Introduction to SOFTWARE ENGINEERING

Nguyễn Huy Hoàng - Phan Gia Phước

[09 . 2021]

Adapted from the Slides of Software Engineering, 10th Ed. by Ian Sommerville

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Chapter 4.

REQUIREMENTS ENGINEERING (cont.)

4.7 (cont.)

UML

What is UML ?

- UML - Unified Modeling Language
- Standard & graphical language

❑ Used for software as well as non software

  - UML is a language (***notation***) for modeling Object Oriented system

- Used for making software blue print
  - *specifying, visualizing, constructing, documenting* the artifacts of software system

Why UML?

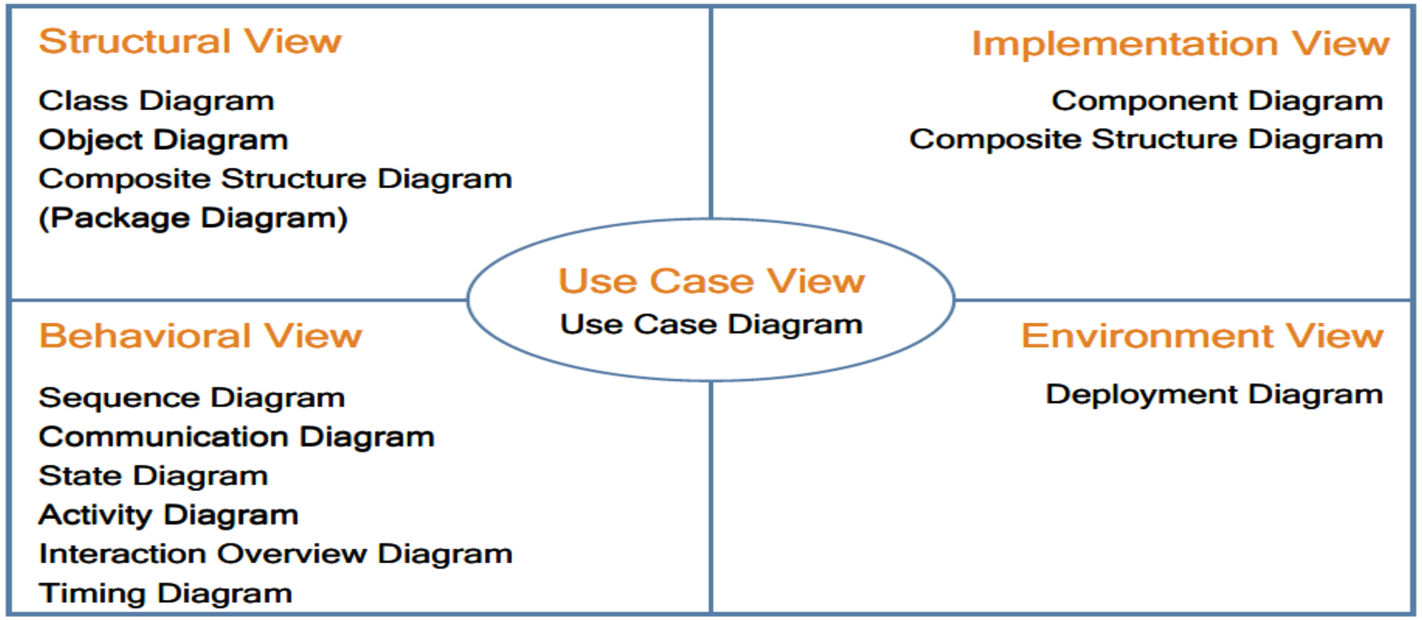
- Graphical notation
  - A picture is worth a thousand words
- *Standard communication* language
- Provides multiple diagrams for capturing different *Architectural Views*
- Promotes component reusability

UML is a standard language for *visualizing, specifying, constructing, and documenting* software systems

UML Architectural Views and Diagrams

- UML defines 13 diagrams that describe 4+1 architectural views

  4+1 architectural views model was proposed by Philippe Kruchten, IBM



4.7 (cont.)

USE-CASE MODELING

Objectives

- When you complete this module, you should be able to:
  - Define actor, use case, and use-case model
  - List the benefits of use cases
  - Explain how use cases fit into a requirements management process and the software development lifecycle

Use cases involve a shift in thinking

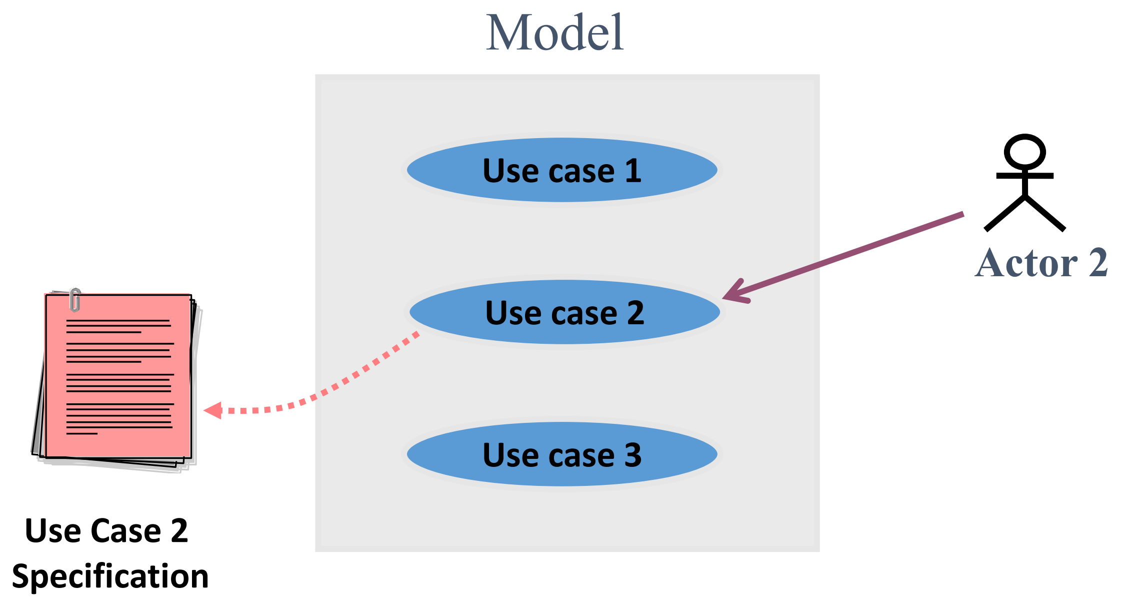
From a focus on the function of a system

To a focus on the value a system must deliver

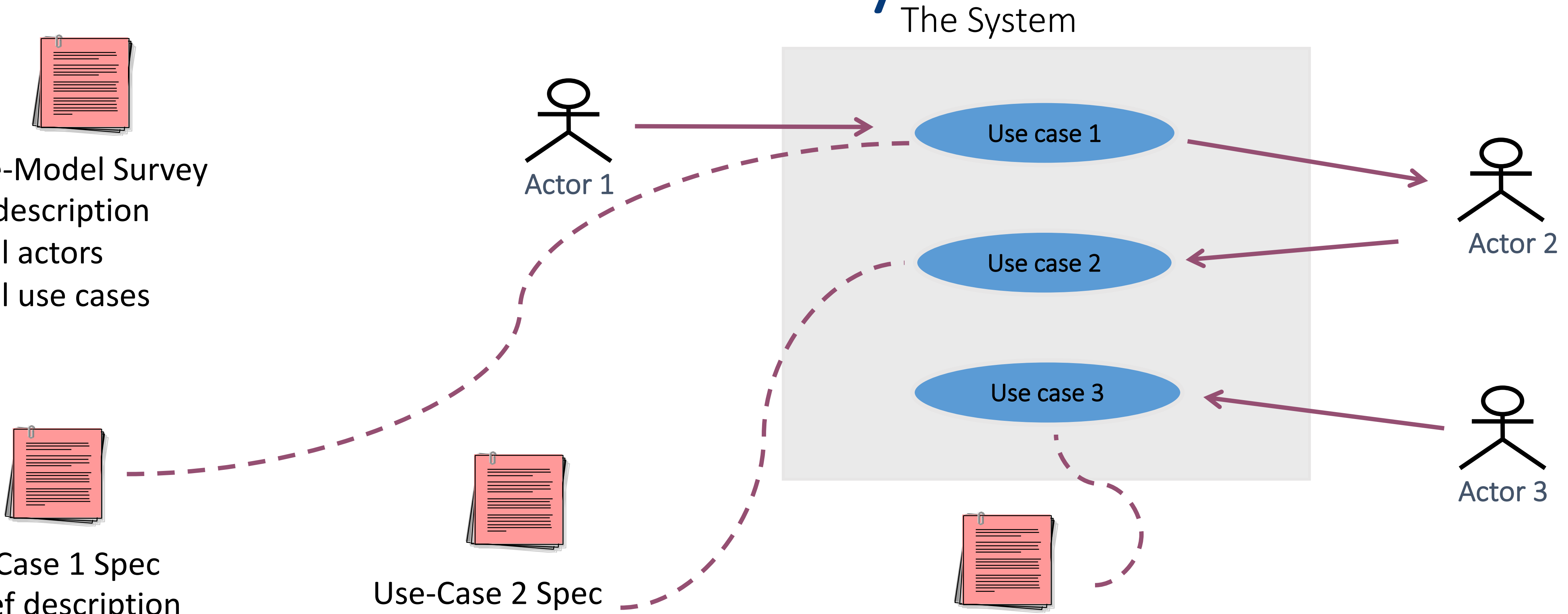
for its stakeholders

What Is Use-Case Modeling?

- Links stakeholder needs to software requirements.
- Defines clear boundaries of a system.
- Captures and communicates the desired behavior of the system.
- Identifies who or what interacts with the system.
- Validates/verifies requirements.
- Is a planning instrument.



A Use-Case Model is Mostly Text



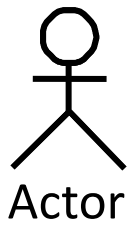
Use-Case-Model Survey
- survey description
- list of all actors
- list of all use cases

Use-Case 1 Spec
- brief description
- flow of events

Use-Case 2 Spec
- brief description
- flow of events

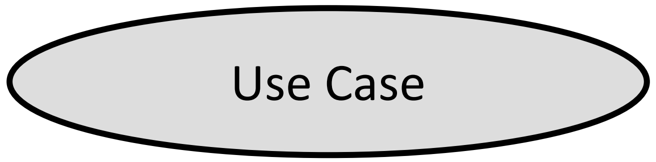
Use-Case 3 Spec
- brief description
- flow of events

Major Use-Case Modeling Elements



Actor

Someone/something outside the system, acting in a role that interacts with the system

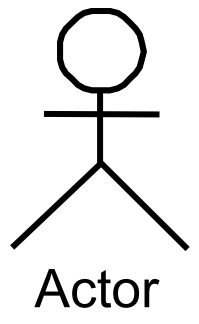


Use case

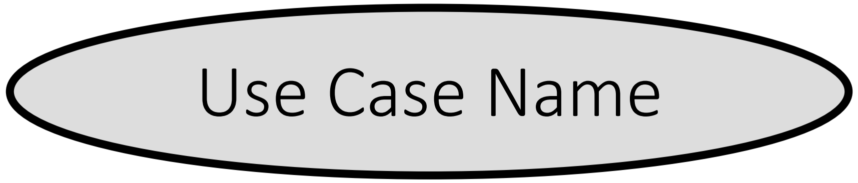
Represents something of value that the system does for its actors

Actor

- An actor represents a role that a human, hardware device, or another system can play in relation to the system
- An actor is external to the system
- A complete set of actors describes all of the ways in which outside users communicate with the system



What Is a Use Case?



A use case defines a sequence of actions

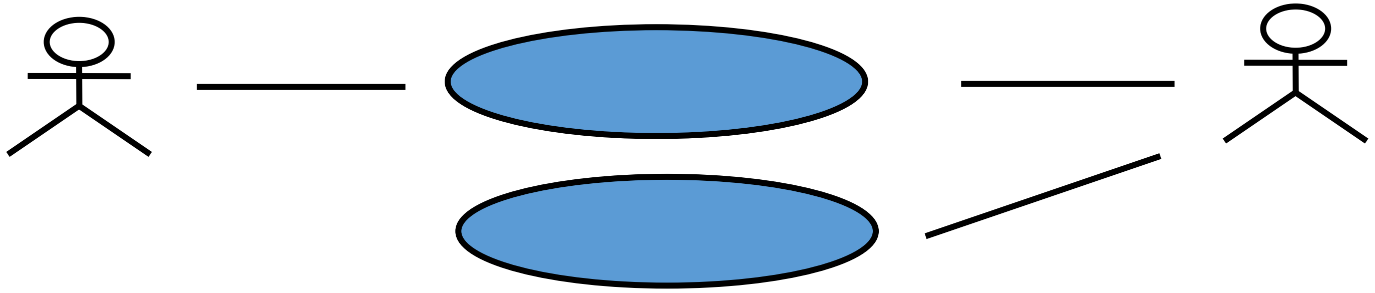
performed by a system

that yields an observable result of value

to an actor.

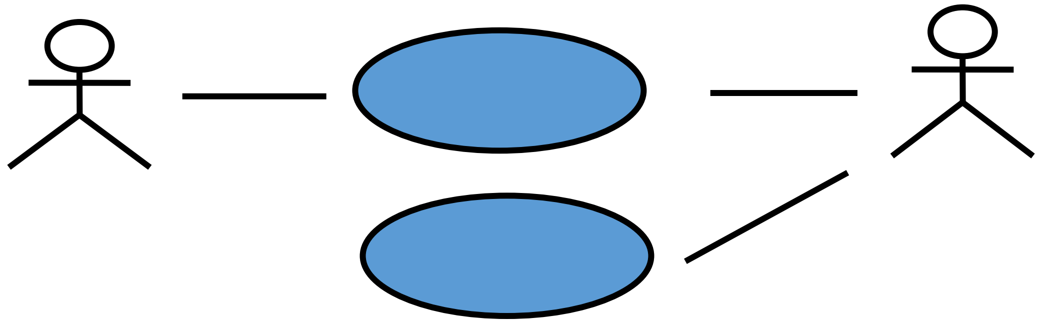
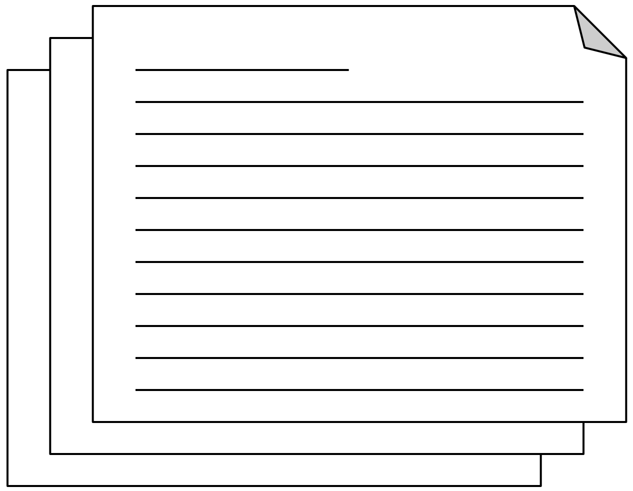
Use cases contain software requirements

- Each use case
  - Is a coherent unit of functionality provided by a system
  - Describes sequences of actions that the system takes to deliver something of value to an actor
  - Models a dialog between the system and actors
  - Is a complete and meaningful flow of events from the perspective of a particular actor



