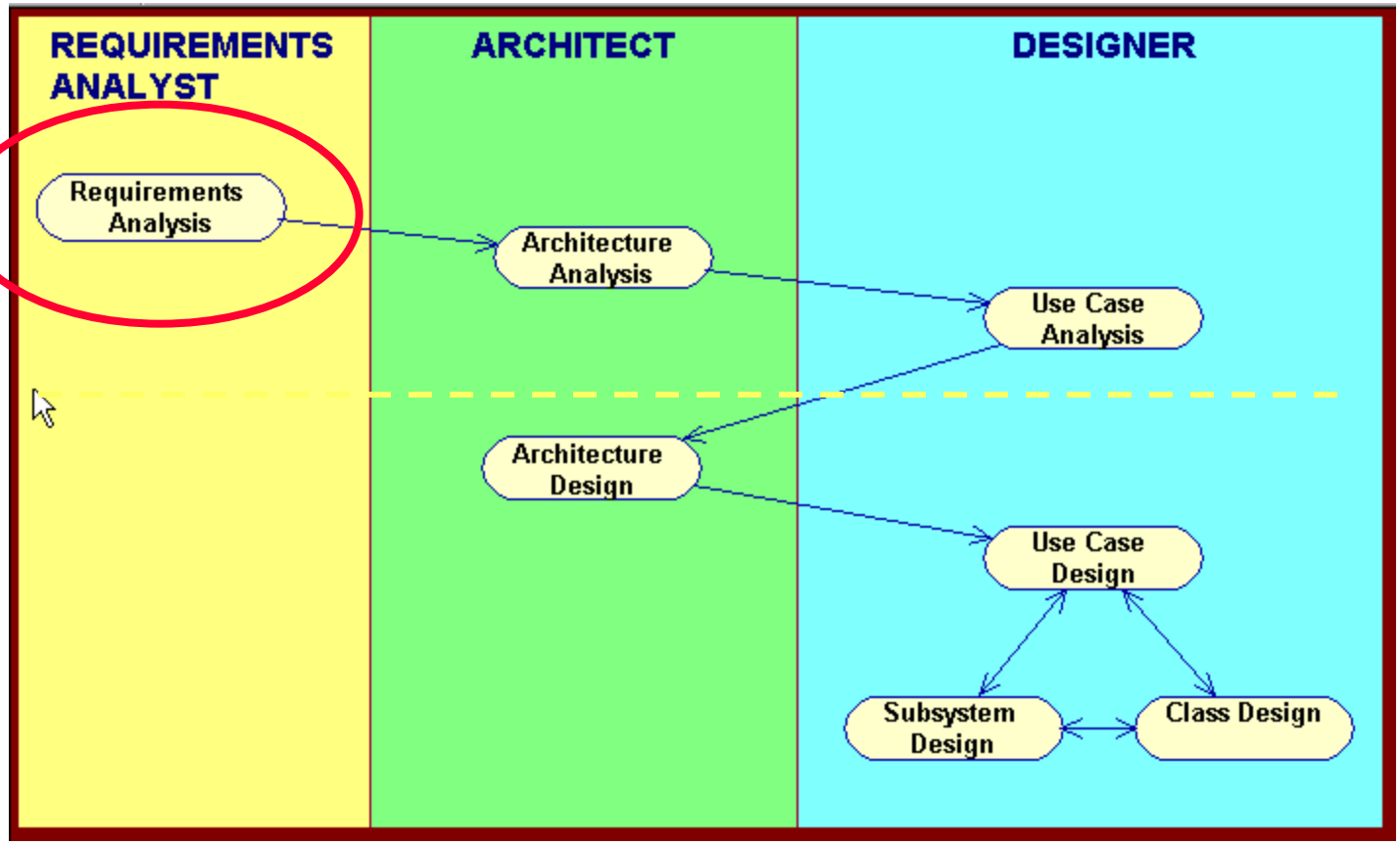
Lesson 5

The Use Case Approach To Software Requirements:
Unity At The Basis Of Diversity

Basic RUP OOAD Activities



Module Main Points

Theme: Use cases are a widely adopted approach to capturing functional system requirements.

1. The main elements of a RUP Software Requirements Specification:
   - use case models,
   - supplementary specifications,
   - glossary.

