Software Design: Modeling with UML

System Analysis and Design

Class 03

Mustafa Hasan

Model

A partial abstract representation of a real-world system

An inexpensive way to analyze, communicate, test, and document our understanding of the system

Types of Models

Computational

- Computer simulations representing time-varying behavior of a system

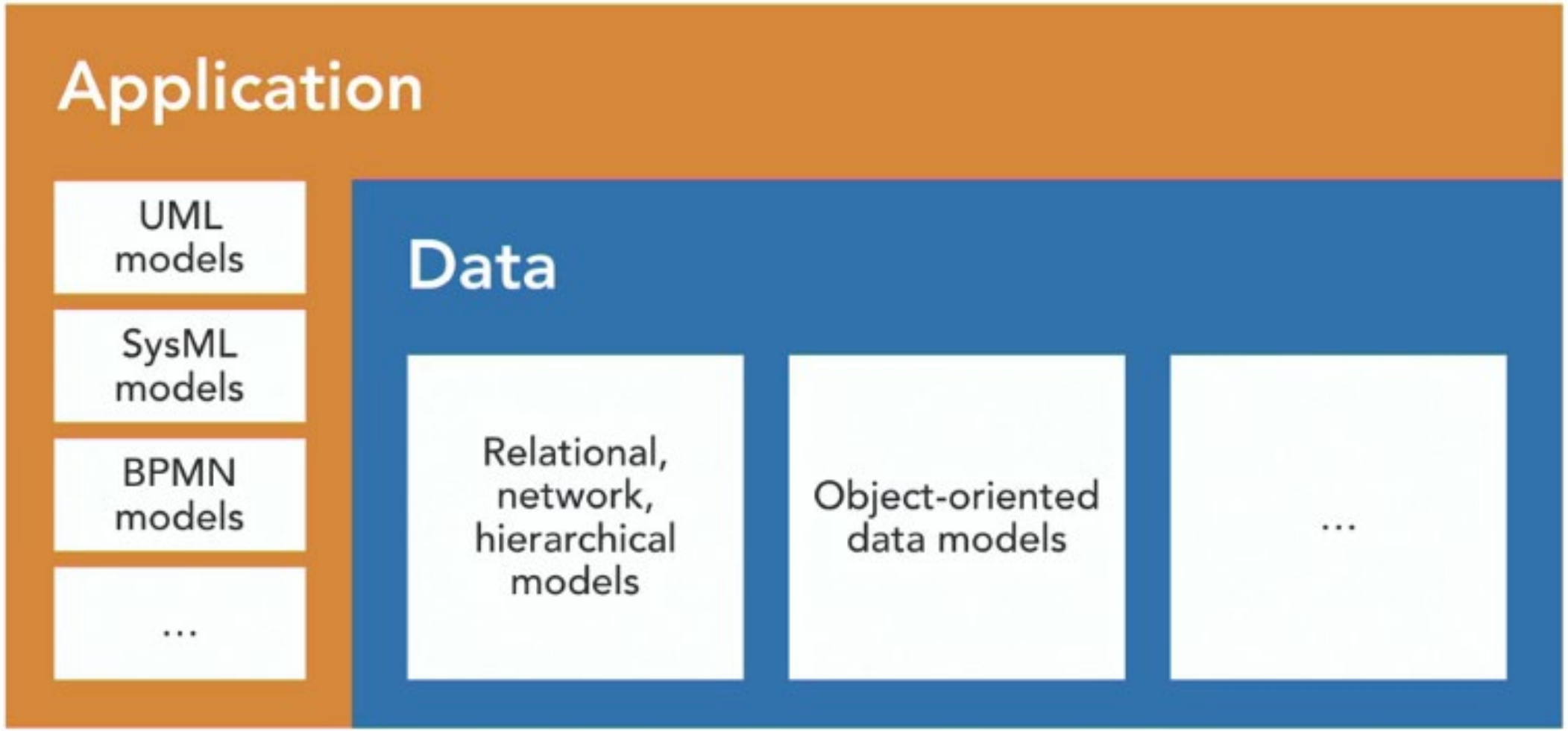
Analytical

- Mathematical models of relationships among variables in a system

Nonanalytical/descriptive

- Describe components and their relationships in a system

Models of Software

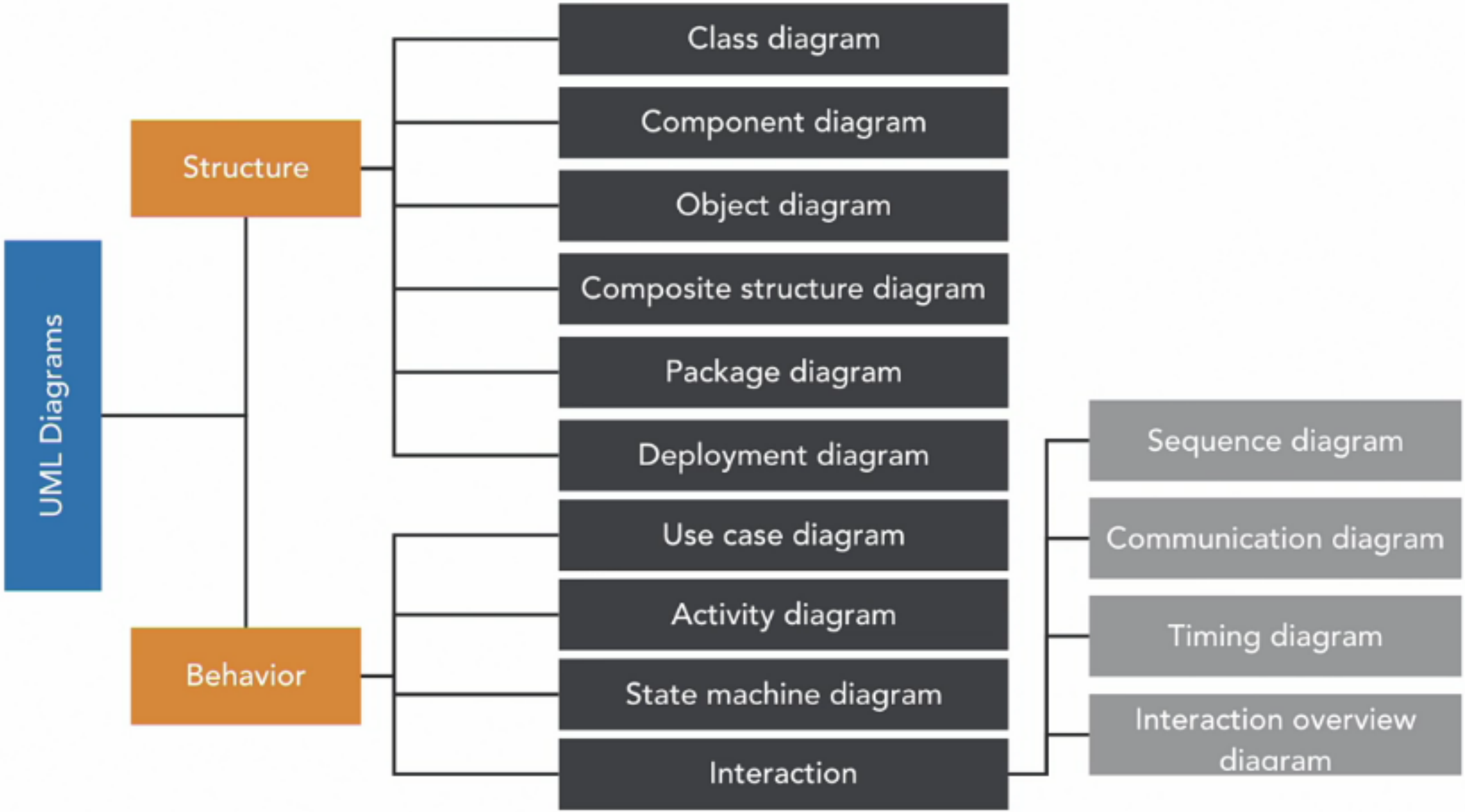


Unified Modeling Language

A family of graphical notations to describe and design software systems, especially those using an object-oriented approach

