Vietnam National University of HCMC
International University
School of Computer Science and Engineering

Object – Oriented Analysis and Design

Use-case Modeling

Instructor: Le Thi Ngoc Hanh, Ph.D

ltnhanh@hcmiu.edu.vn

Outline

- Unified Modeling Language (UML)
- Types of UML
- Use Case Diagram
- Reading: [R1]-Chapter 6

/* Credit images: uml-diagram.org */

Unified Modeling Language

What is UNIFIED MODELING LANGUAGE™ UML® ?

The **unified modeling language** (**UML**) is a general-purpose visual modeling language that is intended to provide a standard way to visualize the design of a system.[1]

UML provides a standard notation for many types of diagrams which can be roughly divided into three main groups: behavior diagrams, interaction diagrams, and structure diagrams.

The creation of UML was originally motivated by the desire to standardize the disparate notational systems and approaches to software design. It was developed at Rational Software in 1994–1995, with further development led by them through 1996.[2]

Source: https://en.wikipedia.org/wiki/Unified_Modeling_Language

Why UML?

“a user logs into the system”

```
#include<stdio.h>
int main() {
   int a, b, sum;
  /*ask user to enter number */
  printf("[illegible]");
 scanf("%d %d", &a, &b);
 sum = a + b;
 printf("Sum : %d", sum);
 return(0);
}
```

- Ensure consistent designs across teams.
- Ensure a high-level design is properly executed in code.

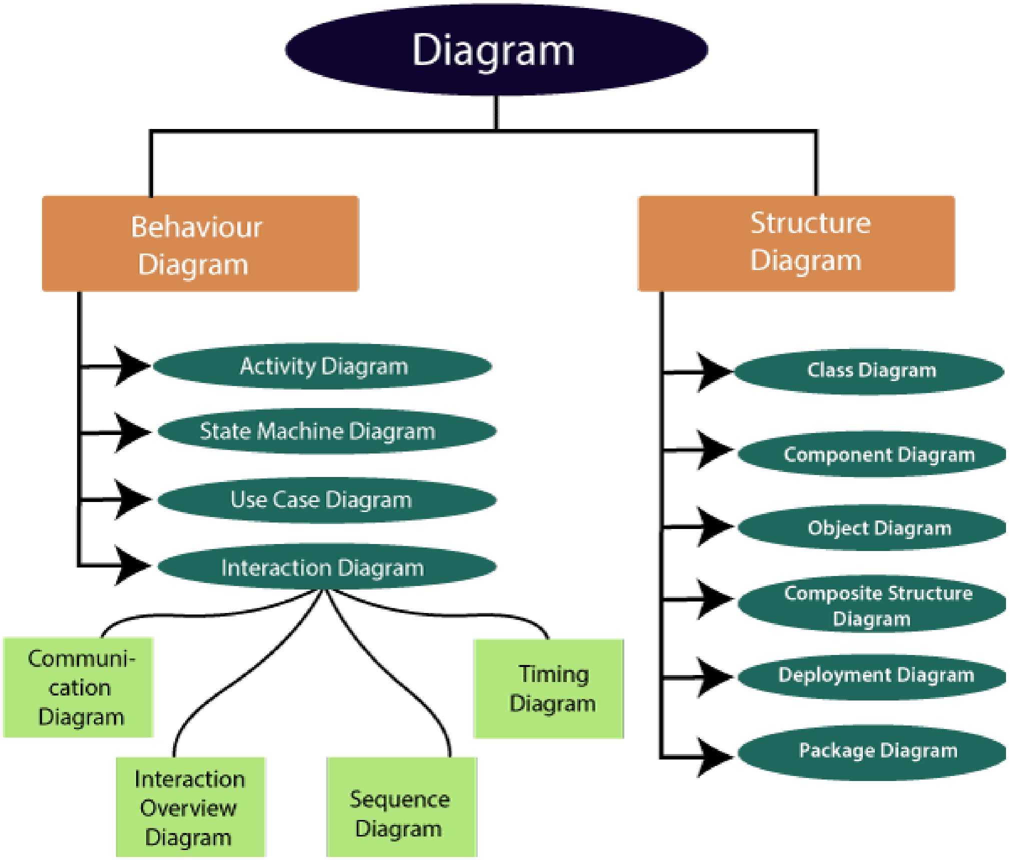
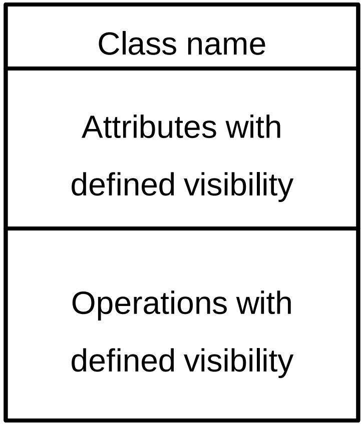