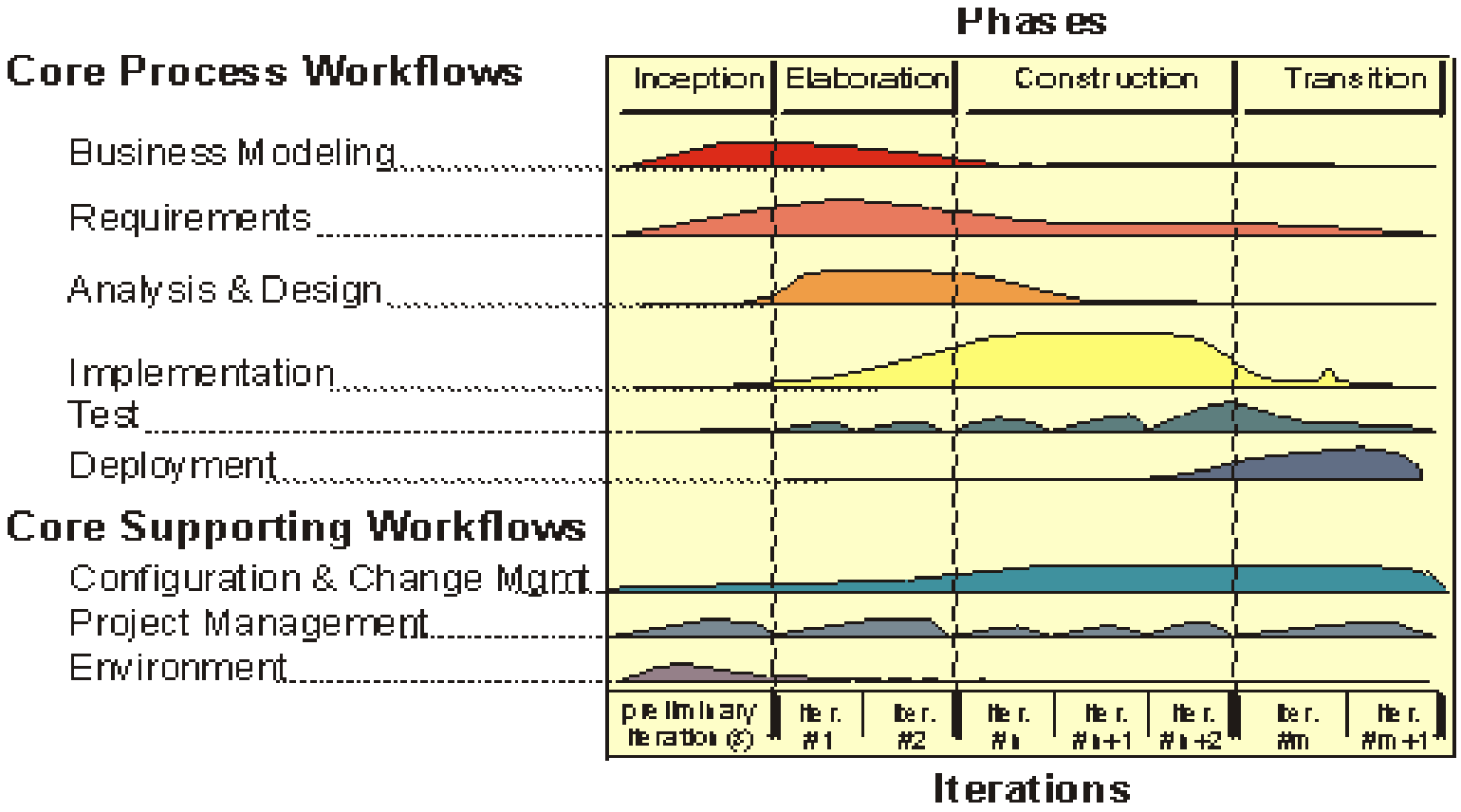
2. A use case is a sequence of actions performed by an actor interacting with the system to achieve a goal, showing how the goal might succeed or fail to be reached

3. Use cases are described in terms of flows and scenarios. A scenario is a single path through a use case. A flow is a set of scenarios that result in the same sort of outcome (e.g., success vs failure flows).

Requirements Analysis Objectives

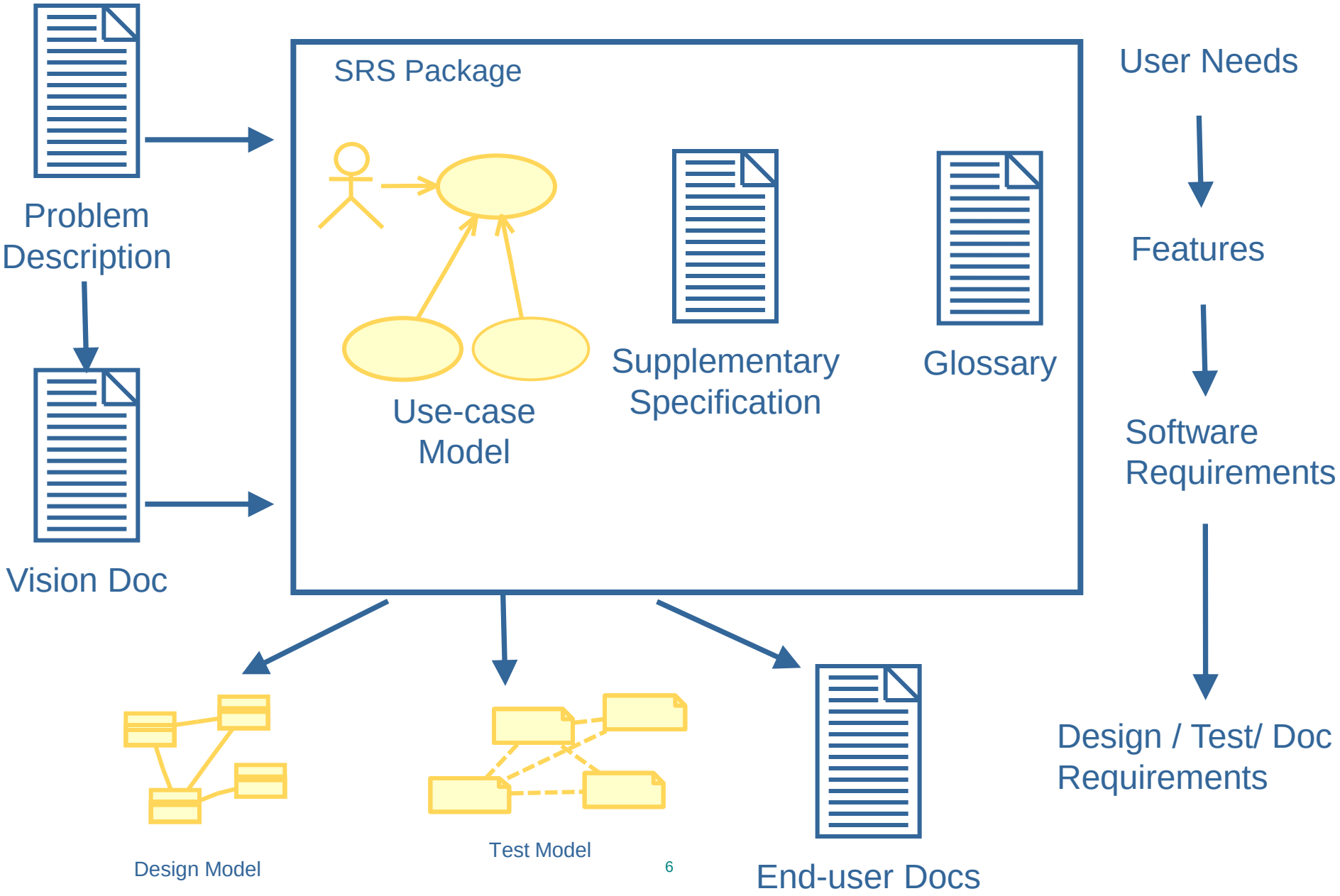
- Understand the content and purpose of the components of a modern Software Requirements Specification
  - Use cases
  - Supplementary Specifications
  - Requirements Glossary
- Understand how use cases are developed, structured, and used
- Understand how use cases integrate and drive the entire software lifecycle