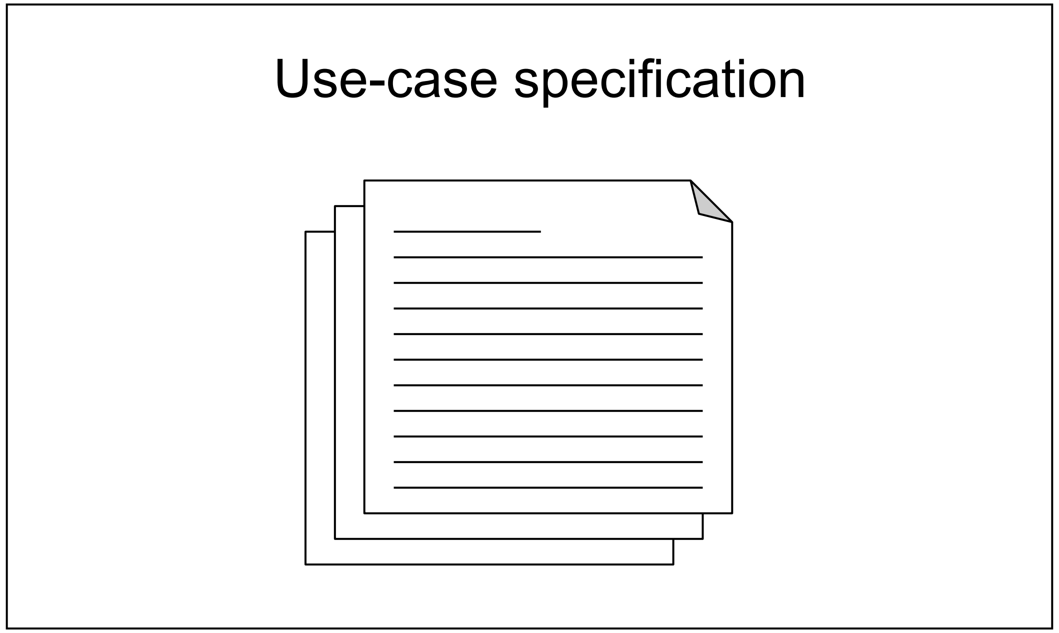
Capture a use-case model

- A use-case model is comprised of:

Use-case diagrams (visual representation)	**Use-case specifications** (text representation)
	

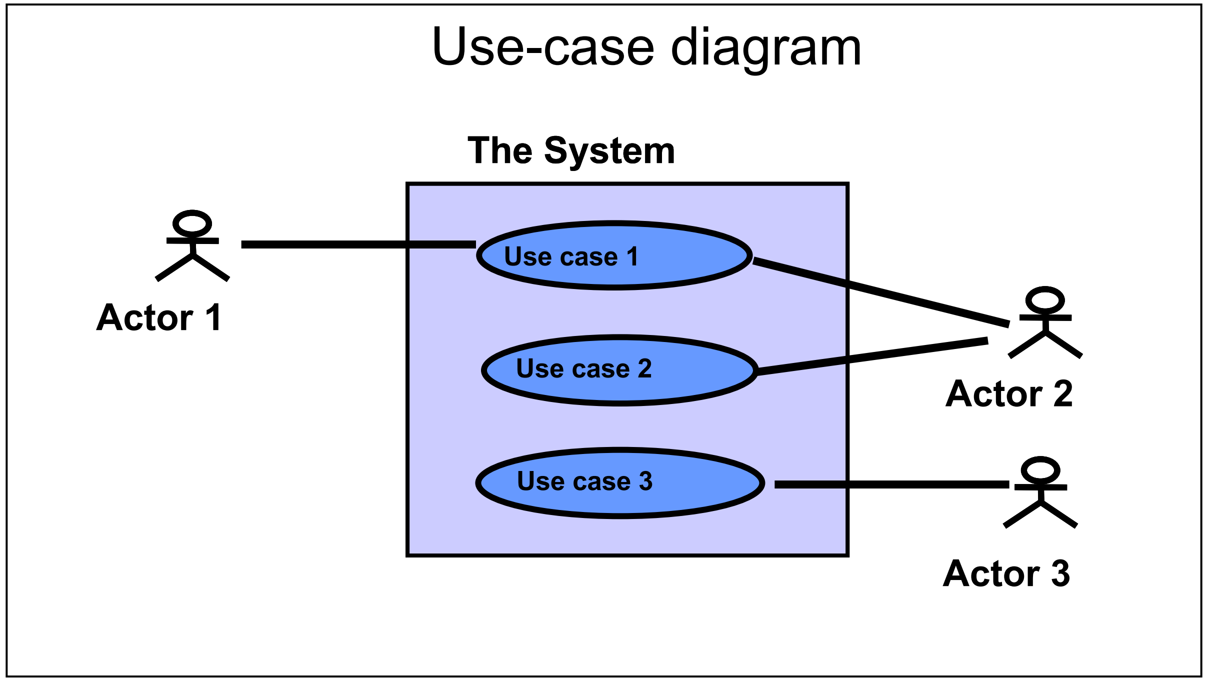
Use-case specification

- A requirements document that contains the text of a use case, including:
  - A description of the flow of events describing the interaction between actors and the system
  - Other information, such as:
    - Preconditions
    - Postconditions
    - Special requirements
    - Key scenarios
    - Subflows



Use-case diagram

- Shows a set of use cases and actors and their relationships
- Defines clear boundaries of a system
- Identifies who or what interacts with the system
- Summarizes the behavior of the system



Review

- What is an actor?
- What is a use case?
- What are some of the benefits of using use cases?
- What is a use-case model?
- What is a use-case diagram?
- What is a use-case specification?
- How does use-case modeling fit into the requirements management process?
- How does use-case modeling fit into the software development lifecycle?

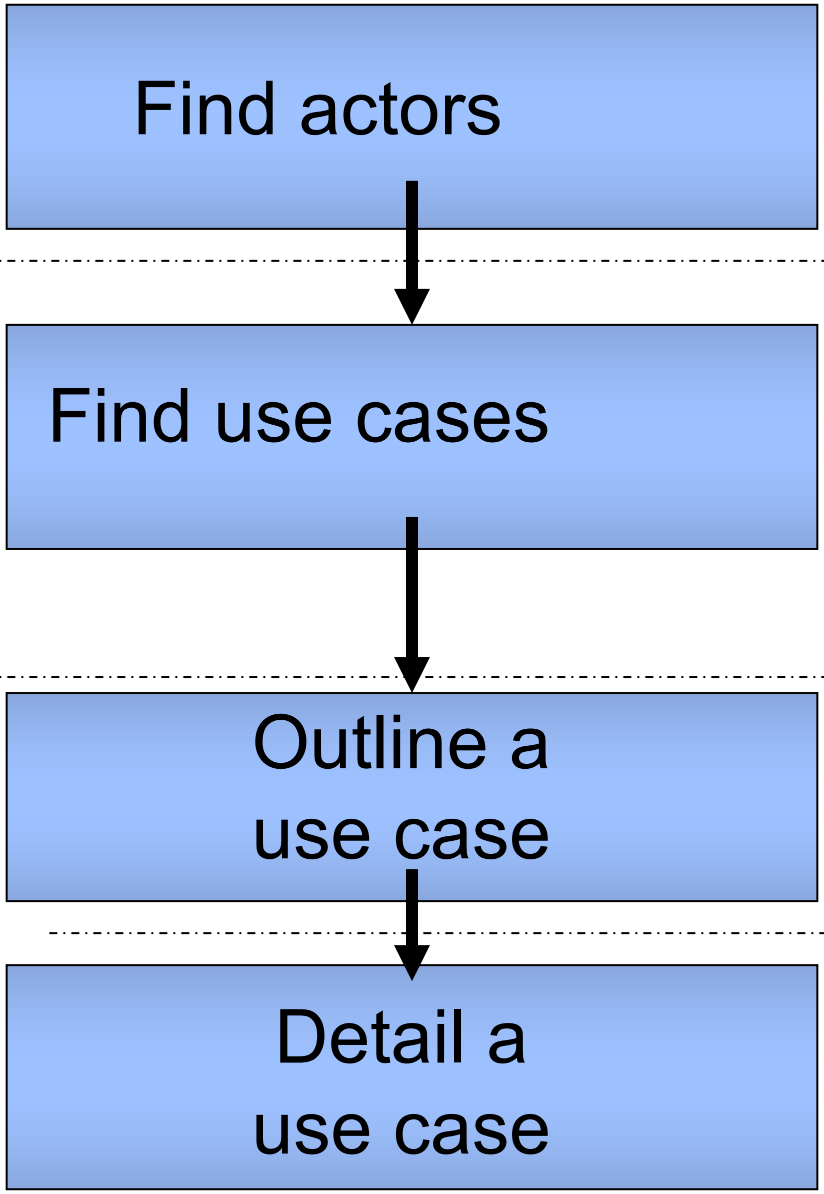
Use-case Model

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Objectives

- When you complete this module, you should be able to:
  - Describe the use-case writing process
  - Find and describe actors
  - Find and describe use cases
  - Relate use case

Process of writing use cases



Brief description: This use case allows a Registrar to close the registration process. Courses that do not have enough students are cancelled. The Billing System is notified for each student in each course that is not cancelled, so the student can be billed.

Close Registration Outline

-Flow of events

  -Step by step

Close Registration Use-Case Specification

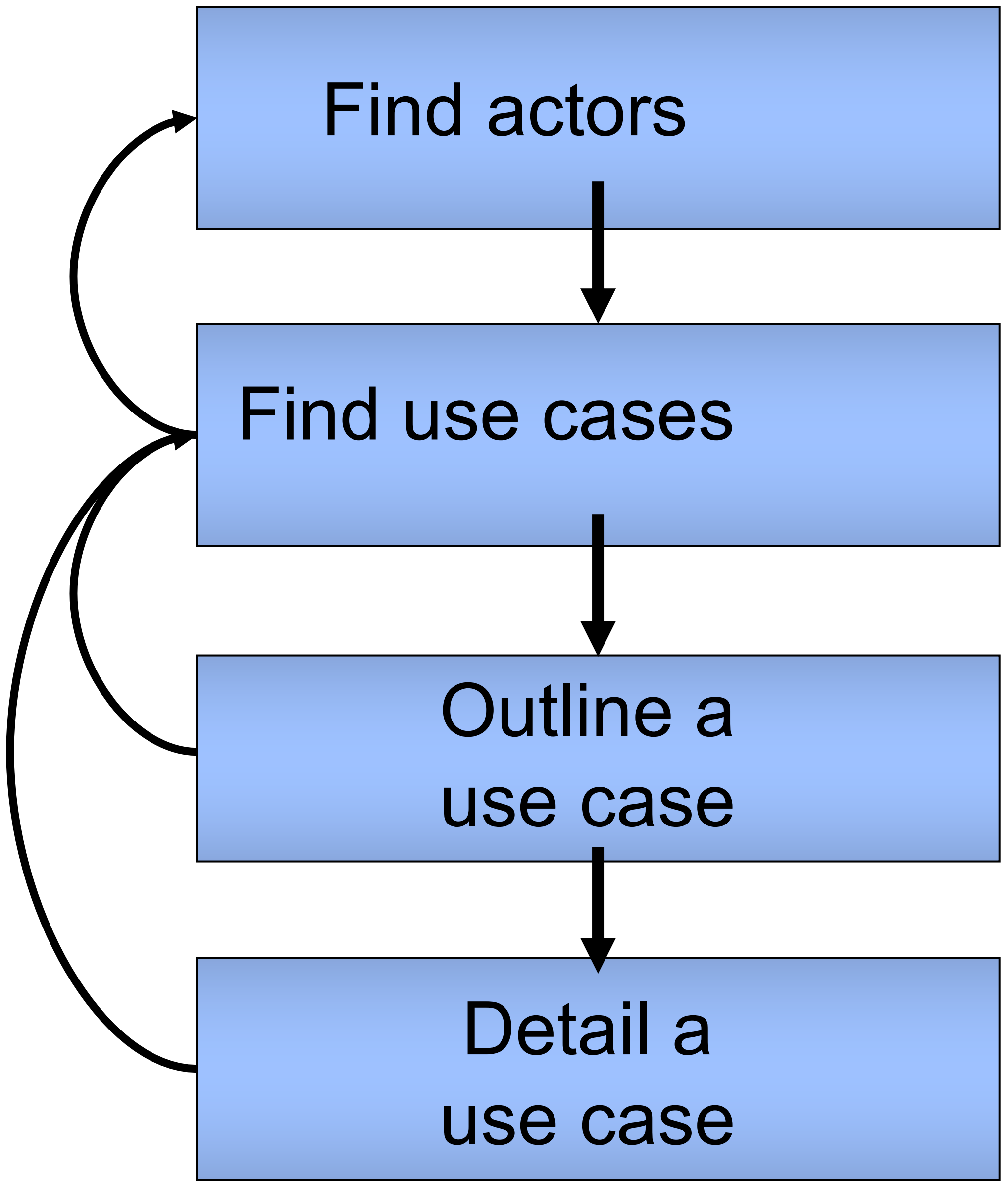
-Detailed Flow of Events

-Special Requirements

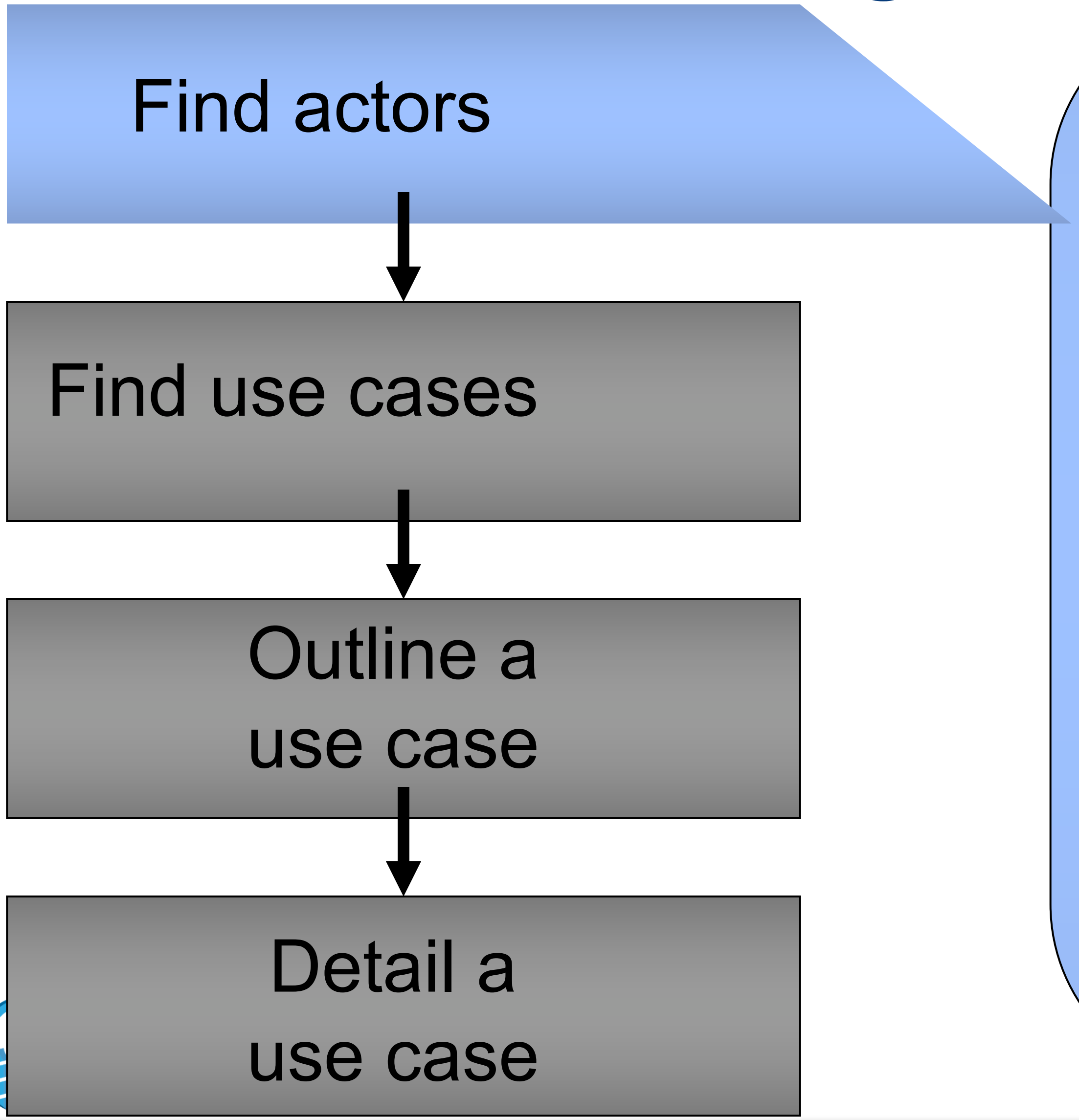
-Pre/Postconditions

Process of writing use cases (cont.)

