chi tiết, chỉ tập trung vào các nội dung chính (quan trọng)
- Phân tách: Phân chia vấn đề thành các phần độc lập để khảo sát riêng biệt
- Quy chiếu: Phân chia các khía cạnh khác nhau và mô tả chúng một cách riêng biệt

# Nguyên tắc mô hình hóa

- Mô-đun hóa: Lựa chọn các cấu trúc ổn định theo thời gian để dễ định vị khi thay đổi
- Mẫu: Cấu trúc của một mô hình đã có và được sử dụng (xuất hiện) trong nhiều ứng dụng khác nhau

# Ngôn ngữ (ký pháp) mô hình hóa

- Ngôn ngữ tự nhiên
- Ngôn ngữ (ký pháp) bán hình thức
- Ngôn ngữ (ký pháp) hình thức

# Ký pháp mô hình hóa

## ➤ Ngôn ngữ tự nhiên:

- Diễn cảm và linh hoạt: hữu ích cho suy diễn và lập các mô hình ký hiệu dễ đọc
- Khó để nắm bắt được các quan hệ mấu chốt

# Ký pháp mô hình hóa

## ➤ Ký pháp bán hình thức

- Nắm được cấu trúc và một số ngữ nghĩa
  - Có thể thực hiện (một số) hoạt động, kiểm tra tính nhất quán, ảnh động,
  - Gần như là trực quan – cho phép chuyển thông tin một cách nhanh chóng đến các dạng đối tác khác nhau
- Ngôn ngữ hình thức UML phù hợp ở đây

# Chọn ký pháp cho việc mô hình hóa

## ➤ Ký pháp hình thức

- Ngữ nghĩa chính xác, có thể suy luận rộng:  
các mô hình dựa trên cơ sở toán
- Các mô hình rất chi tiết



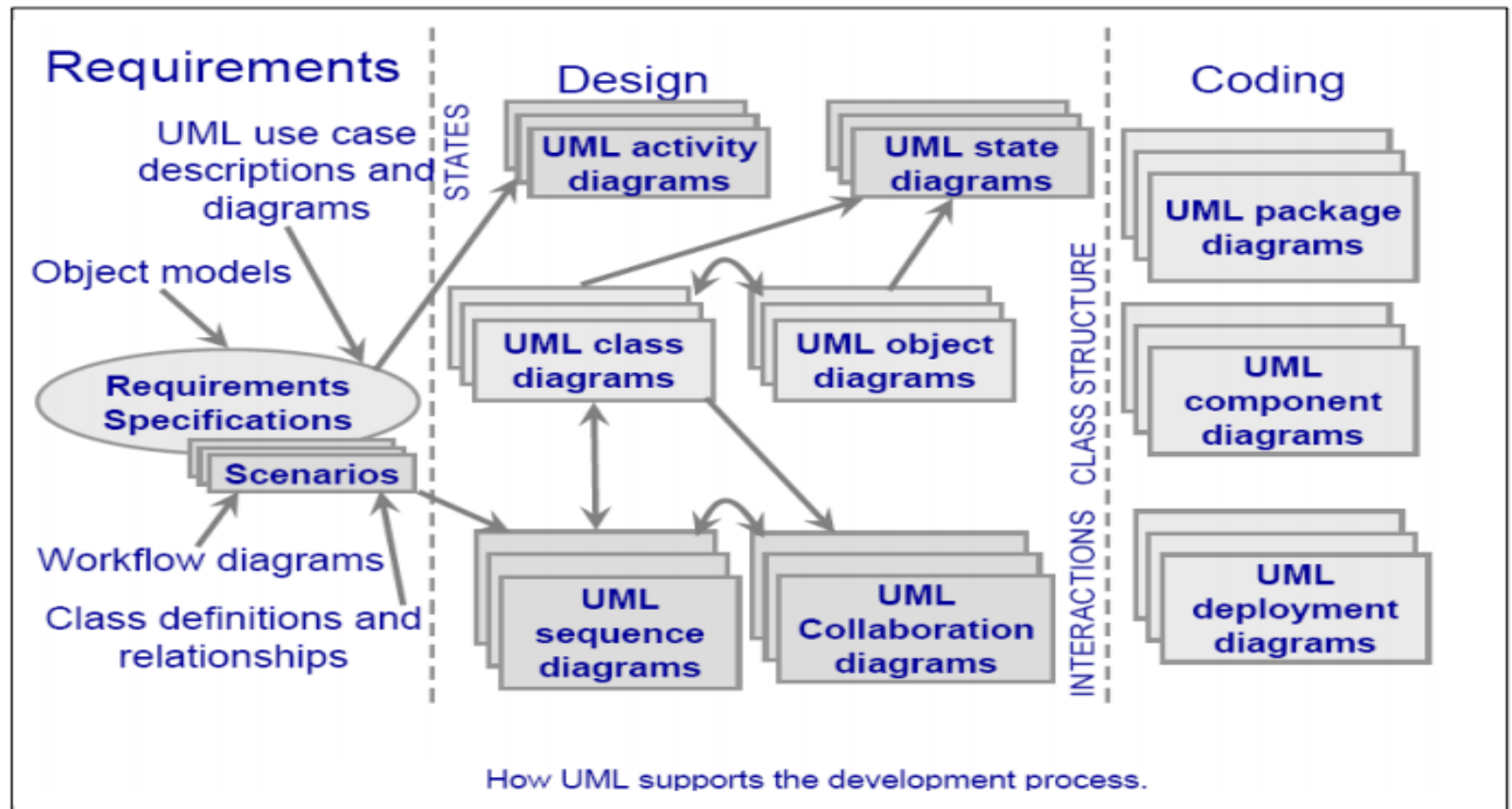
# MÔ HÌNH HÓA TƯƠNG TÁC HỆ THỐNG

# UML (Unified Modeling Language)

- Là một công nghệ chuẩn cho việc mô hình hóa phần mềm hướng đối tượng,
- Là kết quả của sự thống nhất hệ thống ký hiệu của 3 phương pháp hướng đối tượng tiêu biểu
  - ✓ OMT (James Rumbaugh) - Object Modeling Technique
  - ✓ OOSE (Ivar Jacobson) - Object-Oriented Software Engineering
  - ✓ Booch91 (Grady Booch)
- Được hỗ trợ bởi một số CASE tools (Rational ROSE, TogetherJ)
- Bạn có thể mô hình 80% của hầu hết vấn đề bằng cách dùng chỉ khoảng 20% UML,



# Vai trò của UML trong dự án phần mềm



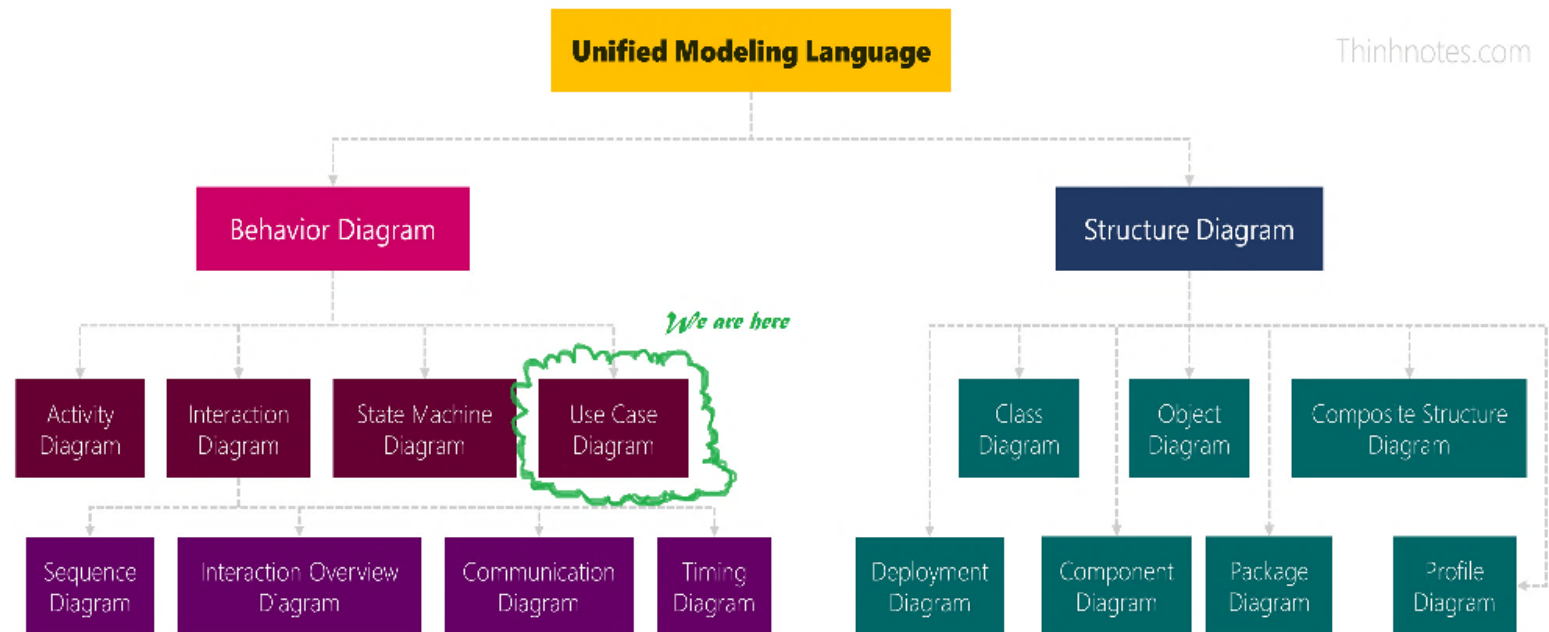
# NỘI DUNG

- Tương tác với hệ thống
  - Con người sẽ tương tác với hệ thống thế nào?
  - Hệ thống tương tác với các hệ thống khác như thế nào?
- Use case (UC)
  - Giới thiệu mô hình UC
  - Định nghĩa các tác nhân
  - Định nghĩa các tình huống (cases)
  - Các đặc tính mở rộng
- Biểu đồ tuần tự:  
Trình tự thời gian của các sự kiện trong UC

## Use case (UC)

*Use Case là kỹ thuật dùng để mô tả sự tương tác giữa người dùng và hệ thống với nhau, trong một môi trường cụ thể và vì một mục đích cụ thể.*

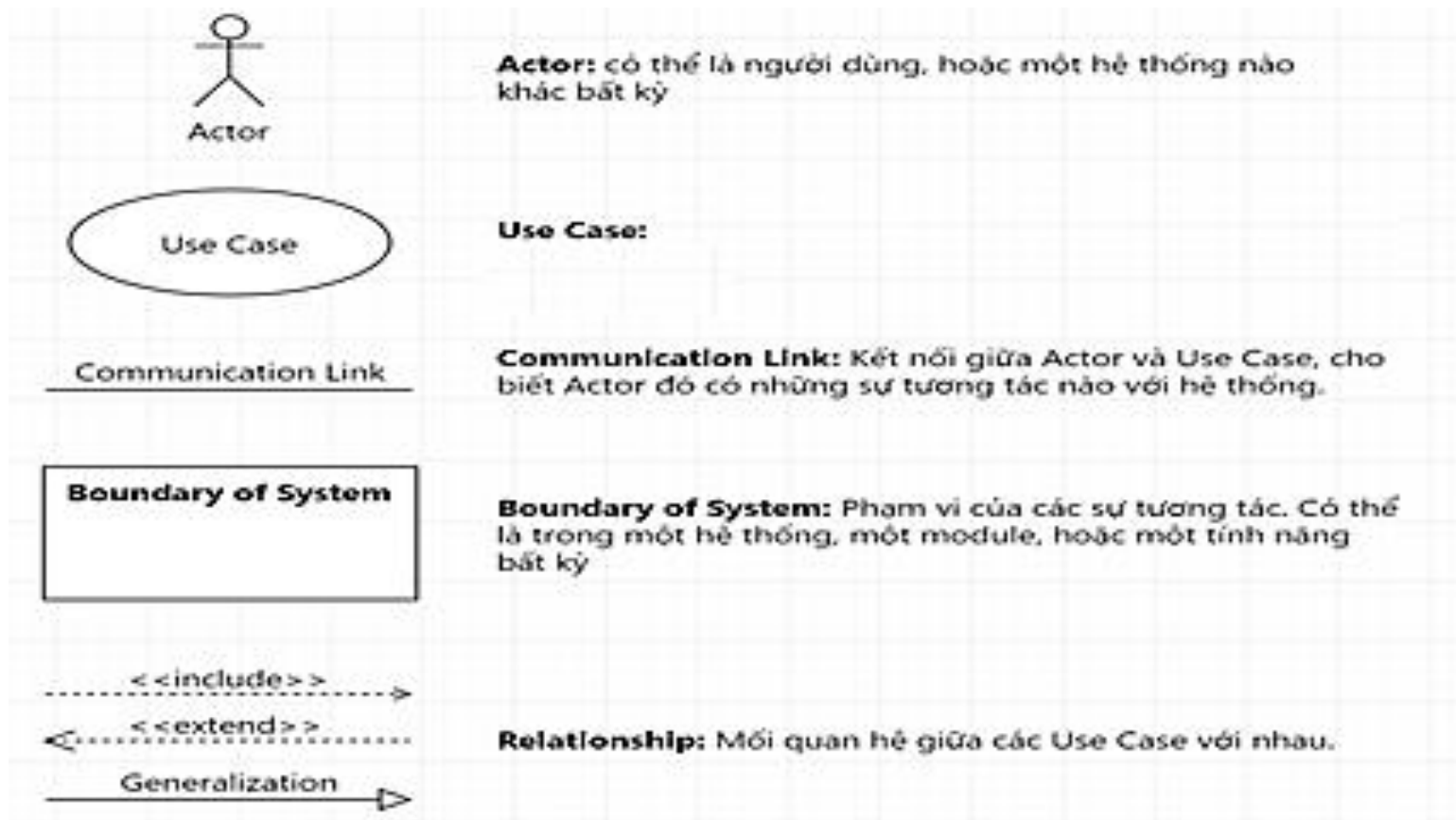
# Use cases (UCs)