Role of Requirements Analysis in Life Cycle



- Inception: identify most use cases, detail critical ones
- Elaboration: complete about 80% of use cases

Software Requirements Specification



Better Approach to Requirements

- Previous requirements documents often arbitrary collection of paragraphs
  - Poor fit with both business reengineering and implementation
- State requirements in terms of how end-users use the system
  - Easier for end-users to understand and relate to

Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

What Is a Use Case?

- A way in which a user interacts with a system in order to achieve some goal
- Goal and the use case can often be used interchangeably
- Example: Withdraw money use case
  - Bank Customer identifies himself or herself, and system verifies
  - Bank Customer chooses from which account to withdraw money and specifies how much to withdraw, and system deducts the amount from the account and dispenses the money

Summary: Use Cases Drive Overall Development

- Use cases form the basis of a modern SRS document. Organizing the requirements document around user goals and actions facilitates user understanding of the requirements and the eventual ***validity and acceptance testing*** of the finished system.

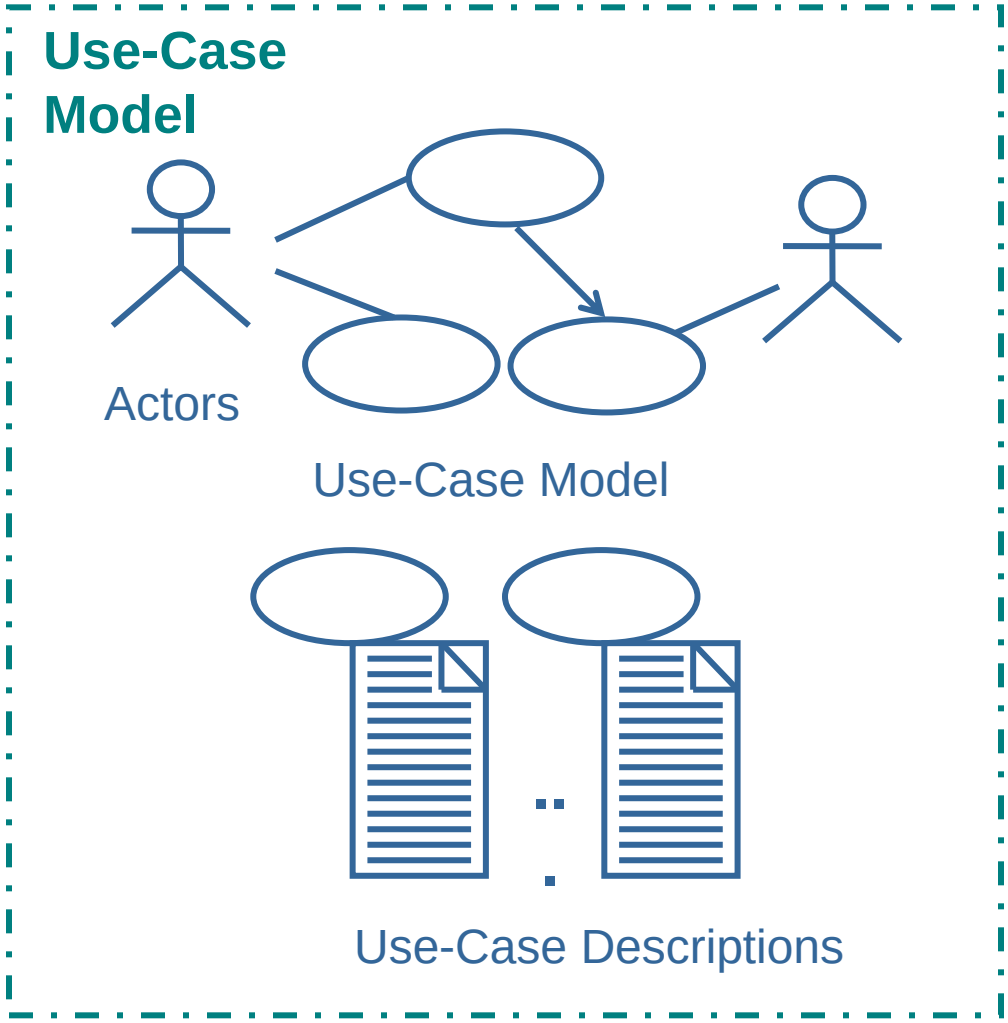
Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