Important Note:
Use case writing is an *iterative* process



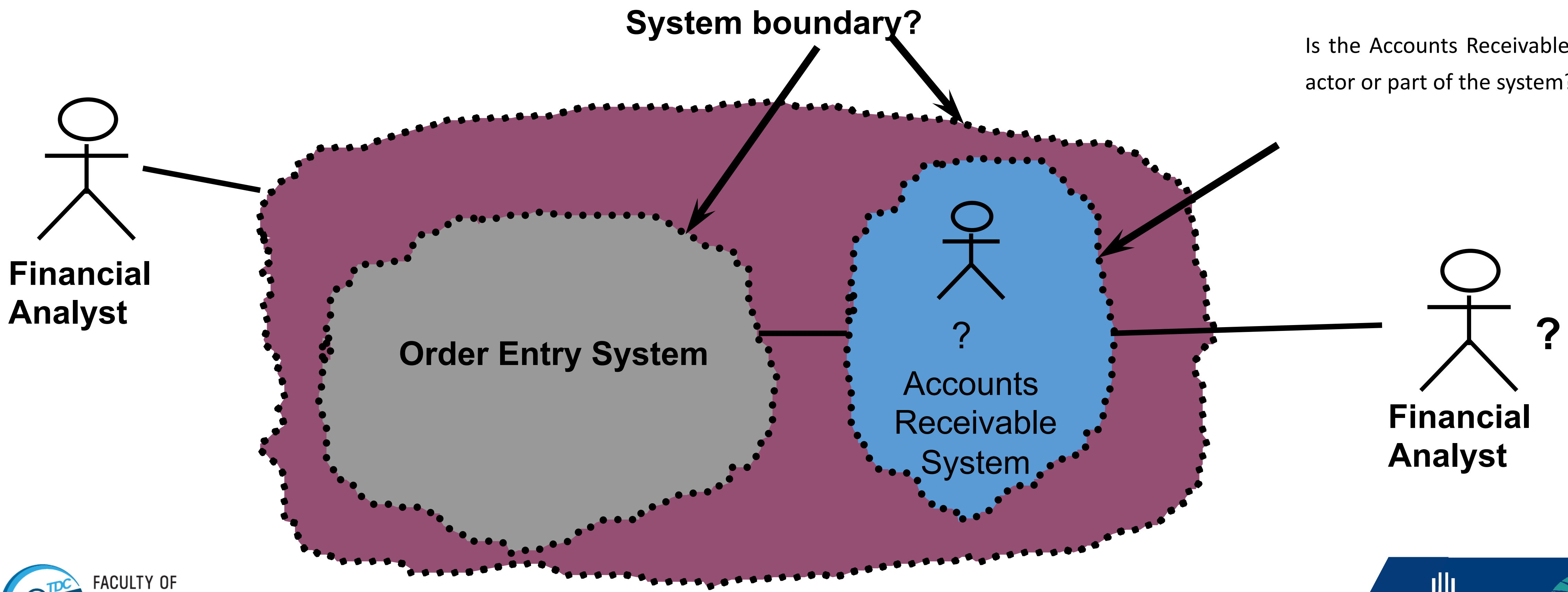
Process of writing use cases



- Name and briefly describe the actors that you have found

Actors and the system boundary

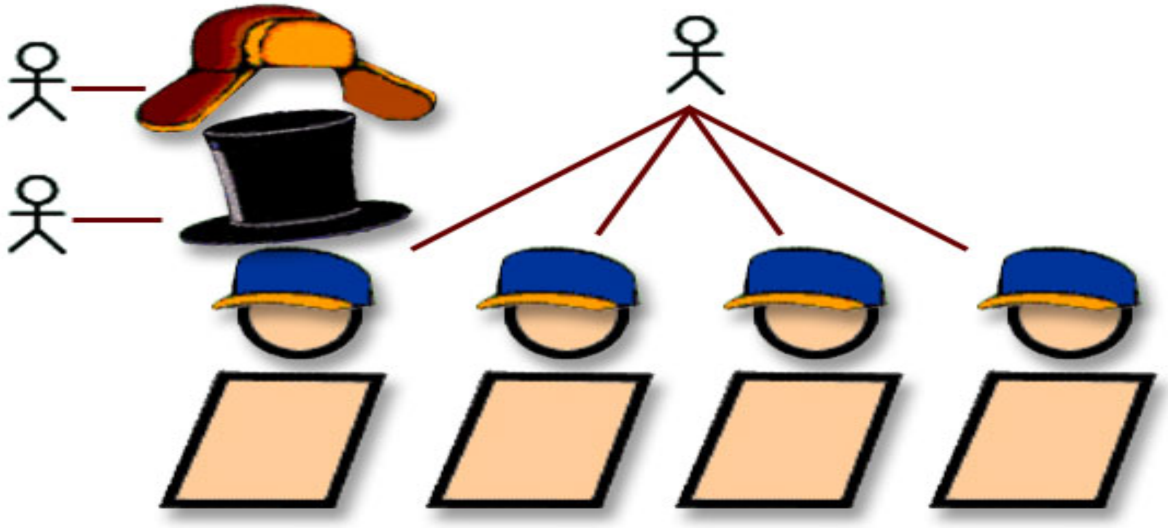
- Determine what the system boundary is
- Everything beyond the boundary that interacts with the system is an instance of an actor



Is the Accounts Receivable System an actor or part of the system?

Actors and roles

- An actor represents a role that a human, hardware device, or another system can plan in relation to the system.

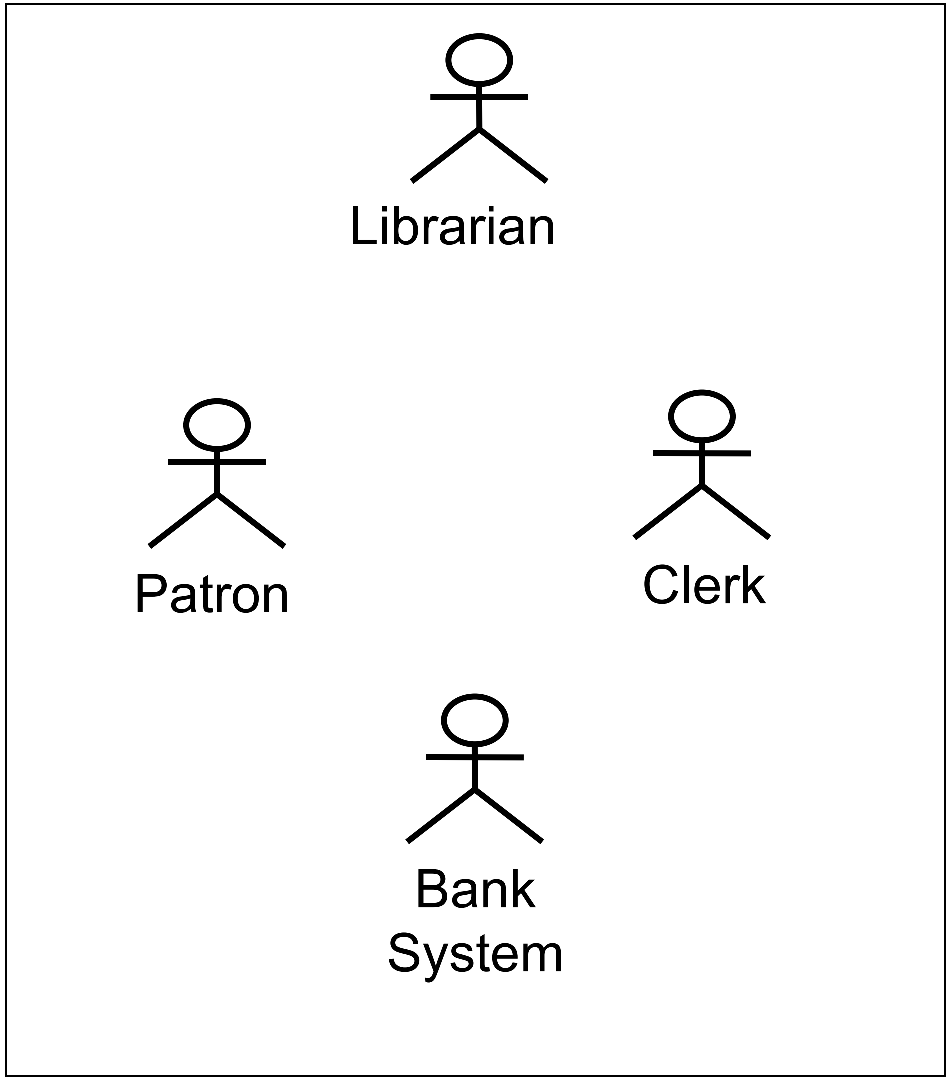


Find actors

- Who or what uses the system?
- Who or what gets information from this system?
- Who or what provides information to the system?
- Where in the company is the system used?
- Who or what supports and maintains the system?
- What other systems use this system?

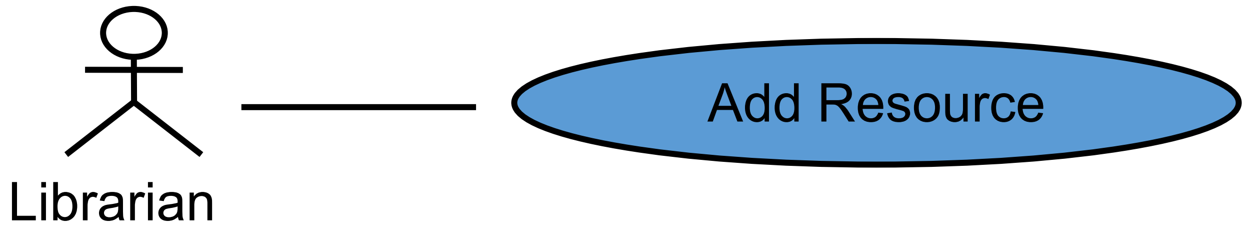
Name the actor

- Actor names should clearly convey the actor's role
- Good actor names describe their responsibilities



Describe the actor

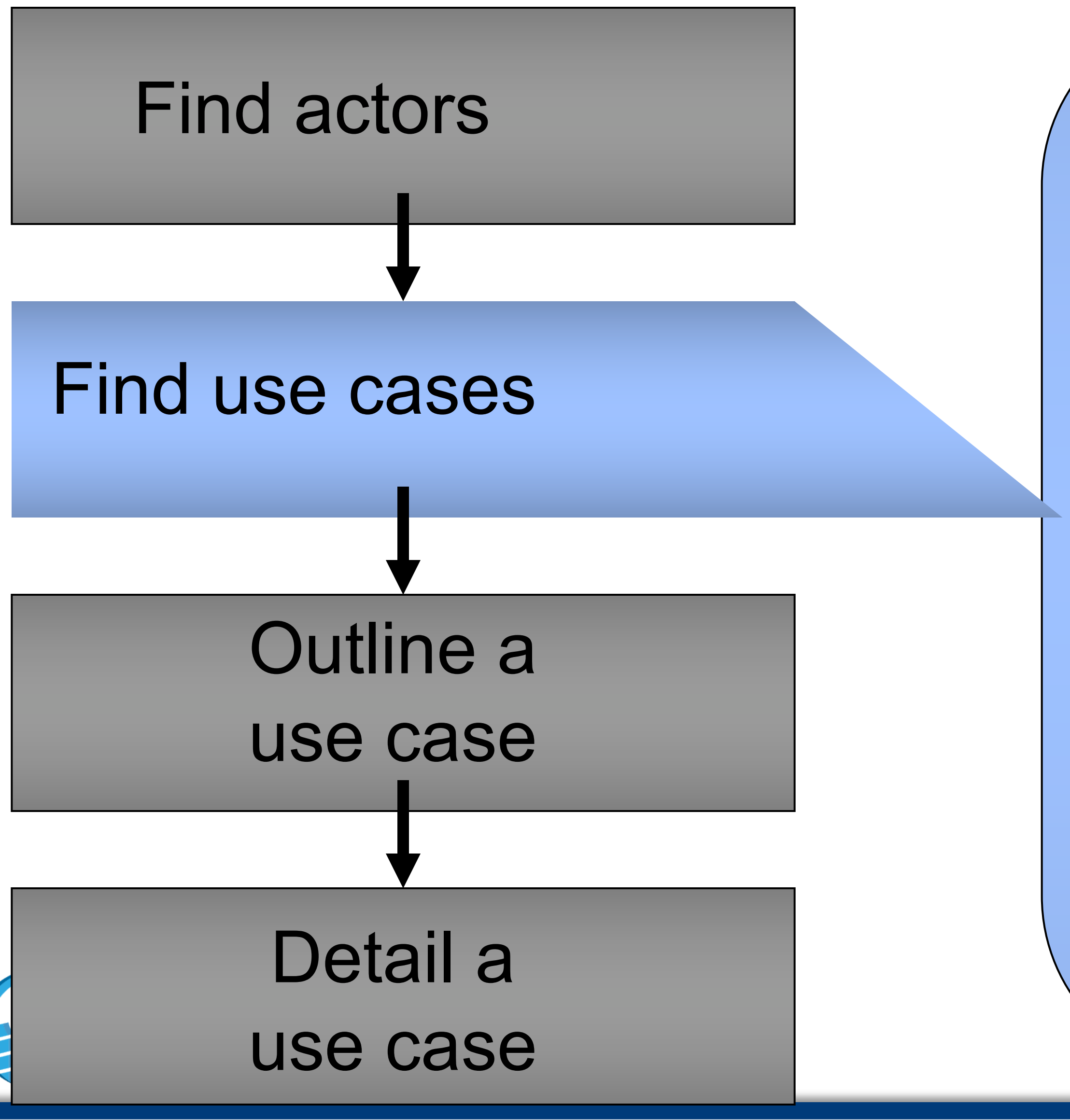
Name	**Librarian**
Brief description	A person who adds a resource into a library
Relationships with use cases	



