LECTURE 3 REQUIREMENTS GATHERING

LEK HSIANG HUI

READINGS

Beginning Software Engineering chapter 4

https://onlinelibrary-wiley-com.libproxy1.nus.edu.sg/doi/pdf/10.1002/9781119209515.ch4

LEARNING OBJECTIVES

At the end of this lecture, you should understand:

- What Requirements are and the types of Requirements
- Some of the Requirements Gathering approaches
- How to draw Use Case Diagrams

INTRODUCTION TO REQUIREMENTS

REQUIREMENTS

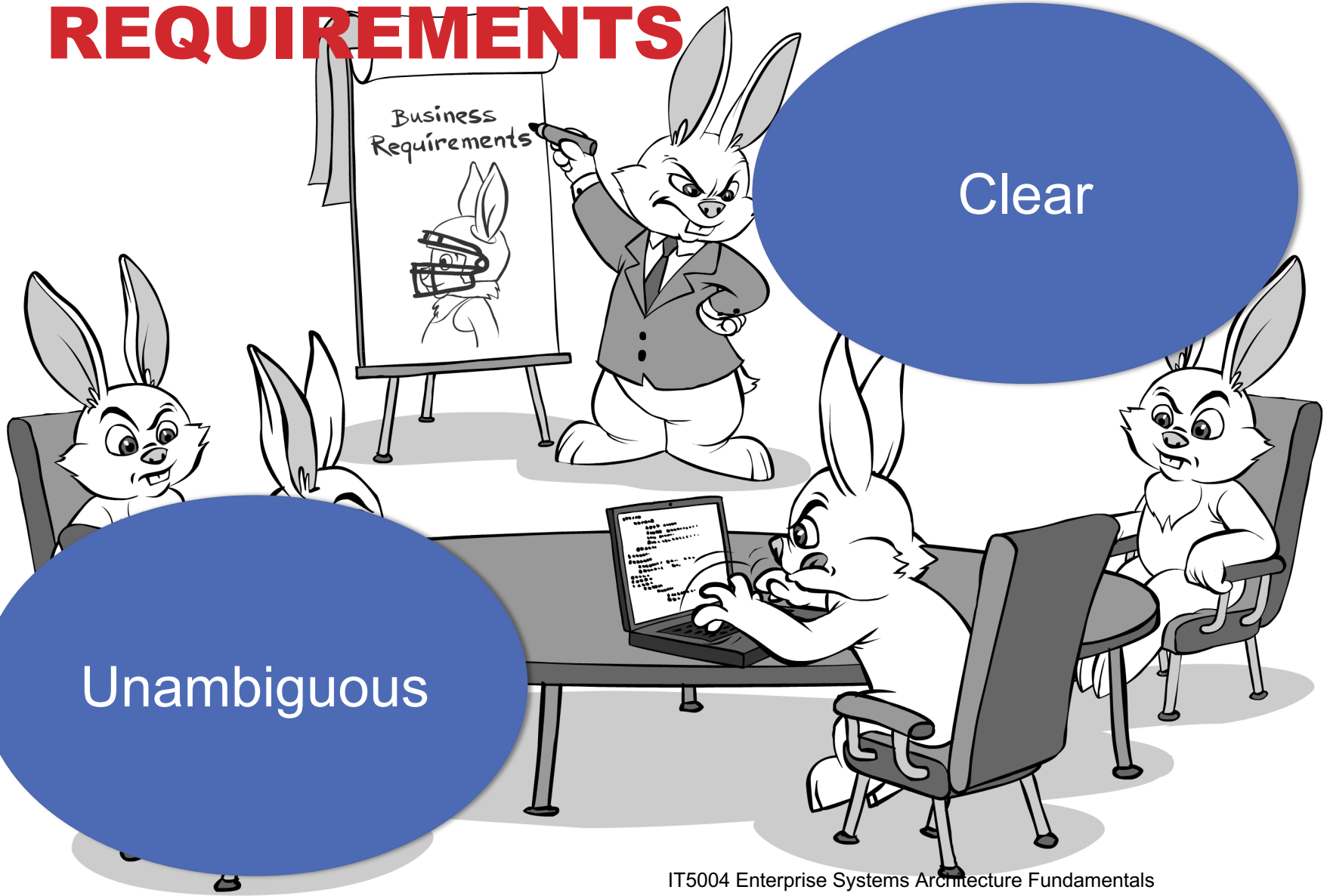
Requirements are features, functionalities, constraints, and qualities that a software must possess

One of the reasons why we use Activity Diagrams is to allow us to discover the requirements