Diagram
Behaviour Diagram
Structure Diagram
Activity Diagram
State Machine Diagram
Use Case Diagram
Interaction Diagram
Communi-cation Diagram
Interaction Overview Diagram
Sequence Diagram
Timing Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram

Notations in UML


Class name
Attributes with defined visibility
Operations with defined visibility

Visibility:

+ Public

# Protected

- Private

Role of Use Case Diagram in UML

Usefulness of Use cases

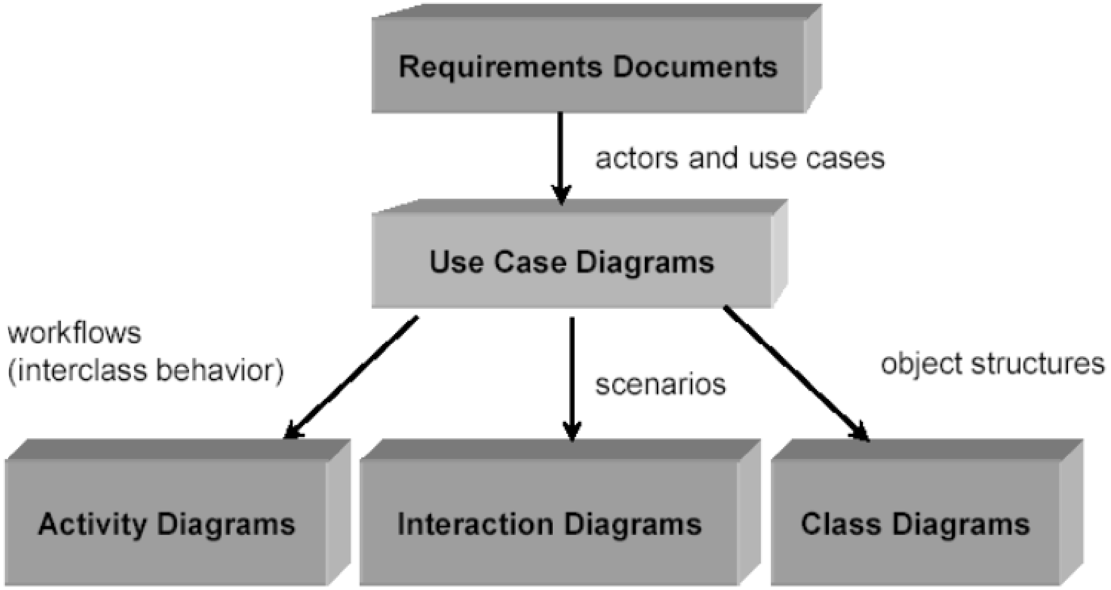
*One of the most important things during **analysis** is to **discover** all the potential **use cases**.*

Unified Modeling Language (UML) delivers the **use cases** to analyze the requirements of a system.

- Use cases are typical **interactions** a **user** has with the **system**.
- A use case indicates a **function** that the user can **understand** and that has **value** for the user.
- Use cases can **vary** considerably in **size**.

Use cases are an essential tool in **requirements capturing** and in **planning** and **controlling** an iterative **project**.

Role of Use Case Diagram in UML



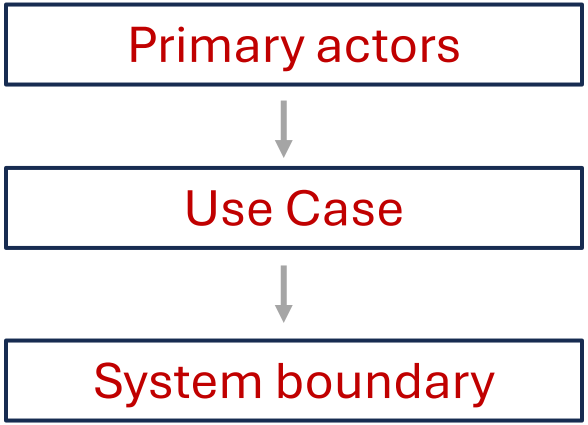
Components in a Use Case Diagram

- Actors:

An actor is external to a system, interacts with the system, may be a human user or another system, has a goal and responsibilities to satisfy in interacting with the system.