Review

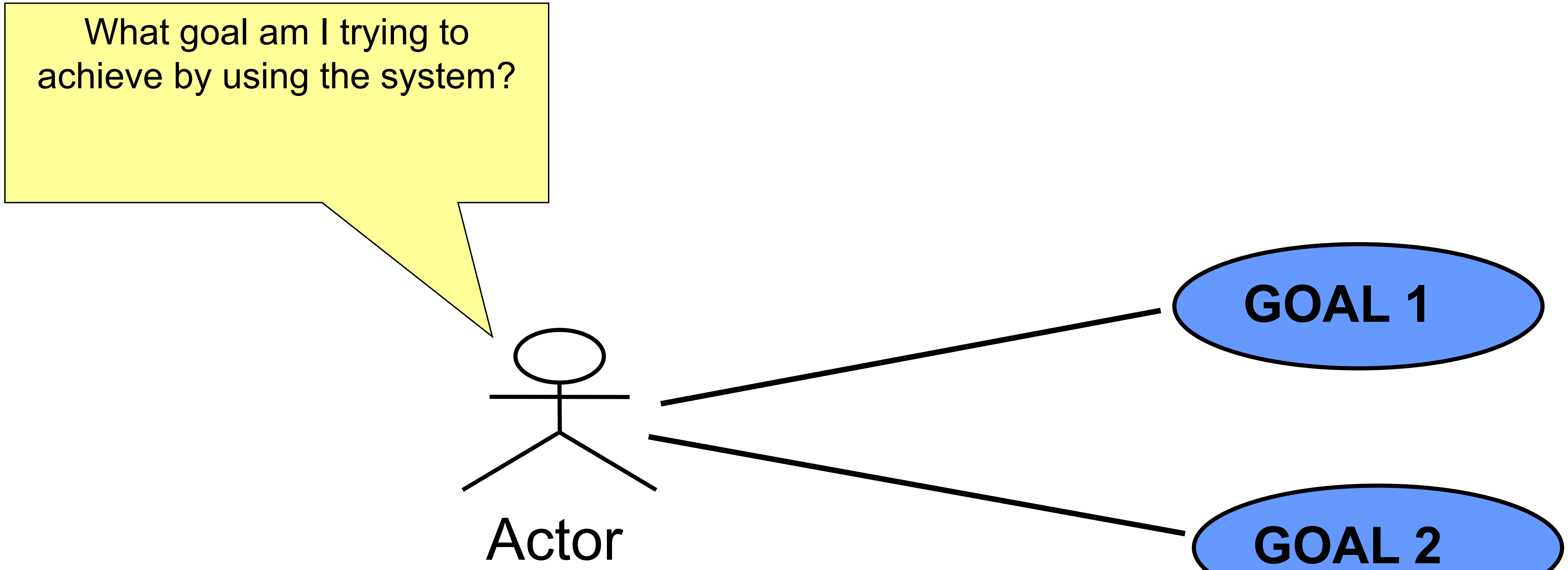
- How do you find actors?

Process of writing use cases



- Name and briefly describe the use cases that you found
- Create a use-case diagram
- Assess business values and technical risks for use cases

Find use cases

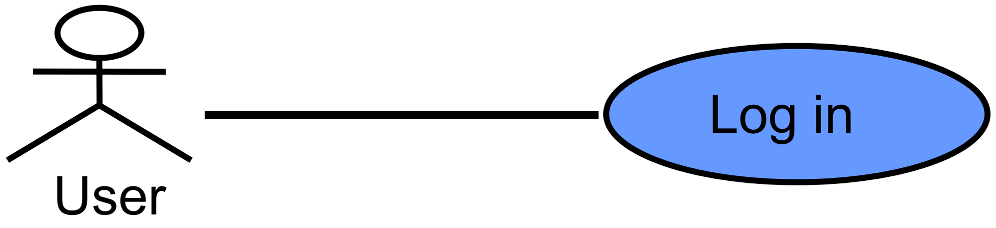


Find use cases (cont.)

- What are the goals of each actor?
  - Why does the actor want to use the system?
  - Will the actor create, store, change, remove, or read data in the system? If so, why?
  - Will the actor need to inform the system about external events or changes?
  - Will the actor need to be informed about certain occurrences in the system?

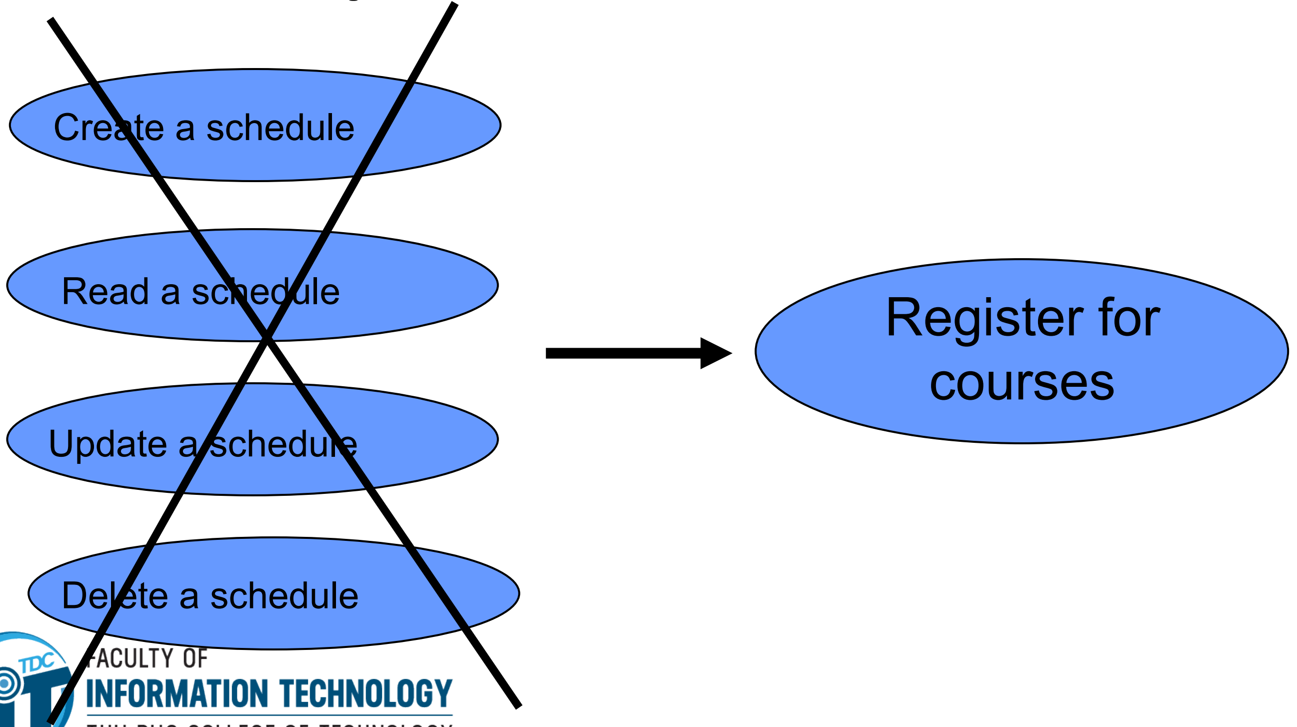
Is Log in a use case?

- By UML definition, log in is not a use case, because it does not produce results of value to an actor.
- However, in many cases, there is a need to capture log in separately because it:
  - Captures more and more complex behaviors (security, compliance, customer experience)
  - Is included in other use cases
- Recommendation: Make an exception and capture log in as a separate use case.



CRUD Use Cases

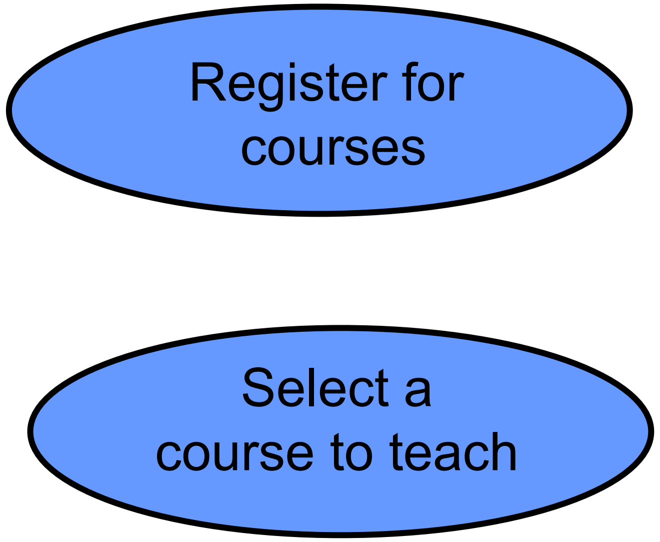
- A CRUD use case is a Create, Read, Update, or Delete use case
- Remove CRUD use cases **if they are data- management use cases that do not provide results that are of value to actors**



- **Do not confuse use cases with functions**
- **Focus on value**

Name the use case

- A use case name should:
  - Be unique, intuitive, and self-explanatory
  - Define clearly and unambiguously the observable result of value gained from the use case
  - Be from the perspective of the actor that triggers the use case
  - Describe the behavior that the use case supports
  - Start with a verb and use a simple verb-noun combination

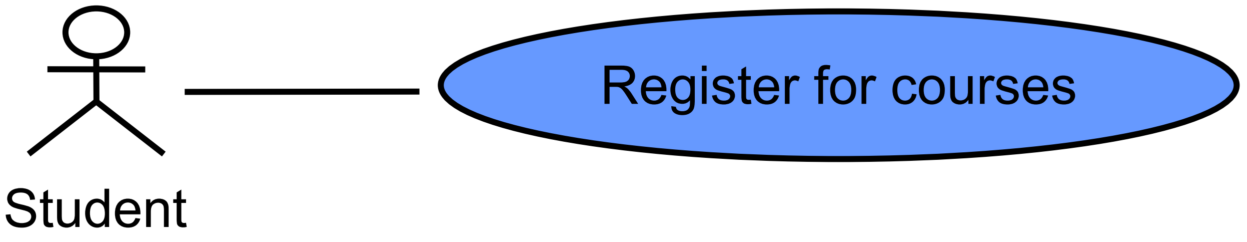


Guideline: Conduct a survey to learn whether customers, business representatives, analysts, and developers all understand the names and descriptions of the use cases

Describe a use case (text description)

Name	**Register for Courses**
Brief description	The student selects the courses they wish to attend to the next semester. A schedule of primary and alternate courses is produced.

Relationships with actors



Checkpoints for actors

- Have you found all of the actors?
- Have you accounted for and modeled all roles in the system's environment?
- Is each actor involved with at least one use case?
- Do any actors play similar roles in relation to the system? If so, merge them into a single actor.

Checkpoints for use cases

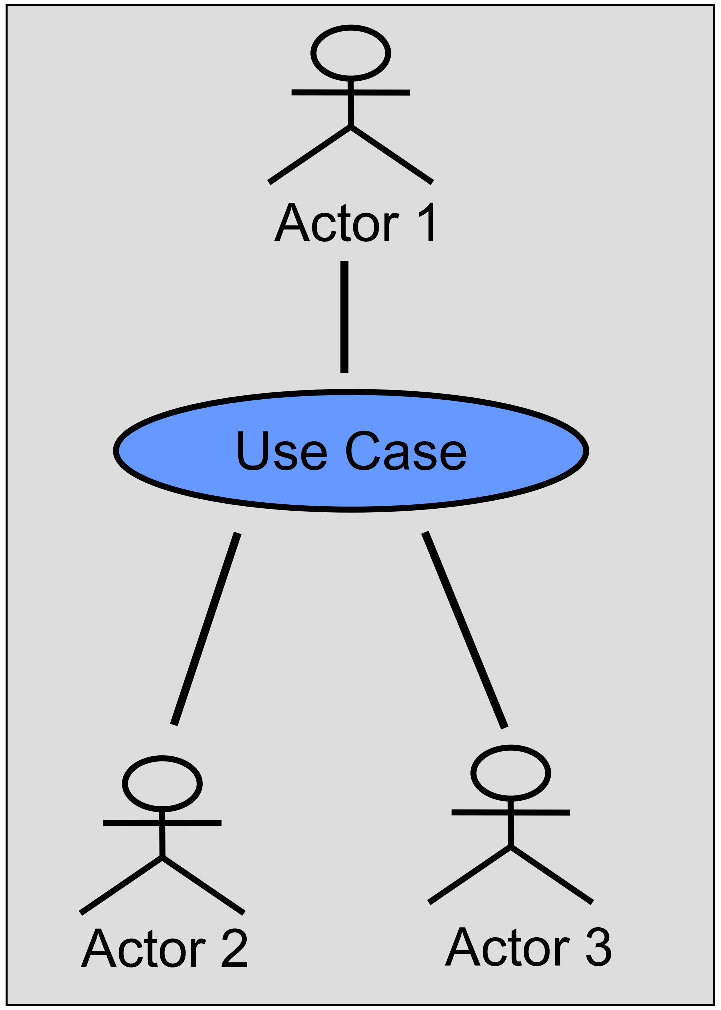
- The use-case model presents the behavior of the system; it is easy to understand what the system does by reviewing the model.
- All use cases have been identified; the use cases collectively account for all required behavior.
- The use-case model contains no superfluous behavior; all use cases can be justified by tracing them back to a functional requirement.

Checkpoints for use cases (cont.)

- The use cases have unique, intuitive, and explanatory names so that they cannot be confused at a later stage. If not, change their names.
- Customers and users understand the names and descriptions of the use cases.
- The brief description gives a true picture of the use case.
- Each use case is involved with at least one actor.
- No use cases have very similar behaviors or flows of events.

