COMP 3111
SOFTWARE ENGINEERING

LECTURE 8
SYSTEM REQUIREMENTS CAPTURE

SYSTEM REQUIREMENTS CAPTURE OUTLINE

- ✓ System Requirements Capture Overview
  - Life-cycle Role
  - Importance of Requirements Capture
  - Why Requirements Capture is Difficult

- ➔ **System Requirements Capture Activities**
  - ✓ **Capture Data Requirements: Domain Modeling**
  - ☞ **Capture Functional Requirements: Use-Case Modeling**
  - **Capture Nonfunctional Requirements**
  - **Validate System Requirements**

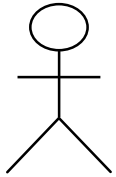
USE-CASE MODELING

- Captures the system behavior *from the users' point of view*.
- Developed incrementally *in cooperation with the domain model*.
- Helps in:
  - capturing data and functional requirements.
  - planning iterations of development.
  - validating the system.

☞ **USE CASES DRIVE THE DEVELOPMENT EFFORT.**

All required functionality is described in the use cases.

USE-CASE MODELING: ACTORS



An *actor* represents something outside the system that interacts *directly* with it.

Can be a person or another system.

Provides input to or receives output from the system.

A role a user can play → multiple roles per user;
multiple users per role.

☞ **Actors are usually the source for discovering use cases.**

An actor is a stereotype of a UML class:

an actor is a classifier; a specific user or system is an instance.

USE-CASE MODELING: IDENTIFYING ACTORS

- Who or what uses the system?
- What roles do they play in the interaction?
- Who gets and provides information to the system?
- What other systems interact with this system?
- Who installs, starts and shuts down, or maintains the system?

☞ **It is possible to have both a domain model class and an actor that represent the same thing.**

☞ **Input/output devices are NEVER actors!**

For each actor, briefly describe the role it plays when interacting with the system.

ASU USE-CASE MODEL: ACTORS

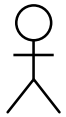
At the beginning of each term, students may request a course catalogue containing a list of course offerings needed for the term. Information about each course, such as instructor, department, and prerequisites are included to help students make informed decisions.

The new system will allow students to select four course offerings for the coming term. In addition, each student will indicate two alternative choices in case a course offering becomes filled or is canceled. No course offering will have more than forty students or fewer than ten students. A course offering with fewer than ten students will be canceled. Once the registration process is completed for a student, the registration system sends information to the billing system so the student can be billed for the term.

Instructors must be able to access the online system to indicate which courses they will be teaching, and to see which students signed up for their course offerings.

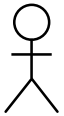
For each term, there is a period of time that students can change their schedule. Students must be able to access the system during this time to add or drop courses.

ASU USE-CASE MODEL: ACTORS



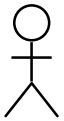
Student

A person who is registered to take courses at the university.



Billing System

An external system responsible for billing students.

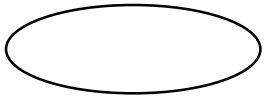


Instructor

A person who is part of the teaching staff of the university.

ASU USE-CASE MODEL

USE-CASE MODELING: USE CASE



use-case name

A *use case* is a specific way of using the system by performing some part of its functionality.

Describes the interaction that takes place between an actor and the system and what the system must do.

☞ Considered from the actor's viewpoint.

Constitutes a *complete sequence* of events/actions.

Always initiated by an actor.

- Initially, only consider normal sequence of events/actions.

☞ **Ignore alternative events/actions.**

USE-CASE MODELING: SCENARIO

A *scenario* is a concrete, focused, informal description of a *single use* of the system from the viewpoint of a *single actor*.

☞ **It is an actual attempt to carry out a use case.**

Note: There are two viewpoints of use-case modeling:

1. top-down: start with use cases, refine with scenarios.
2. bottom-up: start with scenarios, abstract to use cases.

☞ **In reality, the use-case specifier uses both viewpoints.**

A use case is a stereotype of a UML class:
use case is a classifier; scenario is an instance

USE-CASE MODELING:
IDENTIFYING USE CASES AND SCENARIOS

- What are the tasks that an actor wants the system to perform?
- What information does an actor access (create, store, change, remove or read) in the system?
- Which external changes does an actor need to inform the system about?
- Which events does an actor need to be informed about by the system?
- How will the system be supported and maintained?