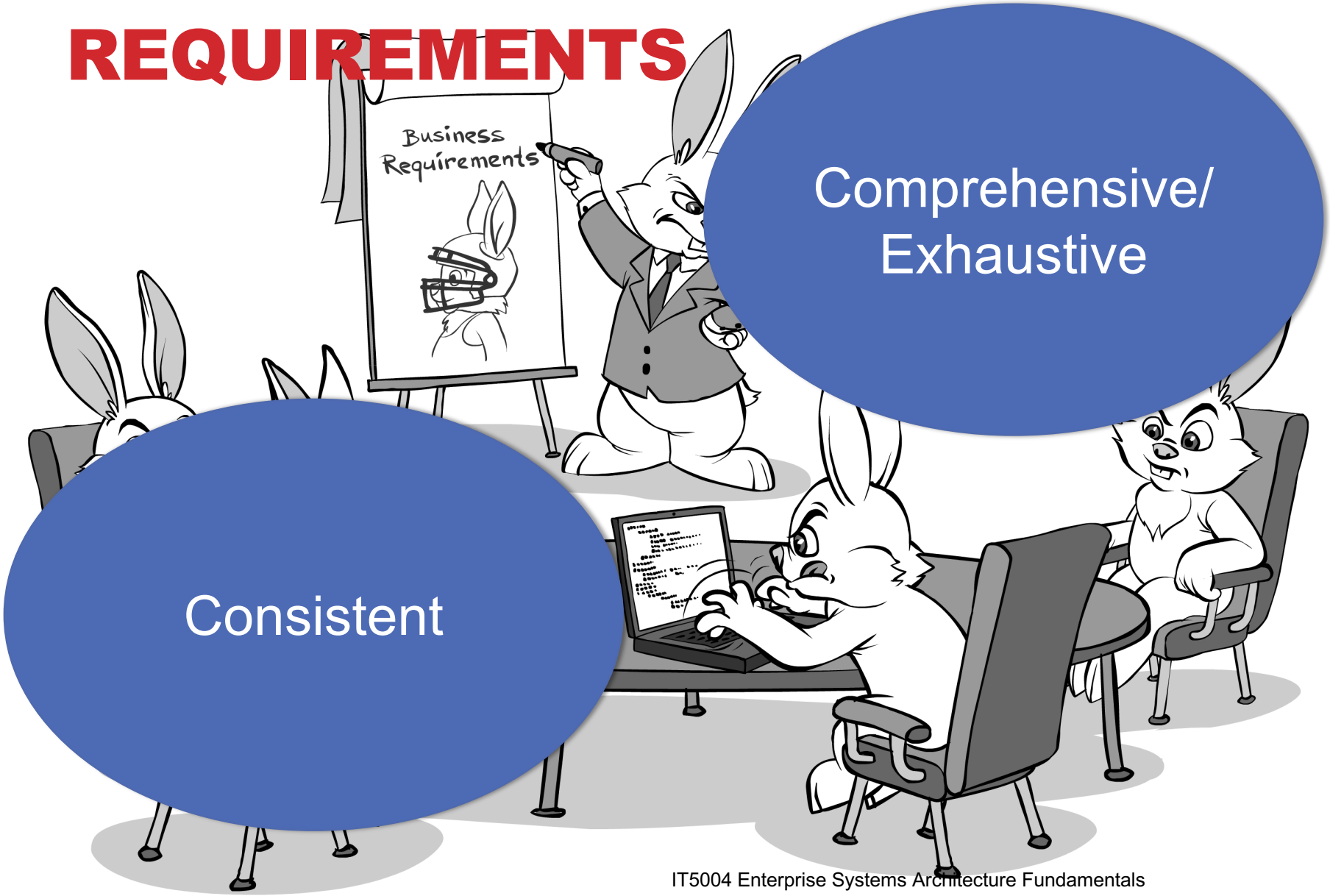
Requirements guides the developments:

- It helps to define the project scope
- Can be used to verify that the finished system addresses the business needs