- Use Cases:
  - Identify functional requirements
  - Describe actions performed by a system
  - capture interations between the system and actors
- Relationships: Actors are connected to the use cases with which they interact by a line which represents a relationship between the actors and the use cases.

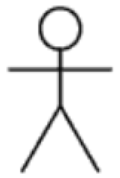
Basic Procedure in Use Case Diagram

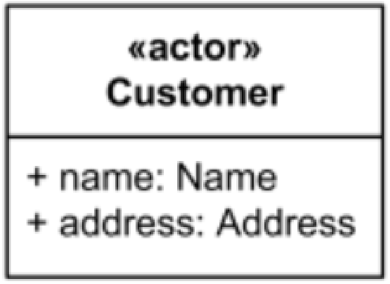


Actors

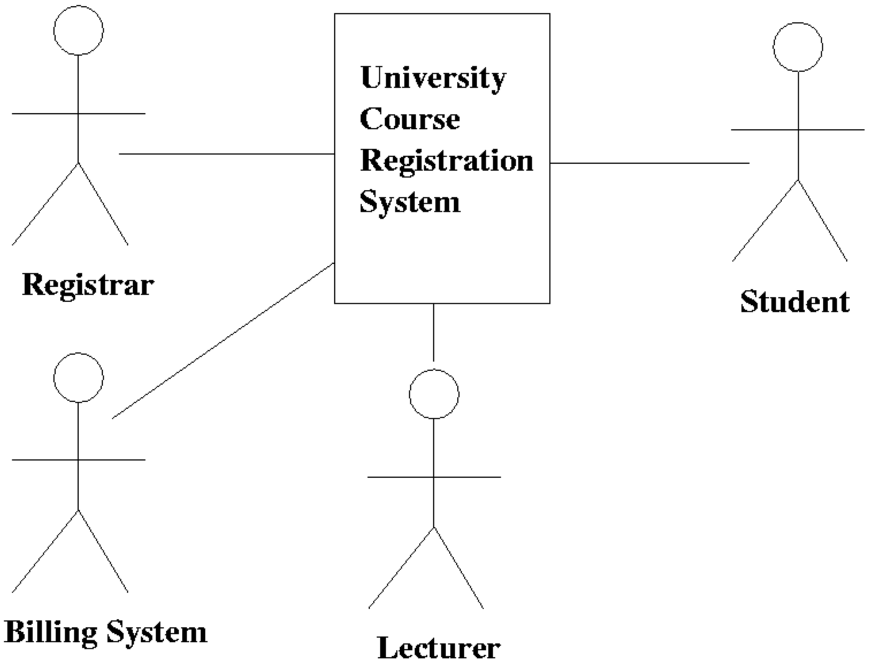
- An actor is a role that a user or other system plays with respect to the system to be developed.
- A single actor in a use case diagram can represent multiple users (or systems).
- A single user (or system) also may play multiple roles.
- Actors don't need to be human, e.g., an external system that needs some information from the current system.