☞ **State a use case name from the perspective of the actor as a present-tense, verb phrase in active voice.**

☞ **Provide a description of the purpose of the use case and an outline of its functionality.**

☞ **Use application domain terms in descriptions (i.e., from the glossary/data dictionary).**

ASU USE-CASE MODEL: USE CASES

At the beginning of each term, **student**s may request a course catalogue containing a list of course offerings needed for the term.

functionality: Student: ~~*request course catalogue*~~ → *browse course offering*
Someone: ~~*prepare course catalogue*~~ → *prepare course offering*

Information about each course, such as **instructor**, department, and prerequisites are included to help **student**s make informed decisions.

functionality: Part of *prepare course offering* functionality.

The new system will allow **student**s to select four course offerings for the coming term.

functionality: Student: *select course offering*

ASU USE-CASE MODEL: USE CASES

In addition, each **student** will indicate two alternative choices in case a course offering becomes filled or is canceled.

functionality: Student: ~~*select alternate choices*~~ → *select course offering*
Someone: *cancel course offering*

No course offering will have more than forty **student**s or fewer than ten **student**s.

functionality: None (constraint on functionality)

A course offering with fewer than ten **student**s will be canceled

functionality: No new functionality in this statement

Once the registration process is completed for a **student**, the registration system sends information to the **billing system** so the student can be billed for the term.

functionality: Billing System: *receive billing information*

ASU USE-CASE MODEL: USE CASES

Instructors must be able to access the online system to indicate which courses they will be teaching, and to see which **student**s signed up for their course offerings.

functionality: Instructor: *select courses to teach*
Instructor: *request enrollment list*

For each term, there is a period of time that **student**s can change their schedule.

functionality: Student: *change course offering*

Students must be able to access the system during this time to add or drop courses.

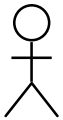
functionality: Student: *add course offering*
Student: *drop course offering*

ASU USE-CASE MODEL: ACTOR DESCRIPTIONS

Registrar

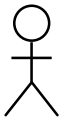
Discovered from domain experts.

A person who is responsible for maintaining the curriculum information inclusive of students, professors and courses. The Registrar uses the Course Registration System to prepare the course catalogue for the coming term and to manage course offerings.



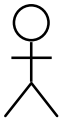
Student

A person who is registered to take courses at the university. A student uses the Course Registration System to register for courses in the current or a future term.



Billing System

An external system responsible for billing students. After a student successfully registers for a term, the Course Registration System sends the billing information to the billing system.



Instructor

A person who is part of the teaching staff of the university. A professor uses the Course Registration System to make choices on the courses to teach and to request course enrolment lists.