Based on standards controlled by Object Management Group (OMG)

Releases: UML 1 (1997), UML 2 (2005), UML 2.5 (2015)


UML Diagrams
Structure
Class diagram
Component diagram
Object diagram
Composite structure diagram
Package diagram
Deployment diagram
Behavior
Use case diagram
Activity diagram
State machine diagram
Interaction
Sequence diagram
Communication diagram
Timing diagram
Interaction overview diagram

1

Structure

Represents static view of the system and its components

2

Behavior

Represents dynamic view of the system and its components

3

Interaction

Represents interaction

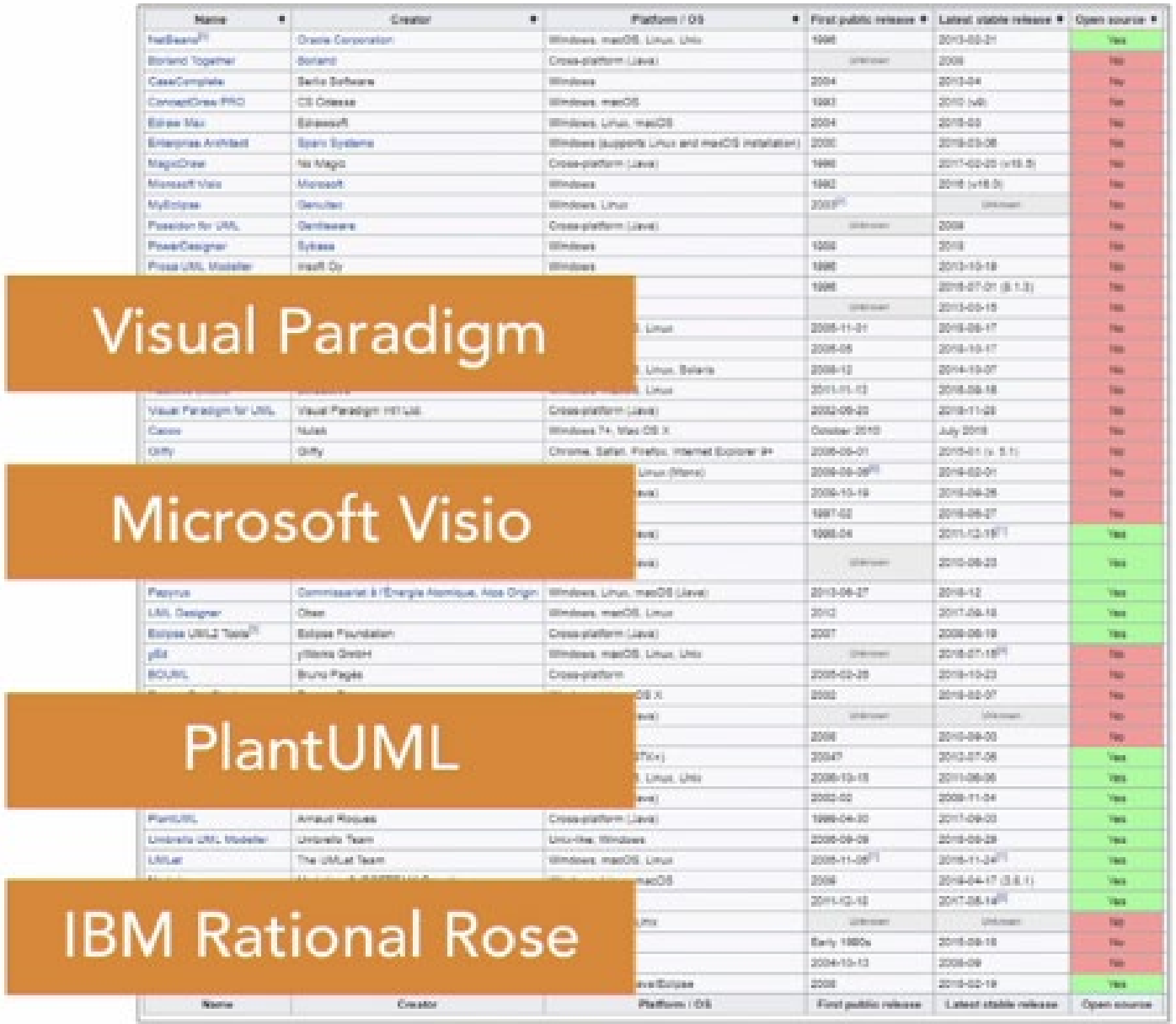
- Among components of the system
- Between system and external actors