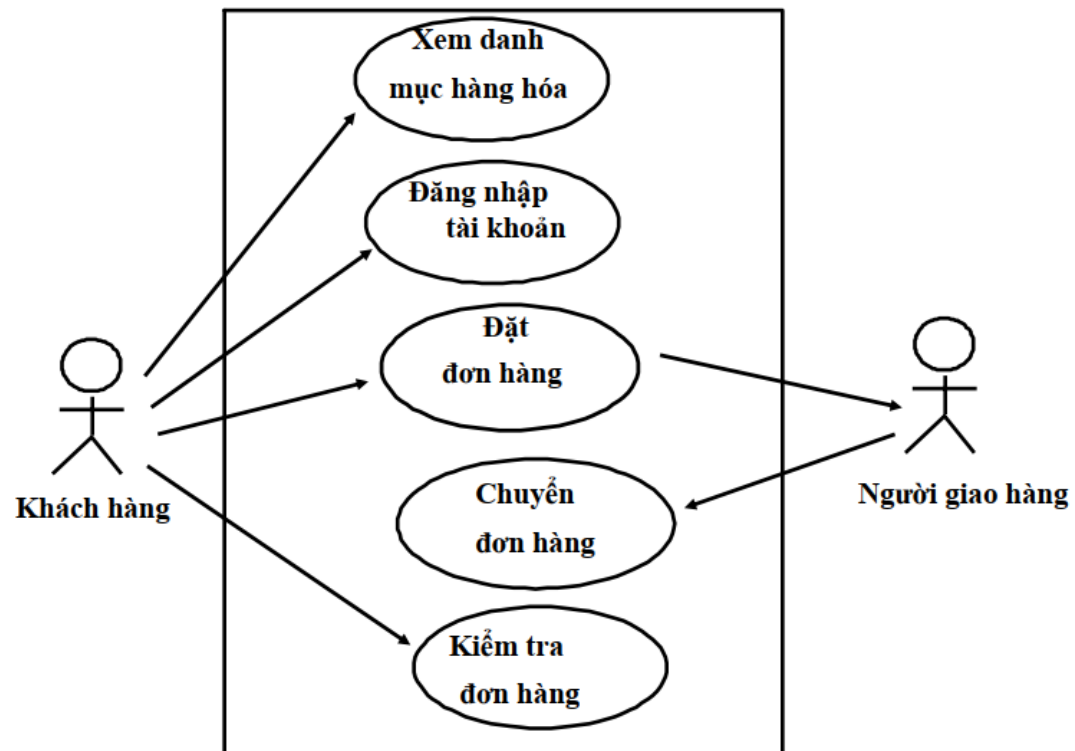
# CÁC THÀNH PHẦN TRONG SƠ ĐỒ UC

- Tác nhân (Actor)
- Ca sử dụng (Use Case)
- Các liên kết (Communication Link)
- Phạm vi (Boundary of System)
- Các mối quan hệ (Relationships)

# CÁC THÀNH PHẦN TRONG SƠ ĐỒ UC



# Ví dụ



# Tác nhân (Actor)

- ✓ Một tác nhân là một người hoặc một đối tượng khác có tương tác với hệ thống.
- ✓ Mọi stakeholder của hệ thống là các ứng cử cho các tác nhân
- ✓ Có thể xác định các tác nhân từ các UC



# Xác định các tác nhân

- ✓ Ai là người sử dụng hệ thống?
- ✓ Ai sẽ là người Admin của hệ thống (tức người cài đặt, quản lý, bảo trì... hệ thống)?
- ✓ Hệ thống này có được sử dụng bởi bất kỳ một hệ thống nào khác không?
- ✓ Ai là người đưa dữ liệu vào hệ thống?
- ✓ Ai là người cần những dữ liệu đầu ra?

## Ví dụ: Xác định Actor từ Stakeholders

Với dự án xây dựng website cho công ty du lịch, trong số các Stakeholders của dự án, ai là tác nhân của hệ thống?

# Ví dụ: Xác định Actor

Trong số các Stakeholders dưới đây, ai là tác nhân của hệ thống

1. Người chủ công ty du lịch (Travel Agency Owner)
2. Người dùng 1 – Người từ U.S. (User 1)
3. Người dùng 2 – Người từ Pháp (User 2)
4. Người phát triển (Developer)
5. Người quản lý nội dung (content Manager)
6. Bộ phận phục vụ khách hàng (Customer Service Representative)
7. Người cung cấp khách sạn (Hotel Provider), Công ty cho thuê xe ô tô (Car Rental Agent) và đại diện hãng hàng không (Airline Representative)

# Actors

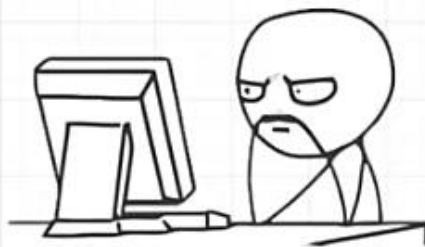
1. Người chủ công ty du lịch (Travel Agency Owner)
2. Người dùng 1 – Người từ U.S. (User 1)
3. Người dùng 2 – Người từ Pháp (User 2)
4. Người phát triển (Developer)
5. Người quản lý nội dung (content Manager)
6. Bộ phận phục vụ khách hàng (Customer Service Representative)
7. Người cung cấp khách sạn (Hotel Provider), Công ty cho thuê xe ô tô (Car Rental Agent) và đại diện hãng hàng không (Airline Representative)

# Use cases (UCs)

- ✓ Các UC được khởi tạo bởi một **tác nhân**.
- ✓ Các UC mô hình hóa **một tương tác** giữa tác nhân và hệ thống.
- ✓ Các UC nắm bắt các yêu cầu chức năng của hệ thống.
- ✓ Mỗi UC biểu biểu một luồng sự kiện hoàn thiện và đầy đủ ý nghĩa.
- ✓ Use Case phải diễn tả được Requirement theo góc nhìn cụ thể từ phía người dùng.

→ *UC chính là các chức năng mà các Actor sử dụng trên hệ thống*

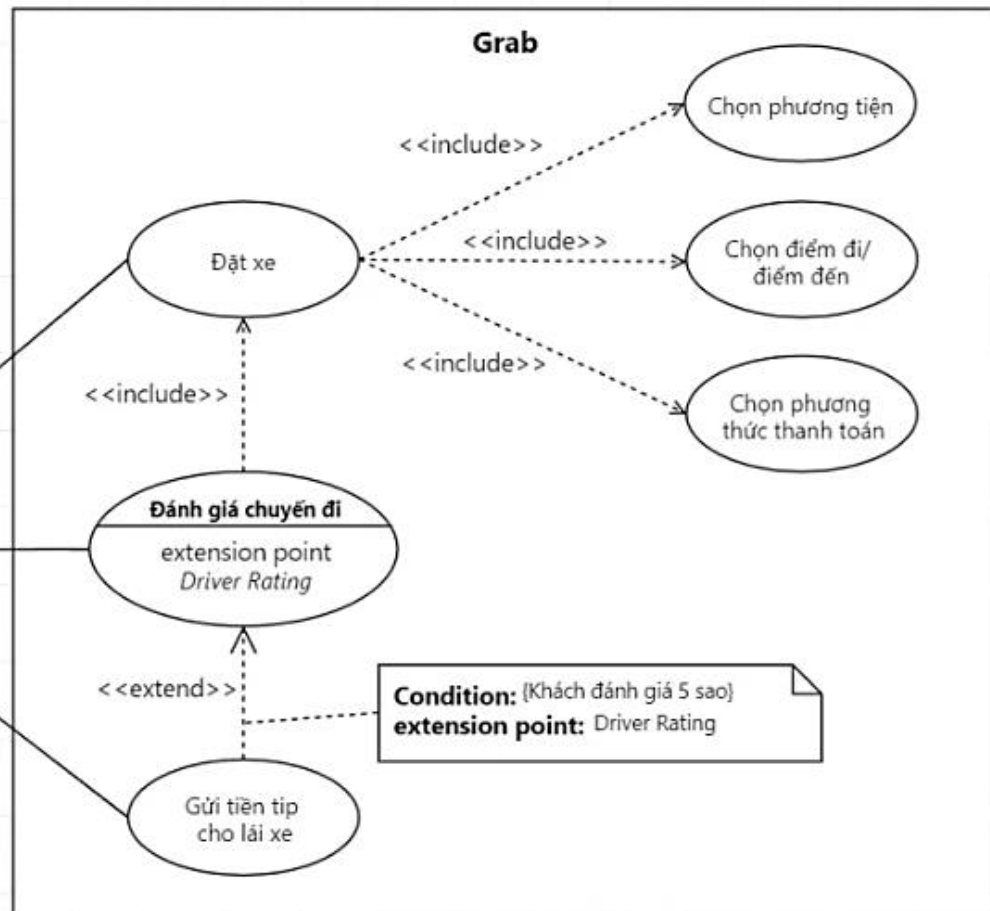
# Quan hệ <<extend>> và <<include>>



Include, extend???



