CSC290 Communication Skills for Computer Scientists

Lisa Zhang

Lecture 4; Oct 1, 2018

Reminder

- Team Charter due Sunday (suggest submit by Friday).
- Topic choices this Friday.
- No lecture / tutorial next week!
  - Thanksgiving and Reading Week
  - Reading week recommendations posted
- Blog post topic posted (due following Sunday)
- Midterm information posted

Unified Modelling Language (UML)

What is UML?

UML is is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.

What is UML used for?

- Analysis and design of software
- Modelling software built using object-oriented languages
- Documentation and communication
- Requirement analysis

Types of UML Diagrams

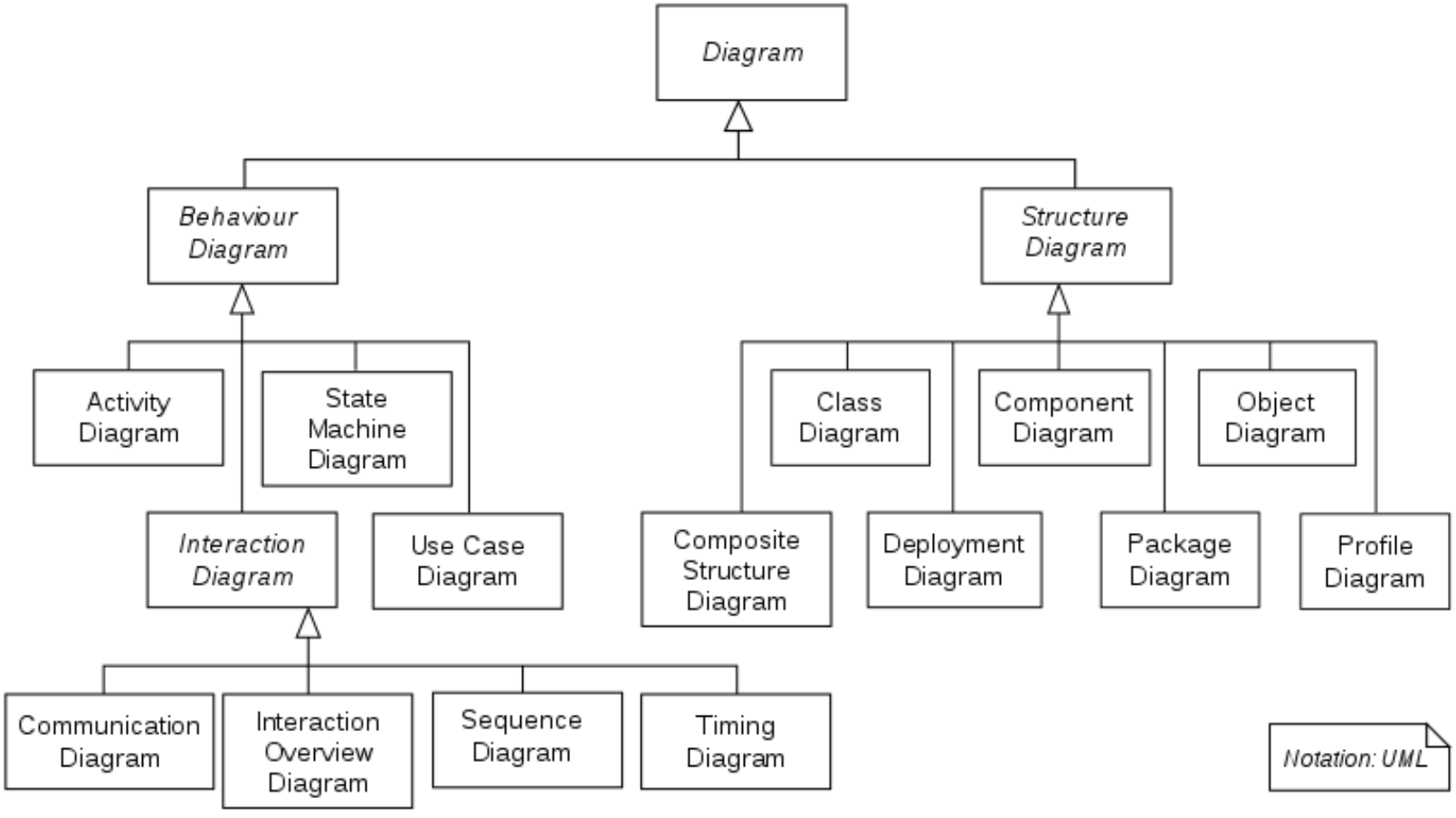


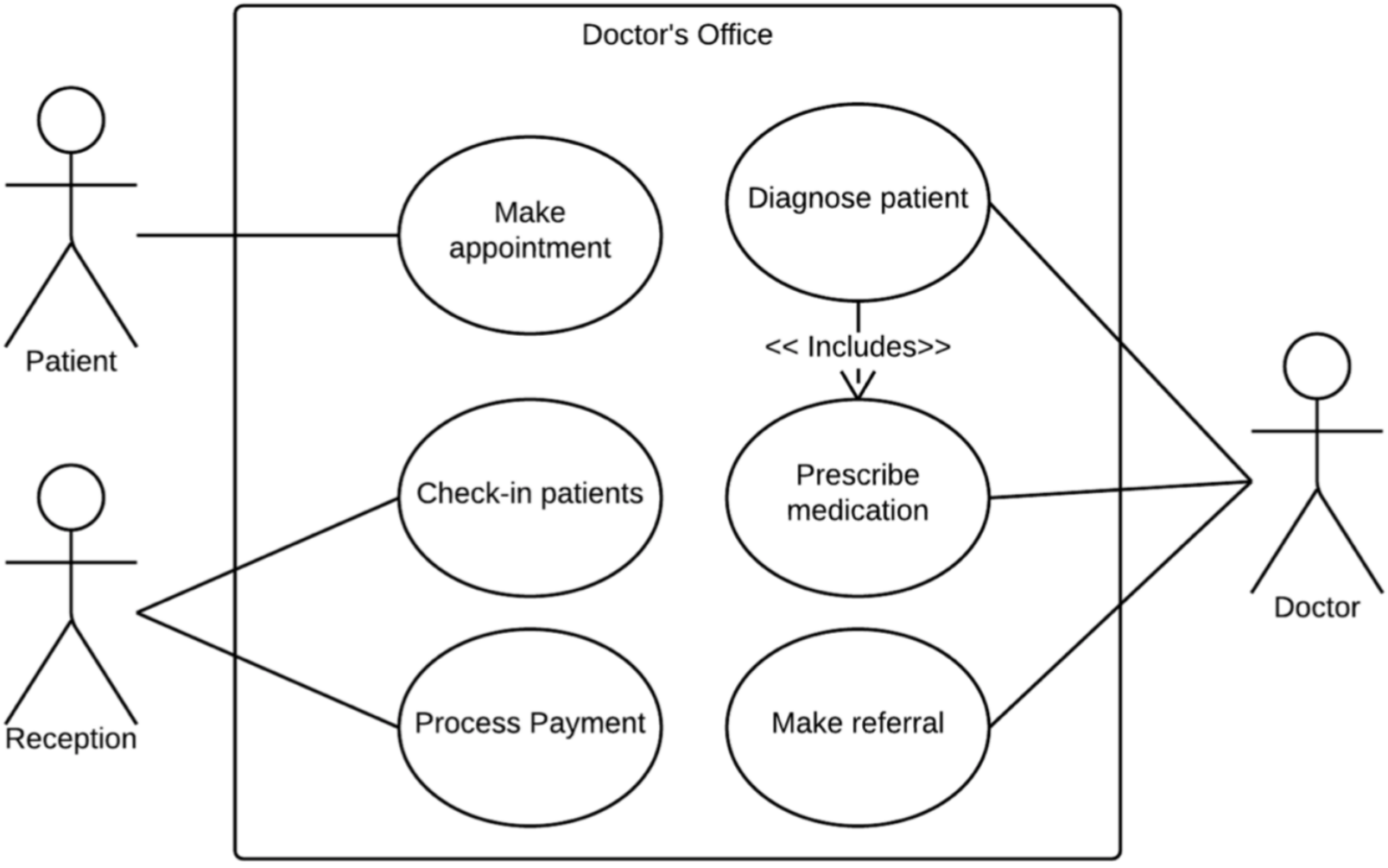
Figure 1: from wikipedia

Our focus

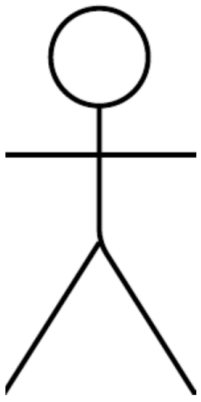
- Use Case Diagram
- Activity Diagram
- Class Diagram