Important Considerations

- **Model selectively:** you need not (and should not) draw all the models to develop a system
- **Model collaboratively:** use models to think, share, learn, and understand together with your team
- **Model smartly:** start rough and refine it as needed, making it as a long-term asset for the team

CASE Tools

- Computer-aided software engineering tools help in various tasks throughout the software development life cycle
- Some key functions of CASE tools: modeling, code generation, reverse engineering, analyzing code complexity, and other metrics

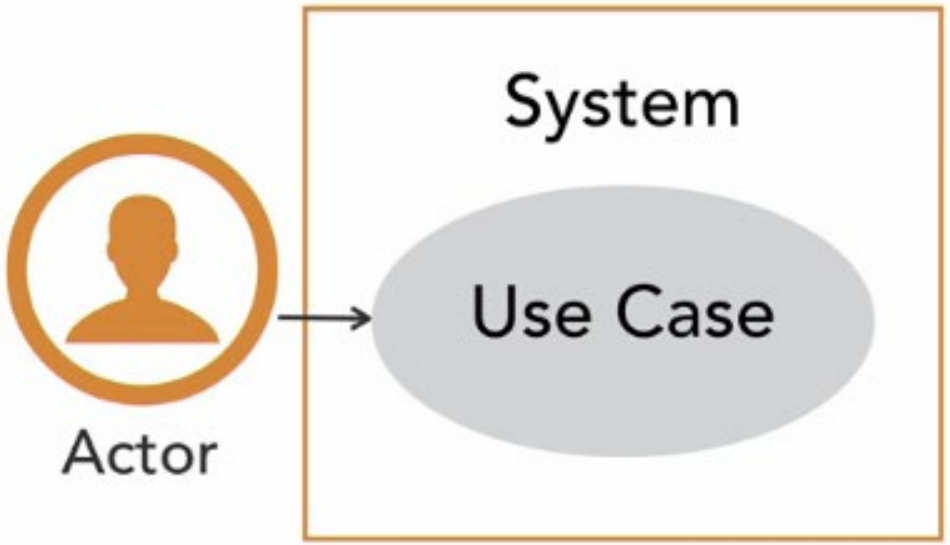


https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Use Case Diagram

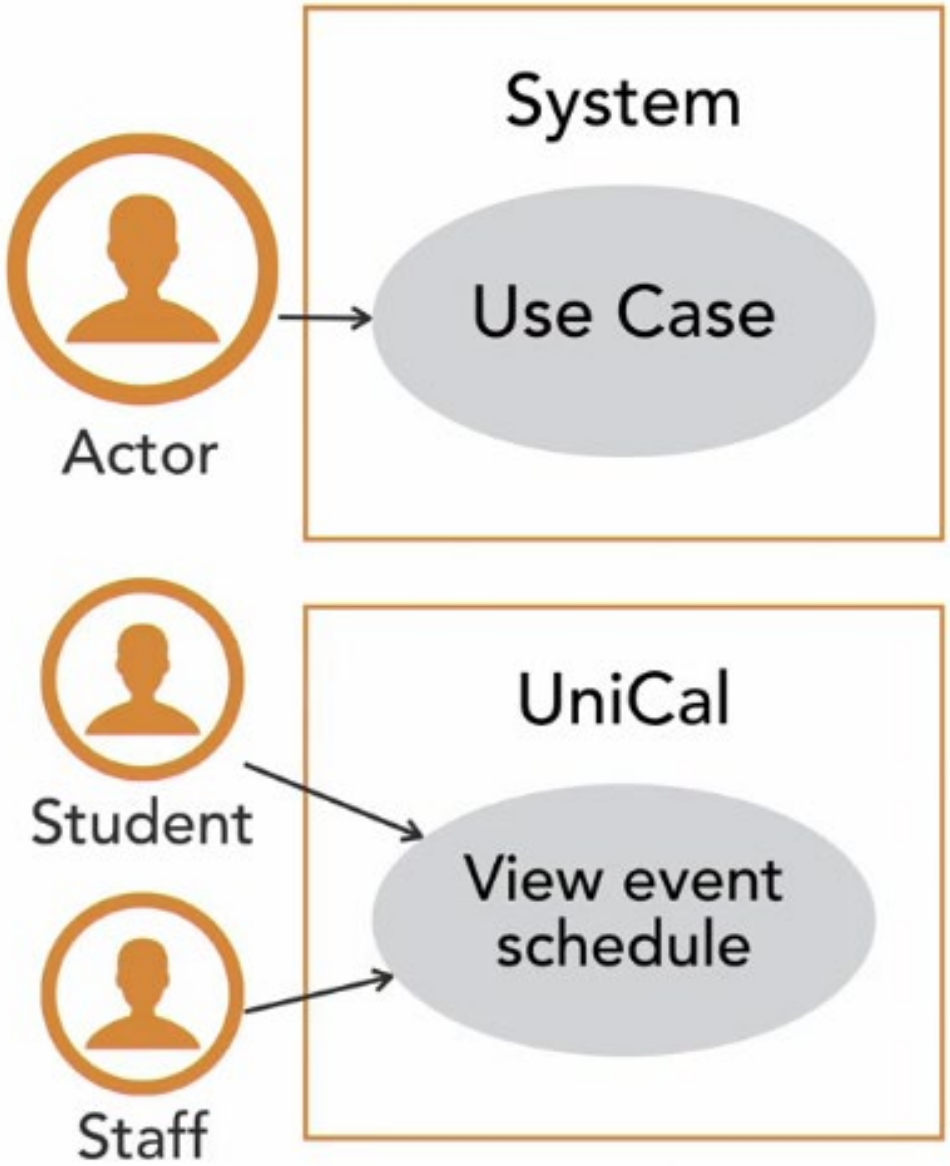
Capture high-level functionality of a system using notations for actors, use cases, and relationships among them