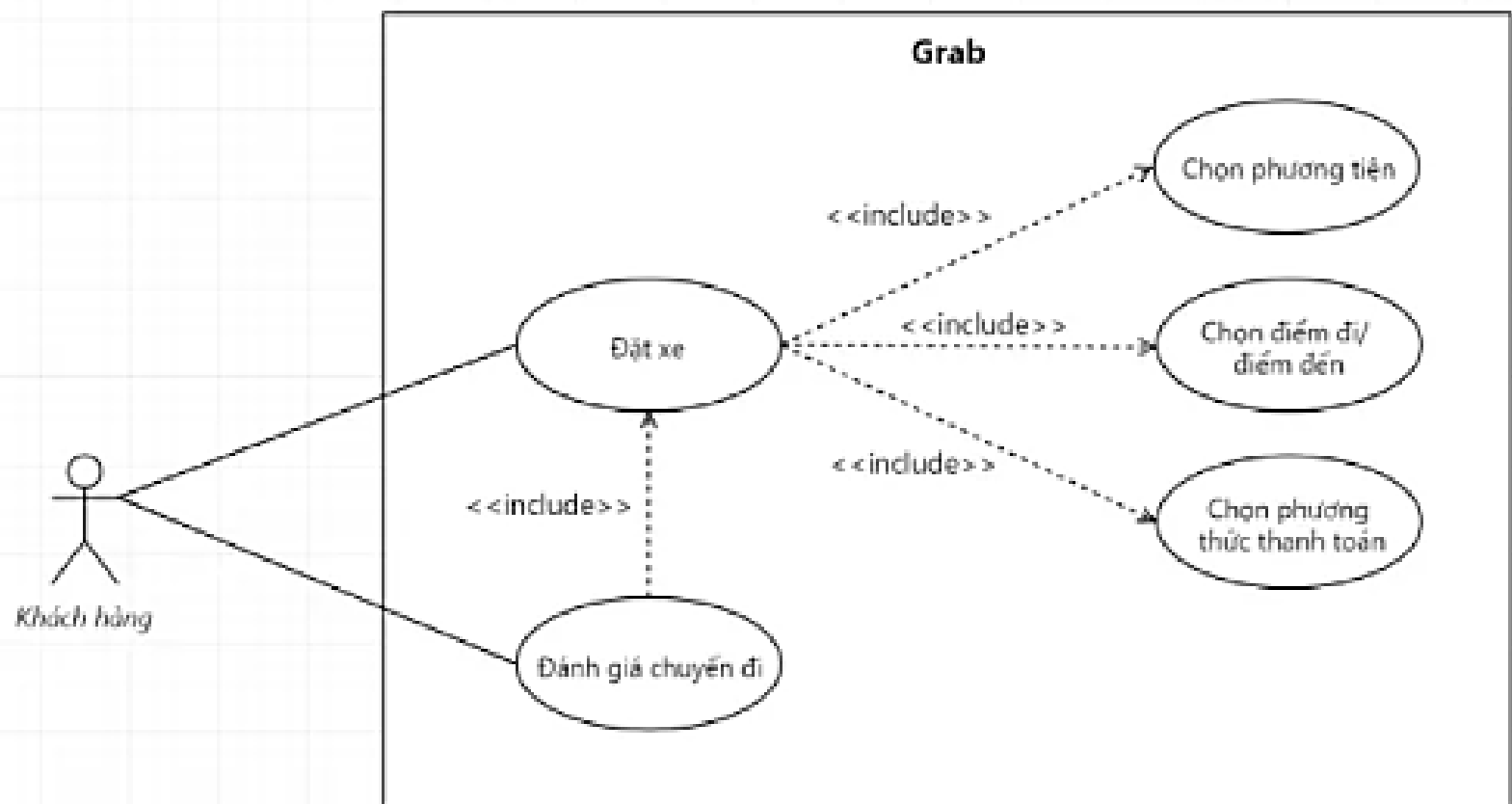
**USE CASE  
DIAGRAM**



# Quan hệ <<extend>> và <<include>>

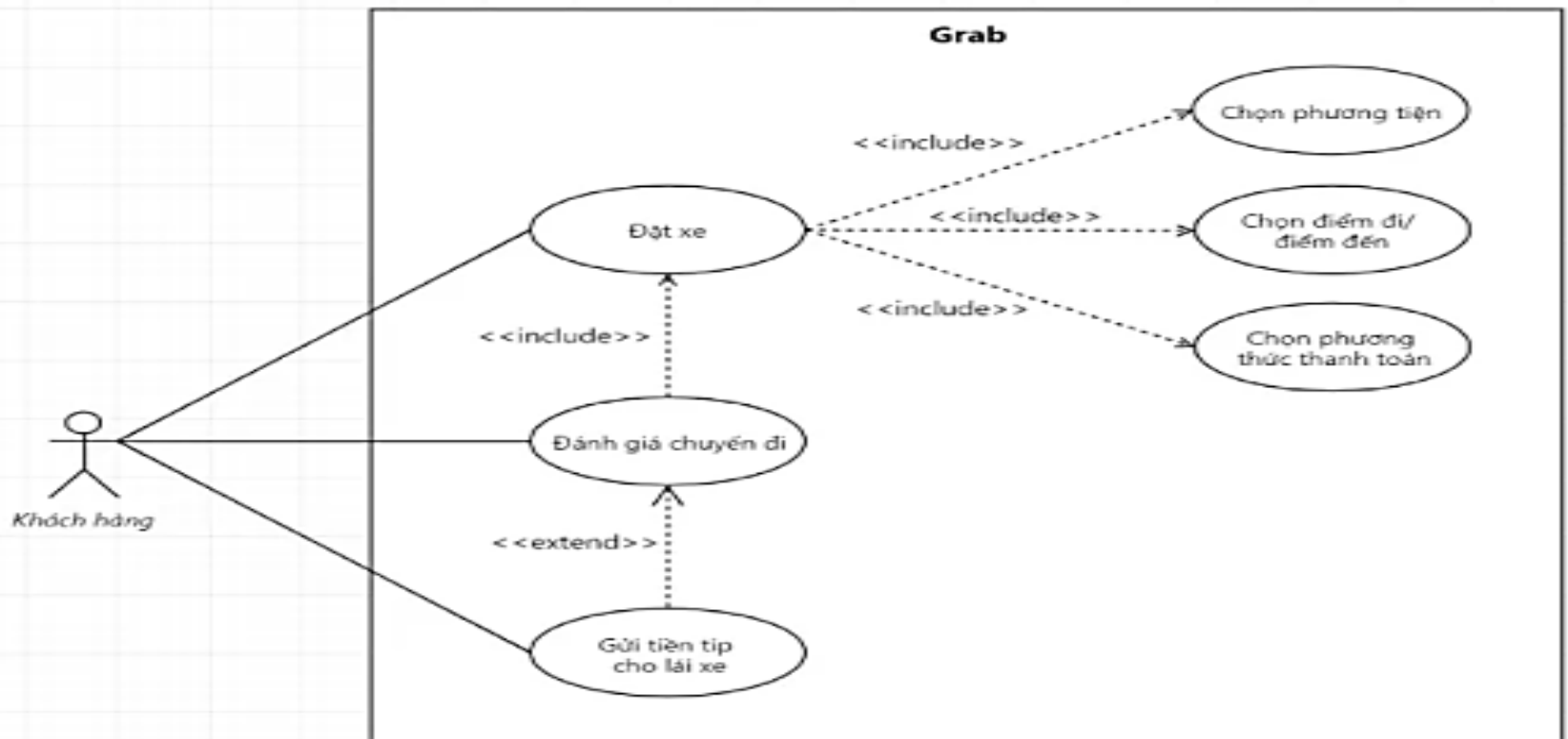
- ✓ Include là mối quan hệ bắt buộc phải có giữa các Use Case với nhau.
- ✓ Extend là mối quan hệ mở rộng giữa các Use Case với nhau.

# Quan hệ <<include>>



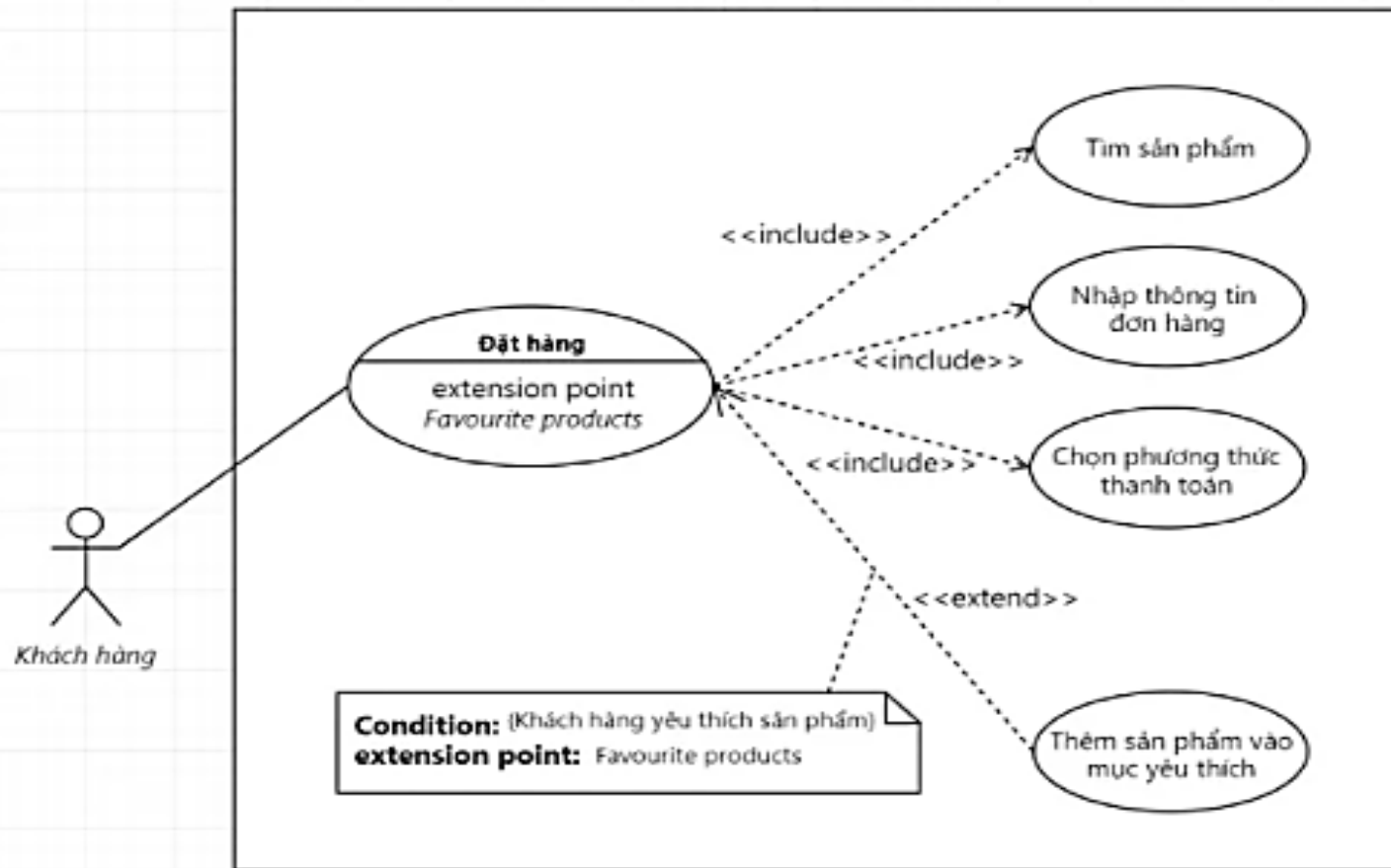


# Quan hệ <<extend>>

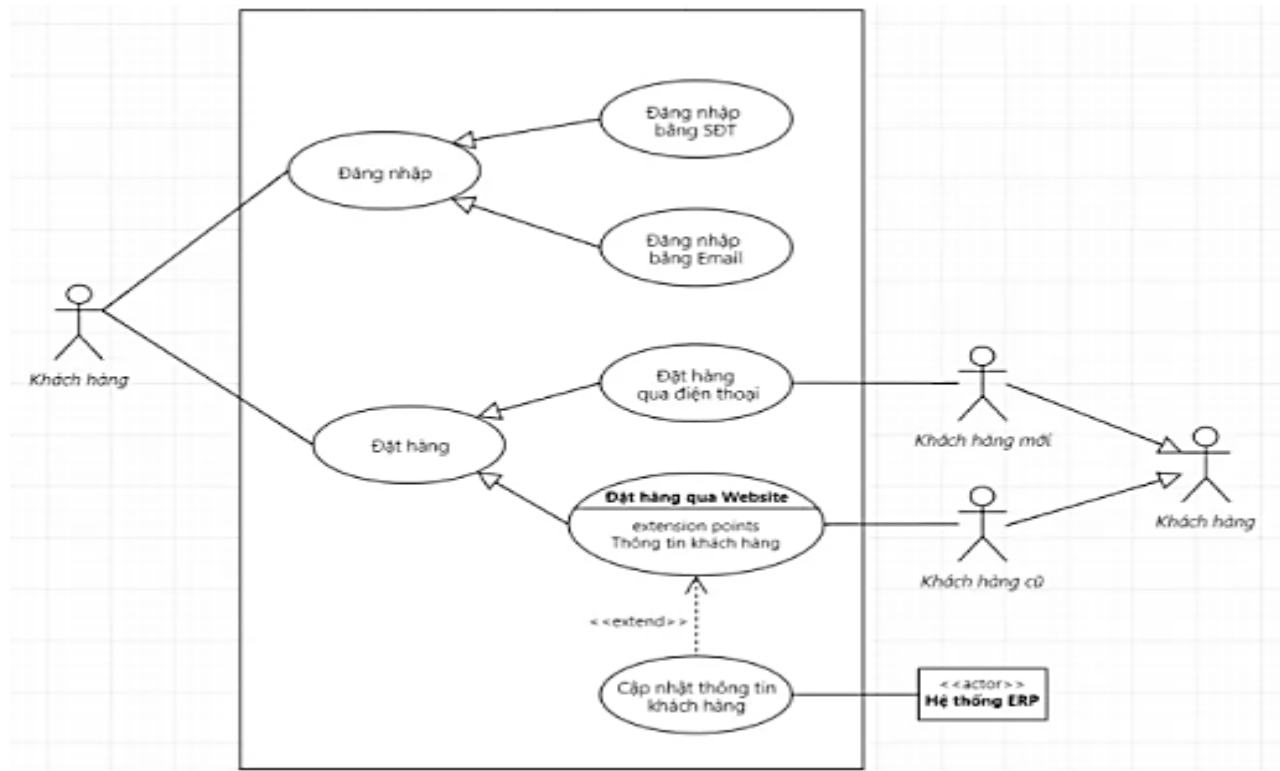


Ví dụ về mối quan hệ Extend giữa các Use Case

# Quan hệ <<extend>>



# Quan hệ <<Generation>>



Ví dụ về quan hệ cha-con (Generalization) trong Use Case.