ASU USE-CASE MODEL: ACTOR FUNCTIONALITY

Registrar – *prepare course offering*

Registrar – *cancel course offering*

Student – *browse course offering*

Student – *select course offering*

Student – *change course offering*

Student – *add course offering*

Student – *drop course offering*

Billing System – *receive billing information*

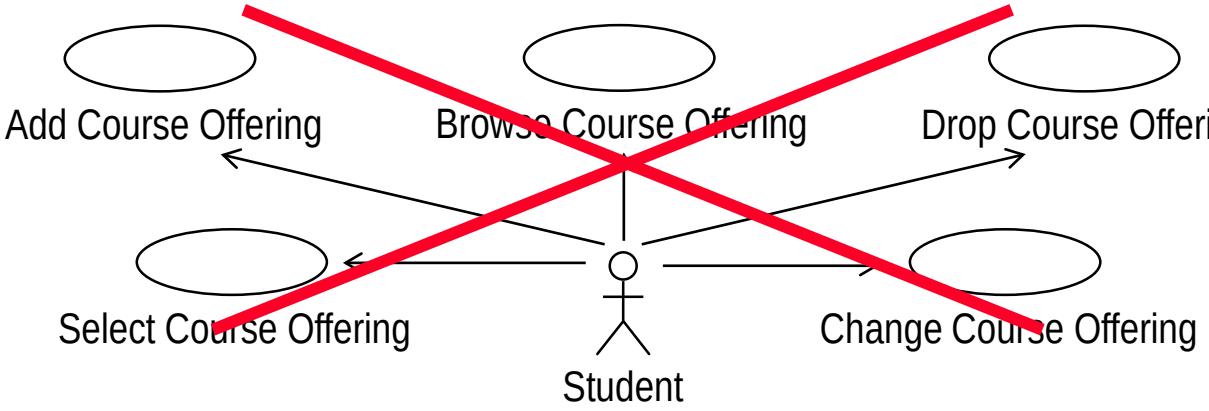
Instructor – *select courses to teach*

Instructor – *request enrollment list*

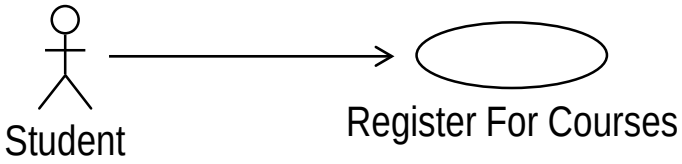
WHAT IS A GOOD USE CASE?

A use case typically represents a major piece of functionality that is complete from beginning to end.

A use case must deliver something of value to an actor.



Are these good use cases?



What is the actor trying to accomplish?

WHAT IS A GOOD USE CASE?

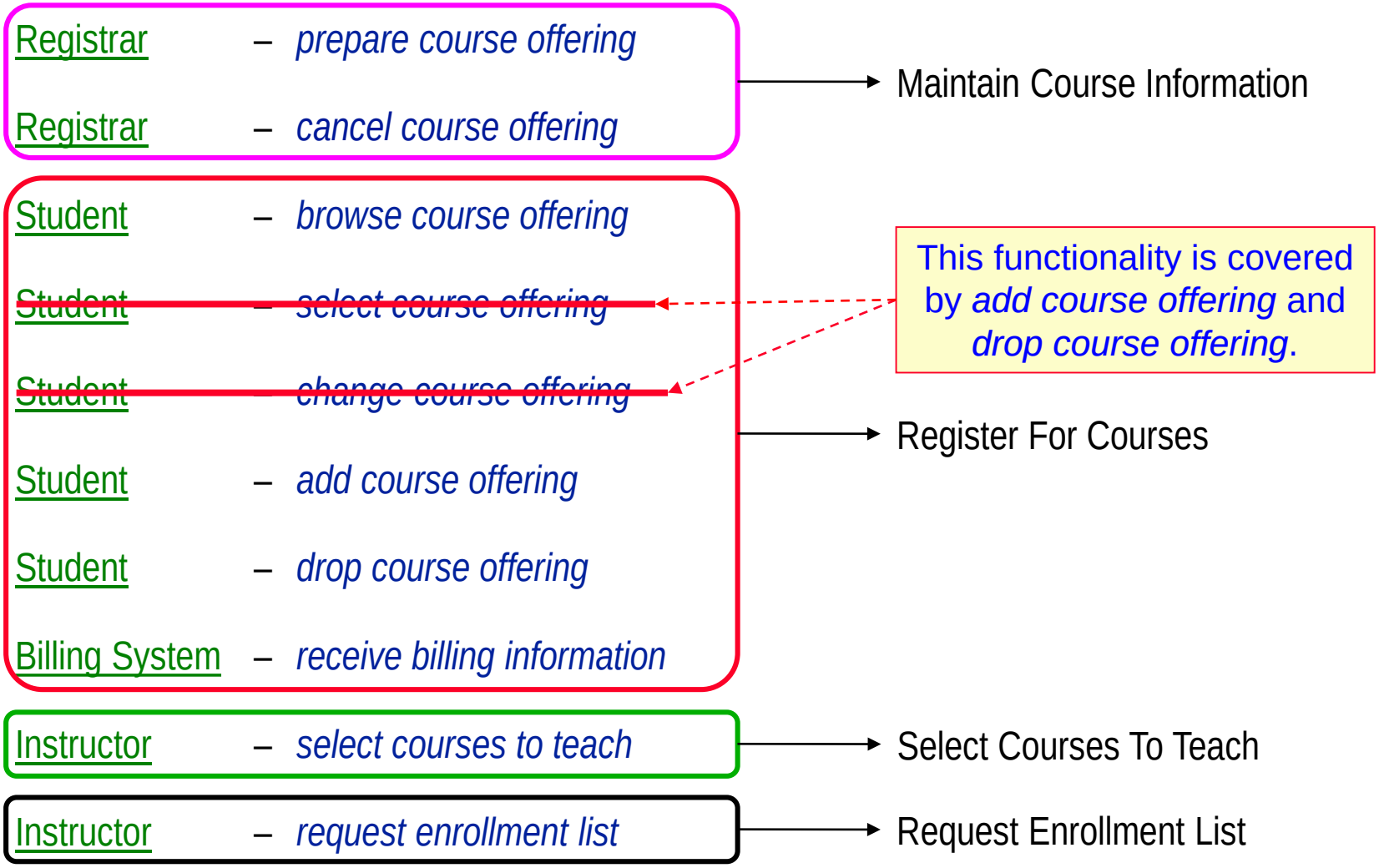
A use case typically represents a major piece of functionality that is complete from beginning to end.