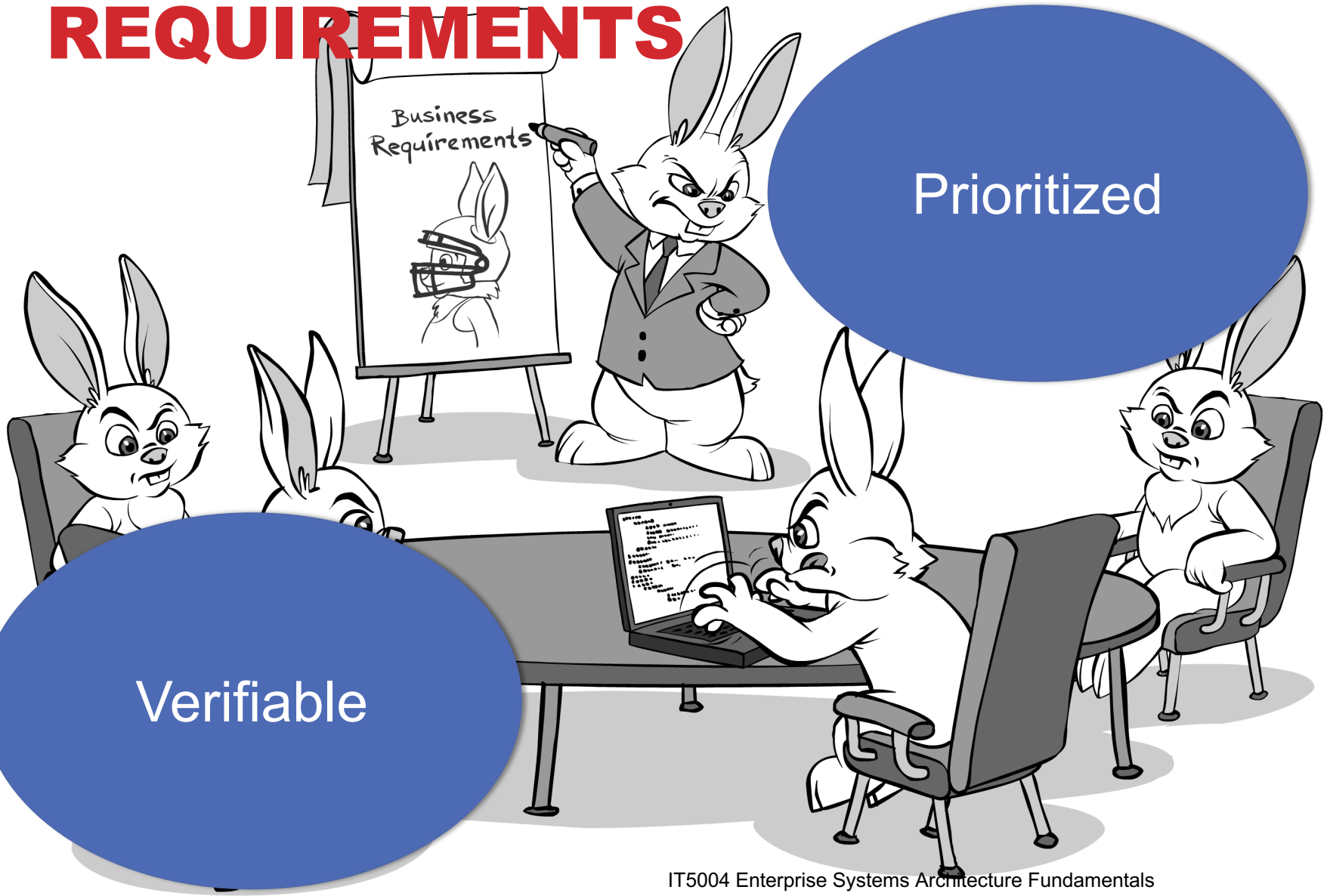
PROPERTIES OF REQUIREMENTS



PROPERTIES OF REQUIREMENTS



PROPERTIES OF REQUIREMENTS



TYPES OF REQUIREMENTS

TYPES OF REQUIREMENTS

Different types of requirements

- Functional Requirements
- Nonfunctional Requirements
- Implementation Requirements

FUNCTIONAL REQUIREMENTS

Describes what the system should do

Features your customers get in restaurant menu ordering application:

- Signup / login
- Profile settings
- My orders
- Can find restaurant based on category
- Can view the list of restaurant
- Search restaurant by category, food name, etc
- Can view full details of restaurant with reviews & menu list
- Can share reviews of restaurant
- Check out process -
- Can add menu to cart
- User selects food delivery now or later
- Updates on delivery & billing details
- Usage of discount voucher
- Select payment option - COD/ credit or debit card

NONFUNCTIONAL REQUIREMENTS

Describes various aspects/quality of the system:

- Performance (speed)
- Security
- Reliability
- Scalability
- Connectivity
- Etc

They describe properties of the system not captured by functional requirements

IMPLEMENTATION REQUIREMENTS

Temporary requirements that are needed in order to transit from using the old system to new system

Does not necessarily involve programming

- E.g. Ensures old database schema is properly migrated to new database schema naming
- Prepare training videos and materials
- Ensure that server is upgraded to 64GB RAM when deploying new system
- Etc

REQUIREMENTS GATHERING APPROACHES

REQUIREMENTS GATHERING APPROACHES

Approaches for gathering requirements include:

- Interviewing end-users
- Understanding business process flow
- Studying existing systems

INTERVIEWING END-USERS

Challenges:

- Need to ask the right questions and learn to listen
- They are not developers and do not think like programmers
- End-users often do not know what they want
- They often do not know what you would need

INTERVIEWING END-USERS

Can use the 5W1H approach

- Who – who is the user?
- What – what should the application/user do?
- When – When is the system used?
- Where – Where is the application used?
- Why – Why is there a need for the system?
- How – How can we solve the problem (user might have some idea)?