Use Case diagram

Use Case diagram



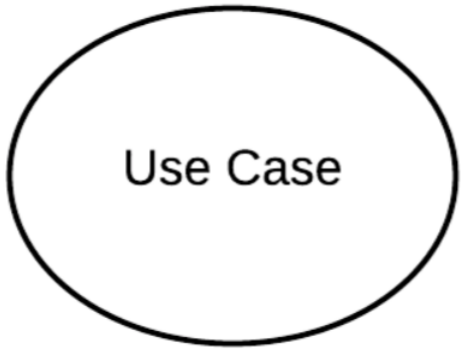
Parts of a Use Case Diagram



Actor: an external entity which interacts with your system

- Carries out or triggers use cases
- Does not have to be a person; could be an external system
- Labelled with a noun or noun phrase (eg. customer, purchaser, administrator)

Parts of a Use Case Diagram

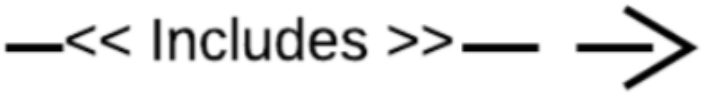


Use case: a set of scenarios tied together by a common user goal

- Can extend or include another use case
- Represents a major piece of system functionality
- Labelled with a descriptive verb-noun phrase:
  - active voice, present tense verb phrase,
  - direct object phrase (eg, renew vehicle registration)
- Avoid vague use case names, like “manage X” or “perform X”.
  - vague, do not resonate with business experts
  - possibly too high level of granularity