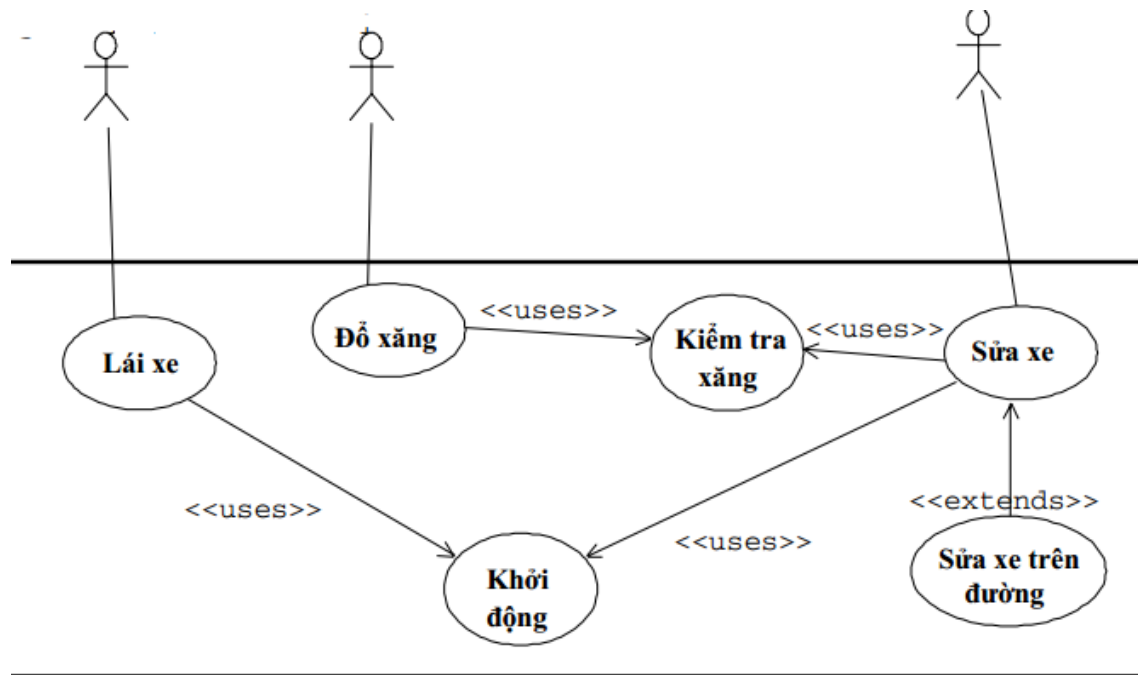
# NHẬN BIẾT CÁC TÁC NHÂN

## Hỏi những câu hỏi sau:

- Ai sẽ là người dùng chính của hệ thống? (tác nhân chính)
  - Ai sẽ cần sự hỗ trợ từ hệ thống để làm các công việc hàng ngày của họ?
  - Ai hoặc điều gì sẽ quan tâm đến những kết quả mà hệ thống đưa ra ?
- Ai sẽ bảo trì, quản lý, điều hành hoạt động của hệ thống ? (tác nhân phụ)
- Hệ thống cần các thiết bị phần cứng nào ?
- Hệ thống cần tương tác với những hệ thống nào khác ?

# VÍ DỤ

**Xác định các tác nhân trong sơ đồ use case sử dụng một chiếc xe**



# NHẬN BIẾT CÁC TÁC NHÂN

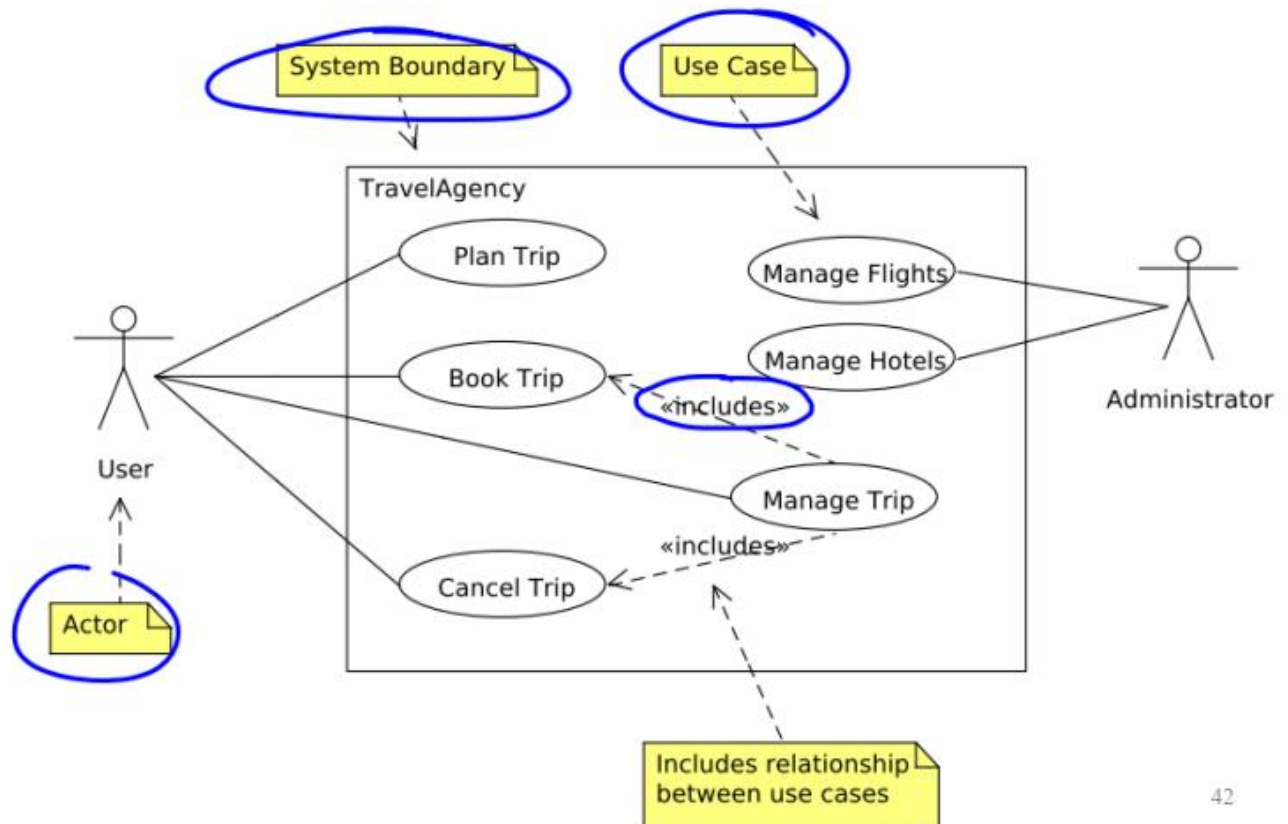
## **Tìm kiếm:**

- Những người trực tiếp sử dụng hệ thống
- Và những người dùng khác – những người cần các dịch vụ từ hệ thống

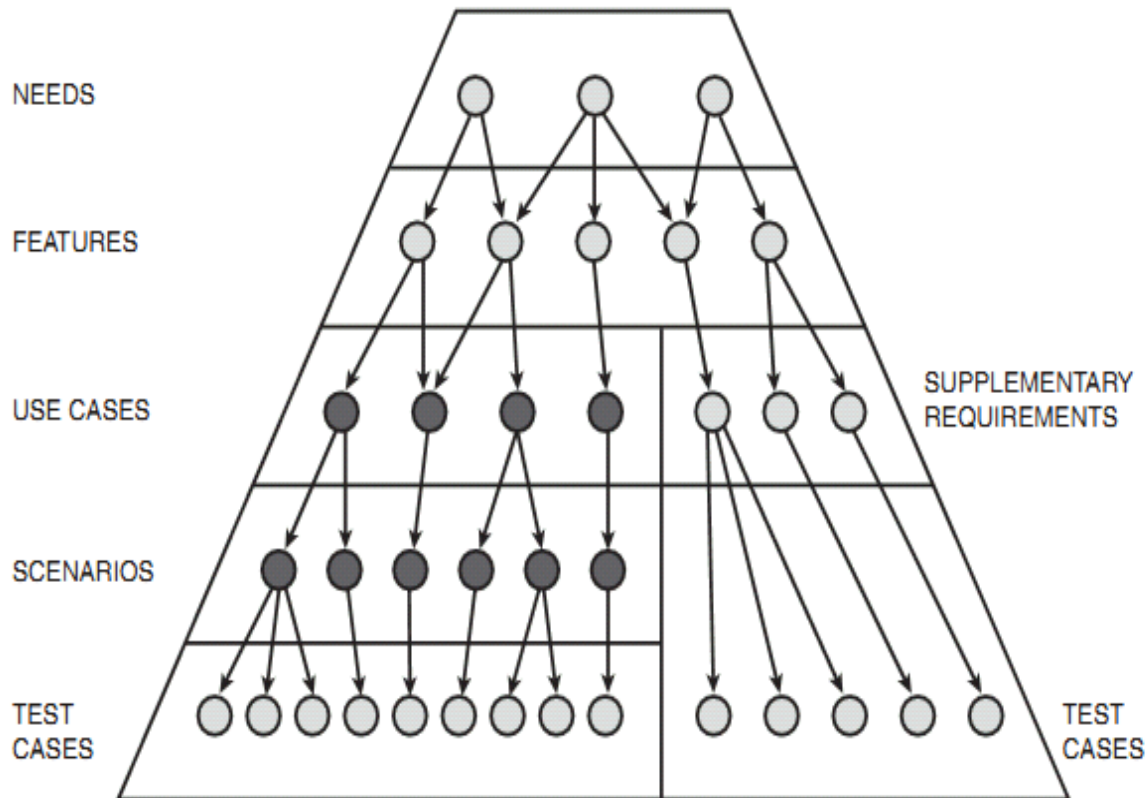
## **Có 3 loại Actor chính:**

- Người dùng. Ví dụ: sinh viên, nhân viên, khách hàng...
- Hệ thống khác.
- Sự kiện thời gian. Ví dụ: Kết thúc tháng, đến hạn...

# Ví dụ:



# MÔ HÌNH HOÁ CÁC YÊU CẦU





## VÍ DỤ

---

**Xác định các tác nhân trong sơ đồ  
use case sắp xếp lịch họp**

# MỘT SỐ CHÚ Ý - Đặt tên

- ✓ Tên Actor: là 1 danh từ, không dùng động từ, và không dùng mệnh đề quan hệ
- ✓ Tên Use Case: phải ghi rõ ràng, rành mạch, nên bao gồm Động từ + danh từ

✓ Ví dụ:

Tác nhân là Khách hàng, Nhân viên,...

UC là: Chuyển tiền nội địa, Thanh toán đơn hàng,...

# MỘT SỐ CHÚ Ý – Phân rã UC

- ✓ Cần phân rã các UC đủ chi tiết để biết các tác nhân có thể thực hiện các tương tác gì với hệ thống
- ✓ Đảm bảo sơ đồ UC đủ chi tiết nhưng cũng phải gọn gàng và thẩm mỹ

# XÁC ĐỊNH USE CASES

- ✓ Chức năng gì mà mỗi tác nhân sẽ mong đợi từ hệ thống?
- ✓ Các tác nhân có cần được thông báo về các sự kiện đang diễn ra trong hệ thống không?
- ✓ Các tác nhân cần cung cấp các thông tin gì cho hệ thống?