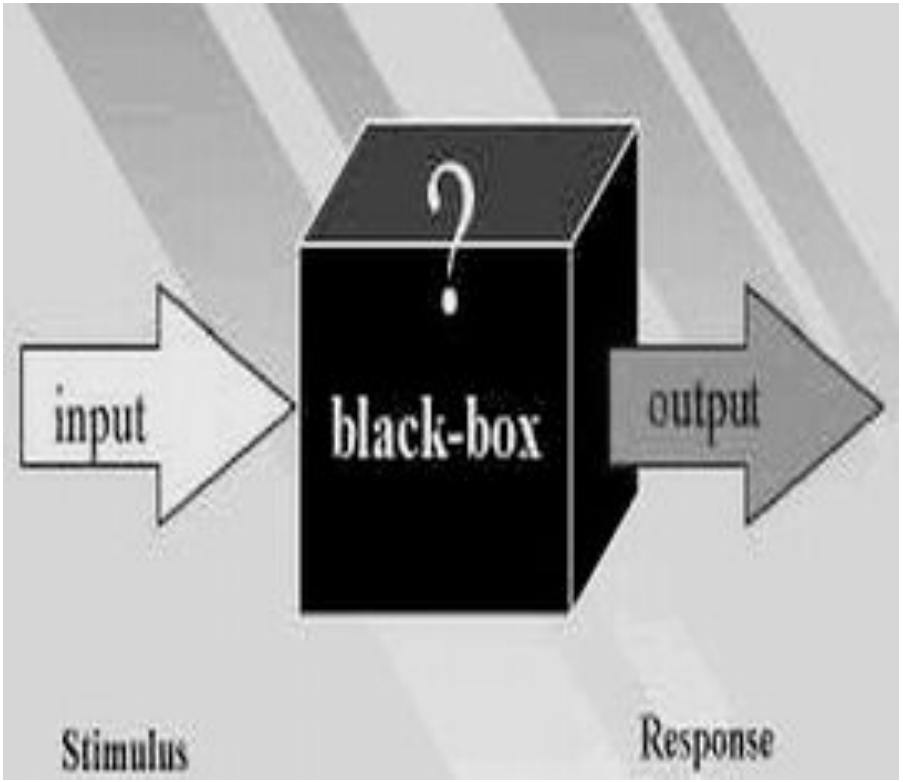
UNDERSTANDING BUSINESS PROCESS FLOW

Often, we can analyze business process flow using events

Event decomposition is a technique that can be used to identify all the events

- Not all events in a business process is relevant
- Only those that should be remembered by the system or those that should trigger processing in the system are important

EVENT DECOMPOSITION



How to identify Events?

- One approach is **Event Decomposition**
- Treat system as a black box

EVENT DECOMPOSITION

List of **use cases** triggered by business events

3 TYPES OF EVENTS

External Events

- Occur outside the system
- **Usually caused by external agent**
  e.g. customer

Temporal Events

- Occurs when system reaches a point (deadline) in **time**

State Events

- Events that occur when something happen inside the system that **trigger the need for processing**

EXTERNAL EVENTS

Examples:

- Customer wants to login
- Customer want to check account details
- Customer want to show the list of transaction
- Customer want to search for items
- Etc

TEMPORAL EVENTS

Examples:

- Time to produce monthly report
- Time to email bill PDF
- Time to send scheduled advertisements
- Etc

STATE EVENTS

Examples (triggered when certain conditions are satisfied):

- Send instant message notification
- Send low stock email notification
- Update delivery status
- Etc

STUDY EXISTING SYSTEMS

Once we have the goals of the system, we can refine the requirements by studying similar systems

- This reduces the chance of missing requirements
- Allows us to rethink about our proposed requirements (might be able to simplify requirements or do differently)

USE CASES

We then document these requirements as use cases (i.e. system functionality)

Example:

- Requirement: **Customer want to show the list of transactions**
- Use Case: **Get list of transactions of customer**