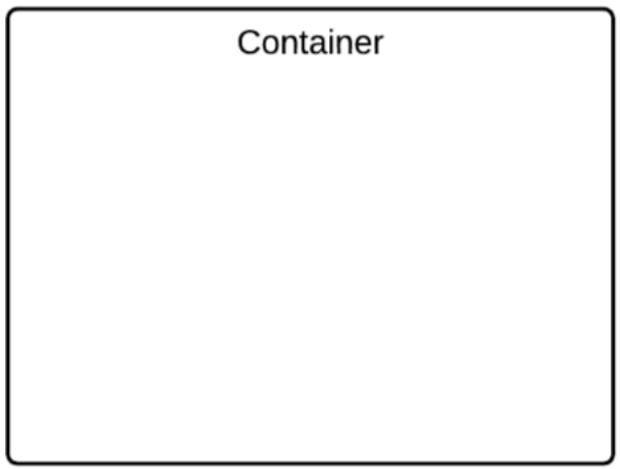
Parts of a Use Case Diagram

Associations: links actor with a use case; non-directional



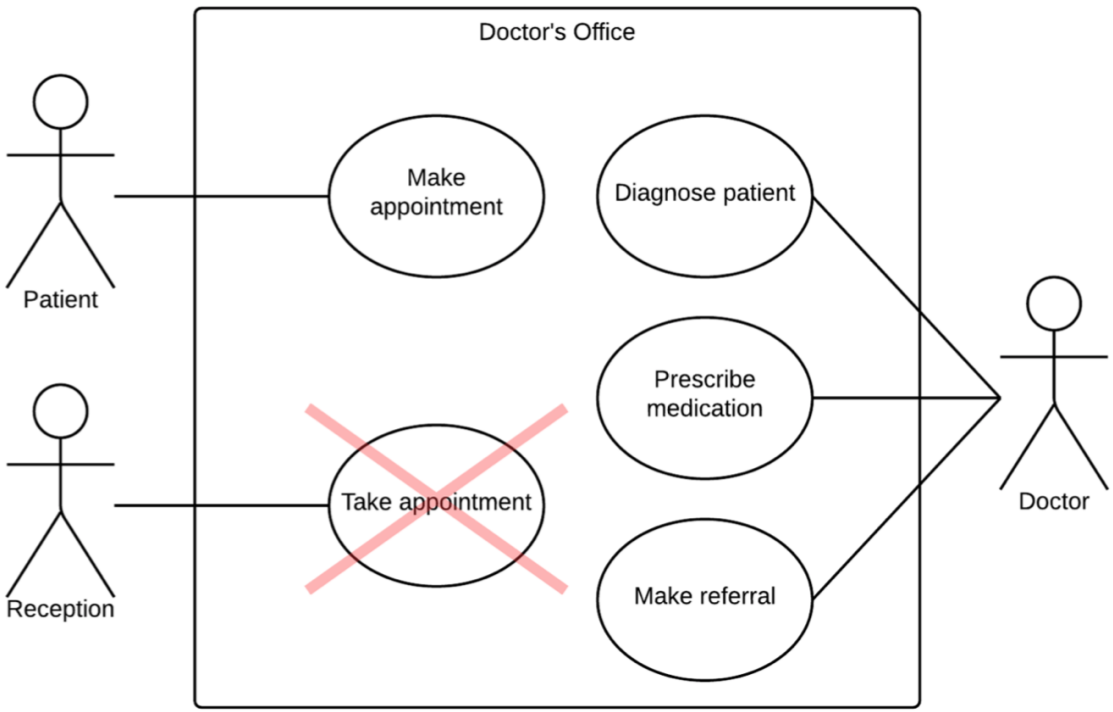
Include or Extend Relationships: associates two use cases; directional; dashed line labelled with relationship