Often drawn by business analysts to depict the summary all use cases in a system



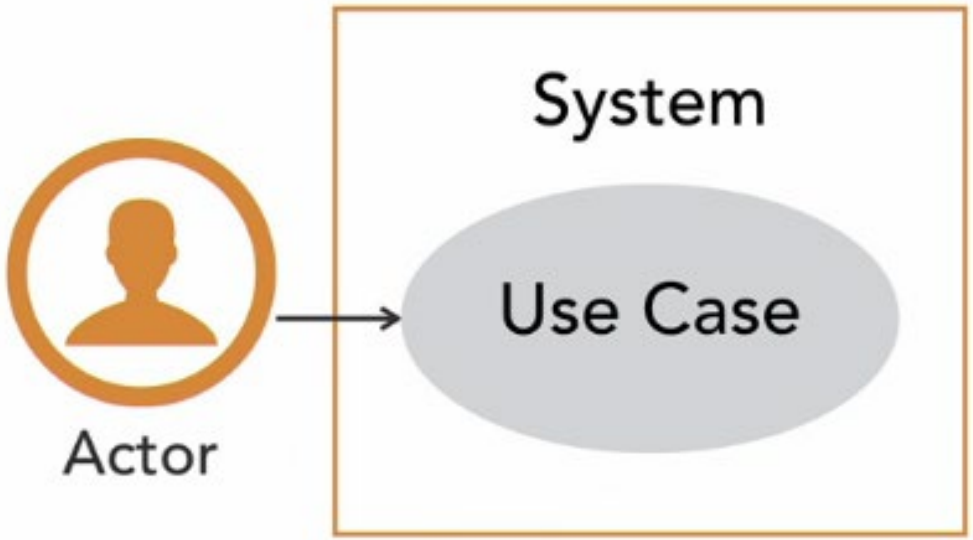
Key Elements

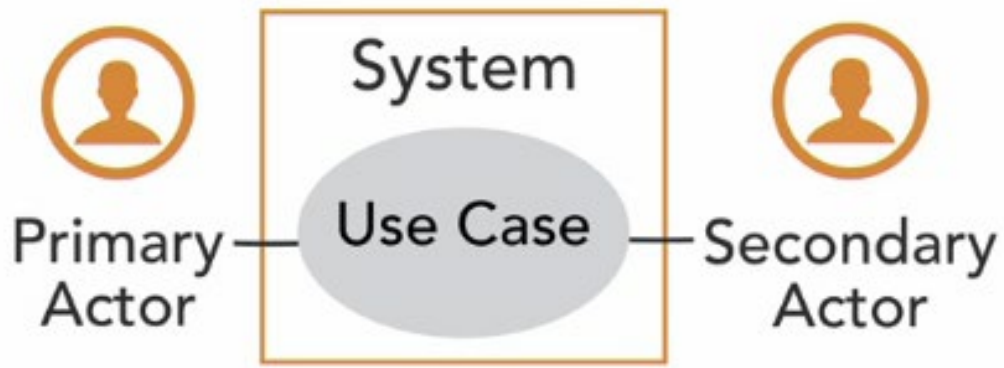
- Use cases
- Systems
- Actors
- Associations



Use Case

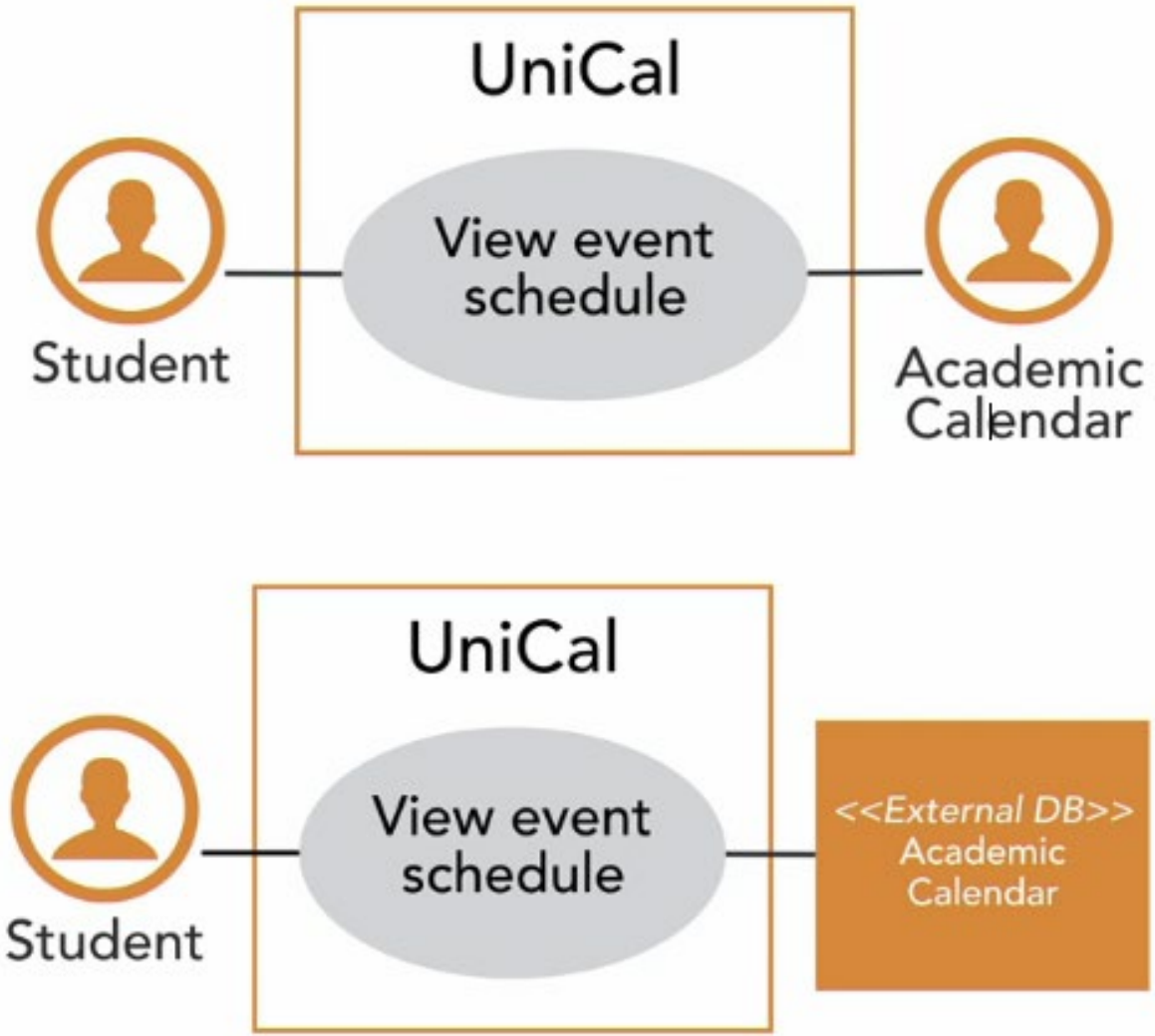
- The use case notation is a bubble that carries use case title
- Associated with actors and possibly other user cases