Use-case diagrams: communicates-association

- A channel of communication between an actor and a use case
- A line represents a communicates-association

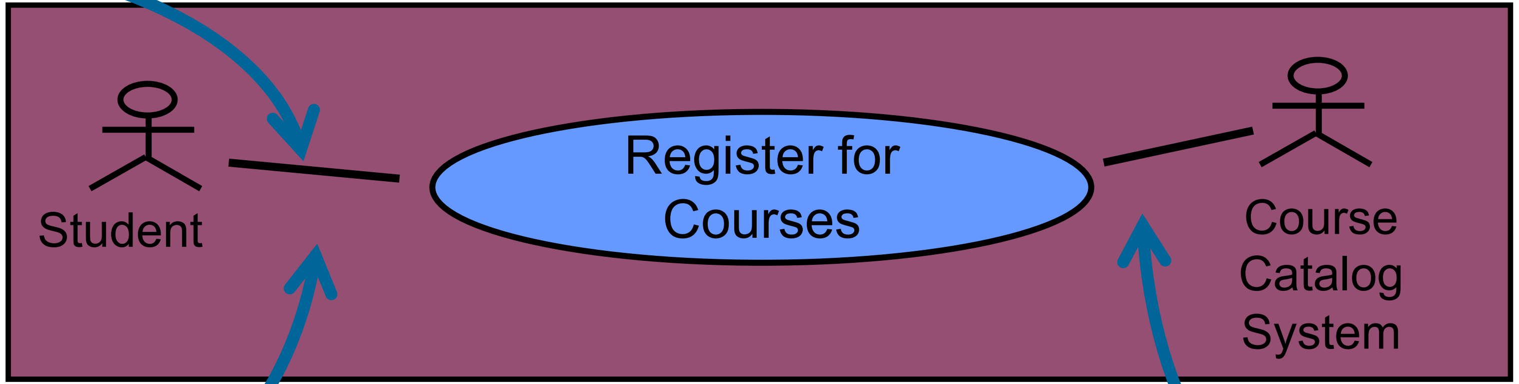


Each communicates-association is a whole dialog

Student logs on to system

System approves logon

Student requests course information



System displays course list

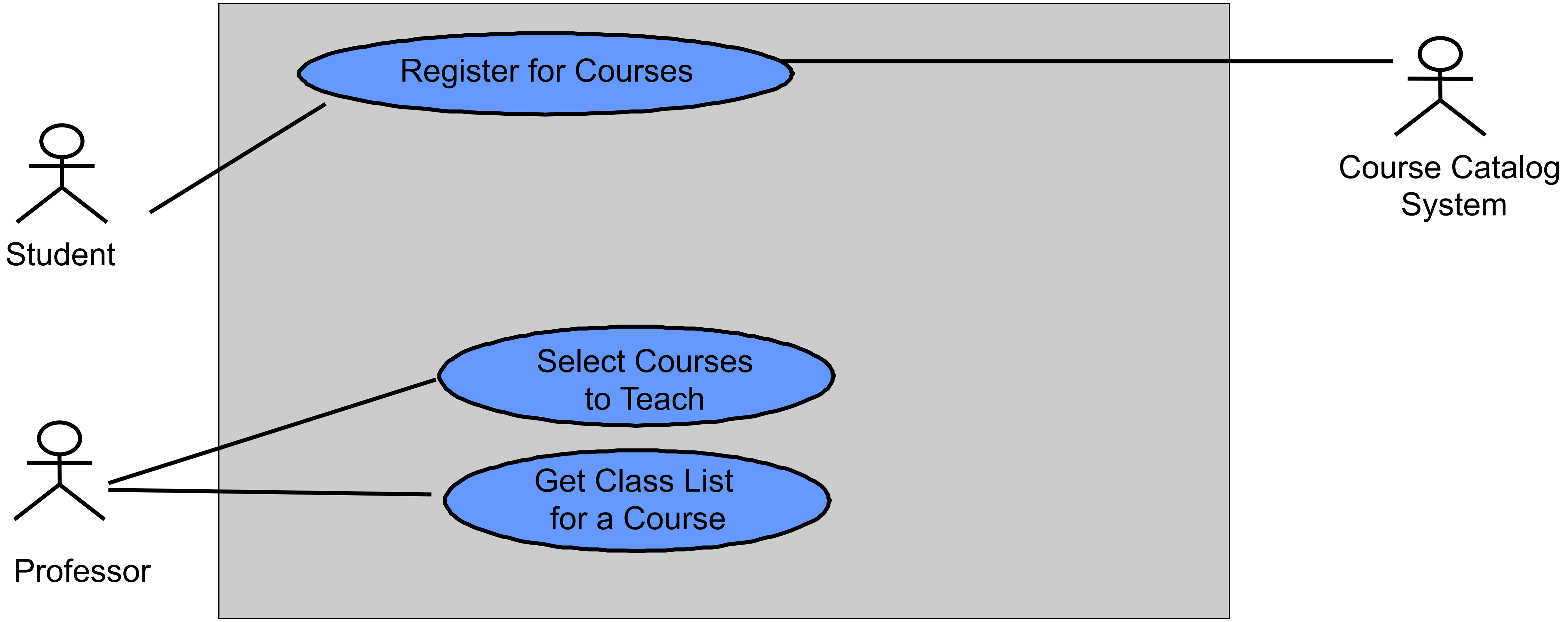
Student selects courses

System displays approved schedule

System transmits request

Course Catalog returns course information

Use-case diagram example



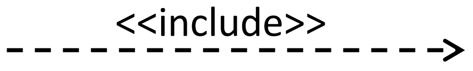
Review

- How do you find use cases?

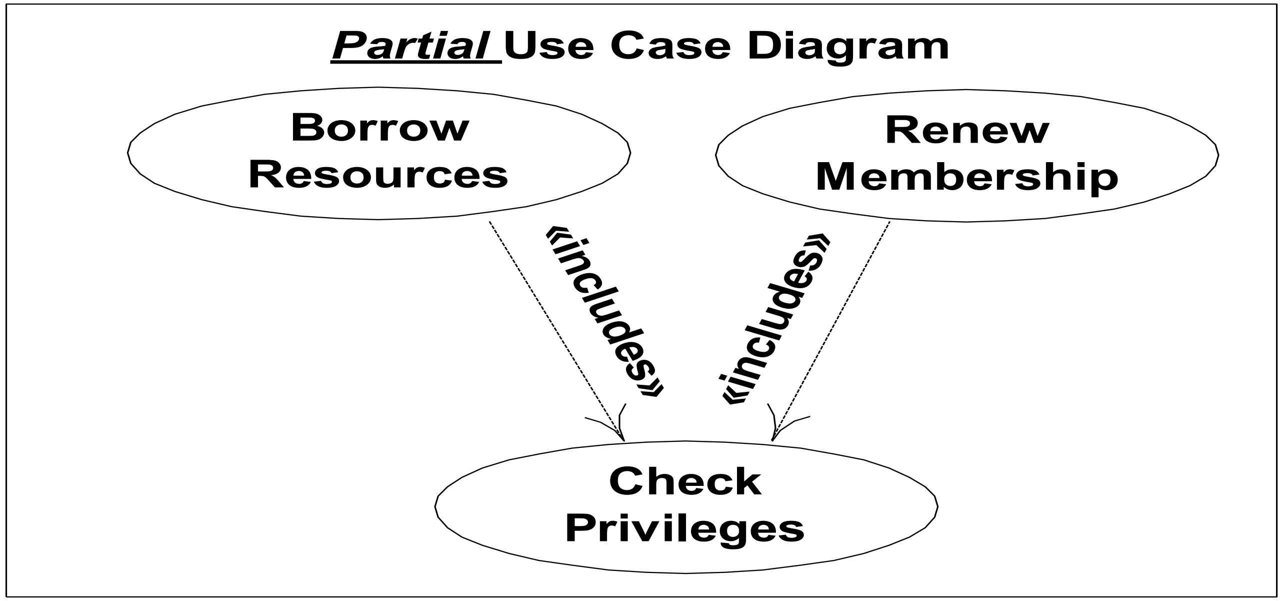
Relating Use Cases

- Use cases can be organized and related to simplify their descriptions.
- Three associations are:
  - Include.
  - Extend.
  - Generalization-specialization.

Include



- includes: “The insertion of additional behavior into a base use case that explicitly describes the insertion.”
  - Used to factor out a shared subprocess.



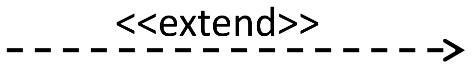
Include (continued)

- When a super-task use case (base use case) uses a subtask use case (an abstract use case).
  - An abstract use case is a factored-out subtask that does not stand on its own as a complete, separate process.
- In the use-case text, the base use case explicitly initiates the abstract use case.
  - "initiates" is a verb favored by some use case writers.
    - Use Case: Borrow Resources
    - ...
    - Step 5. Initiate use case Check Library Privileges
    - ...

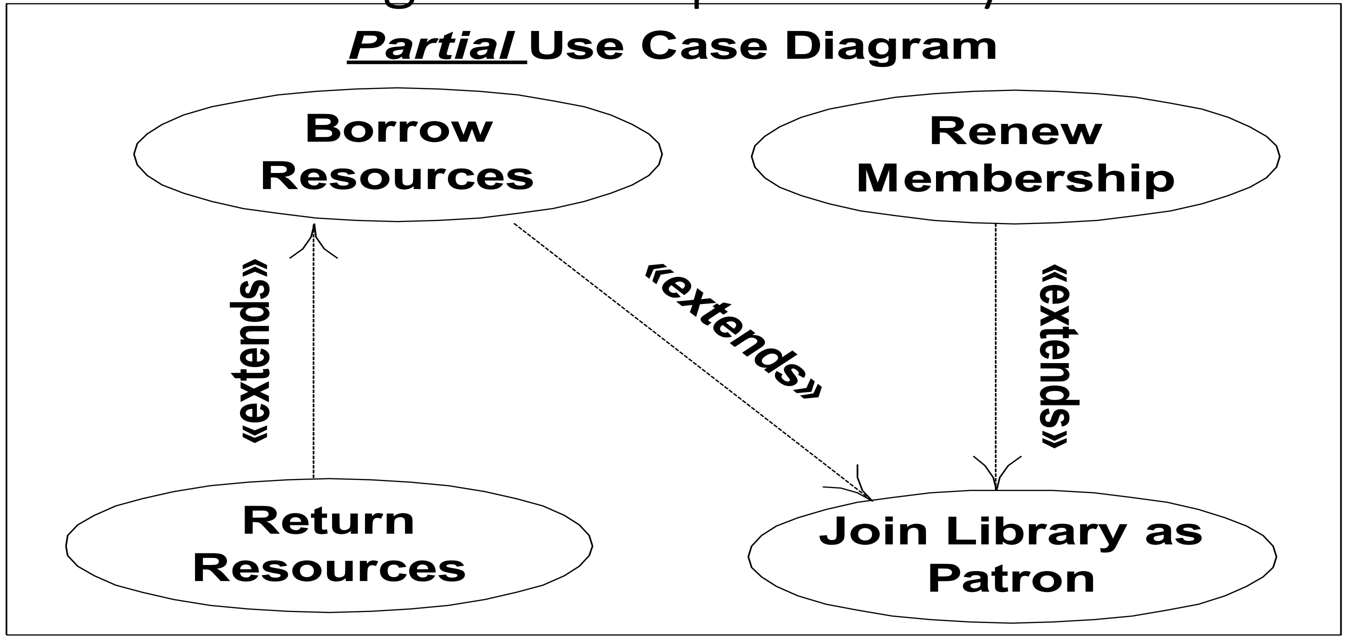
Include (continued)

- When to use?
  - When several use cases share a common subflow that can be factored out.
  - When a use case is too complex and needs factoring into smaller, understandable chunks.

Extend



- extends: "The insertion of additional behavior into a base use case that does not know about it."
- Used to extend an existing self-complete story.



Extend (continued)

- When a second use case extends the story of the first use case.
    - “Chapter 1 followed by Chapter 2.”
- The two use cases are complete on their own.
- The first use case does not know about or refer to the second use case.

Extend (continued)

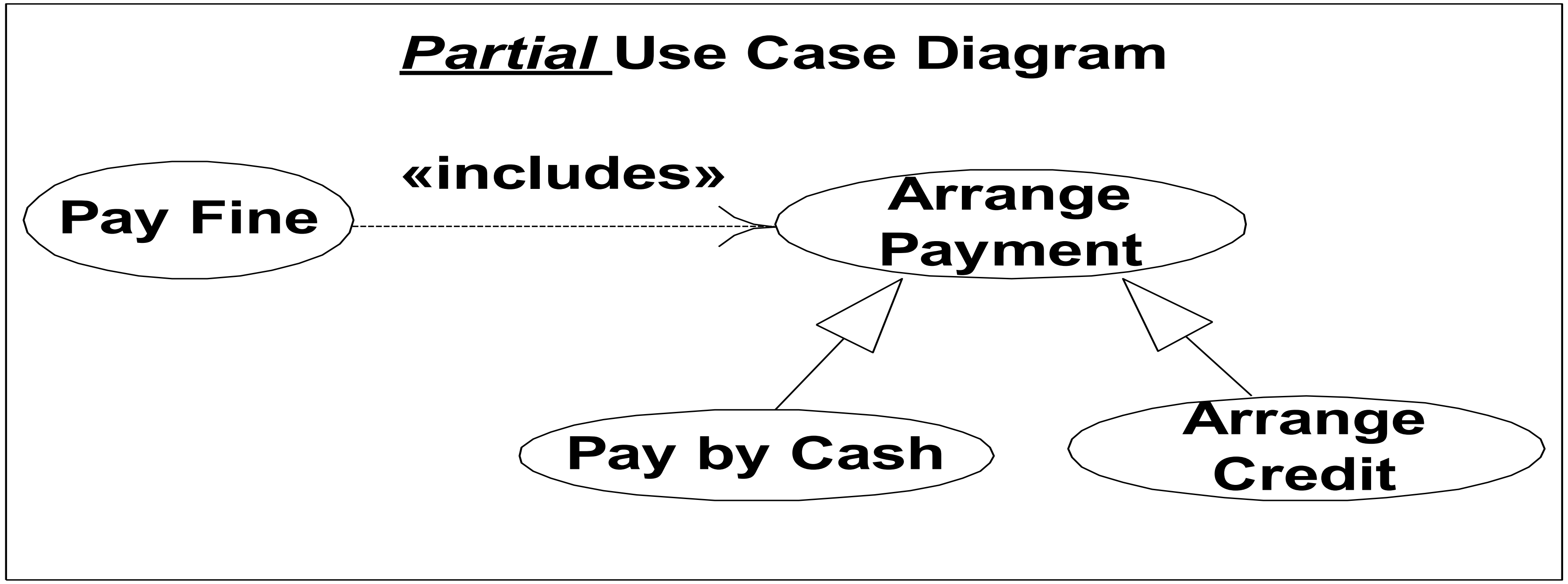
- The second use case can extend the first at its end or at any point - the extension point.
- The second use case might unconditionally extend the first or only under some condition.
- In the use case text of the second, it notes (in the Cross Reference section) that it extends the first.
    - Use Case: Return Resources
    - Cross-References: Extends use case Borrow Resources
    - …

Extend (continued)

- When to use?
  - To add new possible postflows to a complete use case.
  - To show a conditional or alternate subflow.

Generalization-Specialization $\longrightarrow\!\!\!\triangleright$

- "... to add a more specific use case that inherits and adds features to (a more general use case)."