Parts of a Use Case Diagram

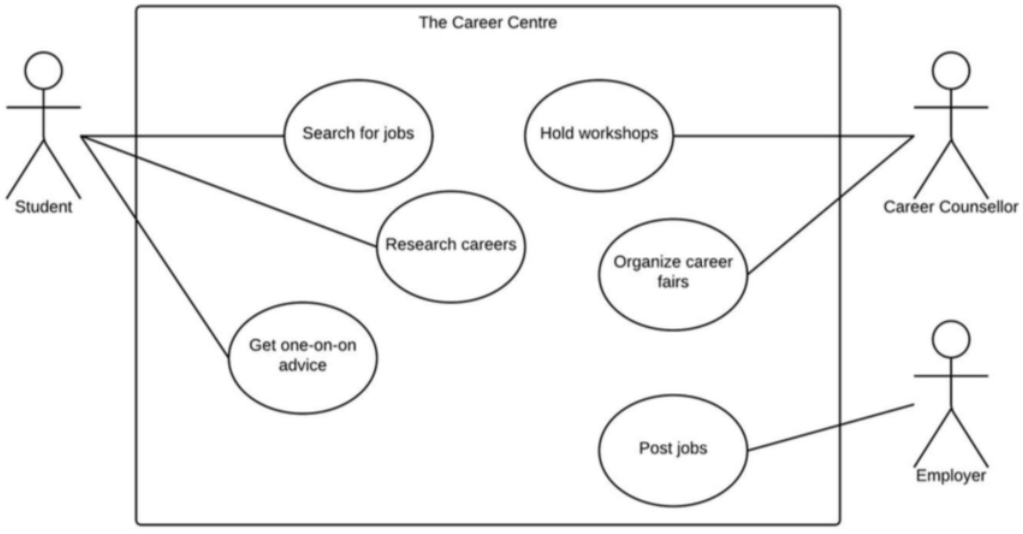


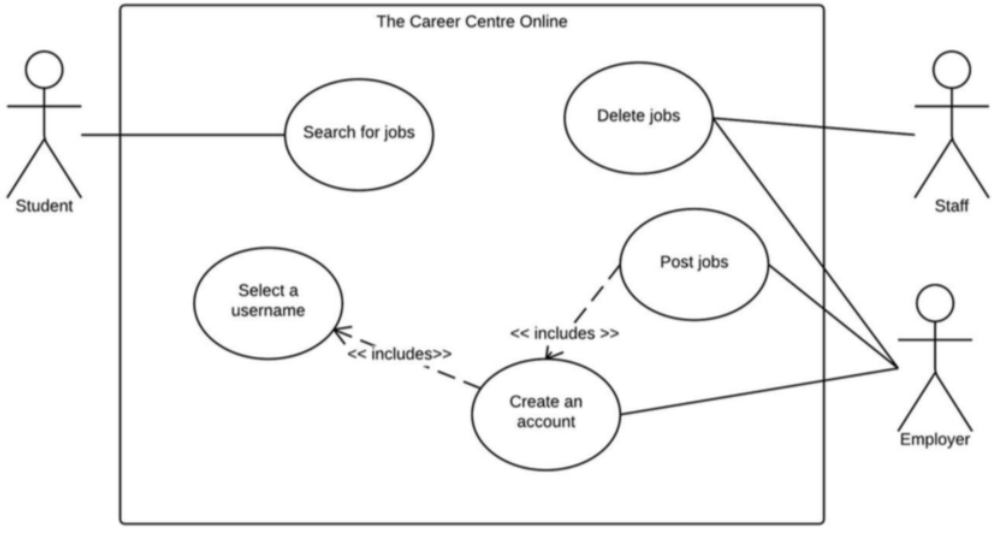
System Boundary A box or border that separates actors from use cases to denote that actors are external to the system

