

UML OVERVIEW

CSE416-S01 - Software Engineering

1

CSE416 - Software Engineering © Robert Kelly, 2026 2

Reference

- Use case
en.wikipedia.org/wiki/Use_case

2

Session Objectives

- Understand the purpose of UML in the design and development of a system
- Understand the use of use case descriptions to identify the detailed functionality of a system
- Begin to transform top-level requirements into use cases

3

What is Modeling?

- Modeling consists of building an abstraction of reality
- Abstractions are simplifications because:
 - They ignore irrelevant details and
 - They only represent the relevant details
- What is relevant or irrelevant depends on the purpose of the model, the audience, and other factors

This is a very difficult decision



4

Why Model Software?

- Software is getting increasingly more complex
 - Some versions of Windows > 40M lines of code

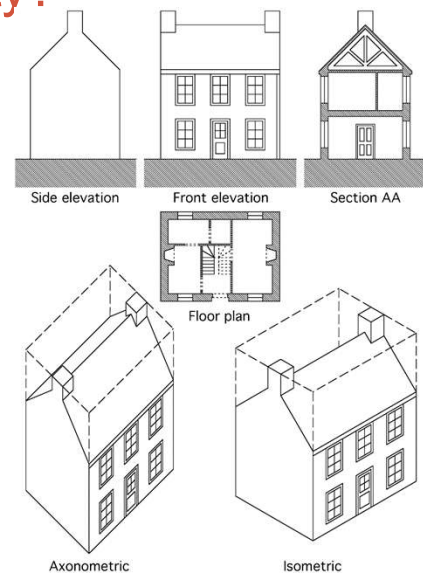
Could you comprehend 40M LOC?
- Modifying a model of a system is much, much easier than modifying software
- We need simpler representations for complex systems
 - Modeling is a way for dealing with complexity

Remember, one course goal is to
think first, code second

5

How Do We Deal With Complexity?

- Break it down into simpler parts
- Example- design specifications for a building
- Helps in
 - getting user/peer feedback
 - Getting approval
 - Avoiding construction problems



6

Systems, Models and Views

- A model is an abstraction describing a system or a subset of a system
- A view depicts selected aspects of a model
- A notation is a set of graphical or textual rules for depicting views
- Views and models of a single system may overlap each other

We often just generate different views, which together constitute a model

7

What is UML?

- UML (Unified Modeling Language)
 - A standard for modeling object-oriented software.
 - Derived from the convergence of notations from three leading OO approaches:
 - OMT (James Rumbaugh)
 - OOSE (Ivar Jacobson)
 - Booch (Grady Booch)
- Supported by several CASE tools
 - Visio
 - Workbench
 - Visual Paradigm
 - Lucidchart

You can model 80% of most problems by using about 20 of % UML (maybe 90/10)

8

UML Approach for CSE416

We will use activity diagrams to model the behavior of the Python parts of the system

- Use cases
 - Describe the functional behavior of the system as seen by the user
 - Great for decomposing a system into buildable units
- Sequence diagrams for server
 - Describe the dynamic behavior between actors/system/objects
 - Helps to define the objects that are needed to implement a use-case
- Class diagrams for server
 - Describe the static structure of the system: Objects, Attributes, Associations
 - Can be revised based on discoveries made from sequence diagrams
- Activity diagrams for preprocessing and SeaWulf

9

Other UML Notations

- UML provides other notations used less often
- Implementation diagrams
 - Component diagrams
 - Deployment diagrams
 - State-chart diagrams (essentially a finite state automaton)

10

Naming Conventions for OO/Java

- Camel case for classes (upper cc) and attributes (lower cc)
- Classes– singular
- Attributes– singular (plural for collections)
- Avoid acronyms and abbreviations except where well known (e.g., PI for Principal Investigator)

Conventions apply very early in the process

Names should describe the application domain, not the implementation approach

Naming conventions are part of the “teamwork” approach to CSE416

11

Use Case

- Used during requirements analysis phase to represent system behavior in response to external drivers (e.g., button click)
- Represents an interaction sequence for a type of functionality
- A use case consists of:
 - Unique name
 - Participating actors
 - Entry conditions
 - Trigger
 - Flow of events (scenario)

Use cases can vary in the level of detail. In CSE416, we use a simple model of a use case

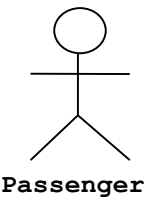
The project use case list is really just a list of use case descriptions, not details

12

CSE416 - Software Engineering © Robert Kelly, 2026 13

Actors

- An actor models an external entity which communicates with the system
- It can be a:
 - User,
 - External system, or
 - Physical environment
- An actor has a unique name
- Example:
 - User of your system



Passenger

Similar to a role

13

CSE416 - Software Engineering © Robert Kelly, 2026 14

Example

- Example of a textual use case
- Design issues:
 - No overlap in use cases (instead think of preconditions)
 - Look for use cases that cover multiple roles (with exceptions that differentiate the roles)
 - Proper size (not too many steps or too few steps)

Sample Use Case – CSE416

Use-case:	Determine Minorities Eligible for an Opportunity District
Primary actor:	General User
Goal in context: <i>one sentence description</i>	Determine eligible minority groups for an opportunity district, resulting in those groups highlighted in GUI
Preconditions:	State is selected User has set threshold for minority population to be eligible for consideration as an opportunity district
Trigger:	User navigates to a tab that displays minority group options (e.g., box & whisker plot request)
Scenario: <i>step by step actions taken by the system to achieve goal in context</i>	<ol style="list-style-type: none"> 1. GUI sends request to server 2. Server accesses current state and minority population threshold (e.g., 400,000) 3. For each of the minority population groups, server determines if the population of the group exceeds the opportunity district threshold. 4. Server returns data that identifies the minority groups whose population exceeds the threshold 5. GUI modifies the GUI components so that the GUI components for the identified minority groups are active and all other minority group components are not active
Exceptions: <i>Special cases that deviate from scenario above</i>	<ol style="list-style-type: none"> 1. User may not have set the threshold. If that is the case, use a default of 400,000 for each minority population threshold.
Priority: <i>Build number (with reasons)</i>	Build 5 (earlier builds can default to making all minority groups eligible)
Open issues: <i>List of open interface questions or unresolved issues</i>	<ol style="list-style-type: none"> 1. Requirement may be modified so that ineligible minority groups are not displayed instead of being greyed out

14

Use Case Class Exercise

- A volunteer (or selected) group will start with a use case description from the project use case list, and develop a detailed text use case
- Helps to start thinking about design issues
- Helps to start thinking about interface issues
 - Parameters passed to server
 - JSON returned from server

You will be required to show some detailed text use cases in your midterm exam

15

Have You Satisfied the Objectives?

- Understand the purpose of UML in the design and development of a system
- Understand the use of use case descriptions to identify the detailed functionality of a system
- Begin to transform top-level requirements into use cases

16