Actor

- A user's role with respect to the system
- May be a human or another system
- Primary actor: whose goal is fulfilled by the use case
- Secondary actor: who is involved in the use case

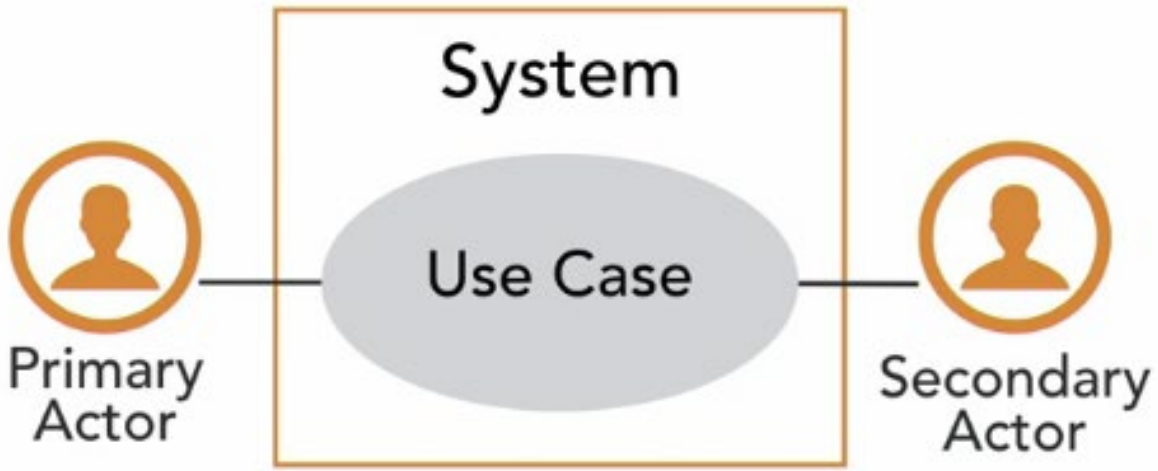


Secondary Actor

Often an external system

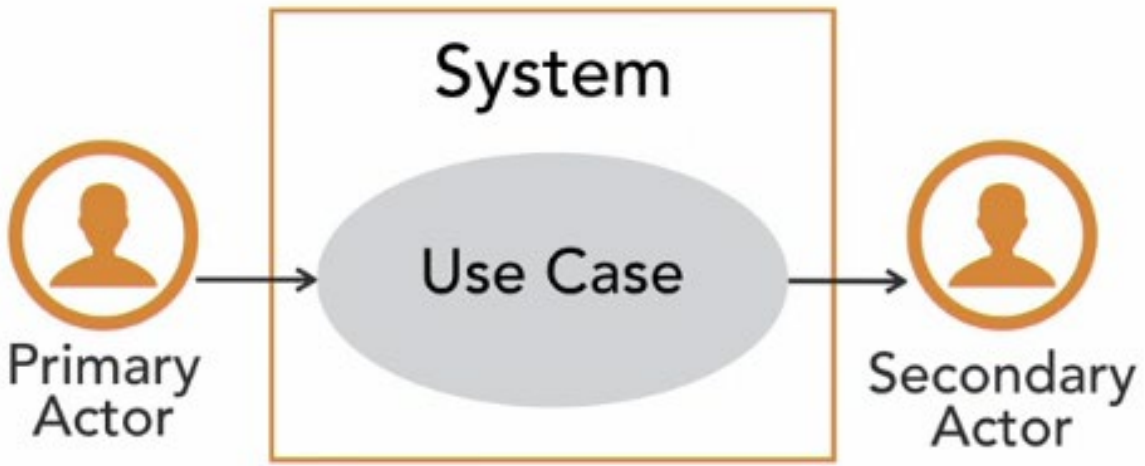
Possible notations

- A nonhuman notation such as a rectangle
- Stereotyped with << >>



Association

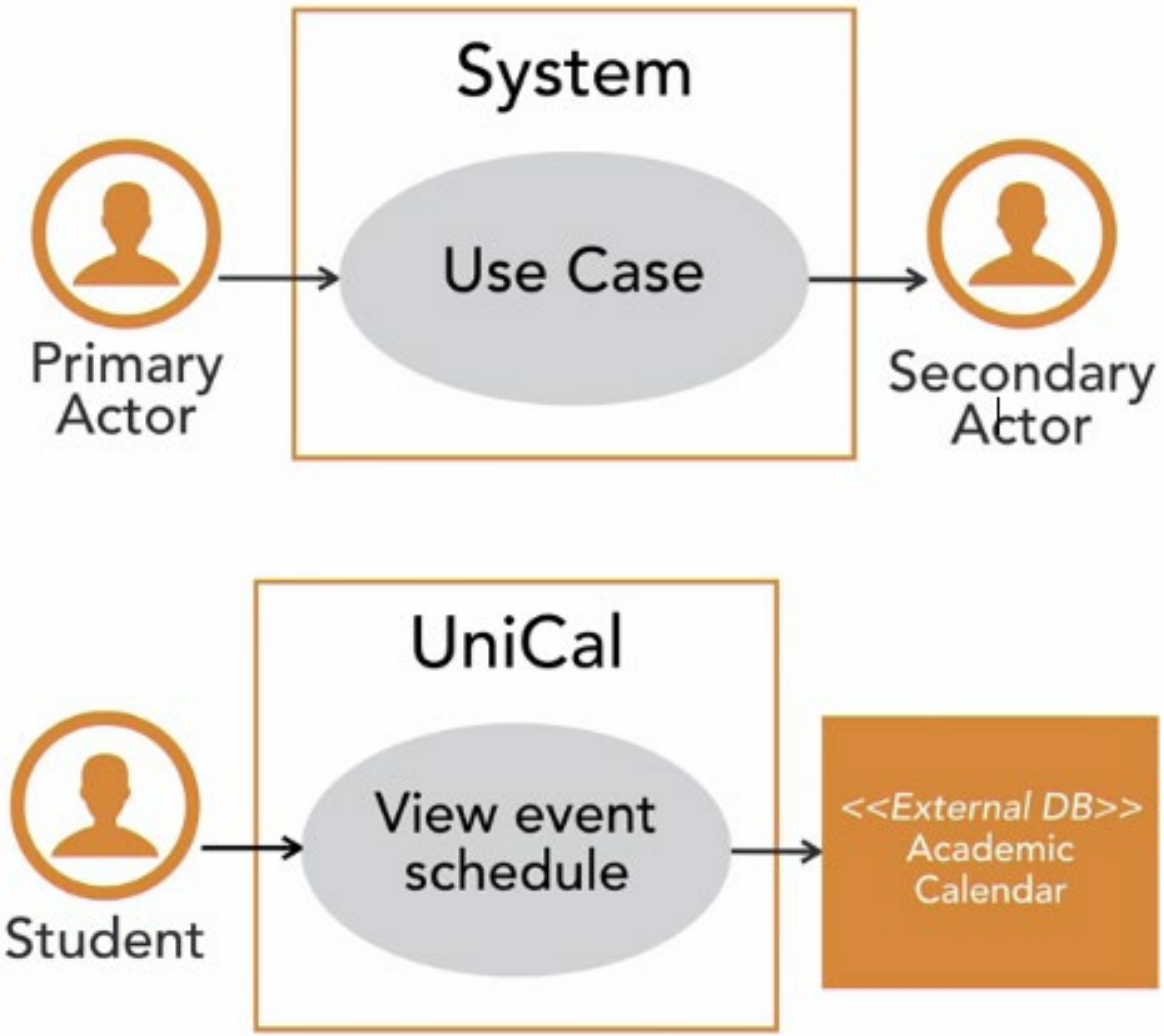
- Between actor and use case
- Between use cases
- Between actors