Pitfall: repeat use case

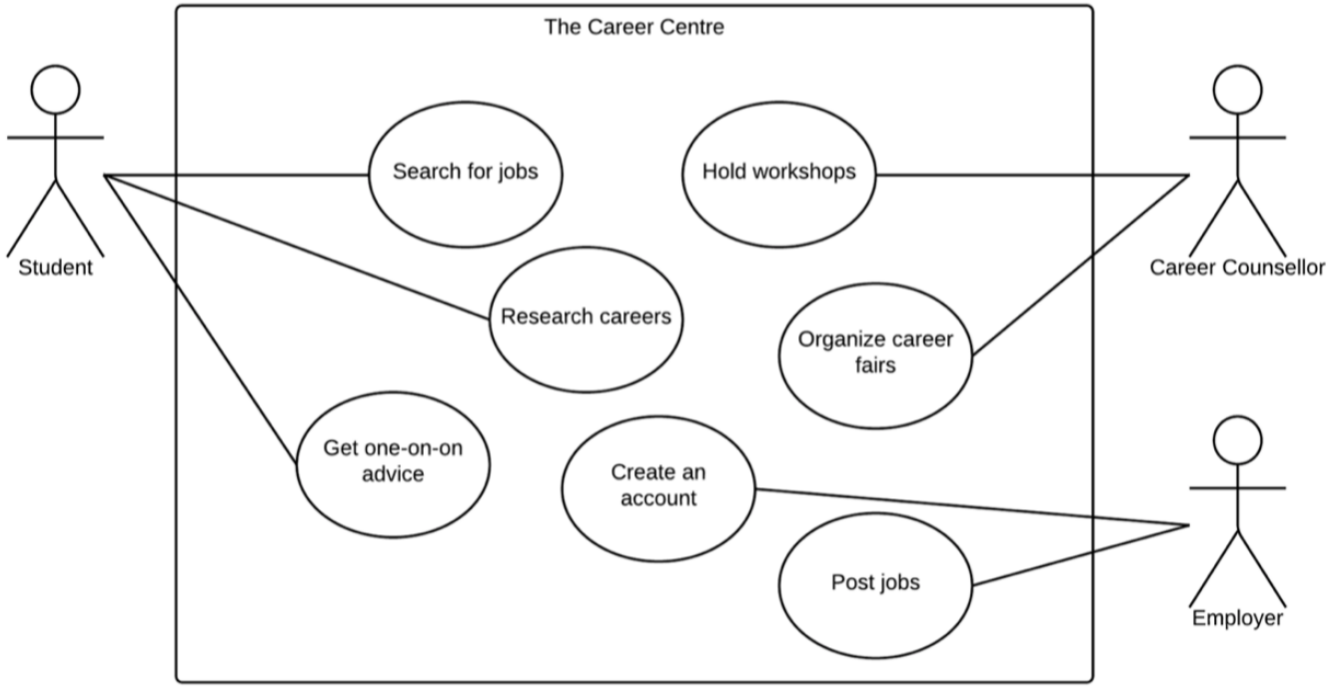


Pitfall: business system vs software system

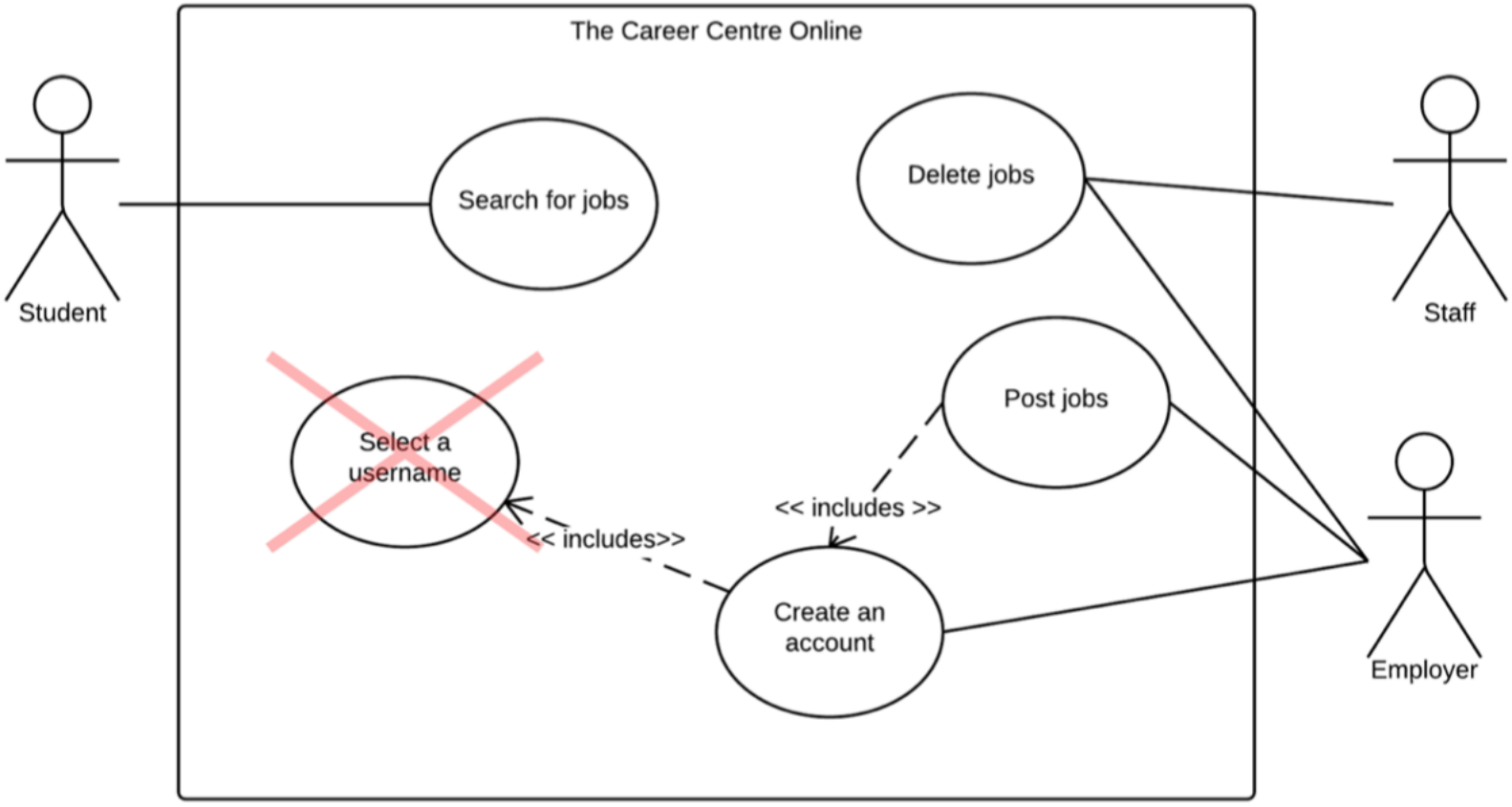




Pitfall: business system vs software system



Pitfall: inconsistent use case level



Exercise

Create a use case diagram for MarkUs.