- Requirement: **Customer want to buy item**
- Use Case: **Create new order**

USE CASE DIAGRAMS

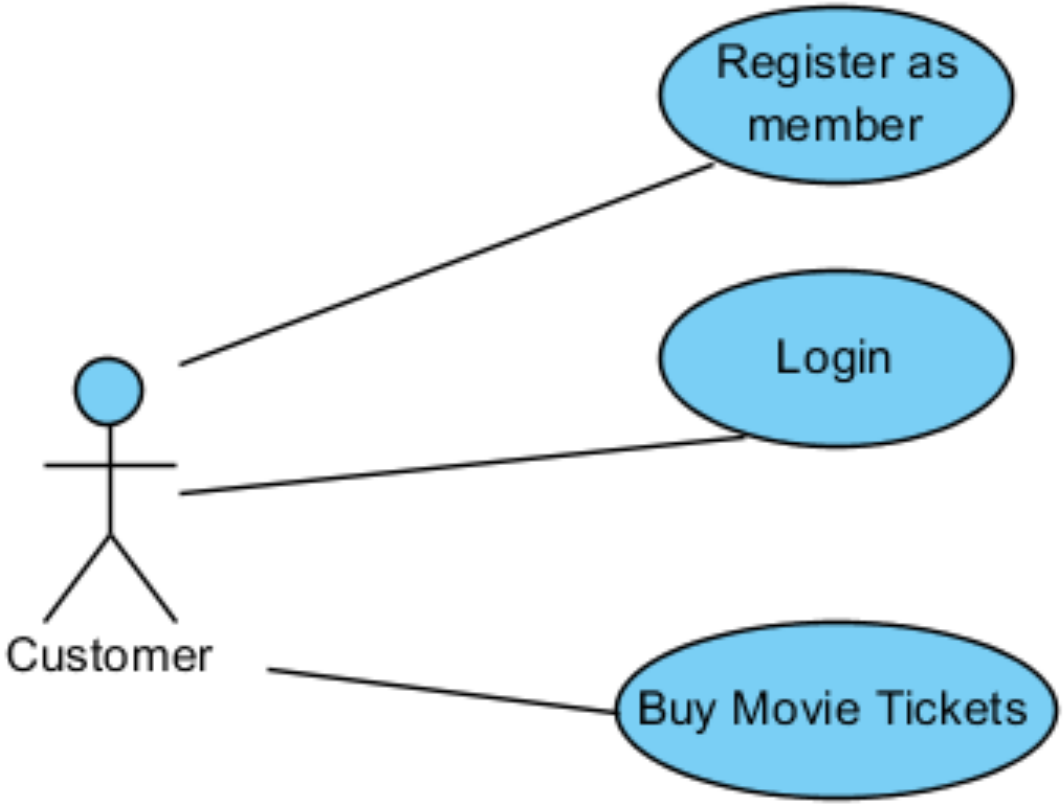
USE CASE DIAGRAMS

Use case diagrams is a modeling technique for documenting the use cases

Diagram consists of 3 main elements

- Actors
- Use cases
- Association between actors to use cases

AN EXAMPLE OF A USE CASE DIAGRAM



IDENTIFY ACTORS

Can be users of the IT system

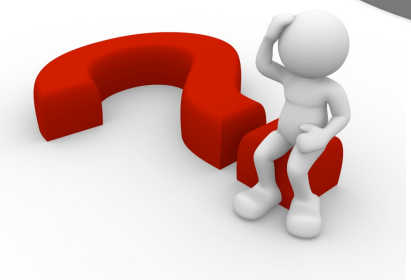
Or other system that interacts with the IT system

QUESTIONS TO ASK

Who are the possible users?

Is there just one system or many subsystems

How does the system interact with other systems?



QUESTIONS TO ASK

What will the system be like?

Should roughly have an idea

QUESTIONS TO ASK

Who are the possible users?

Customers?

QUESTIONS TO ASK

Customers?

How many types of customers are there?

How are the different types of customers using the system?