# XÁC ĐỊNH USE CASES

- ✓ Các tác nhân cần nhận các thông tin gì từ hệ thống?
- ✓ Hệ thống cần được thông báo về các sự kiện gì bên ngoài hệ thống?

Các UC có thể được xác định trong khi hội thảo các yêu cầu

# LẬP TÀI LIỆU USE CASE

## Cho mỗi use case:

- Chuẩn bị tài liệu “luồng sự kiện” (“flow of events”), được viết từ hướng nhìn của một tác nhân.
- Mô tả chi tiết những cái mà hệ thống cần phải làm chứ không chỉ ra hệ thống sẽ làm nó như thế nào?

## Các nội dung đặc trưng :

- Use case bắt đầu và kết thúc như thế nào;
- Tiến trình bình thường của các sự kiện;
- Tiến trình thay phiên của các sự kiện;
- Tiến trình ngoại lệ của các sự kiện;

## Kiểu tài liệu:

- Chọn lựa cách hiển thị use case:
  - Biểu đồ hoạt động (Activity Diagrams) – tốt cho tiến trình công việc
  - Biểu đồ hợp tác (Collaboration Diagrams) – tốt cho thiết kế cấp cao
  - Biểu đồ trình tự (Sequence Diagrams) – tốt cho thiết kế chi tiết

# LUỒNG SỰ KIỆN CHO USE CASE

**Có thể dùng theo mẫu đơn giản như sau:**

- X. Luồng sự kiện cho Use case ABC
- X1. Điều kiện bắt đầu: danh sách những điều kiện phải thỏa mãn trước khi Use case được thực hiện.

*Ví dụ như: một Use case khác phải thực hiện trước khi Use case này được thực hiện hay người dùng phải có đủ quyền để thực hiện Use case này. Không nhất thiết mọi Use case đều phải có điều kiện bắt đầu.*

- X2. Luồng chính: mô tả những bước chính sẽ xảy ra khi thực hiện Use case.
- X3. Các luồng phụ (luồng con).
- X4. Các luồng rẽ nhánh.

Trong đó X là số tự nhiên của Use case trong hệ thống

# Ví dụ: Luồng sự kiện mô tả Use case cho hệ thống rút tiền tự động

## 1.1 Điều kiện bắt đầu.

## 1.2 Luồng chính:

1.2.1 Người dùng đưa thẻ vào máy.

1.2.2. Máy hiển thông báo chào mừng và yêu cầu nhập mã số

1.2.3 Người dùng nhập mã số

1.2.4 Máy xác nhận mã số đúng. Nếu nhập sai mã số, luồng rẽ nhánh E-1 được thực hiện.

1.2.5 Máy hiện ra ba lựa chọn:

Rút tiền: luồng con A-1

Chuyển tiền: luồng con A-2

Thêm tiền vào tài khoản: luồng con A-3

1.2.6 Người dùng chọn rút tiền

## 1.3. Luồng con:

1.3.1 Luồng con A-1:

1.3.1.1 Máy hỏi số lượng tiền cần rút

1.3.1.2 Người dùng nhập số tiền cần rút : Máy kiểm tra trong tài khoản có đủ tiền không. Nếu không đủ luồng rẽ nhánh E-2 được thực hiện

....

## 1.4. Luồng rẽ nhánh:

1.4.1 E-1: Người dùng nhập sai mã số :Máy thông báo là người dùng đã nhập sai mã số yêu cầu người dùng nhập lại hoặc hủy bỏ giao dịch.

1.4.2 E-2: Không đủ tiền trong tài khoản...



# VÍ DỤ

Xác định luồng sự kiện mô tả Use case sắp xếp lịch họp

# Luồng sự kiện mô tả Use case- sắp xếp lịch họp

- **Tên UC:** Sắp xếp lịch họp
- **Tác nhân:** Người xếp lịch, Người tham gia họp
- **Điều kiện bắt đầu:** Có yêu cầu cuộc họp, người xếp lịch đăng nhập hệ thống.
- **Luồng sự kiện chính:**
  1. Người xếp lịch đăng ký lịch họp.
  2. Hệ thống thông báo tới những người tham gia.
  3. Người tham gia xác nhận lịch họp.
  4. Các phản hồi được gửi tới người xếp lịch.
  5. Nếu đủ thành phần tham gia, người xếp lịch chốt lịch họp. Hệ thống cập nhật lịch họp.
  6. Lịch họp được gửi tới tất cả những người tham gia.
  7. Kết thúc UC (lịch họp đã được sắp xếp)

Luồng sự kiện mô tả Use case- sắp xếp lịch họp

Xác định luồng con và luồng rẽ  
nhánh của UC sắp xếp lịch họp

# MÔ HÌNH HÓA TRÌNH TỰ CỦA CÁC SỰ KIỆN

## Các đối tượng “làm chủ” thông tin và hành vi

- Chúng có các thuộc tính và phương thức liên quan đến trách nhiệm của chúng.
- Chúng không “biết” thông tin về các đối tượng khác, nhưng có thể gọi đến.
- Để thực hiện tiến trình công việc, các đối tượng phải hợp tác.
  - ...bằng cách gửi thông báo đến một cái khác để gọi những phương thức từ chúng
- Các đối tượng chỉ có thể gửi thông báo đến cái khác nếu chúng “biết” cái khác
  - nếu có một quan hệ kết hợp giữa chúng

# MÔ HÌNH HÓA TRÌNH TỰ CỦA CÁC SỰ KIỆN

## Mô tả một Use Case dùng Biểu đồ trình tự

- Biểu đồ trình tự chỉ ra từng bước những cái bao gồm trong một use case
  - Những đối tượng nào liên quan tới use case
  - Các đối tượng đó tham gia như thế nào trong chức năng
- Có thể cần một vài biểu đồ trình tự để mô tả duy nhất một use case.
  - Mỗi biểu đồ trình tự mô tả một kịch bản có thể cho use case
- Biểu đồ trình tự ...
  - ... sẽ giữ dễ dàng để đọc và hiểu.
  - ... không bao gồm những kiểm soát logic phức tạp

# BÀI TẬP

---

Mỗi nhóm lựa chọn 1 Actor, 1 UC của actor đó và viết luồng sự kiện cho UC đã chọn

## BÀI TẬP THỰC HÀNH

- Xác định các Tác nhân (Actors), các Use case (UCs). Vẽ biểu đồ UC.
- Mô tả các UC chính bằng luồng sự kiện.

# MÔ HÌNH HOÁ ĐỐI TƯỢNG



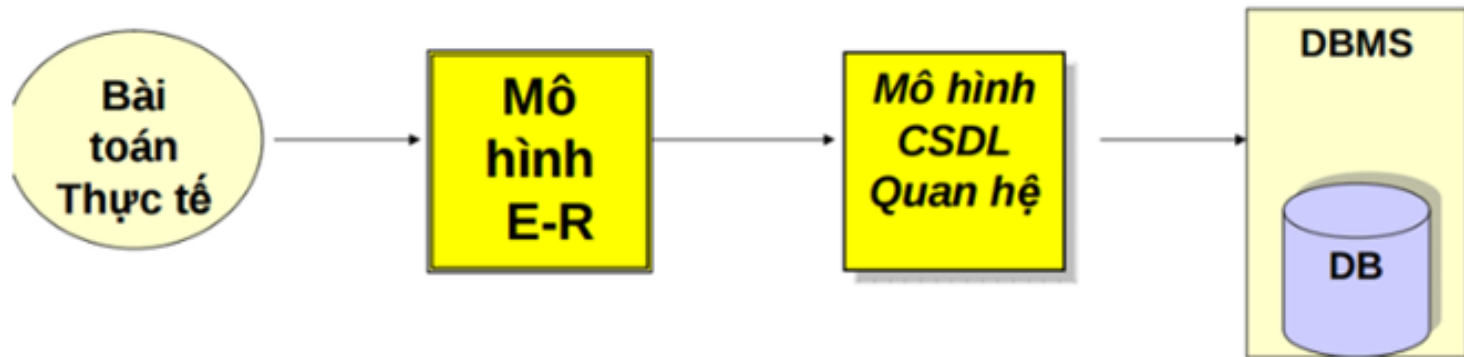
# Nội dung

- Mô hình thực thể quan hệ
- Mô hình hóa đối tượng bằng ULM



# **Mô hình thực thể quan hệ (Entity Relationship Modelling)**

# Mô hình thực thể quan hệ



# Mô hình thực thể quan hệ

Các thành phần:

- Thực thể (Entities)
- Quan hệ (Relationships)
- Thuộc tính (Attributes)

# Thực thể

- Thực thể (Entity): Là khái niệm mô tả một lớp các đối tượng có đặc trưng chung mà chúng ta cần quan tâm.
  - Các thực thể là đối tượng cụ thể hoặc trừu tượng: như Sinh viên, Khách hàng, ...
  - Trong sơ đồ thì thực thể thường được ký hiệu là hình chữ nhật

**Sinhvien**

**khachhang**

# Thực thể yếu

Thực thể yếu: X là thực thể yếu nếu sự tồn tại của X phụ thuộc vào sự tồn tại của thực thể Y.  
Được ký hiệu bằng hình chữ nhật kép

Ví dụ: Người nhà (người phụ thuộc) của nhân viên trong công ty

# Bản ghi

Bản ghi: là một đối tượng cụ thể của lớp các đối tượng đó.

Ví dụ: Sinh viên Nguyễn Văn Anh là đối tượng cụ thể của thực thể Sinh viên

**SINHVIEN**








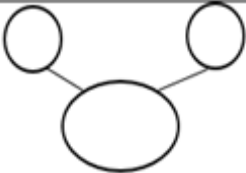
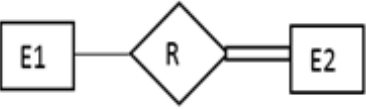


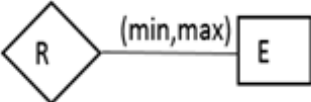
<b>Masv</b>	<b>Tensv</b>	<b>Que</b>
Sv1	Nguyễn Văn Anh	Hà Nội
Sv2	Phạm Ngọc Bình	Hải phòng

# Thuộc tính (Attribute)

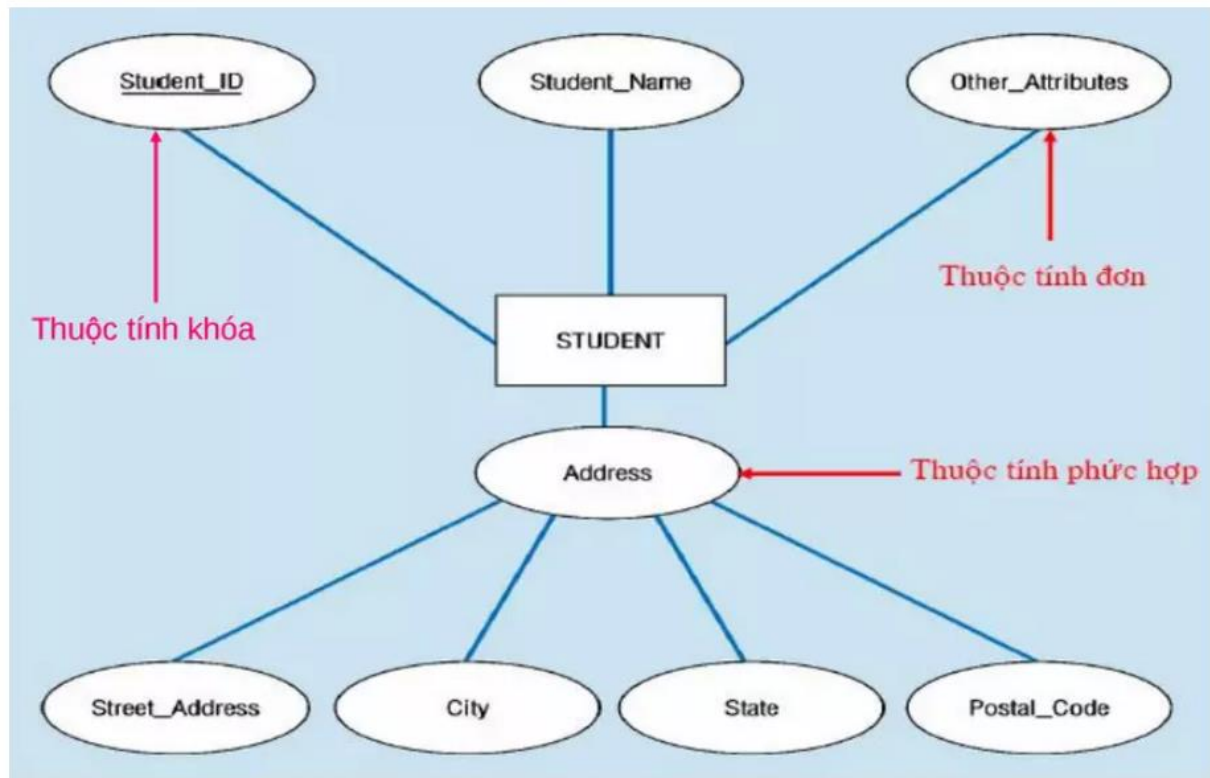
- Là các tính chất, đặc điểm chung của lớp đối tượng. Nó là một giá trị dùng để mô tả một đặc trưng nào đó của một thực thể.
- Thuộc tính có thể là: đơn (singled, đa trị/lặp (multiple-valued), suy diễn (derived attribute), thuộc tính hợp thành...