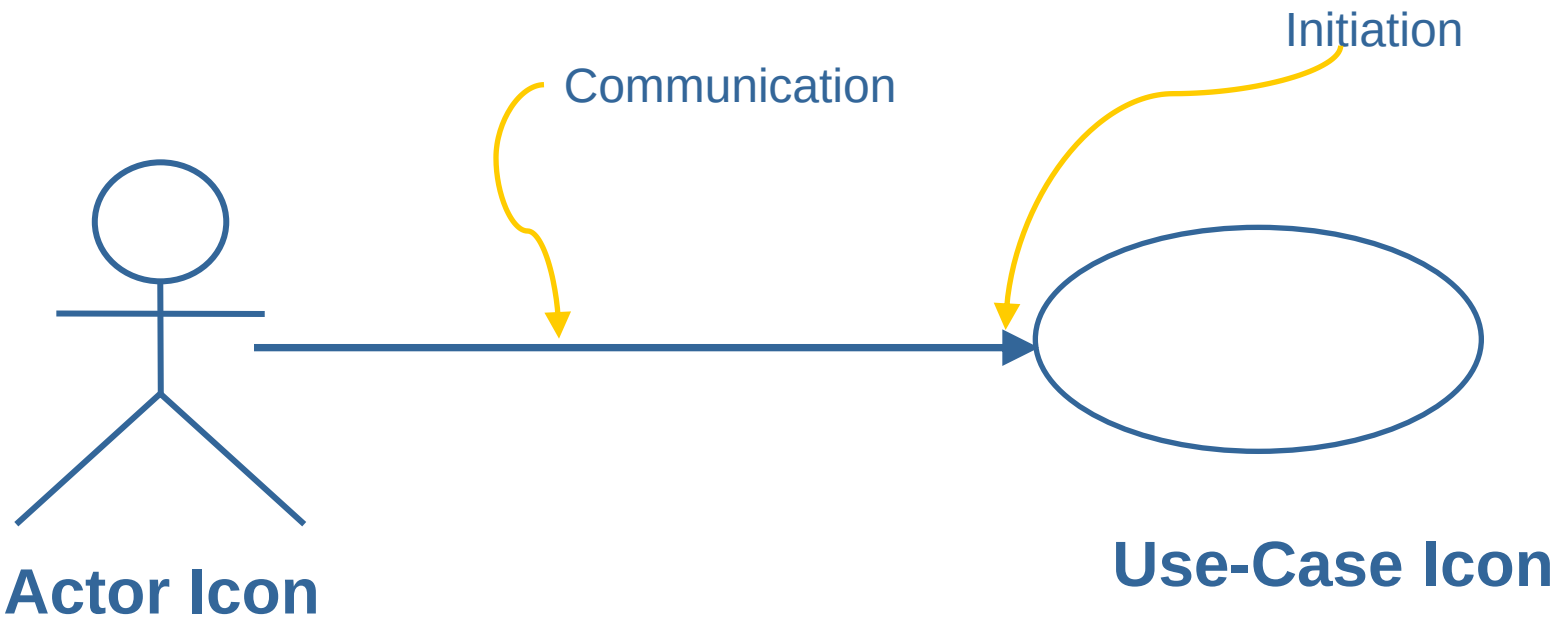
Use-Case Model = Diagrams and Descriptions

- The use-case model is composed of :

1. use-case diagram(s)
2. use-case descriptions

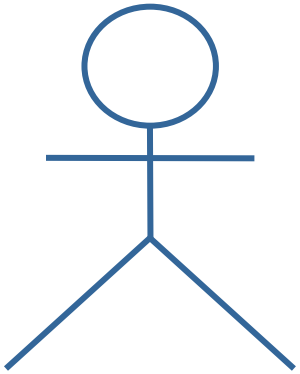


Use Case Diagram Basic Components



Use-Case Modeling: Actors

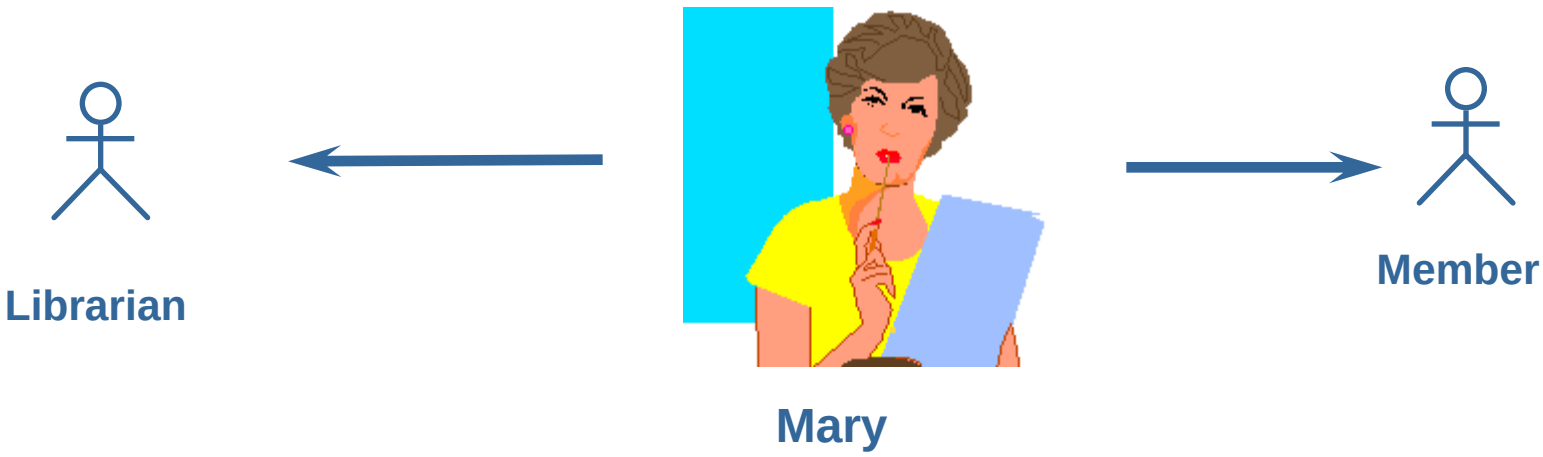
- The use case model starts in the Inception Phase with the identification of actors and principal use-cases.
- Actors are not part of the system--they represent anyone or anything that must interact with the system