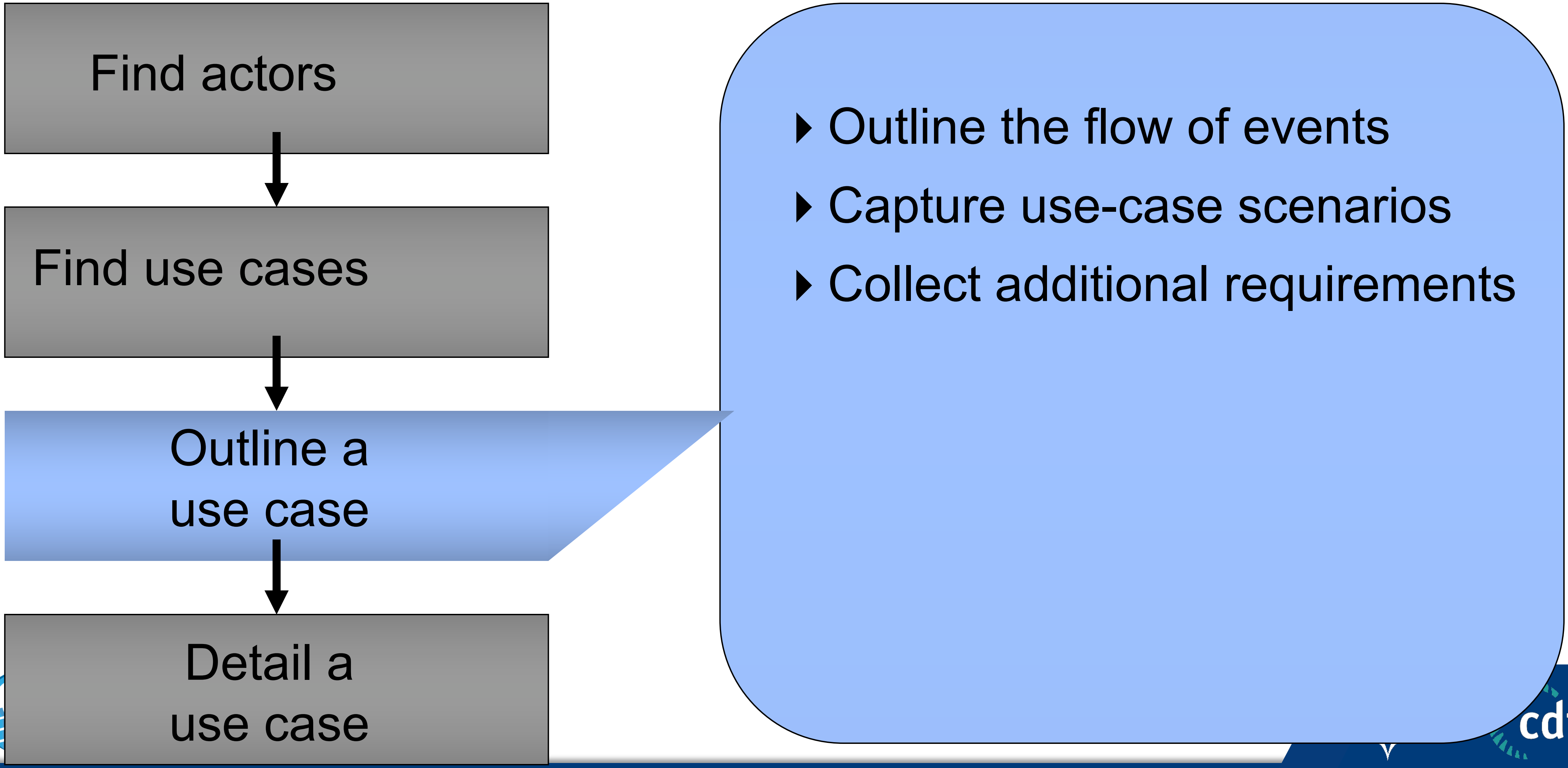
Generalization-Specialization (continued)

- Example:
  - Use Case: Pay Fine
  - ...
  - Step 3. Initiate use case Arrange Payment
  - …
  -
  - Use Case: Arrange Payment
  - Step 1. Collect payment
  - Step 2: Verify payment
  - Step 3: Record payment

Generalization-Specialization (continued)

- Use Case: Pay by Cash
- Cross References: Specializes Arrange Payment
- ...
- -
- Use Case: Arrange Credit
- Cross References: Specializes Arrange Payment
- ...

Process of writing use cases



- Outline the flow of events
- Capture use-case scenarios
- Collect additional requirements

Outline each use case

- An outline captures use case steps in short sentences, organized sequentially

<u>Use Case Name</u>

Brief Description

Basic Flow

1. First step
2. Second step
3. Third step

Alternative Flows

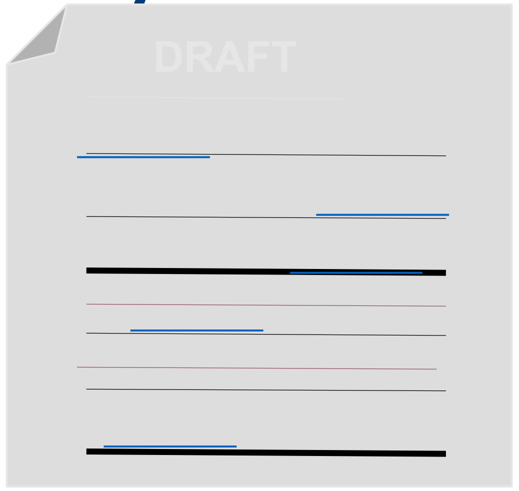
1. Alternative flow 1
2. Alternative flow 2
3. Alternative flow 3

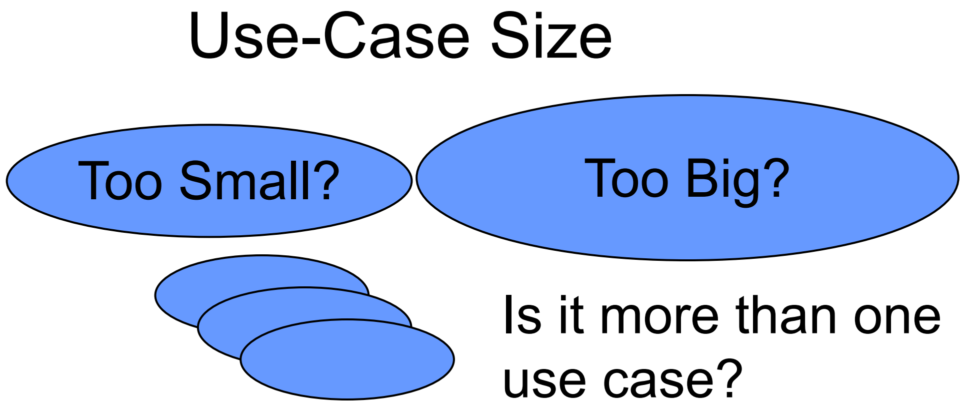
Number and name the steps

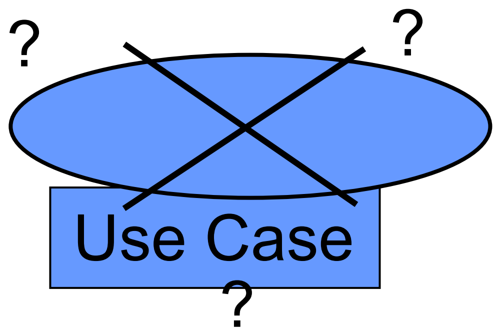
Structure the basic flow into steps

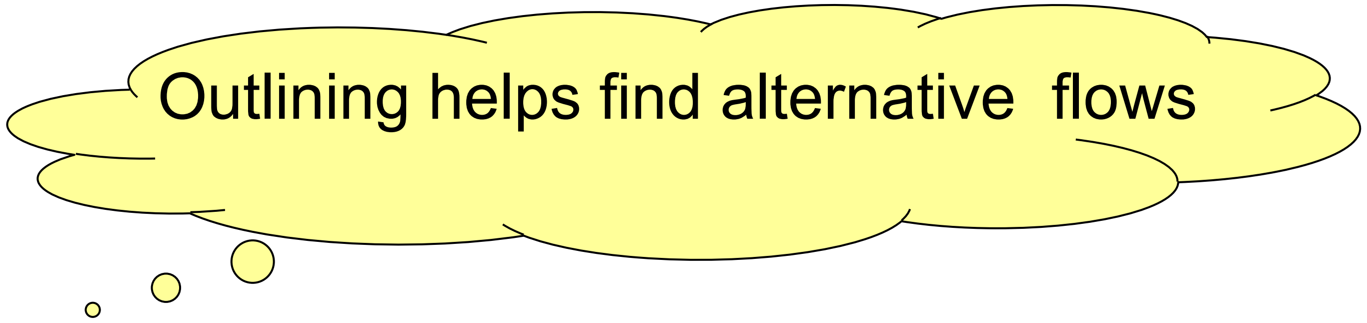
Identify alternative flows

Why outline use cases?



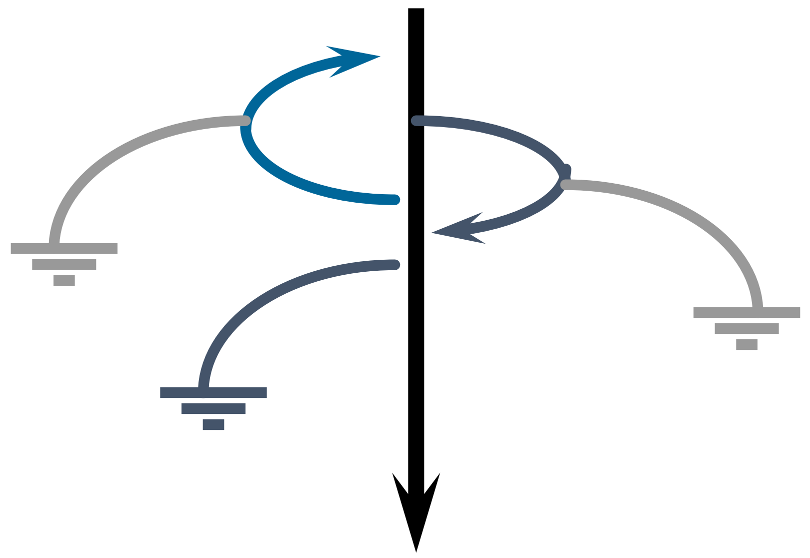






Flows of events (basic and alternative)

- A flow is a sequential set of steps
- One basic flow
  - Successful scenario from start to finish
- Many alternative flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows



Outline the flows of events

- Basic flow
  - What event starts the use case?
  - How does the use case end?
  - How does the use case repeat some behavior?
- Alternative flows
  - Are there optional situations in the use case?
  - What odd cases might happen?
  - What variants might happen?
  - What may go wrong?
  - What may not happen?
  - What kinds of resources can be blocked?

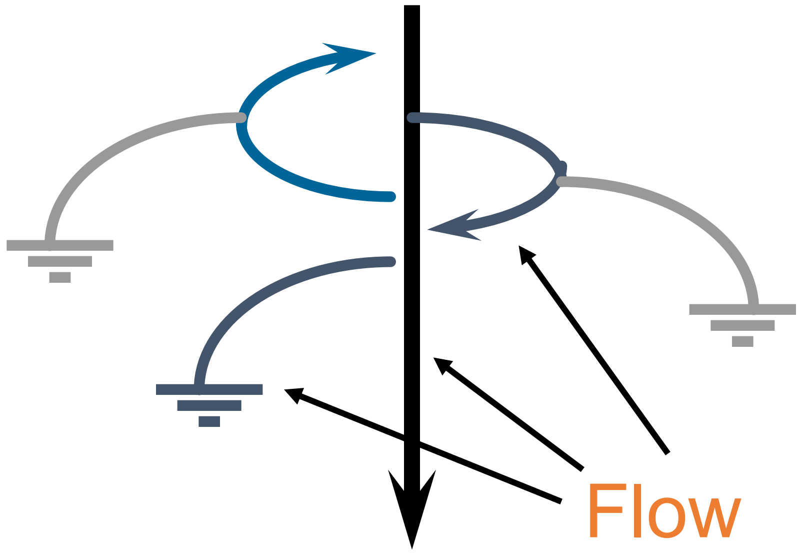
Step-by-step outline: Register for Courses

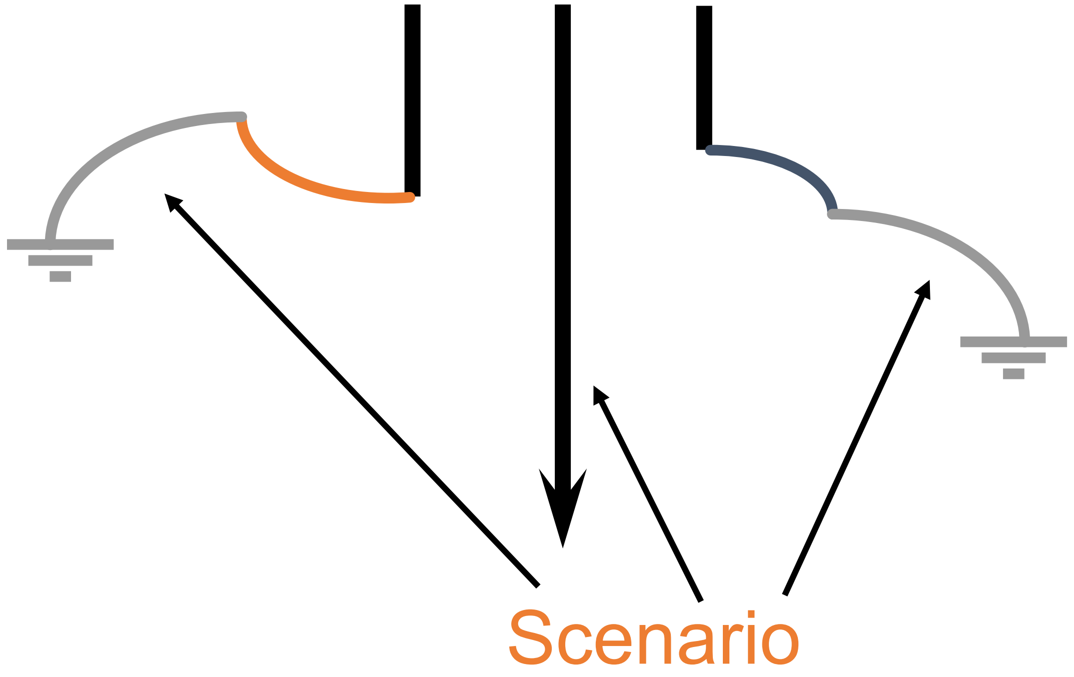
- Basic Flow
  - 1. Student logs on.
  - 2. Student chooses to create a schedule.
  - 3. Student obtains course information.
  - 4. Student selects courses.
  - 5. Student submits schedule.
  - 6. System displays completed schedule .
- Alternative Flows
  - A1. Unidentified student.
  - A2. Quit.
  - A3. Cannot enroll.
  - A4. Course Catalog System unavailable.
  - Can we allow students to register if the Course
  - Catalog is unavailable?
  - A5. Course registration closed.

What are other alternatives?

What is a use-case scenario?

- An instance of a use case
- An ordered set of flows from the start of a use case to one of its end points





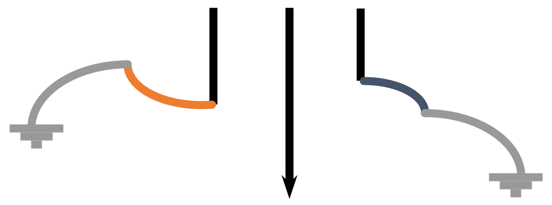
Note: This diagram illustrates only some of the possible scenarios based on the flows.

Why capture use-case scenarios?

- Help you identify, in concrete terms, what a system will do when a use case is performed
- Make excellent test cases
- Help with project planning
- Useful for analysis and design

How to capture use-case scenarios

- Capture scenarios in the Use-Case Specification in their own section
- Give each scenario a name
- List the name of each flow in the scenario
  - Place the flows in sequence
- Example:
- Use Case: Register for Courses
- Scenario: Quit before registering
  - Flows: Basic Flow, Quit



Outline: Register for Courses

Basic Flow of Events

1. Student logs on.
2. Student chooses to create a schedule.
3. Student obtains course information.
4. Student selects courses.
5. Student submits schedule.
6. System displays completed schedule.

Alternative Flows

A1. Unidentified student.
A2. Quit.
A3. Cannot enroll.
A4. Course Catalog System unavailable.
A5. Course registration closed.
…

Scenarios

1. Register for courses: Basic Flow
2. Unidentified Student: Basic Flow, Unidentified Student
3. Quit before registering: Basic Flow, Quit

…

What are other scenarios?