# Ký hiệu trong mô hình ER

Symbol	Meaning	Symbol	Meaning
	Thực thể/ENTITY		Thuộc tính/ATTRIBUTE
	Thực thể yếu/WEAK ENTITY		Thuộc tính khóa /KEY ATTRIBUTE
	Mối liên kết/RELATIONSHIP		Thuộc tính đa trị /MULTIVALUED
	Mối liên kết xác định/IDENTIFYING RELATIONSHIP		Thuộc tính tổ hợp /COMPOSITE ATTRIBUTE
	E2 tham gia toàn bộ trong R		Thuộc tính suy diễn /DERIVED ATTRIBUTE
	Liên kết 1:N		Lực lượng của E trong R

# Ví dụ về thực thể và thuộc tính



# Các quan hệ (Relationships)

- Các kết nối logic giữa hai hoặc nhiều thực thể.

Ví dụ: **Cư trú** là một quan hệ có thể tồn tại giữa **Thành phố** và **Nhân viên**

- Một thể hiện của một quan hệ là một thể hiện n-tuple của thực thể.

Ví dụ: **bộ 2 thành phần (Nam, Cần Thơ)**, là một thể hiện trong quan hệ **Cư trú**

# Xây dựng mô hình ER

Bước 1: Liệt kê, chính xác hóa và lựa chọn thông tin cơ sở

- Xác định một từ điển bao gồm tất cả các thuộc tính (không bỏ sót bất cứ thông tin nào).
- Chính xác hóa các thuộc tính đó. Thêm các từ cần thiết để thuộc tính đó mang đầy đủ ý nghĩa, không gây nhầm lẫn, hiểu nhầm.
- Chú ý: Để lựa chọn các đặc trưng cần thiết, duyệt từ trên xuống và chỉ giữ lại những thuộc tính đảm bảo yêu cầu sau:
  - Thuộc tính cần phải đặc trưng cho một lớp các đối tượng được xét.
  - Chọn một thuộc tính một lần, nếu lặp lại thì bỏ qua.
  - Một thuộc tính phải là sơ cấp (nếu giá trị của nó có thể suy ra từ các thuộc tính khác thì bỏ qua).

# Xây dựng mô hình ER...

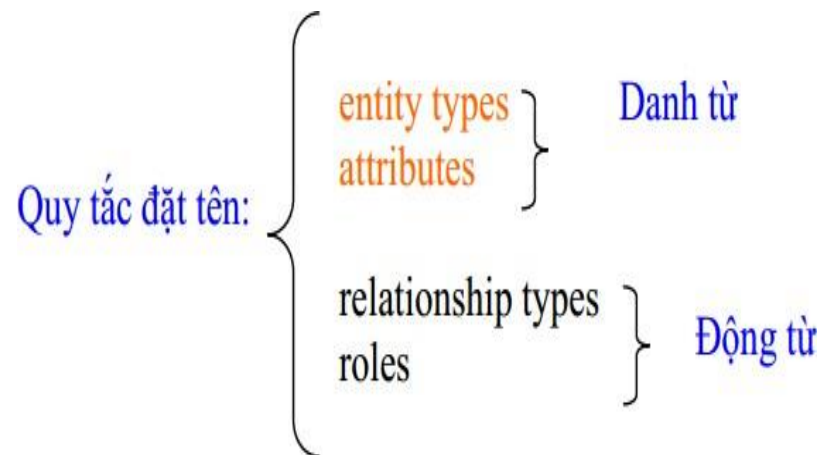
Bước 2: Xác định các thực thể, thuộc tính.

- Duyệt danh sách các thuộc tính từ trên xuống để tìm ra thuộc tính tên gọi. Mỗi thuộc tính tên gọi sẽ tương ứng với một thực thể.
- Gán các thuộc tính cho từng thực thể.
- Xác định thuộc tính định danh cho từng thực thể

# Xây dựng mô hình ER...

Bước 3: Xác định các mối quan hệ và các thuộc tính riêng

- Xét các thuộc tính còn lại, tìm tất cả các động từ (ứng với thuộc tính đó).
- Với mỗi động từ, trả lời các câu hỏi: Ai? Cái gì? Ở đâu? Khi nào? Bằng cách nào?



# Xây dựng mô hình ER...

Bước 4: Vẽ sơ đồ mô hình thực thể - liên kết, xác định lực lượng tham gia liên kết cho các thực thể.

Bước 5: Chuẩn hóa sơ đồ và thu gọn sơ đồ

- Chuẩn hóa: Loại bỏ các thuộc tính lặp, nhóm lặp và các thuộc tính phụ thuộc thời gian -> sơ đồ chỉ còn các thực thể đơn, thuộc tính đơn.
- Thu gọn sơ đồ: Nếu một thực thể có tất cả các đặc trưng:
  - Là thực thể treo: chỉ tham gia vào một mối quan hệ và chứa 1 thực thể duy nhất (có thể có thuộc tính thứ 2 thêm vào làm định danh).
  - Mối quan hệ là bậc hai và không có thuộc tính riêng.
  - Mối quan hệ là 1: N hay 1:1

# Ví dụ

Quy tắc nghiệp vụ của hệ thống CSDL COMPANY như sau:

1. Công ty có nhiều phòng/ban (DEPARTMENTS). Mỗi phòng/ban có tên (name), mã số (number) duy nhất và có một nhân viên (employee) làm quản lý (manages) phòng/ban. Chúng ta lưu lại ngày bắt đầu (start date) làm quản lý phòng/ban của nhân viên đó.
2. Mỗi phòng/ban có thể có nhiều địa điểm khác nhau (locations).
3. Mỗi phòng/ban điều hành một số dự án (PROJECTs). Mỗi dự án có tên (name), mã số (number) duy nhất và chỉ có một địa điểm (location).



## Ví dụ...

4. Với mỗi nhân viên (EMPLOYEE), chúng ta lưu lại những thông tin sau: tên (name), số bảo hiểm xã hội (social security number), địa chỉ (address), lương (salary), giới tính (sex), ngày sinh (birth date).
5. Mỗi nhân viên làm việc ở một phòng/ban, nhưng có thể làm việc cho nhiều dự án. Chúng ta lưu lại số giờ làm việc (the number of hours per week) của từng nhân viên trong từng dự án.
6. Chúng ta lưu lại thông tin về người quản lý trực tiếp (direct supervisor), của mỗi nhân viên. Người quản lý trực tiếp cũng là một nhân viên.
7. Mỗi nhân viên có những người phụ thuộc vào họ (DEPENDENTs). Mỗi người phụ thuộc ta lưu lại thông tin về tên (name), giới tính (sex), ngày sinh (birth date) và quan hệ (relationship)

# Yêu cầu

Xác định các thực thể và các mối quan hệ giữa các thực thể, xây dựng mô hình thực thể liên kết

# Thực hiện

Bước 1: Liệt kê, chính xác hóa và lựa chọn thông tin

Bước 2: Xác định các thực thể, thuộc tính:

**DEPARTMENT** (Name, Number, Locations, NumberOfEmployees)

**EMPLOYEE** (Ssn, name(Fname, Mname, Lname), Bdate, sex, Address, Salary)

**PROJECT** (Name, Number, Location)

**DEPENDENT** (Name, Sex, BirthDate, Relationship)

# Thực hiện

Bước 3: Xác định các mối quan hệ và các thuộc tính riêng

1. MANAGES: là kiểu liên kết **1:1** giữa EMPLOYEE và DEPARTMENT.

- Lực lượng tham gia liên kết của EMPLOYEE là bộ phận, vì không phải nhân viên nào cũng tham gia quản lý.
- Lực lượng tham gia của DEPARTMENT là toàn bộ, vì tại bất kỳ thời điểm nào một phòng cũng có một nhân viên làm quản lý.
- Thuộc tính StartDate được gắn vào kiểu liên kết để ghi lại thời điểm bắt đầu làm quản lý của nhân viên cho phòng đó.

# Thực hiện

2. WORKS\_FOR: là kiểu liên kết **1:N** giữa DEPARTMENT và EMPLOYEE.

Cả hai kiểu thực thể này đều có lực lượng tham gia toàn bộ vào liên kết.

3. CONTROLS: là kiểu liên kết **1:N** giữa DEPARTMENT và PROJECT.

Lực lượng tham gia của PROJECT là toàn bộ, của DEPARTMENT là bộ phận.

# Thực hiện

4. CONTROLS: là kiểu liên kết **1:N** giữa DEPARTMENT và PROJECT.

Lực lượng tham gia của PROJECT là toàn bộ, của DEPARTMENT là bộ phận.

5. WORKS\_ON: là kiểu liên kết **M:N** giữa EMPLOYEE và PROJECT

Một dự án có nhiều nhân viên làm việc và một nhân viên có thể làm việc cho nhiều dự án.

Thuộc tính Hours là thuộc tính của kiểu liên kết được dùng để ghi lại số giờ mỗi nhân viên làm việc cho một dự án nào đó.

Cả hai kiểu thực thể này có lực lượng tham gia toàn bộ.