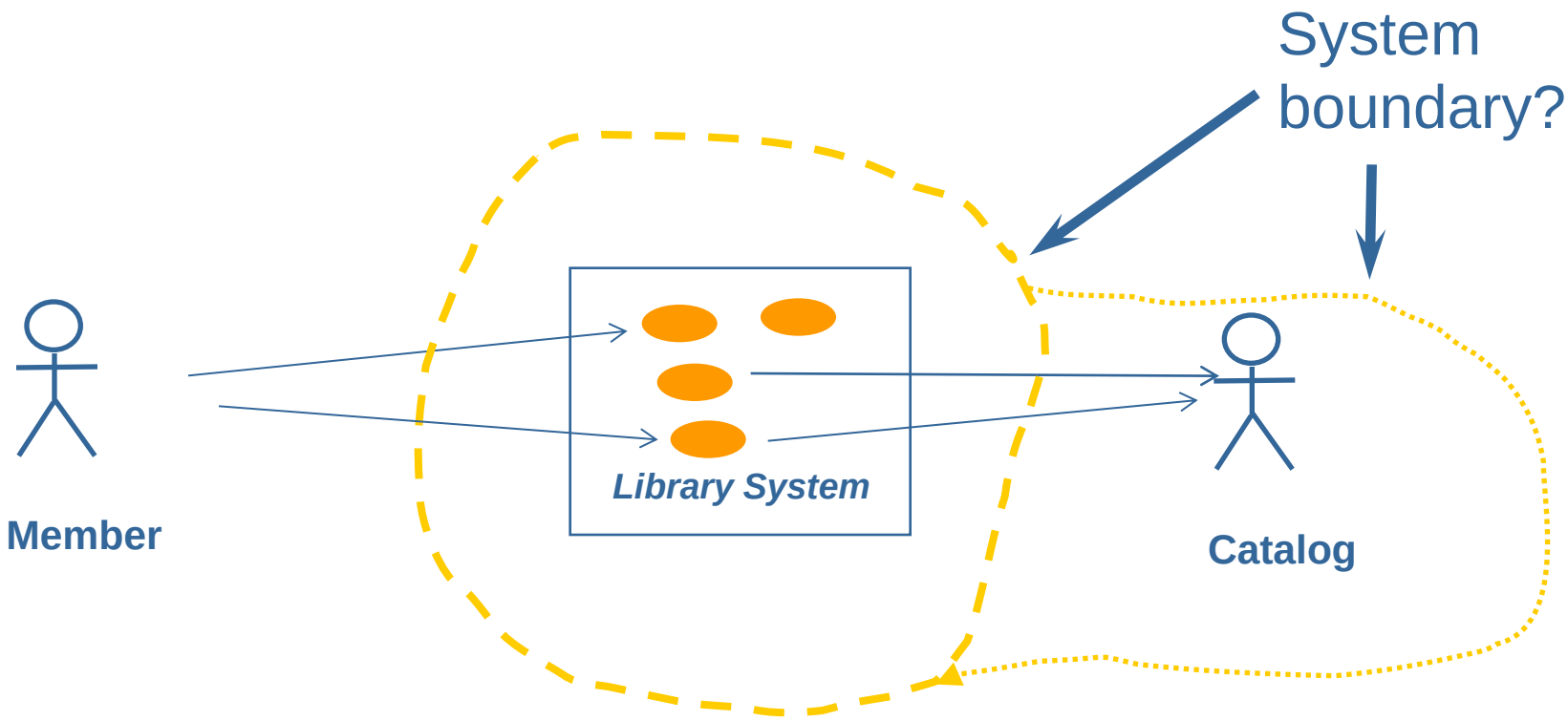
Actor Icon

A User May Have Different Roles

- The same person might appear as different actors
- Does the person use the system in a significantly different manner in the different roles?