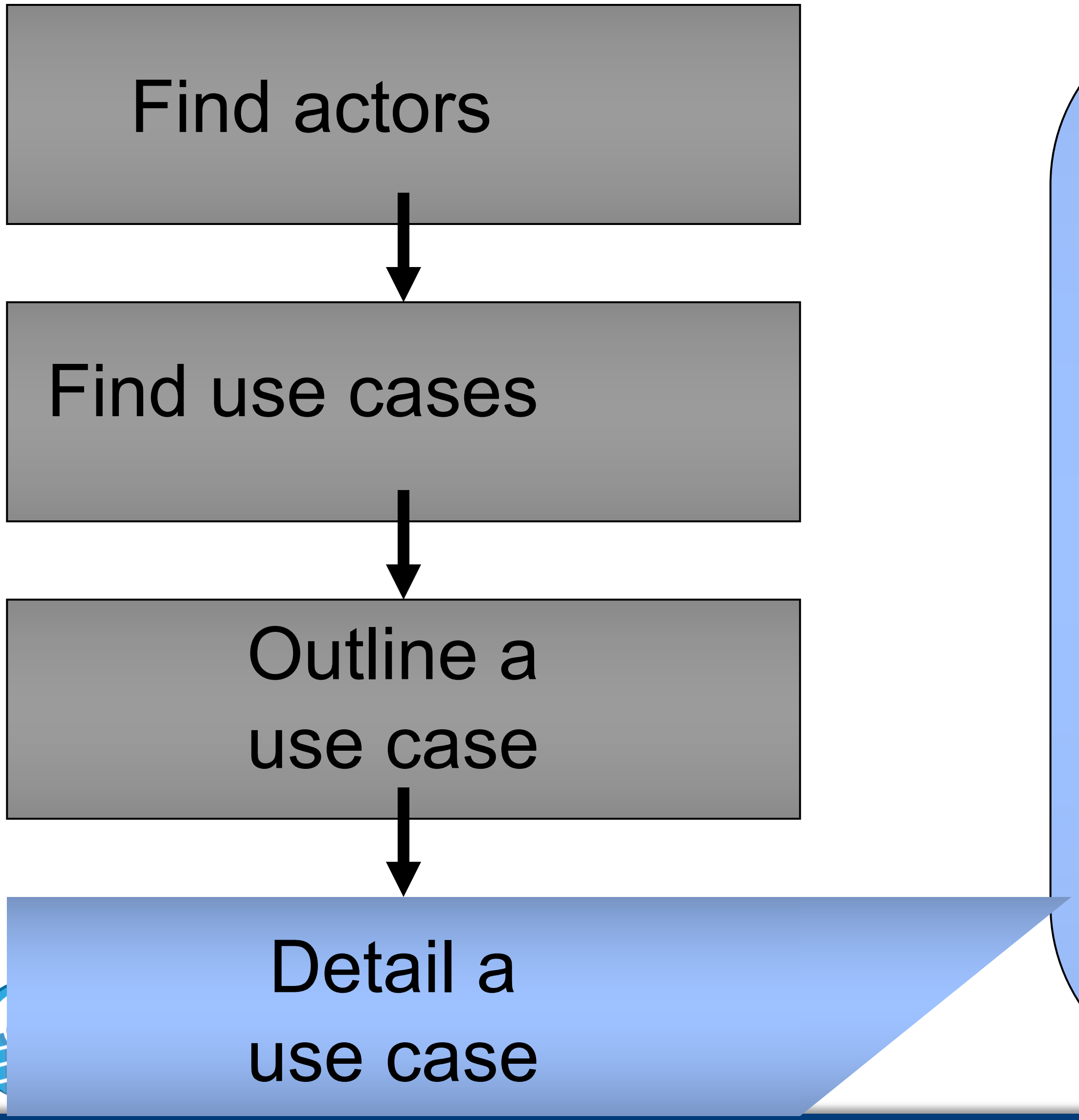
Checkpoints for use cases

- Each use case is independent of the others
- No use cases have very similar behaviors or flows of events
- No part of the flow of events has already been modeled as another use case

Review

- What is the basic flow?
- What is an alternative flow?
- What is a scenario?
- Why do you capture use-case scenarios?
- Where do you collect requirements other than use cases?

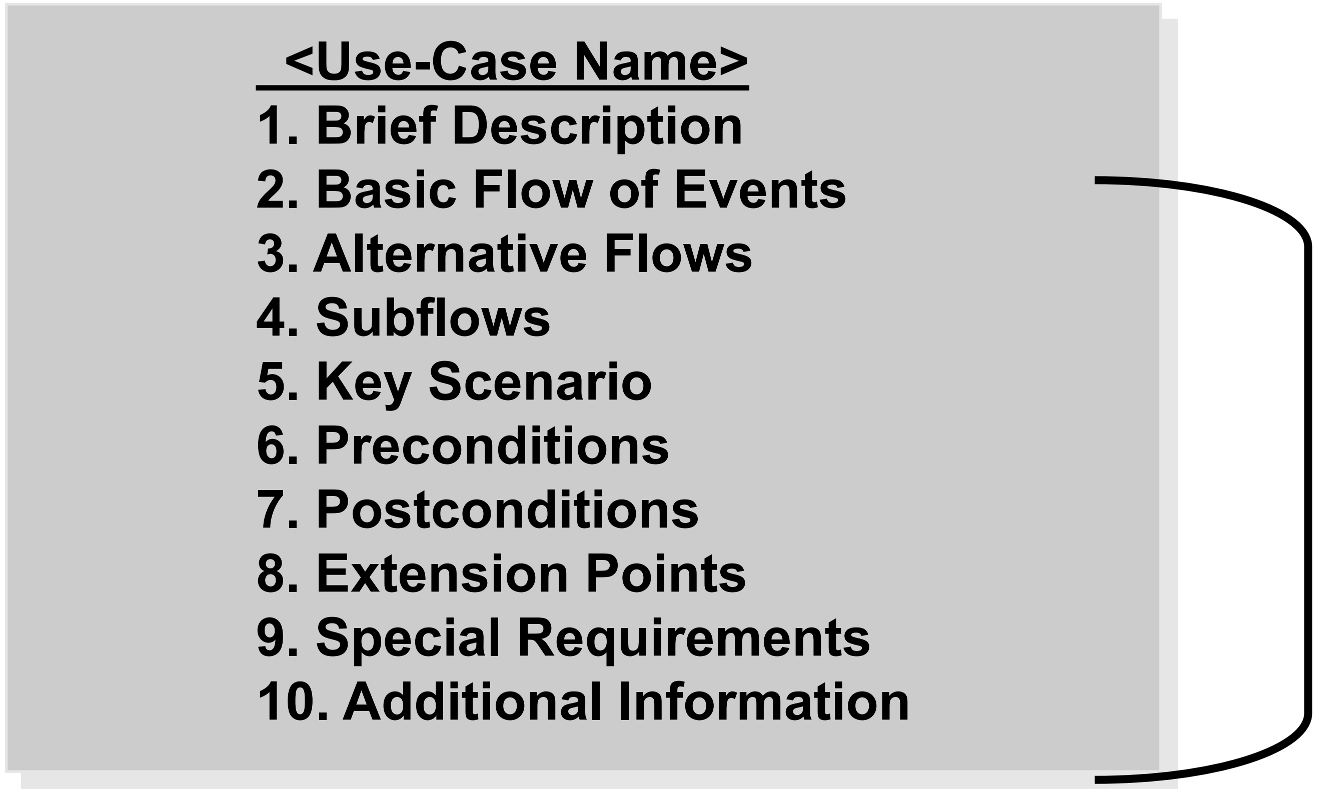
Process of writing use cases



- Detail the flow of events
- Structure the flow of events
- Specify additional use case properties

Detail a use case

You found actors and use cases, then outlined the use cases. Next, you add detail.



<Use-Case Name>

1. Brief Description
2. Basic Flow of Events
3. Alternative Flows
4. Subflows
5. Key Scenario
6. Preconditions
7. Postconditions
8. Extension Points
9. Special Requirements
10. Additional Information

Detail the basic flow of events

Structure the flow into steps

Number and title each step

Describe the steps

Register for Courses

1.1 Basic Flow

1. **Log On**.

This use case starts when someone accesses the Course Registration System and chooses to register for courses. The system validates that the person accessing the system is an authorized student.

2. **Select "Create a Schedule "**.

The system displays the functions available to the student. The student selects "Create a Schedule ".

3. **Obtain Course Information**.

The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student .The student can search the list by department, professor, or topic to obtain the desired course information .

4. **Select Courses**.

The student selects four primary course offerings and two alternate course offerings from the list of available offerings course offerings.

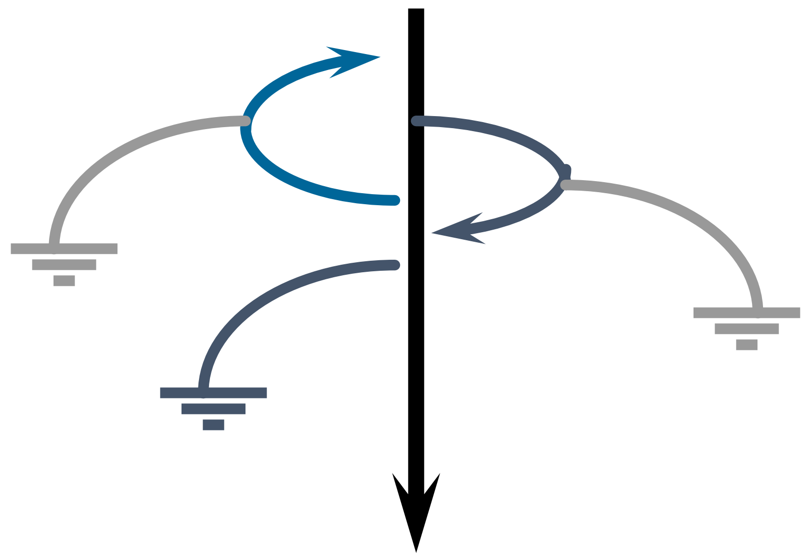
…

Phrasing of steps

- Use the active voice
  - Say: “The Professor provides the grades for each student”
  - Instead of: “When the Professor has provided the grades”
- Say what triggers the step
  - Say: “The use case starts when the Professor chooses to submit grades”
  - Instead of: “The use case starts when the Professor decides to submit grades ”.
- Say who is doing what (use the Actor name)
  - Say: “The Student chooses …”
  - Instead of: "The user chooses …"
  - Say: “The System validates …”
  - Instead of: "The choice is validated …"

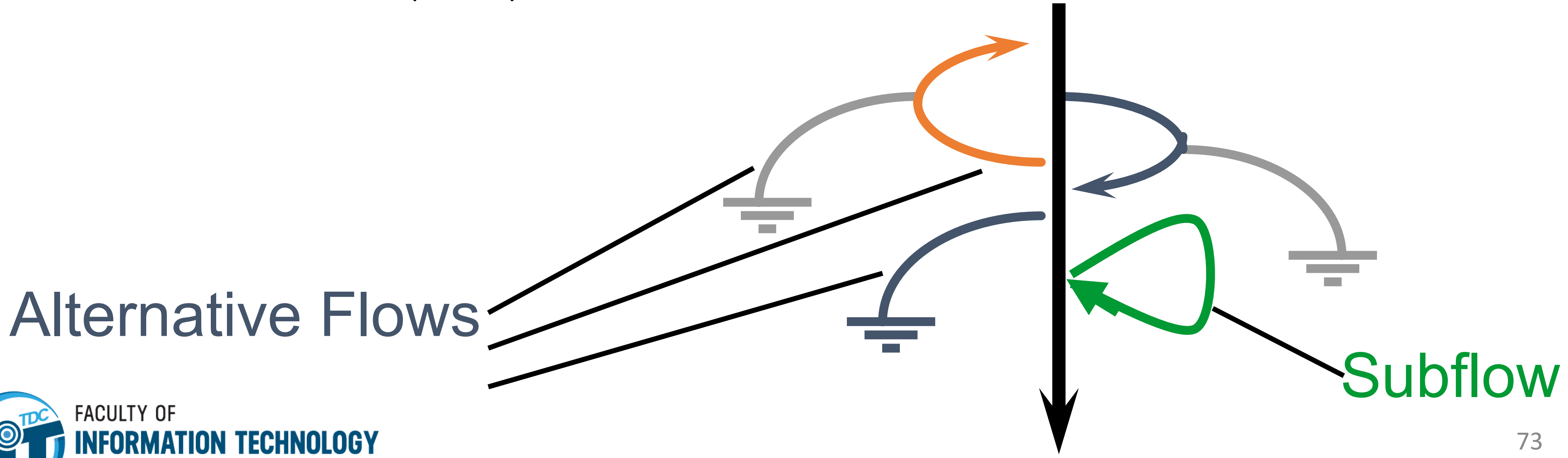
Review: Flows of events (basic and alternative)

- One basic flow
  - Happy day scenario
  - Successful scenario from start to finish
- Many alternative flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows

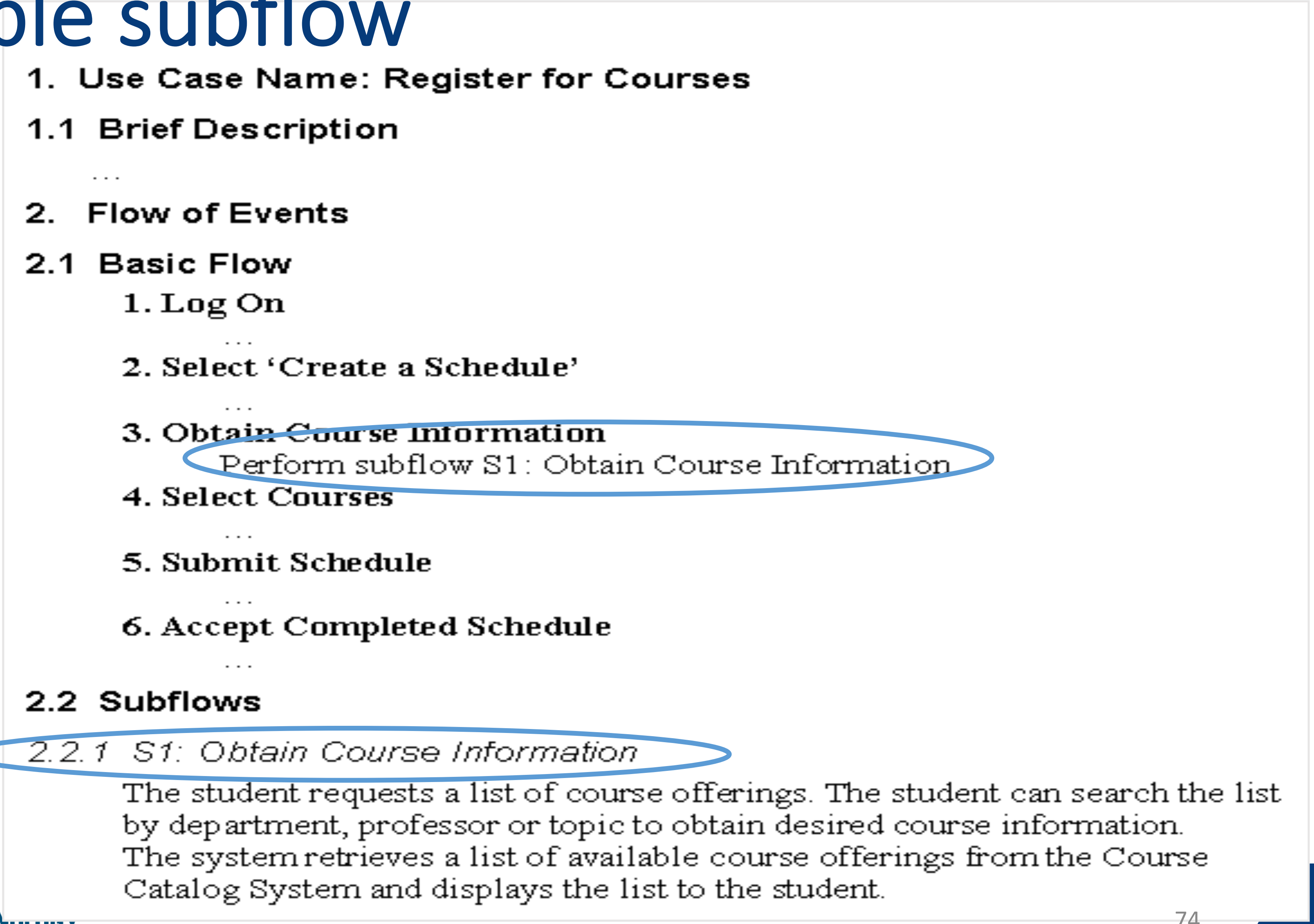


Subflows

- If flows become unwieldy, break individual sections into self-contained subflows
- Subflows
  - Increase clarity
  - Allow internal reuse of requirements
  - Always return to the line after they were called
  - Are called explicitly, unlike alternative flows



Example subflow



1. Use Case Name: Register for Courses

1.1 Brief Description

...