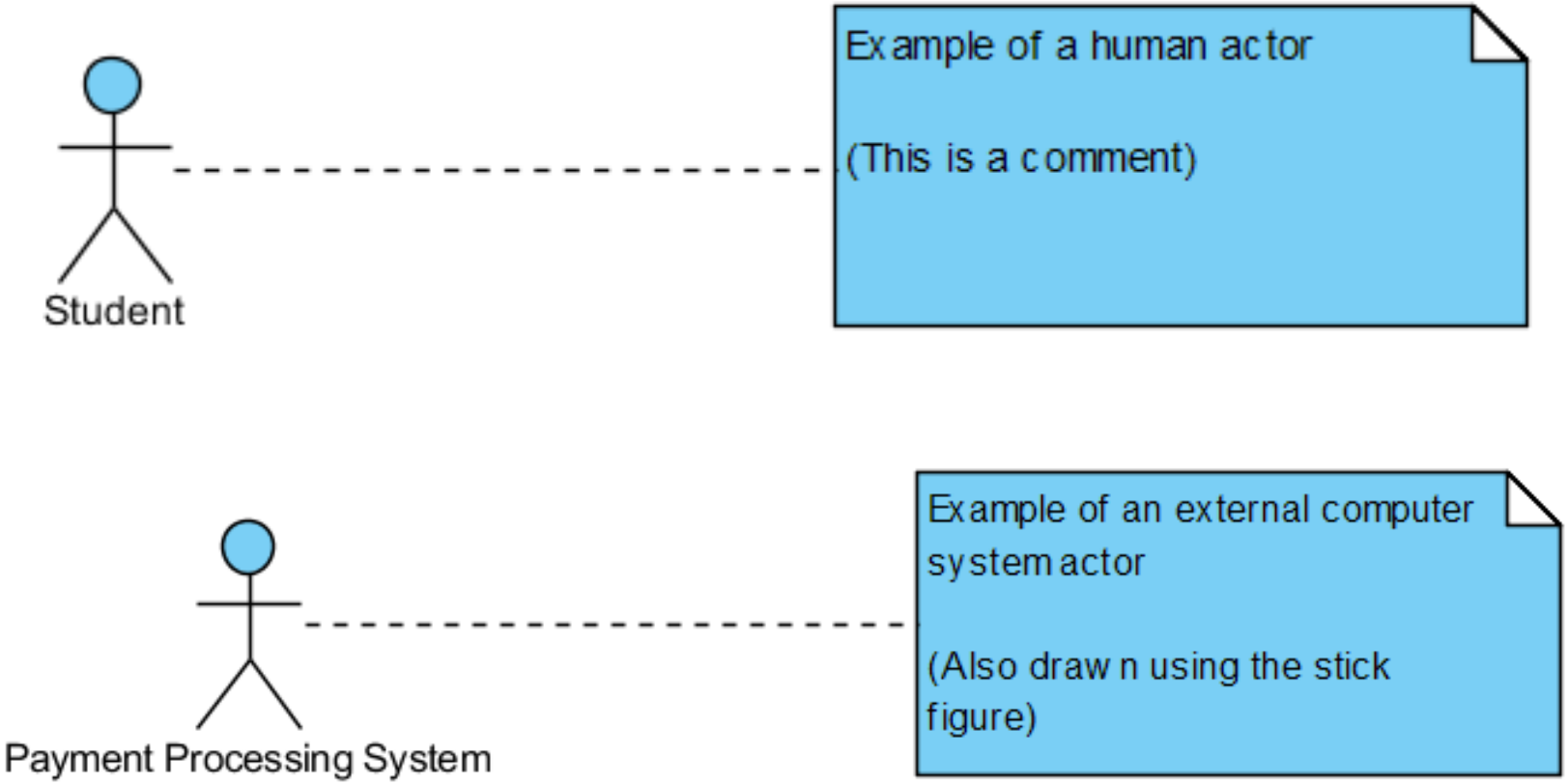
QUESTIONS TO ASK

Employees?

How are the managers using the system?

How is the accountant using the system?

ACTOR


Example of a human actor
(This is a comment)
Student
Example of an external computer system actor
(Also drawn using the stick figure)
Payment Processing System

GENERALIZE ACTOR

After identifying the actors

Generalize them

- i.e. find relations amongst these different types
- e.g. a graduate student is a more specific kind of student

Note that actors can also be systems

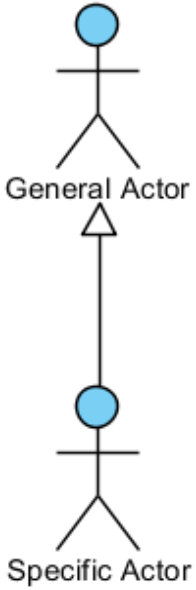
ROLE MAP

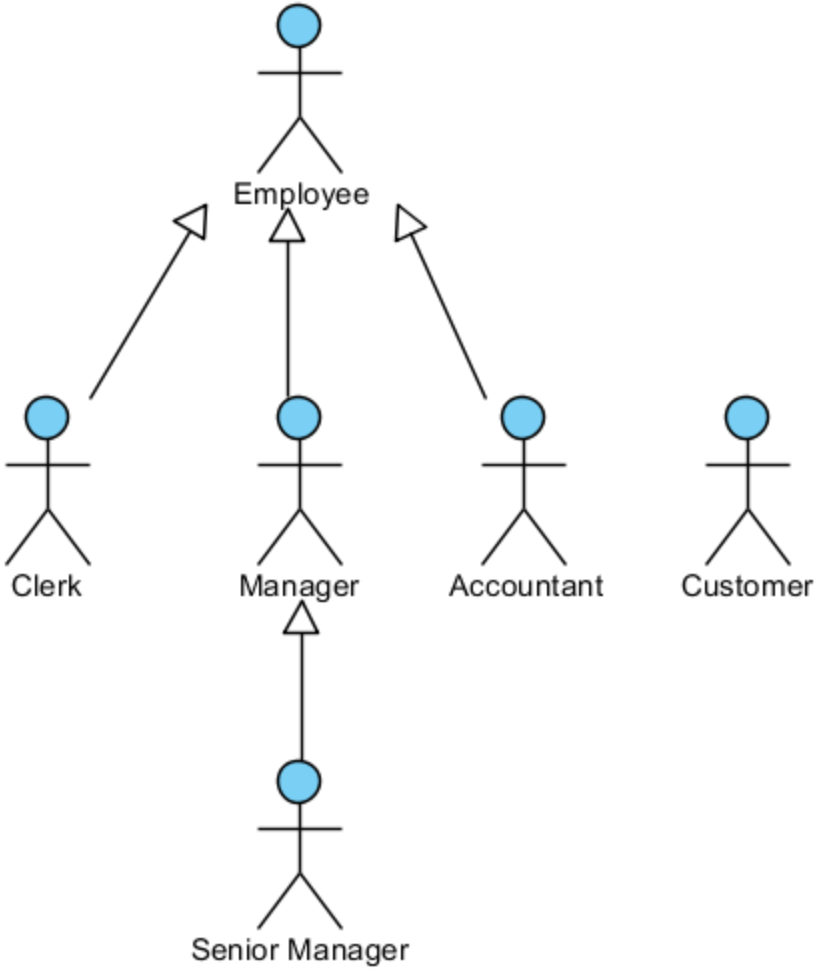
Another diagram used to standardize the treatment of users and external systems