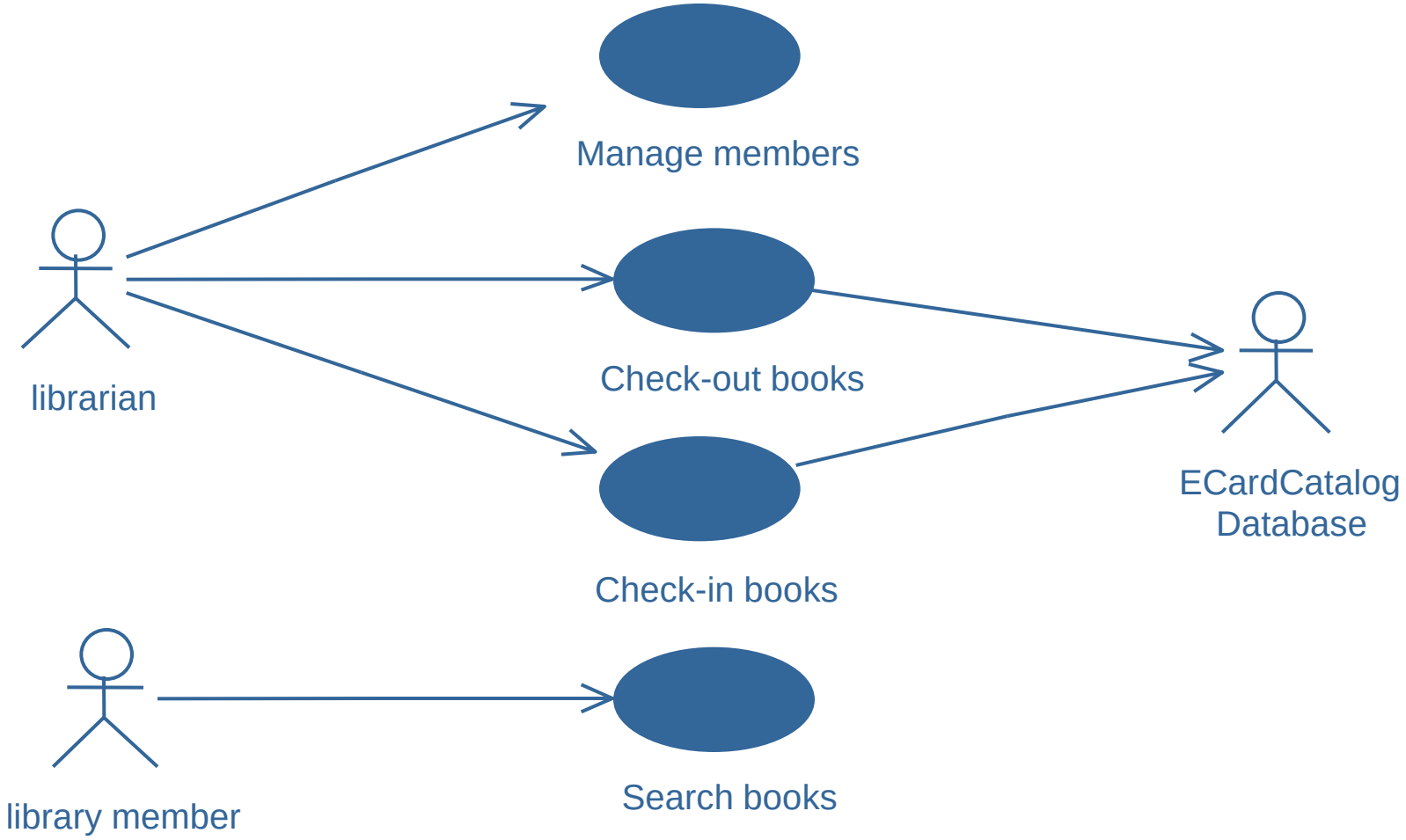
Actors are External to the System



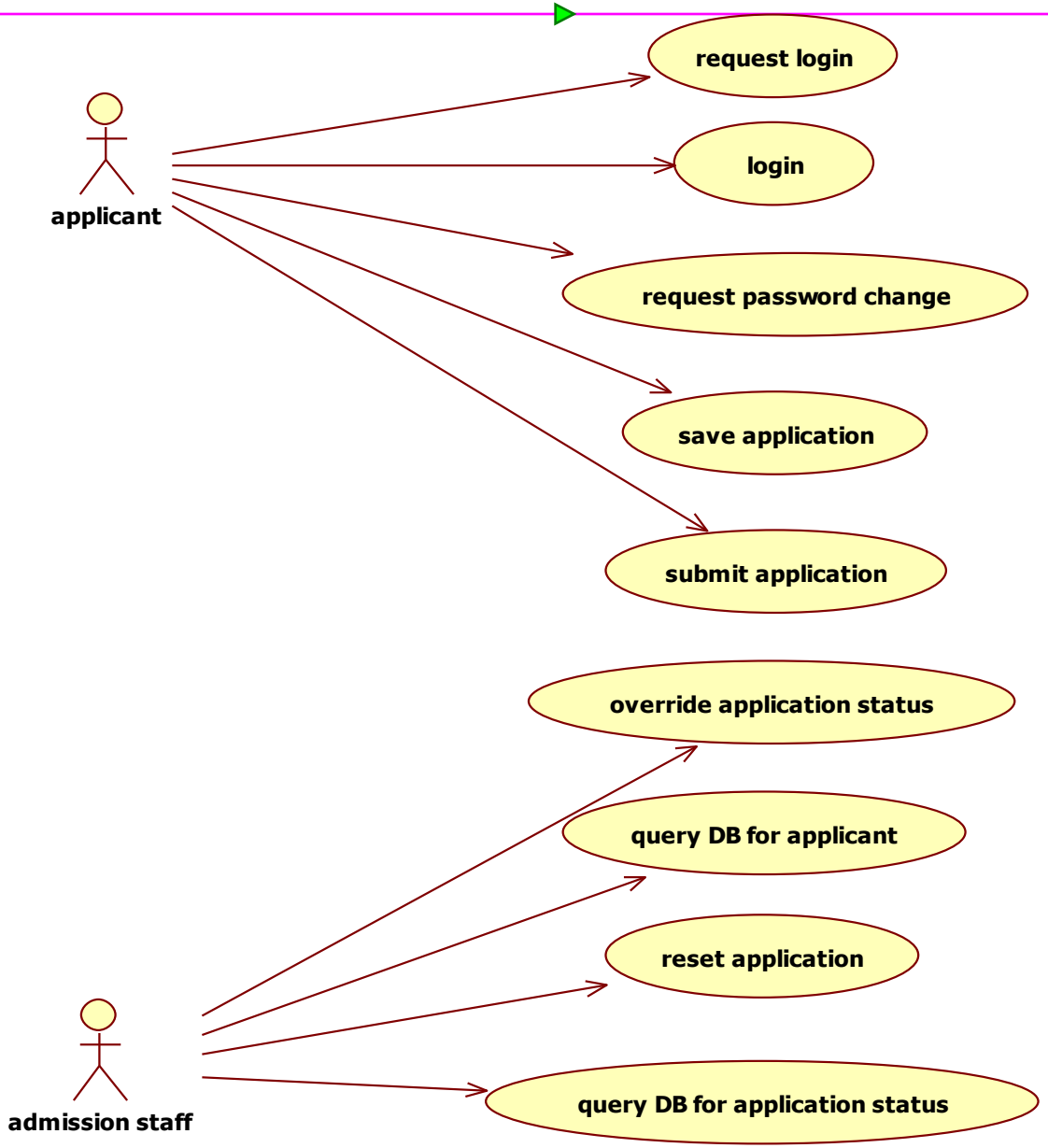
Example: List of Actors and Use-Cases

- Actors
  - Library Member
  - ECardCatalog
  - Librarian
- Use cases
  - Search book
  - Check-in books
  - Check-out books
  - Manage members