2. Flow of Events

2.1 Basic Flow

1. Log On

...

2. Select 'Create a Schedule'

...

3. Obtain Course Information

Perform subflow S1: Obtain Course Information

4. Select Courses

...

5. Submit Schedule

...

6. Accept Completed Schedule

...

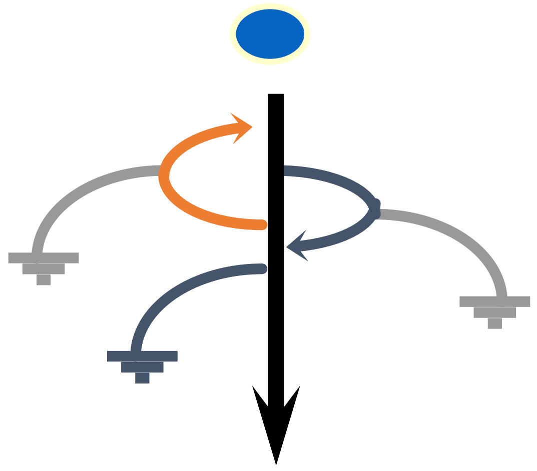
2.2 Subflows

2.2.1 S1: Obtain Course Information

The student requests a list of course offerings. The student can search the list by department, professor or topic to obtain desired course information. The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student.

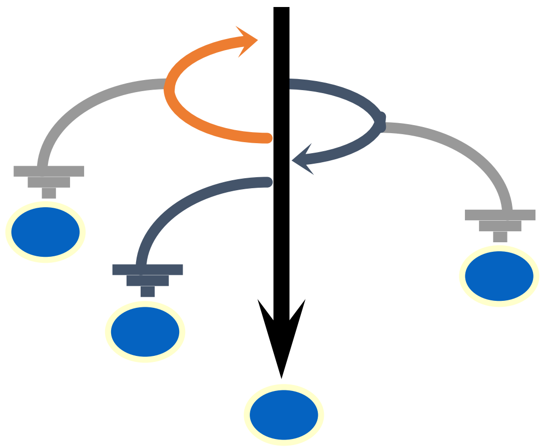
Preconditions

- Describe the state that the system must be in before the use case can start
  - Simple statements that define the state of the system, expressed as conditions that must be true
  - Should never refer to other use cases that need to be performed prior to this use case
  - Should be stated clearly and should be easily verifiable
- Optional: Use only if needed for clarification
- Example
  - Register for Courses use case
  - Precondition:
  - The list of course offerings for the semester has been created and is available to the Course Registration System
  - Student has logged into the Course Registration System

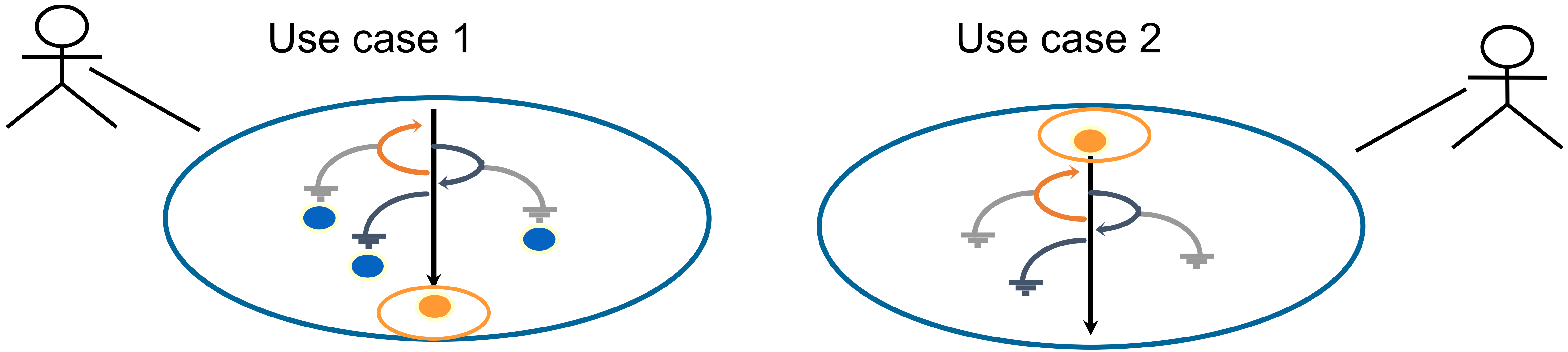


Postconditions

- Describe the state of the system at the end of the use case
  - Use when the system state is a precondition to another use case, or when the possible use case outcomes are not obvious to use case readers
  - Should never refer to other, subsequent use cases
  - Should be stated clearly and should be easily verifiable
- Optional: Use only if needed for clarification
- Example:
  - Register for Courses use case
  - Postcondition: At the end of this use case either the student has been enrolled in courses, or registering was unsuccessful and no changes have been made to the student schedules or course enrollments



Sequence use cases with pre- and postconditions



Use cases do **not** interact with each other.
However, a postcondition for one use case
can be the same as the precondition for another.

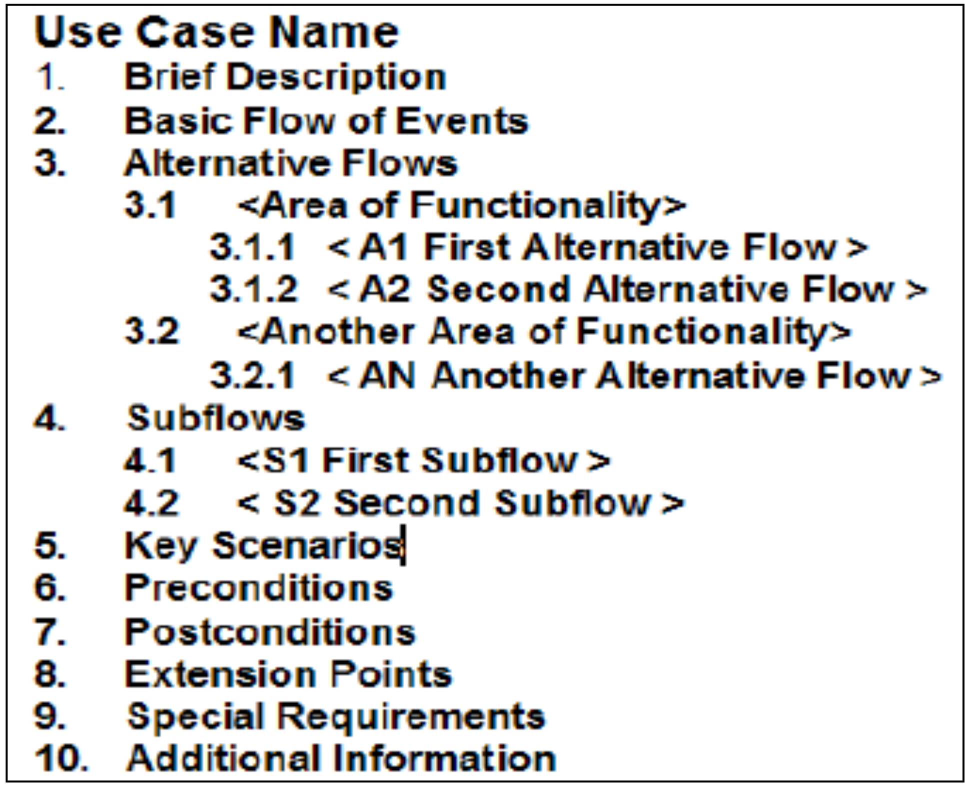
Other use case properties

- Special requirements
  - Related to this use case, not covered in flow of events
  - Usually nonfunctional requirements, data, and business rules
- Extension points
  - Name a set of places in the flow of events where extending behavior can be inserted
- Additional information
  - Any additional information required to clarify the use case

RUP style summary

- Basic flow
  - Steps are numbered and named
  - Steps do not reference alternative flows
  - Shows the main actor succeeding in that actor's main goal
- Alternative flows
  - Have names
  - May have steps

RUP Use-Case Specification Template



Use Case Name

1. Brief Description
2. Basic Flow of Events
3. Alternative Flows
   - 3.1 <Area of Functionality>
     - 3.1.1 < A1 First Alternative Flow >
     - 3.1.2 < A2 Second Alternative Flow >
   - 3.2 <Another Area of Functionality>
     - 3.2.1 < AN Another Alternative Flow >
4. Subflows
   - 4.1 <S1 First Subflow >
   - 4.2 < S2 Second Subflow >
5. Key Scenarios
6. Preconditions
7. Postconditions
8. Extension Points
9. Special Requirements
10. Additional Information

FACULTY OF INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Use case checkpoints

- The actor interactions and exchanged information is clear
- The communication sequence between actor and use case conforms to the user's expectations
- How and when the use case's flow of events starts and ends is clear
- The subflow in a use case is modeled accurately
- The basic flow achieves an observable result for one or more actors

Review

- What are the steps to detailing a use case?
- Give a few examples of best practices in phrasing use case steps?
- What is a subflow, and when should you use one?
- What are pre- and postconditions, and when should you use them?

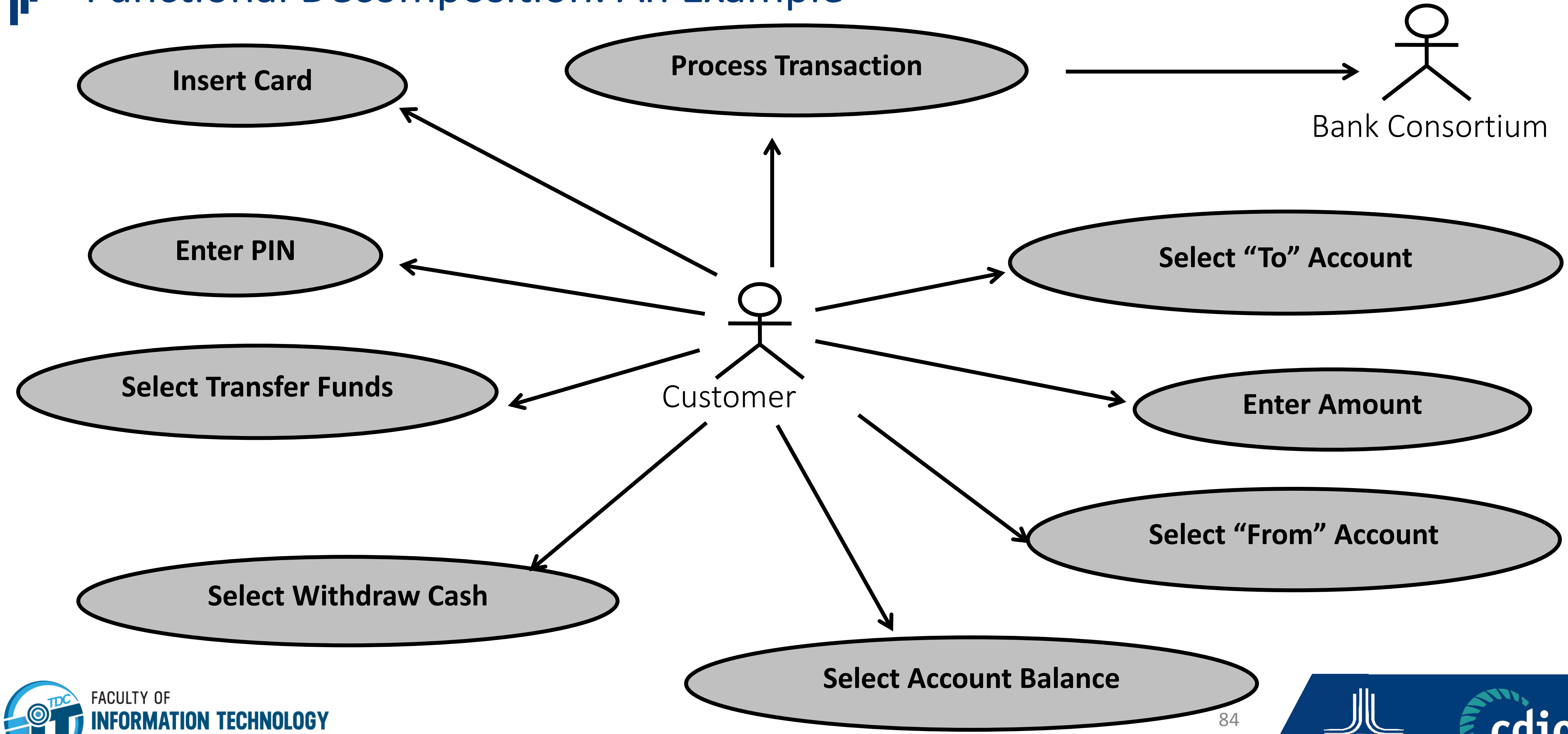
What is NOT a use case

- Functional decomposition
- User interface specification
- System design specifications

Functional Decomposition

- Is breaking down a problem into small, isolated parts.
  - The parts work together to provide the functionality of the system.
    - Often do not make sense in isolation.
- Use cases:
  - Are NOT functional decomposition.
  - Keep the functionality together to describe a complete use of the system.
  - Provide context.

Functional Decomposition: An Example



Avoid Functional Decomposition

- Symptoms
  - Very small use cases
  - Too many use cases
  - Uses cases with no result of value
  - Names with low-level operations
    - "Operation" + "object"
    - "Function" + "data"
    - Example: "Insert Card"
  - Difficulty understanding the overall model

- Corrective Actions
  - Search for larger context
    - "Why are you building this system?"
  - Put yourself in user's role
    - "What does the user want to achieve?"
    - "Whose goal does this use case satisfy?"
    - "What value does this use case add?"
    - "What is the story behind this use case?"

Thanks for your attention!

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Phone:	(+848) 22 158 642
Email:	fit@tdc.edu.vn
Website:	fit.tdc.edu.vn
Facebook:	facebook.com/tdc.fit
Youtube:	youtube.com/fit-tdc