Actors - Notation





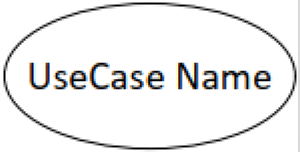
Actors - Example

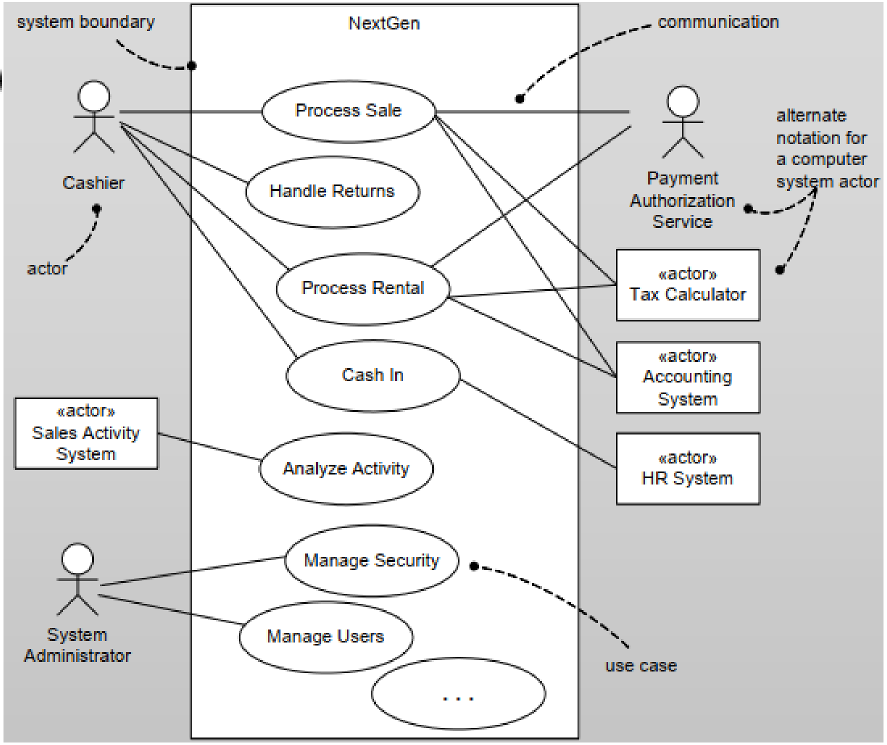


Use Case

- The “case aspect” in a use case represents a high-level description of a desired action in which the actor is involved.
- Each UseCase specifies a unit of useful functionality that the subject provides to its users.

Use Case - Notation

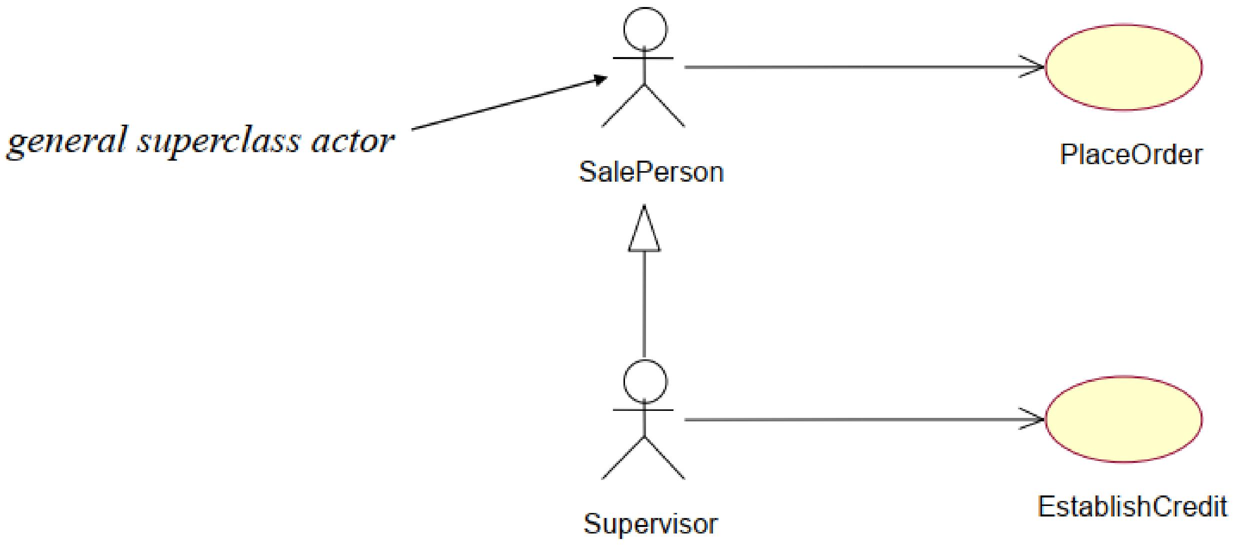




Relationships

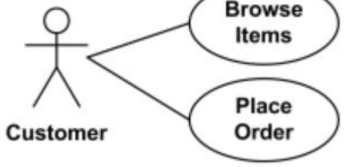
Types	Description	Symbols
association	Relationship between an actor and a use case	———————
extend	Relationship between two use cases	<<extend>> ------------------>
generalization	Relationship between two actors	———————▷
include or use	Relationship between two use cases	<<include>> ------------------>

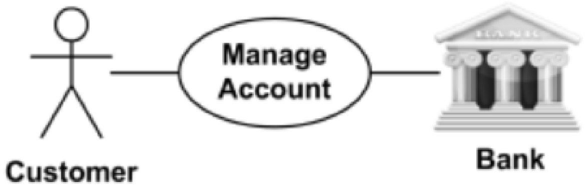
Generalization



Association

- An association between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other.
- Only binary associations are allowed between actors and use cases.