Example: Use-Case Model: Use-Case Diagram



Example: Use-Case Diagram for an application system



Group Exercise

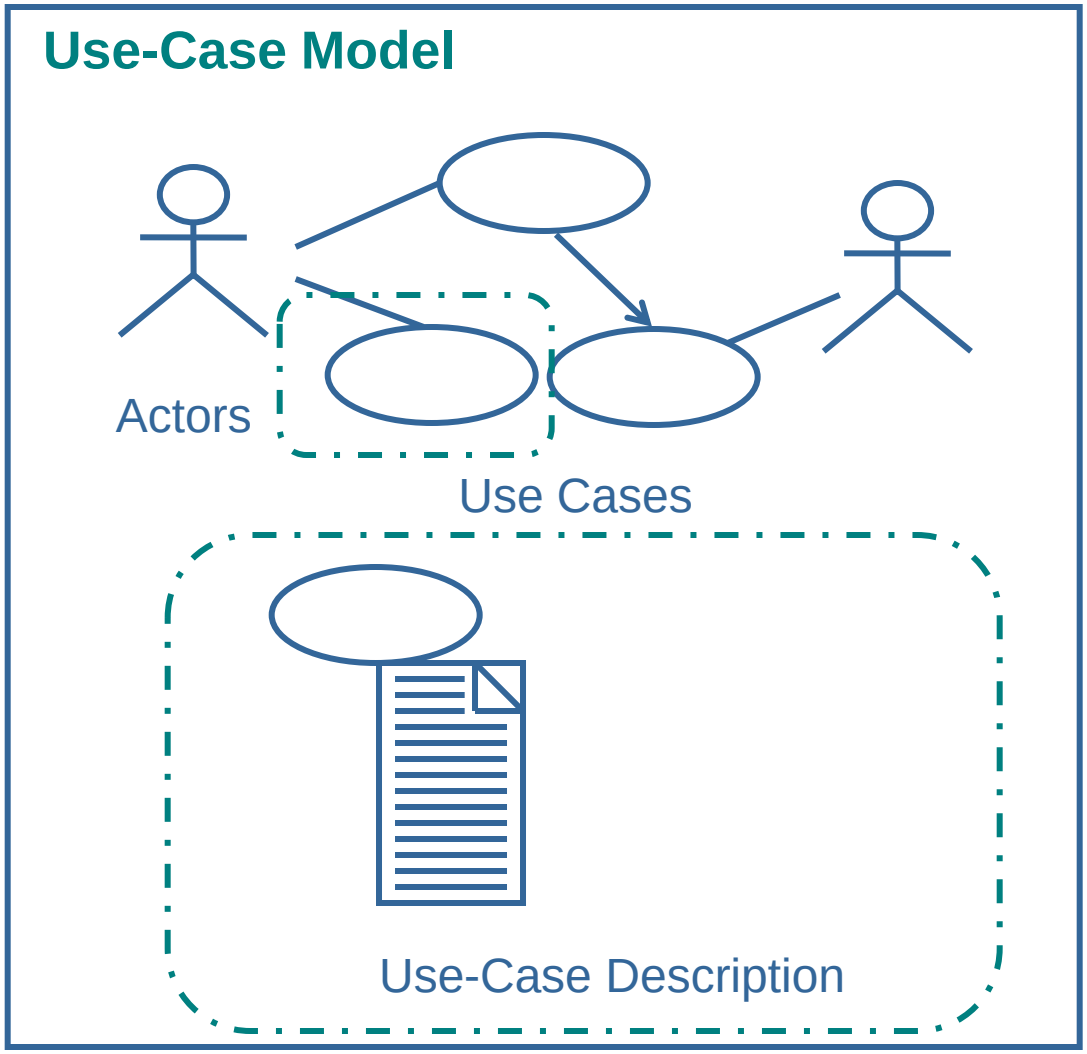
- Each group create a first draft use case diagram for our MUMSched project
- Think about actors and possible use cases

Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

Use Case Descriptions

- Name
- Brief description
- Flows of Events
- Relationships
- Activity and State diagrams
- Use-Case diagrams
- Special requirements
- Preconditions
- Postconditions
- Other diagrams



What Are Scenarios ?

- A scenario is an instance of a use case
- One path
- Like one boat on the river

Use-Case Flows of Events

- A use case “flow” is a collection of scenarios that have a common structure
- Like several boats floating together on the current
- A use case has one normal, *basic flow* and possibly several other *alternative flows* to handle:
  - Regular variants (business logic alternatives)
  - Odd cases
- **Exceptional flows** handle error situations (like system, DB errors, etc.) Things we might catch with exceptions of some type.

A Scenario or a Flow ?

Post-Condition?

Use-Case Description Example (1)

1. Checkout Book

This use case allows the librarian to check-out books for library members

2. Actors

Librarian

3. Pre-Conditions

User is logged in to the system

4.1 Basic Flow

User Action	System Response
1. The librarian enters the library member's ID and requests the system to retrieve the member information	1. The system retrieves the library member's name, address and phone number together with the list of books he/she has borrowed, and shows this information on the Checkout form. The system also retrieves the current balance that the library member owes the library for overdue books.
2. The librarian enters the ID of the book being checked out, and requests the system to retrieve information on the book.	2. The system retrieves the book title, author list and ISBN number and shows this information on the screen.
3. The librarian requests the system to checkout the book.	3. The checked-out book is added to the members checked-out book list, and fields that show the book information are cleared.

Use-Case Description Example (2)

5. Post-Conditions

New checkout record is added to the Library DB whenever checkout is successful.

6. Business Rules

Only members with a valid memberID can checkout books.

Only books that are found in the system an be checked out.

Students cannot checkout books if outstanding fines > $10.00.

Use-Case Description Example (3)

Alternate Paths

1) Library Member Not Found

If the member id does not exist, the system displays a message that the member id cannot be found.

2) Book Not Found

If the book id does not exist, the system displays a message that the book id cannot be found.

3) Current Fines > $10.00

If the current fines for a student are greater than $10.00 the system displays the minimum amount that the student must pay to be allowed to check out the book and an option to switch to the "User pay fine" use case.

Use-Case Description Example

Note that ***alternate flows*** are used to explain the system response to Business Rule checks that fail.

Business Rules

Only members with a valid memberID can checkout books.

Only books that are found in the system an be checked out.

Students cannot checkout books if outstanding fines > $10.00.

What would be some examples of exceptional flows?

Do we need to create use-case descriptions for those exceptional flows?

Our convention (***also for Test 1)*** is:

In the basic success flow we show the system response checking the business rules and we assume that they pass. We do **not** ignore the business rules in the basic flow but we save **how** we handle business rules failures for the alternate flows.

Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

Use Cases DO's

- Use Cases should be named in a way that suggests the goal of the use case -- that is, the goal of the actor in its interaction with the system for this use case.
- A Use Case should include a set of successful paths from a trigger event to the goal (success scenarios)
- Should also record a set of paths from a trigger event that fall short of the goal (failure scenarios)
- Evolve to more detail over the lifecycle

Use Cases DON'Ts

- A Use Case should not specify user interface design
  - Usually it is too early to commit to interface details
  - Instead, the UI is typically designed on the basis of the requirements.
  - However, Use Cases often offer suggestions about the look of the UI as an aid to understand the flows of the Use Case. These suggestions may be adopted, modified, or rejected completely when the UI is designed later on, but the principles they illustrate will survive in one form or another.
- A Use Case should not attempt to specify implementation detail beyond any technical constraints that may have been imposed on the project.

Recommended Steps for Creating Use Cases

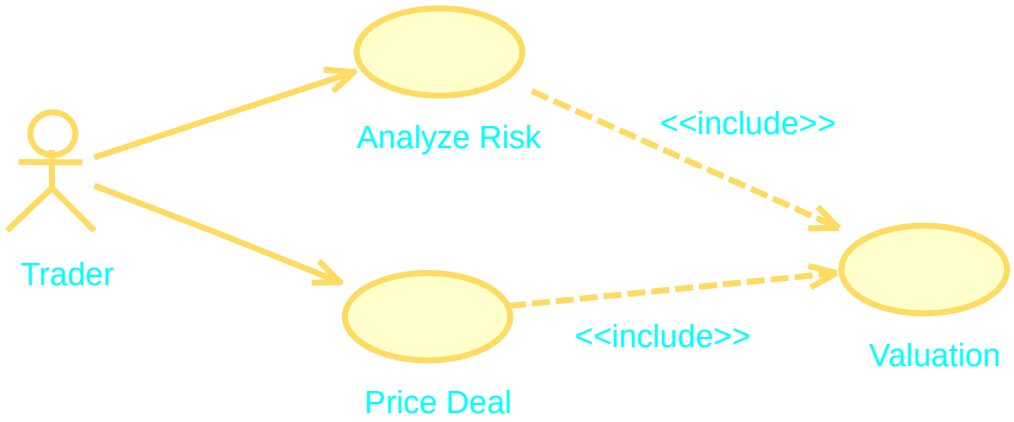
- Identify actors
- Name use cases
  - Start with verb, reflect goal
- Brief description
- Main success scenario
- Pre and post conditions
- Alternate flows
- Exceptional flows—what might go wrong
- Business rules
- Associated non-functional requirements

Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

Complex Use Cases - Relationships: Include

- Common steps used in several use cases
- Subgoals/subflows
- E.g., login, print, valuation



Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

Nonfunctional Requirements

- E.g., performance, security, user sophistication, hardware, software, …
- Attach use-case specific ones to use cases
- Others in supplementary specifications list

Supplementary Specification

- Functionality
- Usability
- Reliability
- Performance
- Supportability
- Design constraints

Supplementary Specification

Supplementary Spec Example (1/2)

Objectives

The purpose of this document is to define non-functional requirements of the Library System. This Supplementary Specification lists the requirements that are not readily captured in the use cases of the use-case model. The Supplementary Specifications and the use-case model together capture a complete set of requirements on the system.

Scope

This Supplementary Specification applies to the Library System .
This specification defines the non-functional requirements of the system; such as reliability, usability, performance, and supportability as well as functional requirements that are common across a number of use cases. (The functional requirements are defined in the Use Case Specifications.).

Supplementary Spec Example (2/2)

Reliability
The main system must be running 95% of the time.

Performance
The system shall support up to 15 simultaneous users against the central database at any given time.

Security
None

Design Constraints
The system shall integrate with an existing legacy system, the ECardCatalog Database, which is a MS Access database running on an Windows NT workstation.

Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

Glossary

- Problem domain terms
- Facilitates common understanding among developers as well as domain experts

Glossary

J139-40

Glossary Example (1/2)

1. Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal *data dictionary*, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

2. Definitions

The glossary contains the working definitions for the key concepts in the Library System.

2.1 ECardCatalog Database

The legacy database that contains all information regarding books in the library.

2.2 Library Member

Person who checks-out books from the library.

Glossary Example (2/2)

2.3 Librarian

Person who works for the library and performs the check-in and check-out procedure. This person also manages library member information.

2.4 Check-out Book

The procedure done if a library member wants to take a library book out of the library.

2.5 Check-in Book

The procedure done if a library member returns a checked out book to the library.

2.6 Overdue Fee

This is the amount that the library member has to pay the library when he/she returns a book later than the maximum allowed check-out days.

Requirements Analysis Topics

- A modern software requirements specification (SRS)
- Use-Case Model
  - Use cases drive entire development lifecycle
  - Use case diagrams
  - Use case descriptions
  - Guidelines for developing use cases
    - Business rules in use cases
    - Structuring complex use cases
- Supplementary Specs
- Glossary
- Review

Review Questions

- What are the main artifacts of requirements analysis, and what are their purposes?
- What is a use case? List examples of use case properties (I.e., the parts of a UC description).
- What is a scenario?
- Name 3 elements of a UC diagram.

Summary Main Points

- Use cases give expression to software requirements and form a unifying thread through the entire software development life cycle.
- Requirements, in the form of use cases, form the basis of a contract between developers and stakeholders. It is agreed:
  a) developers will have fulfilled their responsibility if the software solution meets all requirements
  b) stakeholders can expect developers to provide the features made explicit in requirements, but cannot expect more than this.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

1. Use cases represent the way a user interacts with a system in order to achieve some goal—i.e., the functional requirements of the system.
2. Use cases are a powerful technique for structuring the entire software development process. They naturally keep a lively connection between the users, the developers, and the emerging software solution, and serve to organize the rest of the development stages. Finally, they are used to validate the system in terms of user goals.

3. **Transcendental consciousness** is the unifying basis underlying all our thoughts, feelings, and actions.
4. **Impulses within the transcendental field:** The unified value of pure consciousness established in our activity makes all our activities successful and fulfilling.
5. **Wholeness moving within itself:** In unity consciousness, the diverse parts in any relationship are appreciated as lively expressions of a single unified reality.