# Thực hiện

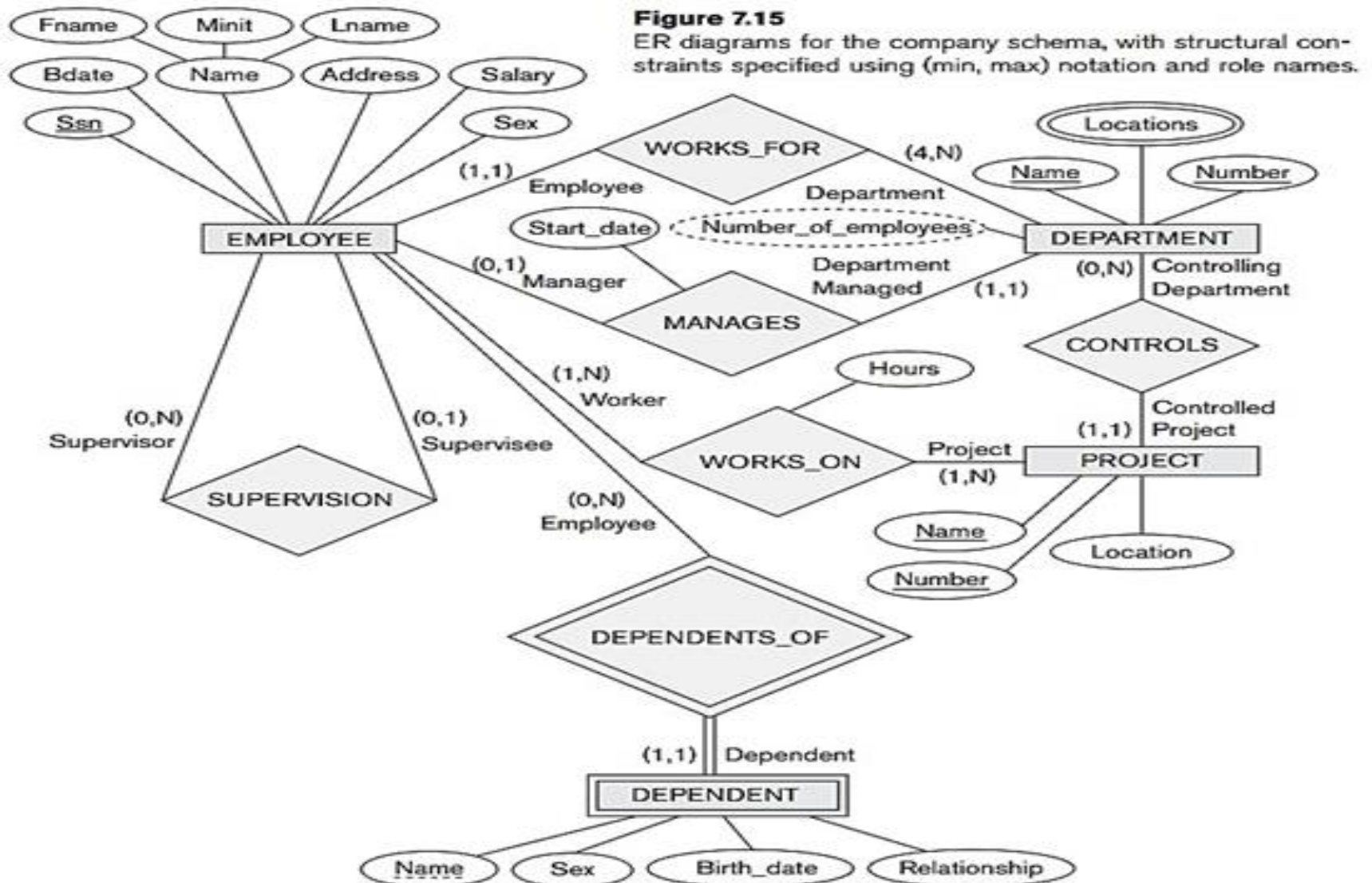
6. `DEPENDENTS_OF`: là kiểu liên kết **1:N** giữa hai kiểu thực thể `EMPLOYEE` và `DEPENDENT`.

Kiểu thực thể `DEPENDENT` là kiểu thực thể yếu, vì nó không tồn tại nếu không có sự tồn tại của `EMPLOYEE`.

Lực lượng tham gia của `EMPLOYEE` là bộ phận, vì không phải nhân viên nào cũng có người phụ thuộc.

Lực lượng tham gia của `DEPENDENT` là toàn bộ vì nó là kiểu thực thể yếu.

# Bước 4: Mô hình ER







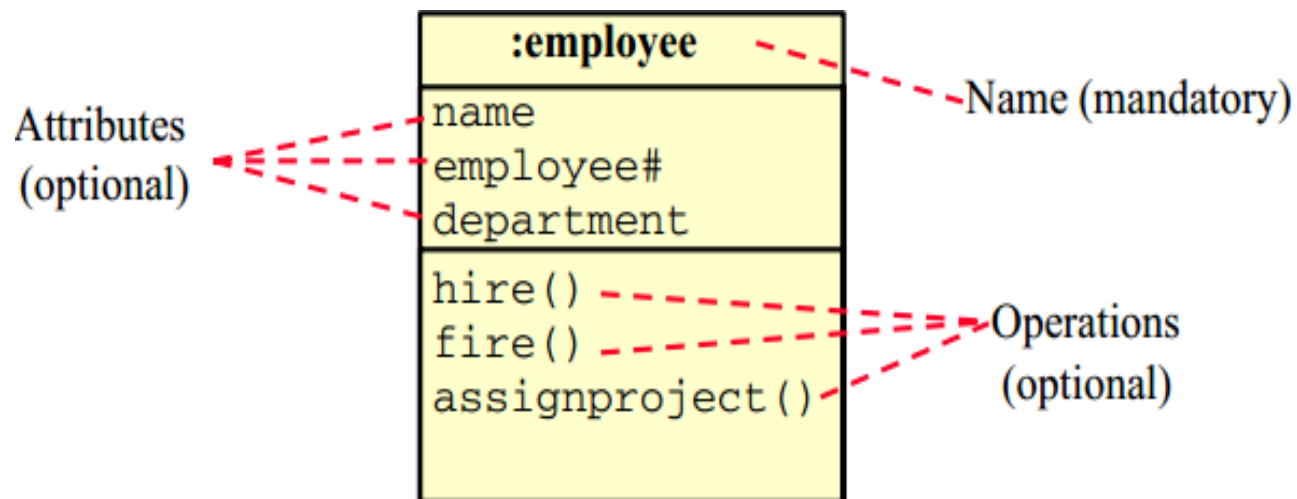
# MÔ HÌNH HOÁ ĐỐI TƯỢNG BẰNG UML (Unified Modeling Language)

# Lớp (class) là gì ?

- Một lớp mô tả một nhóm các đối tượng (objects) với:
  - Các đặc tính tương tự (thuộc tính - attributes),
  - Cùng hành vi ứng xử (phương thức - operations),
  - Quan hệ như nhau đối với các objects khác.
  - Và có chung ngữ nghĩa (“semantics”).

# Ví dụ về lớp

Nhân viên (employee): có 1 tên (name), mã số nhân viên (employee#) và bộ phận trực thuộc (department); một nhân viên thì có thể được thuê (hired), bị sa thải (fired); mỗi nhân viên làm việc trong một hay nhiều dự án (assignproject)



# Tìm lớp từ các yêu cầu

- **Tìm *lớp* từ dữ liệu nguồn:**
  - Tìm các danh từ và cụm danh từ trong mô tả vấn đề của các đối tác chứa trong mô hình nếu họ giải thích một cách tự nhiên hoặc cấu trúc thông tin trong ứng dụng.
- **Tìm *lớp* từ các nguồn khác:**
  - Xem xét các thông tin nền tảng;
  - Những người dùng và các đối tác khác;
  - Các mẫu phân tích;
- **Sẽ rất tốt nếu ban đầu có nhiều ứng viên cho *lớp***
  - Bạn có thể bỏ chúng ngay sau đó nếu chúng hóa ra không hữu ích
  - Quyết định dứt khoát loại bỏ các *lớp* thì tốt hơn là chỉ suy nghĩ về điều đó.

# Đối tượng và Lớp

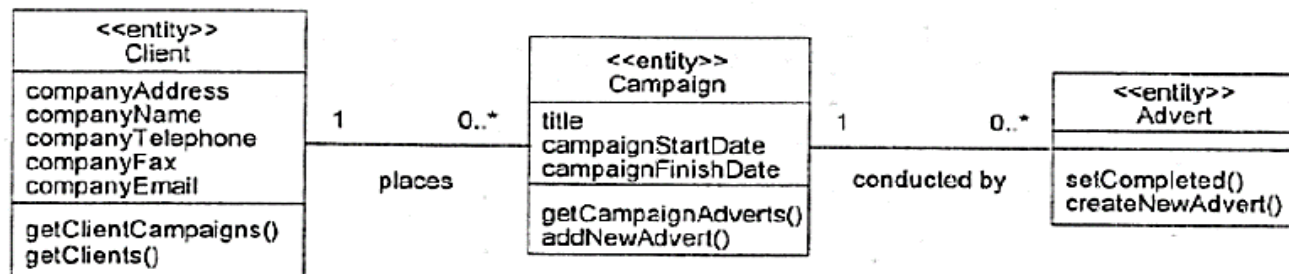
- Các thể hiện của một lớp được gọi là đối tượng.
  - Một đối tượng được trình bày như sau:

Nam : <b>Employee</b>
name: Nam
Employee #: 234609234
Department: Marketing

- Hai đối tượng khác nhau có thể có các giá trị thuộc tính giống nhau (như hai người với tên và địa chỉ giống nhau)
- Đối tượng có quan hệ kết hợp (associations) với đối tượng khác
  - VD: Nam :employee thì có quan hệ kết hợp với đối tượng Mekong1000 :project

# Các kiểu quan hệ giữa các lớp

- **Đối tượng không tồn tại độc lập với những cái khác**
  - Một quan hệ (relationship) diễn tả một sự kết hợp trong số những cái khác.
  - Trong UML, có nhiều kiểu quan hệ khác nhau:
    - Quan hệ kết hợp (Association)
    - Quan hệ tập hợp (Aggregation) và Quan hệ hợp thành (Composition)
    - Quan hệ thừa kế (Generalization)
    - Quan hệ phụ thuộc (Dependency)
    - Quan hệ hiện thực hóa (Realization)
  - Chú ý : Hai quan hệ cuối không hữu ích trong quá trình phân tích yêu cầu
- **Sơ đồ lớp chỉ rõ các lớp và mối quan hệ giữa chúng**



# Quan hệ kết hợp

Đối tượng có mối quan hệ kết hợp (associations) với đối tượng khác

Ví dụ: Nam:employee có quan hệ kết hợp với Mekong1:project

# Bản số (multiplicity) của quan hệ kết hợp

**Bội số quan hệ là số lượng thể hiện của một lớp liên quan tới MỘT thể hiện của lớp khác.**

***Ví dụ:***

Hỏi các câu hỏi về quan hệ kết hợp:

- *Một cuộc vận động (campaign) có thể tồn tại mà không có thành viên trong Ban quản lý hay không ?*
  - ✓ Nếu có, thì quan hệ kết hợp này sẽ là một tùy chọn trong Ban quản lý là 0 hoặc nhiều (0..\*)
  - ✓ Nếu không, thì nó là tùy chọn là 1 hoặc nhiều (1..\*)
  - ✓ Nếu nó cần được quản lý bởi 1 và chỉ 1 thành viên trong Ban thì là chính xác 1 (1)
- *Một câu hỏi khác của quan hệ kết hợp?*
  - ✓ Mỗi thành viên trong Ban quản lý có cần phải quản lý chính xác chỉ một cuộc vận động không?
  - ✓ Không. Vì thế bản số chính xác là 0 hoặc nhiều (0..\*)