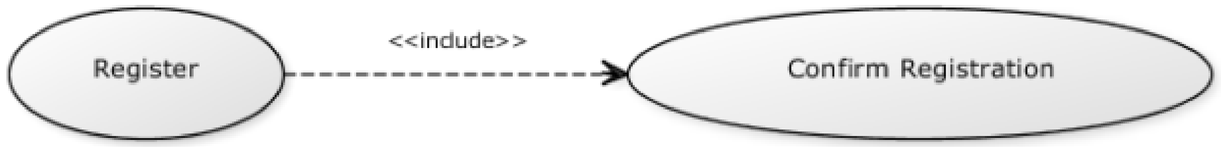


Customer actor is associated with two use cases - Browse Items and Place Order.

Manage Account use case is associated with Customer and Bank actors.

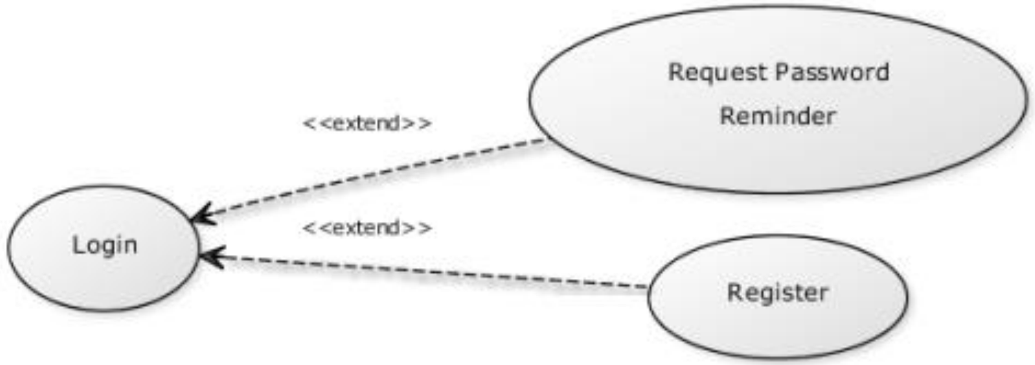
Include

- Use "include" when you are repeating yourself in two or more separate use cases.
- Copying the description of that behavior introduces redundance and may lead to inconsistencies when the behavior is changed.