A use case must deliver something of value to an actor.

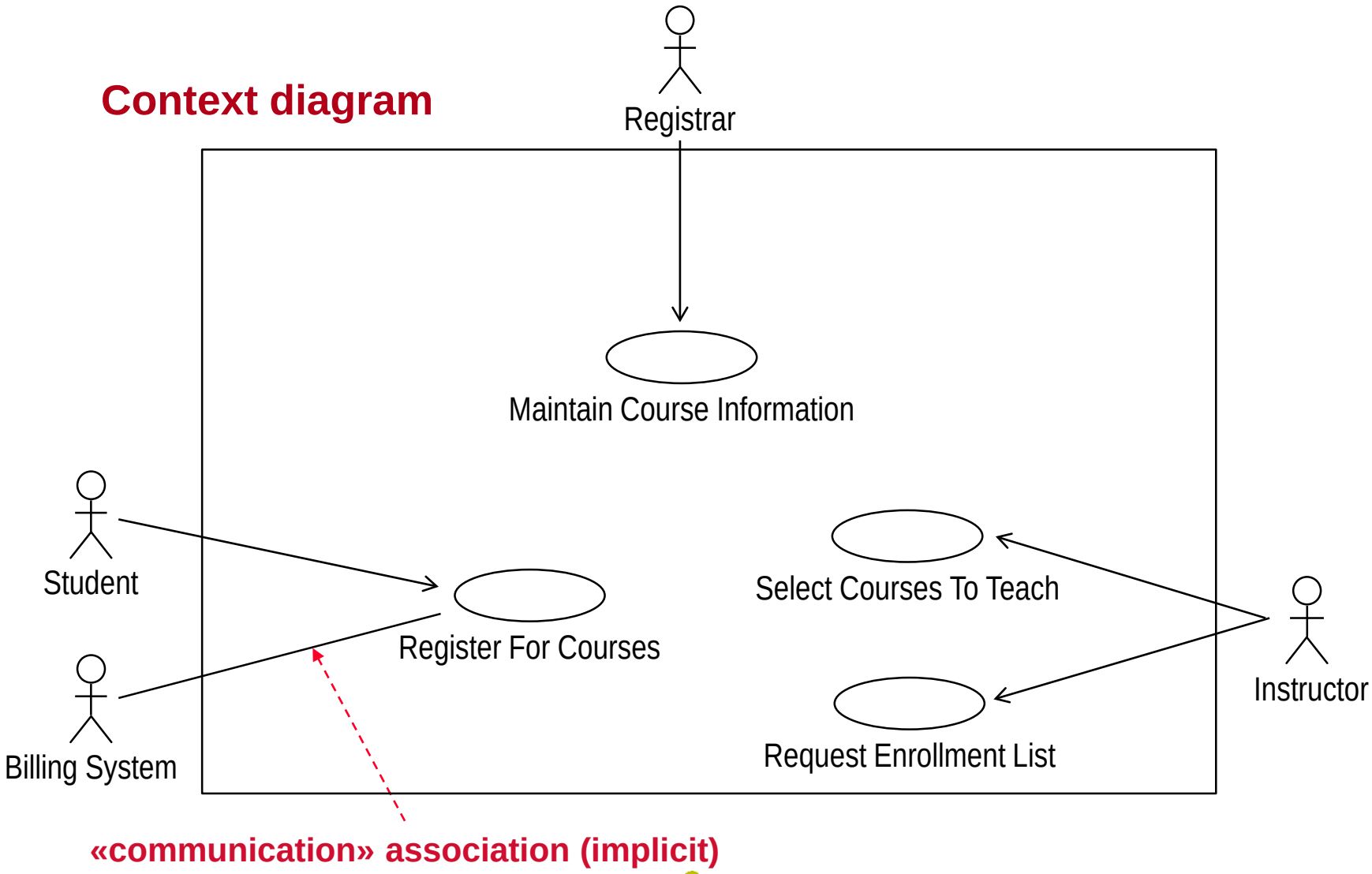
Generally, it is better to have longer and more extensive use cases than smaller ones.