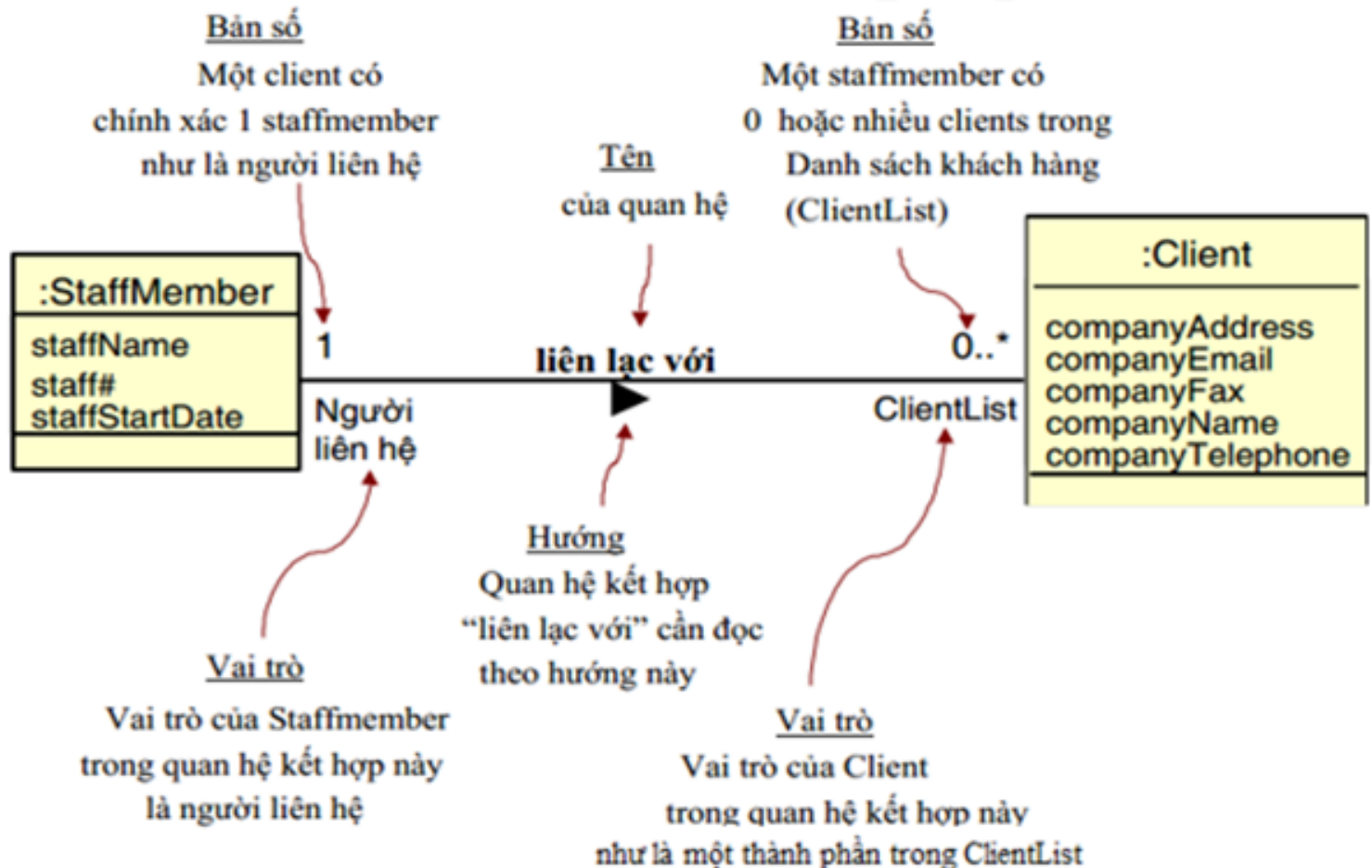


# Bản số (multiplicity) của quan hệ kết hợp

## *Một số ví dụ biểu diễn của bản số:*

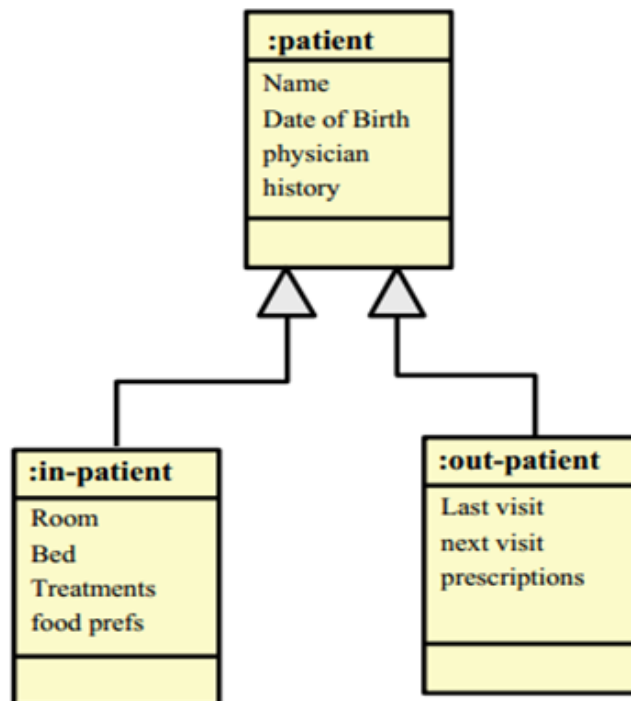
- Tùy chọn (0 hoặc 1)      0..1
- Chính xác 1      1 = 1..1
- 0 hoặc nhiều      0..\* = \*
- 1 hoặc nhiều      1..\*
- Một vùng giá trị      2..6

# Ví dụ về bản số của quan hệ kết hợp



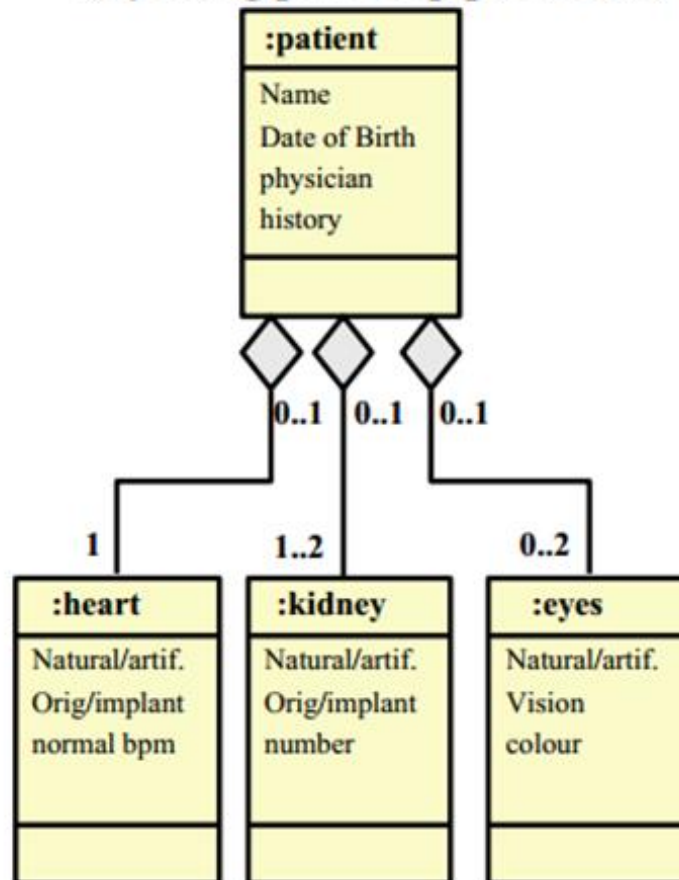
# Ví dụ

## Quan hệ thừa kế (hệ thống phân cấp trừu tượng)



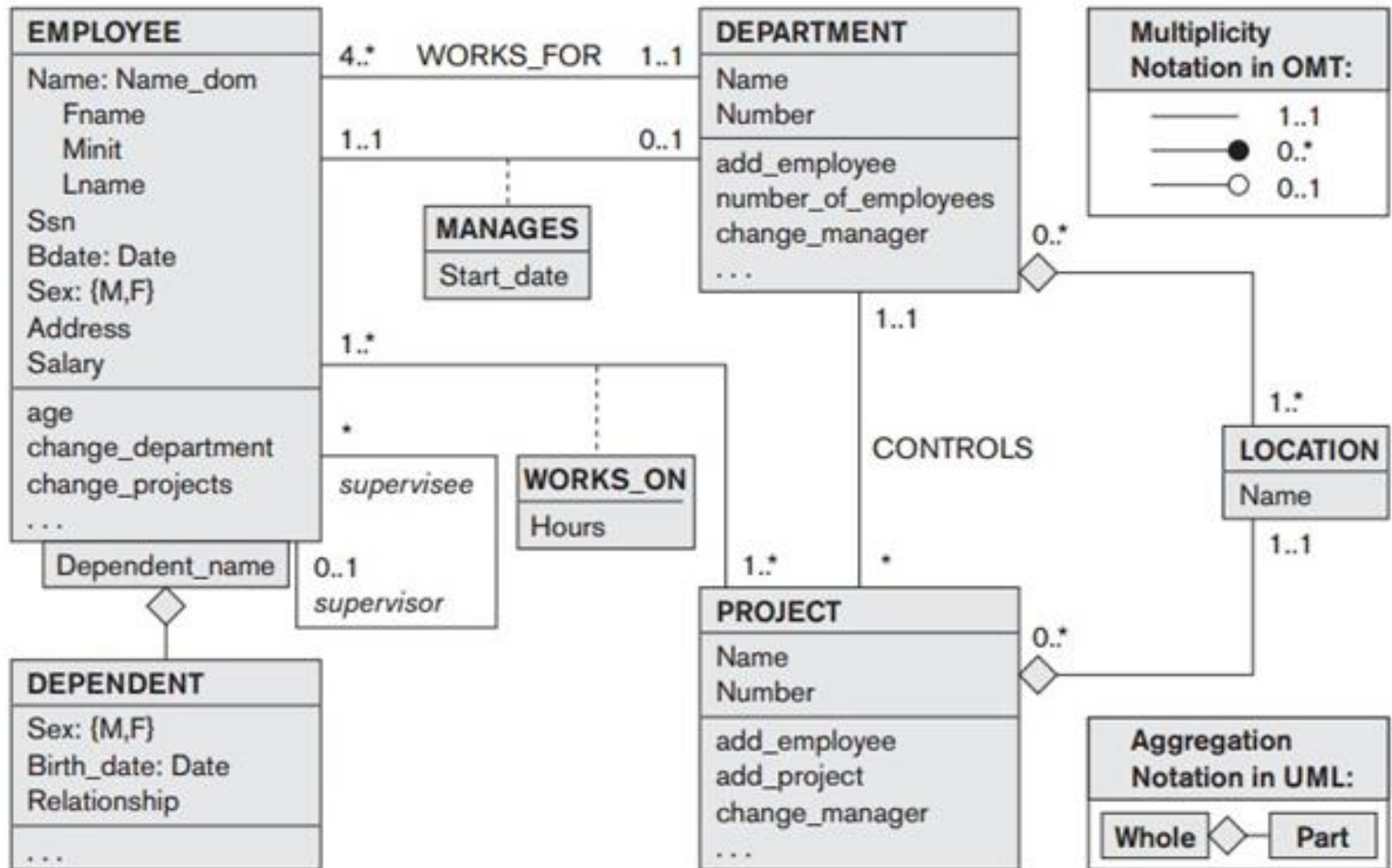
# Ví dụ

## Quan hệ tập hợp (hệ thống phân cấp phân chia)



**Figure 7.16**

The COMPANY conceptual schema  
in UML class diagram notation.



# CHÚ Ý

- **Hiểu rõ các đối tượng trong lĩnh vực ứng dụng**
  - Định nghĩa tất cả các đối tượng mà đối tác nêu ra
  - Quyết định đối tượng nào quan trọng cho thiết kế của bạn
  - Sơ đồ lớp thiết kế tốt khi:
    - ✓ Có quan hệ trực quan giữa các đối tượng trong lĩnh vực
    - ✓ Khảo sát các quy tắc nghiệp vụ và giả thiết thông qua bản số
    - ✓ Đặc tả cấu trúc của thông tin để (cuối cùng) lưu trữ

# Chú ý

- Hướng đối tượng (Object Oriented – OO) là một cách tốt để khảo sát các chi tiết của vấn đề
  - Tránh những chấp vá tự nhiên của cấu trúc
  - Cung cấp một phương thức chặt chẽ để hiểu rõ thực tế

# Chú ý

- Tuy nhiên:
  - OO thiên về thiết kế hơn là phân tích: Trong RE, sơ đồ quan hệ không phản ánh các lớp chương trình
  - Đối với bước phân tích, lược đồ UML được dùng để phác họa chứ không phải bản thiết kế.
  - Nhưng cũng có thể trở thành bản thiết kế khi dùng trong đặc tả



# Chú ý: UML và ERD

- **Sơ đồ ER tương tự như sơ đồ lớp trong UML**

- Sơ đồ lớp nhấn mạnh thứ bậc lớp (class hierarchies) và các phương thức (operations)
- Sơ đồ ER nhấn mạnh các mối quan hệ (relationships) và khóa xác định (identity)

**Nhưng chỉ cần một cho việc phân tích mọi vấn đề cho trước !**

- **ER cung cấp nhiều ký hiệu hơn cho khái niệm cơ sở dữ liệu:**

- Sơ đồ ER cho phép các quan hệ đa chiều (N-ary relationships) (Sơ đồ lớp UML chỉ cho phép quan hệ hai chiều ( binary relationships))
- Sơ đồ ER cho phép các thuộc tính đa giá trị.
- Sơ đồ ER cho phép đặc tả các khóa xác định.

# Chú ý: UML và ERD

- **Sự lựa chọn tùy thuộc vào mục tiêu cài đặt:**
  - Sơ đồ lớp UML cho kiến trúc hướng đối tượng (Object Oriented Architecture)
  - Sơ đồ ER cho CSDL quan hệ (Relational Databases)
  - Nhưng điều này chỉ quan trọng khi bạn dùng chúng cho bản thiết kế. Đối với bản phác thảo, sự tương tự với ký hiệu thì quan trọng hơn

# BÀI TẬP

Bài tập nhóm: Xác định các lớp (các thuộc tính, phương thức) của hệ thống mà các nhóm đăng ký

# Các kiến thức cần nhớ về mô hình hoá yêu cầu

- Tác nhân (Actors),
- Use case (UCs) và Biểu đồ UC.
- Luồng sự kiện.
- Lớp, Biểu đồ lớp (Class),