Between Actor and Use Case

Convention

- Primary actor on the left
- Secondary actor on the right of the system



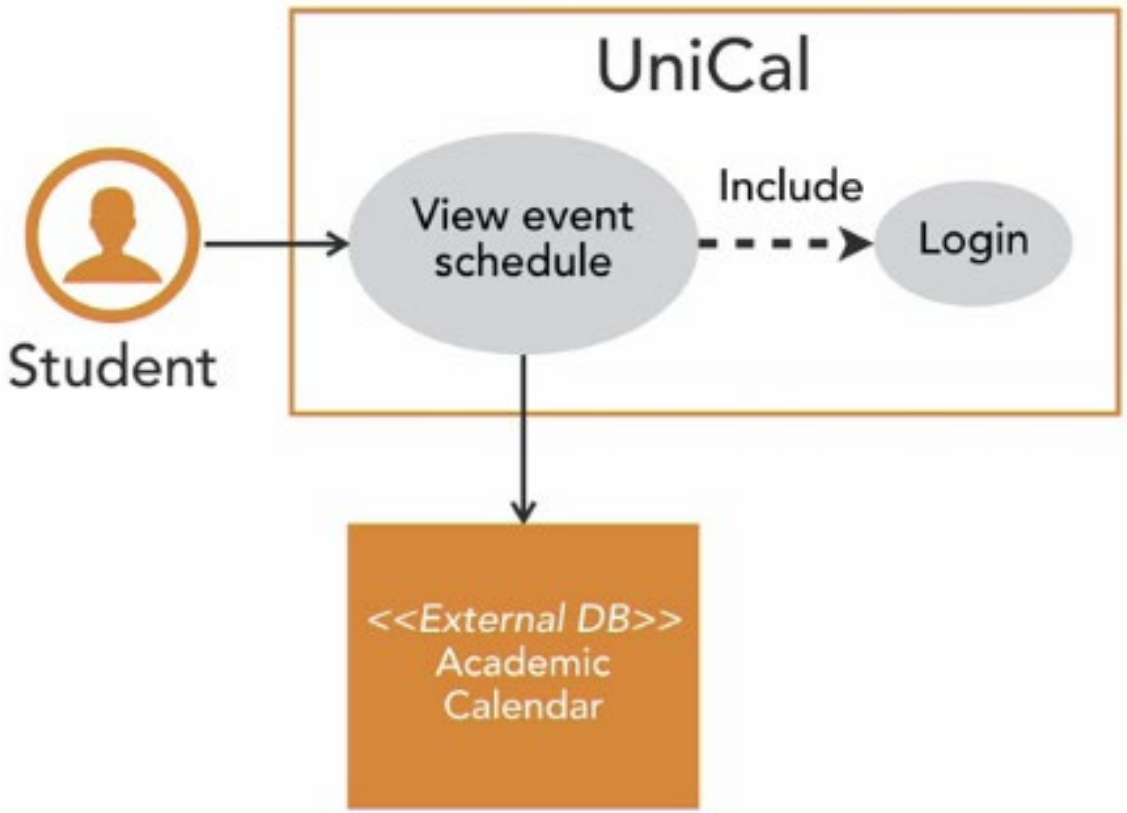
Between Actor and Use Case

Convention

- Primary actor on the left
- Secondary actor on the right of the system

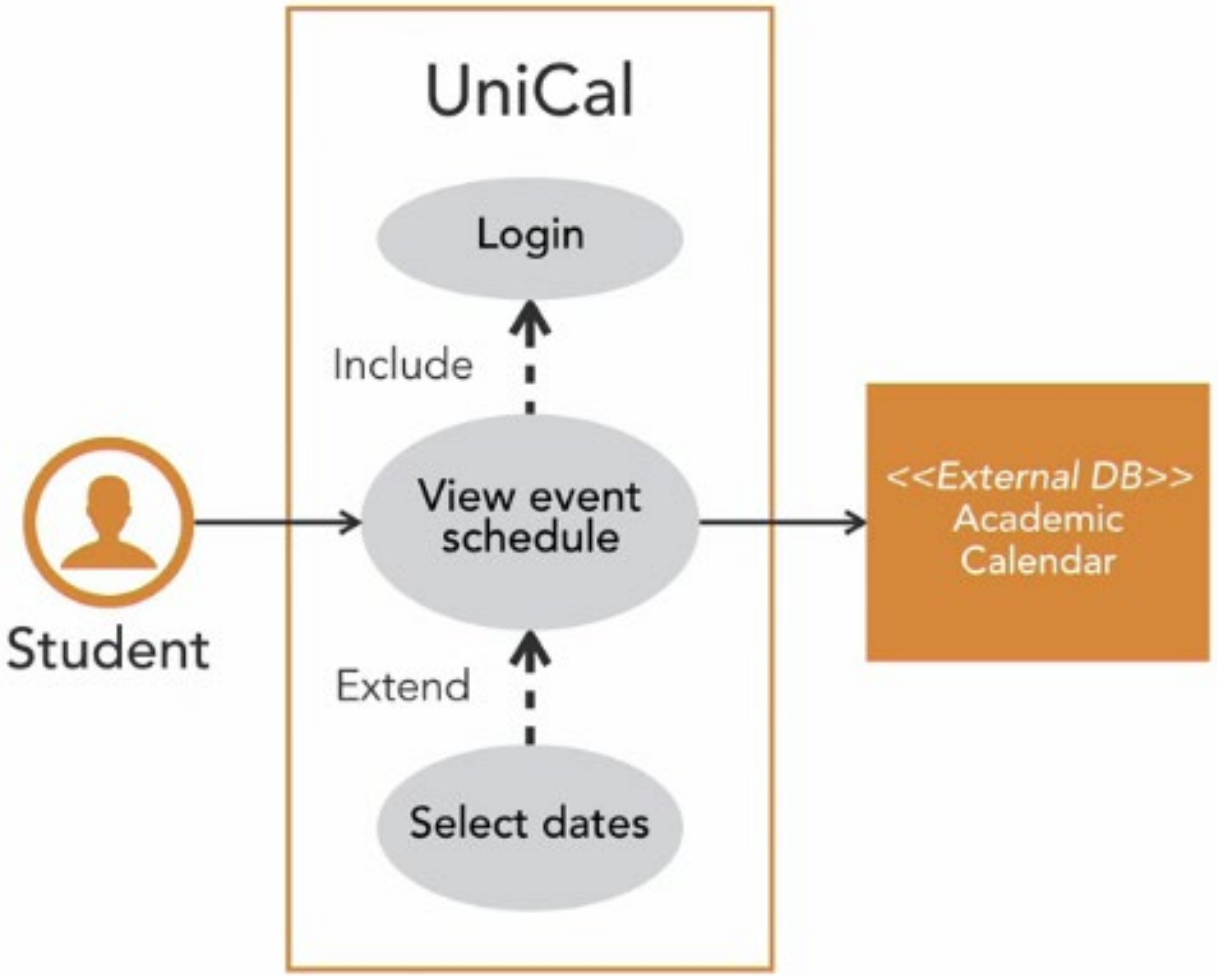