Unlike use-case diagram, only shows actors and how actors are related

ROLE MAP

Recall: Superclass and subclass in OO programming





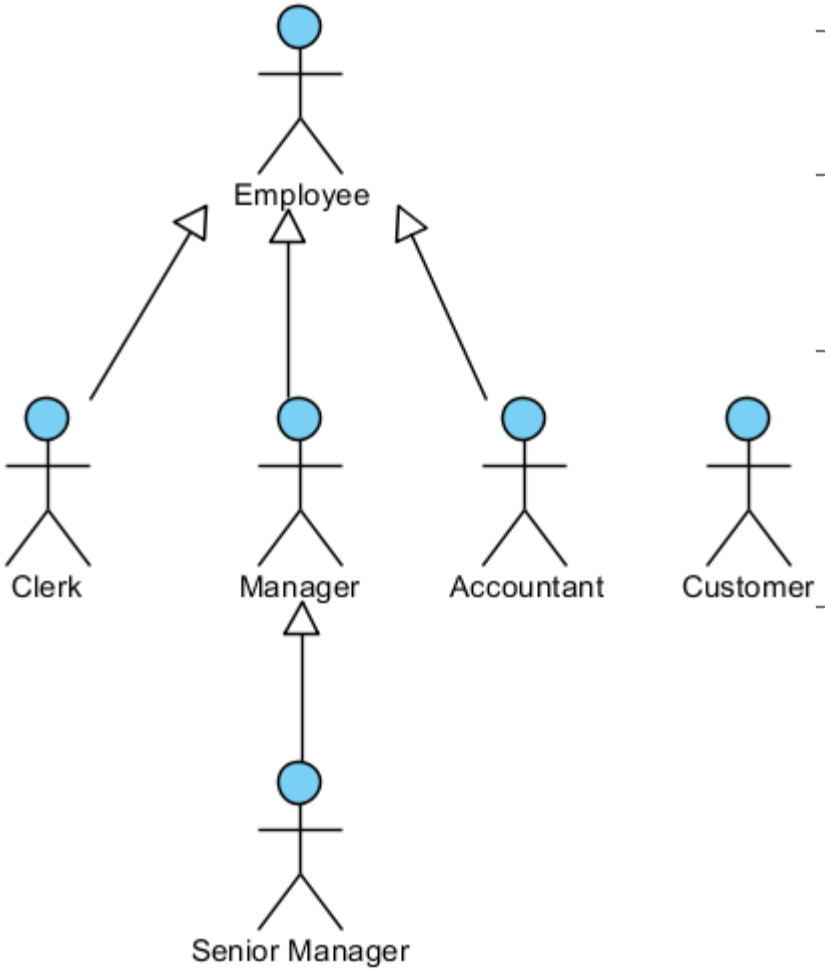
ROLE MAP

The arrow allows us to model actors with overlapping roles

Think of the generalization relationship arrow as "a kind of"

- E.g.
  Clerk, Accountant, Manager are different kinds of Employer
- Senior Manager is a kind of Manager

ROLE MAP



General

More Specific

More Specific
=>
they can do more
things

Even More Specific

GENERALIZATION AND SPECIALIZATION

Specialized actors inherit the ability to do all the things that a generalized actor can do

Think Object-Oriented Programming (OOP)

Clerk extends (inherits) Employee

- Employee = Generalized actor
- Clerk = Specialized actor

WHY DO WE DEFINE GENERALIZED ACTORS?

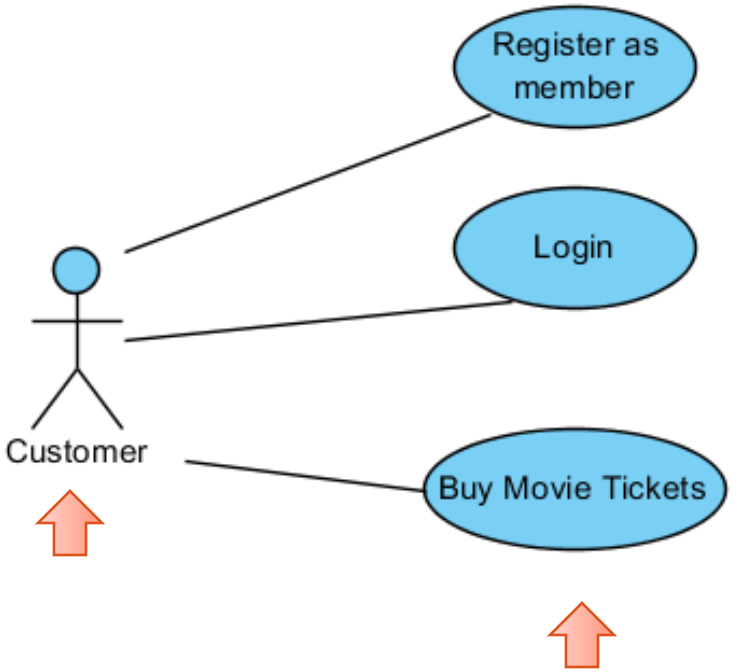
Reduce the number of association lines (just connect to the generalized actor)