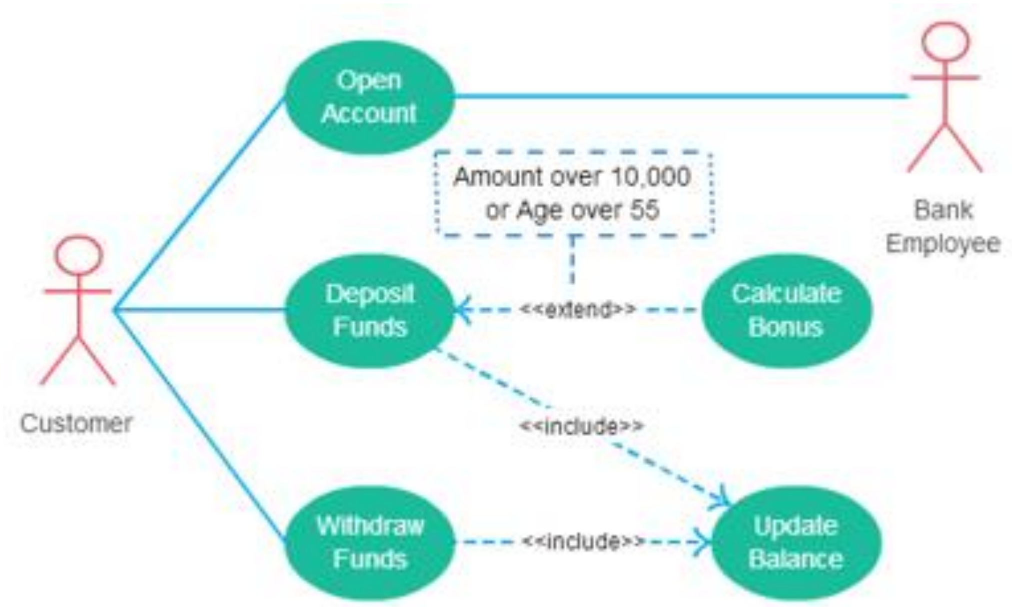


Extend

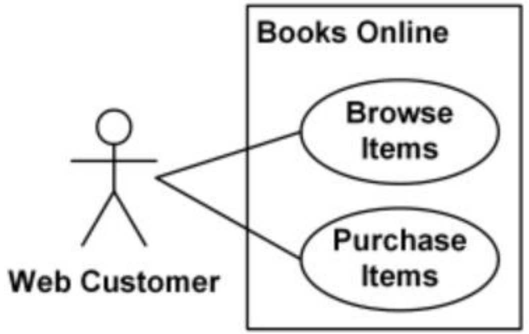
- Extensions are used instead of modeling every extra case by a single use case.
- Extensions are used instead of creating a complex use case that cover all variations.



Extend


Open Account
Amount over 10,000 or Age over 55
Bank Employee
Customer
Deposit Funds
<<extend>>
Calculate Bonus
<<include>>
Withdraw Funds
<<include>>
Update Balance

Subject



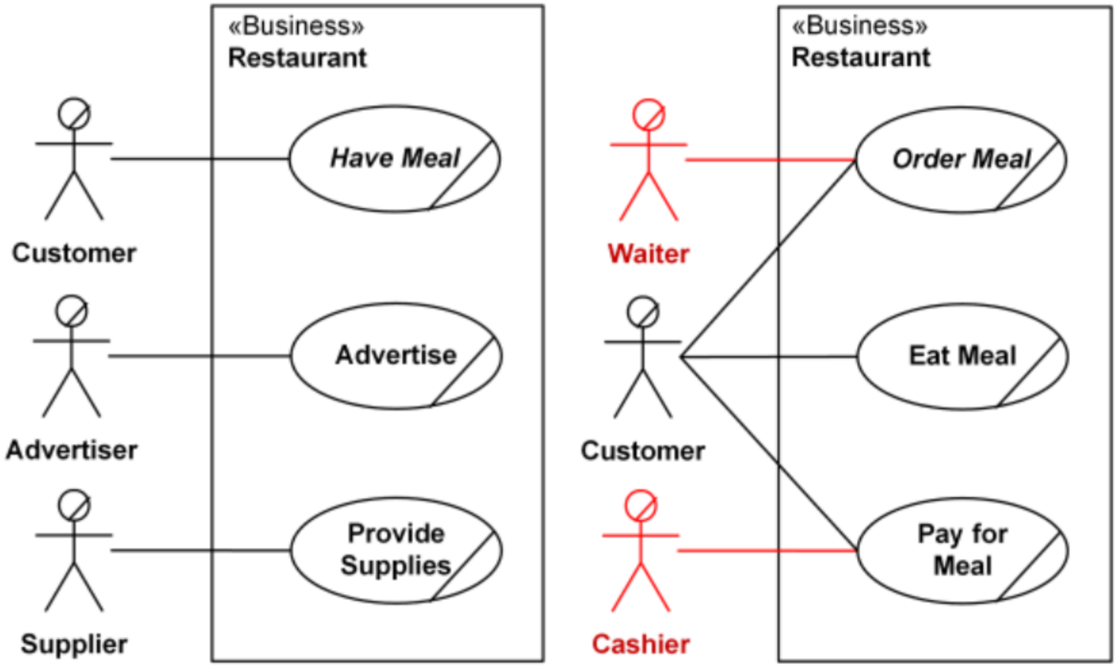
A subject is a classifier (including subsystem, component, or even class) representing a business, software system, physical system or device under analysis, design, or consideration, having some behavior, and to which a set of use cases applies.

Subject

Textual description of the actor will show its boundary!

“We need a system where students can register for courses, view materials, and submit assignments. Professors will upload content and grade students. Admins should manage user accounts and courses.”

Subject


«Business»
Restaurant
Customer
Have Meal
Advertiser
Advertise
Supplier
Provide Supplies
«Business»
Restaurant
Waiter
Order Meal
Customer
Eat Meal
Cashier
Pay for Meal