Arrowheads

- From primary to use case
- From use case to secondary



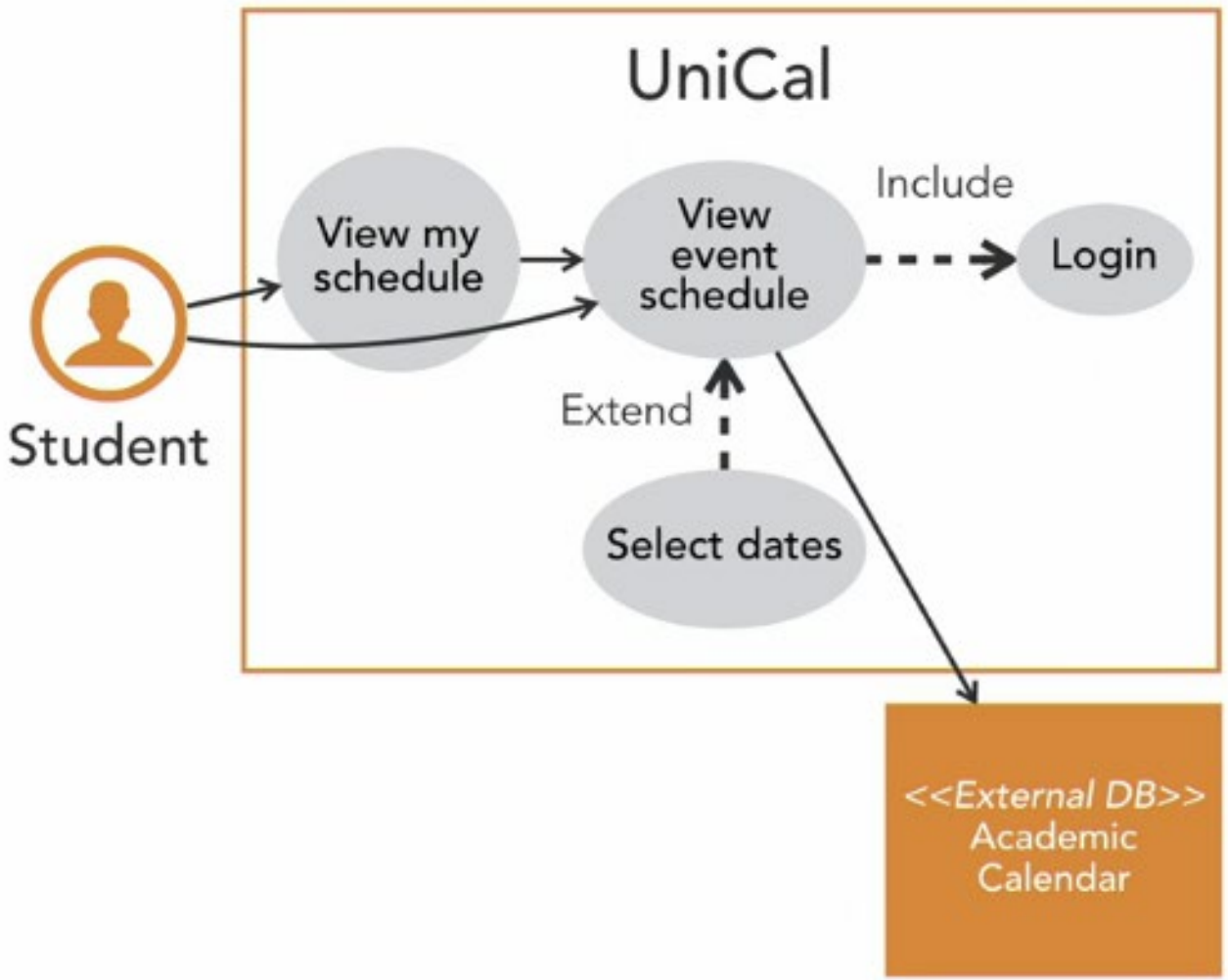
Between Use Cases: Include

- For reusable parts of behavior across two or more use cases



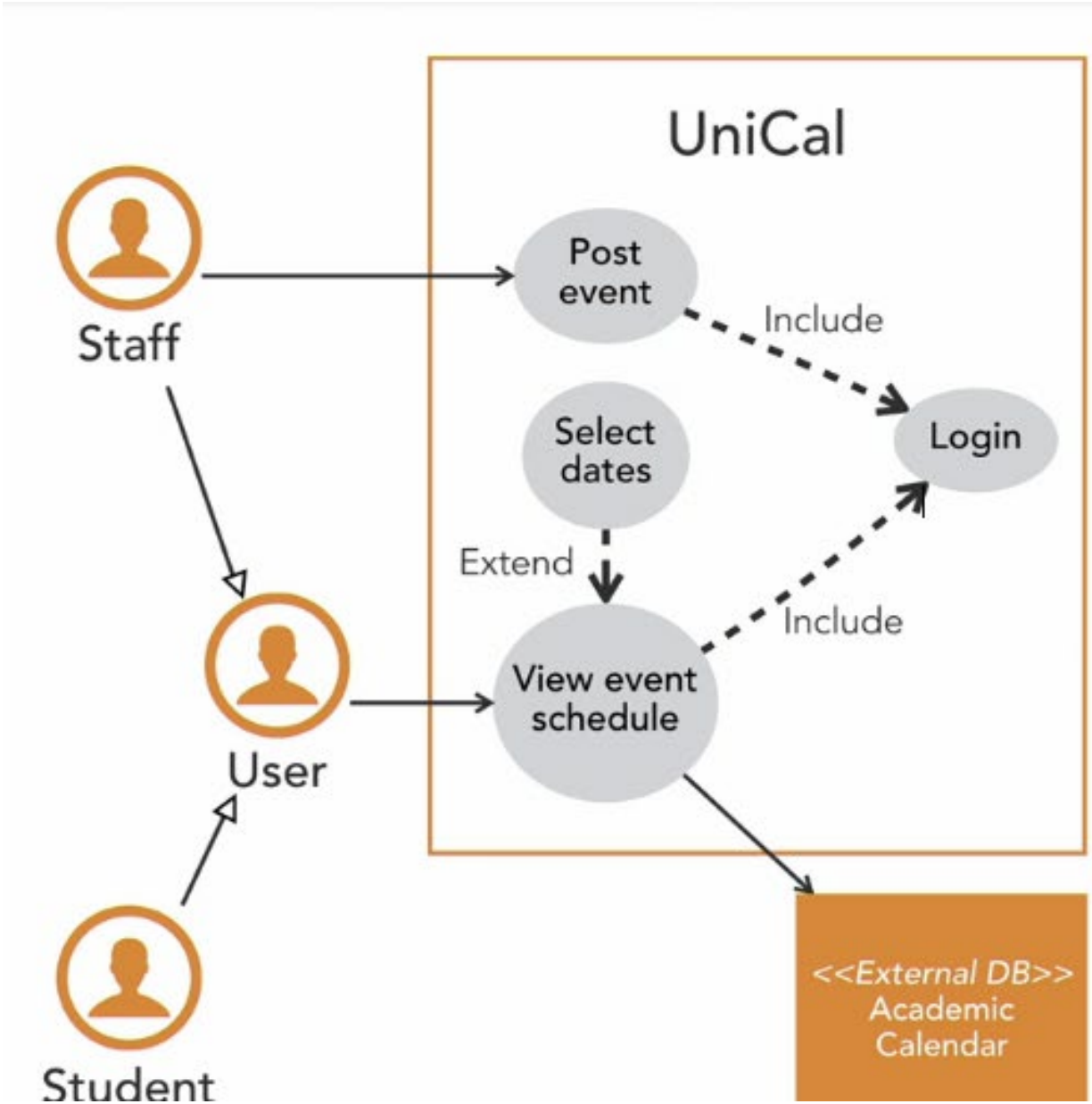
Between Use Cases: Extend

- Optional behavior added to use case
- Helps keep the base use case unchanged while adding more specifics or conditional changes



Between Use Cases: Generalization

- One use case is a special case of another more general use case



Between Actors: Generalization

- Depict generalization – specialization or inheritance relationship between actors