Simplify the drawing of use case diagram

USE CASE MODELING ELEMENTS

Actor: An actor who initiates a use-case interaction.

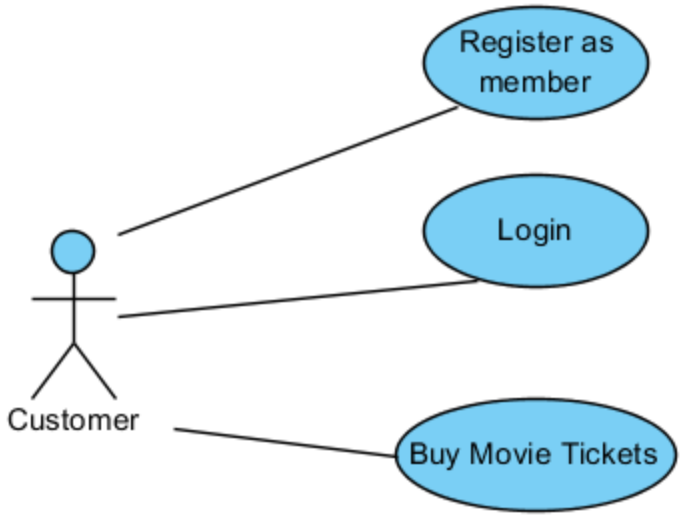
Use case: A user task (indicated as an oval)



Can draw with either a line or arrow pointing to the the use case

IDENTIFY USE CASES

A use case is an interaction between an actor and a computer system



Each use case is named using a verb

E.g. “Register as user”, “Login”, ...

GUIDE FOR IDENTIFYING USE CASES

1. **Need to identify all the actors of this system**
   - Note that actors need not be just human users, they can be **other systems** that interact with this system
2. **A use case must be a system function/interaction**
   - E.g. clerk calling a customer is NOT a use case - it's not a system function
3. **A use case should fulfil a business objective**
   - E.g.
     Actor: Student
     Use case: View list of enrolled courses

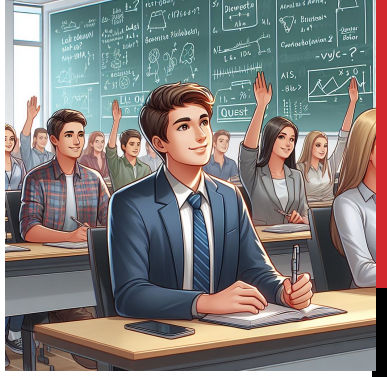
GUIDE FOR IDENTIFYING USE CASES

4. **While a use case is a system function/interaction, need to know how to differentiate between a use case vs a step of a use case**
   - Note that a use case is a high-level system function
   - We will derive the individual steps of a use case in future phases (not now)
   - Do NOT try to list out the individual steps of a use case
   - E.g
     Use case: Update user profile
     Possible steps of the “Update user profile” use case:
     enter name, enter date of birth, enter email, etc
   - Keep in mind that a use case must fulfil a business objective
   - Ultimately whether a system interaction is a use case, or a step depends on the design of the system (will talk more about this later on)

GUIDE FOR IDENTIFYING USE CASES

5. **Use cases are not just a function in code. It should be seen from a point of view of an actor**
   - E.g. “check password during login”, “send email confirmation”, “add to database”, etc are not use cases. They are what the system/code does
6. **Use cases are seen from the perspective of a user rather than from the perspective of the system**
   - i.e. “get a list of chat messages” rather than “receive a chat message”
7. **A use case diagram should not have this system itself as an actor**
   - Usually, we omit time event in the use case diagram
   - But if you really want to draw, you can create an ”actor” called TIME

QUESTION TIME



1. Why do we gather requirements using a use case diagram rather than just come up with a list of system functions?


It’s your turn...

TASK

Come up with the Use Case Diagram for the Canvas system

SUMMARY

Properties of Requirements

Types of Requirements : Functional, Nonfunctional, Implementation Requirements

Requirements Gathering Approaches

Role Map and Use Case Diagrams

WHAT'S NEXT?

Requirements Analysis