We want real and complete use cases, not several sub-cases of a use case.

ASU USE-CASE MODEL:
FUNCTIONALITY ANALYSIS AND GROUPING



ASU USE-CASE MODEL: USE-CASE DIAGRAM



ASU USE-CASE MODEL:
INITIAL USE-CASE SPECIFICATION

Register For Courses

This use case describes the process by which a student registers for a course offering. It provides the capability to create and modify a student's study schedule for the coming term.

Flow of Events

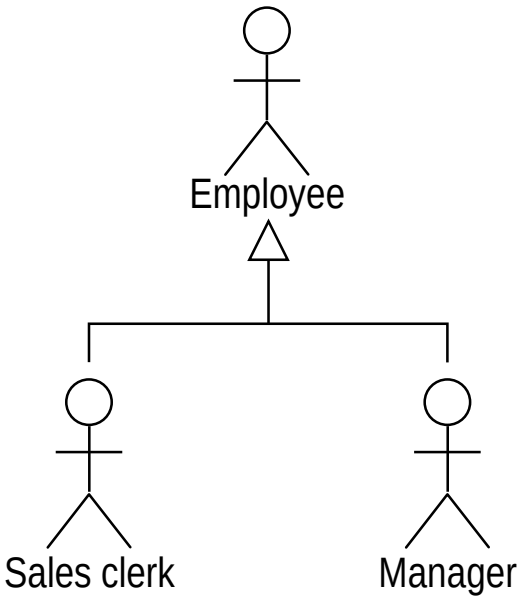
1. Add course offering.
2. Drop course offering.
3. Send billing information.

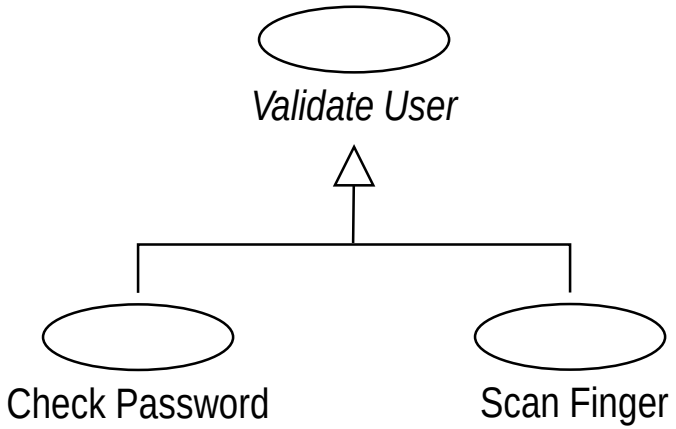
Initially, we just give a brief description of the purpose of a use case and an outline of the steps required to perform the use case.

The steps are refined and described in more detail as we discover more about the functionality required.

USE-CASE MODELING: GENERALIZATION

Since actors and use cases are classifiers, they can be generalized.





Do not use use-case generalization in your project. There is no need to do so and it is often used incorrectly.

The use of generalization represents a design decision!

SYSTEM REQUIREMENTS CAPTURE

USE-CASE MODELING EXERCISE