Group Work

Tuckman's Stages of Group Development

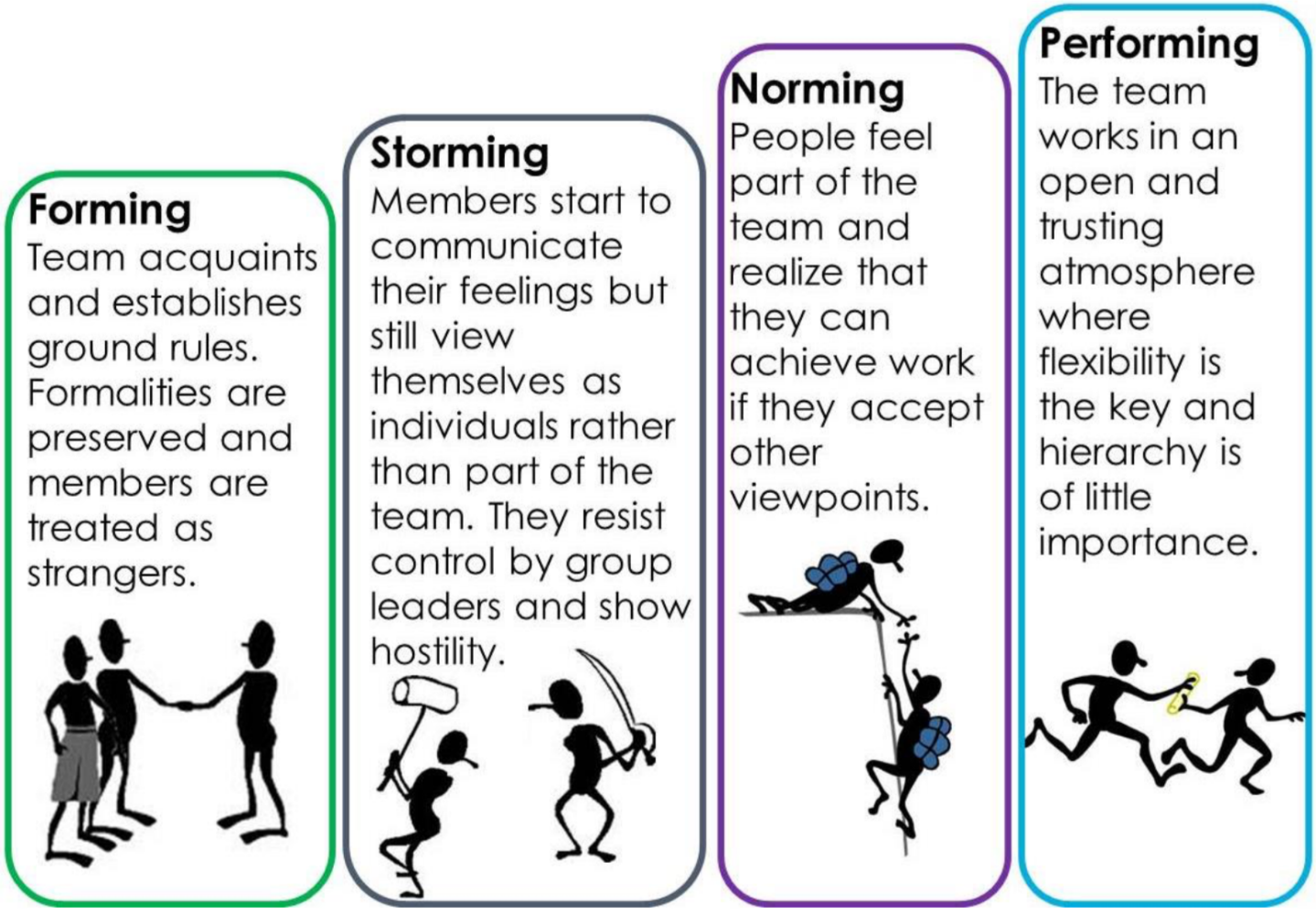


Figure 2: Modified from http://wheatoncollege.edu/sail/files/2011/12/groupDevelopment.jpg

Group Development

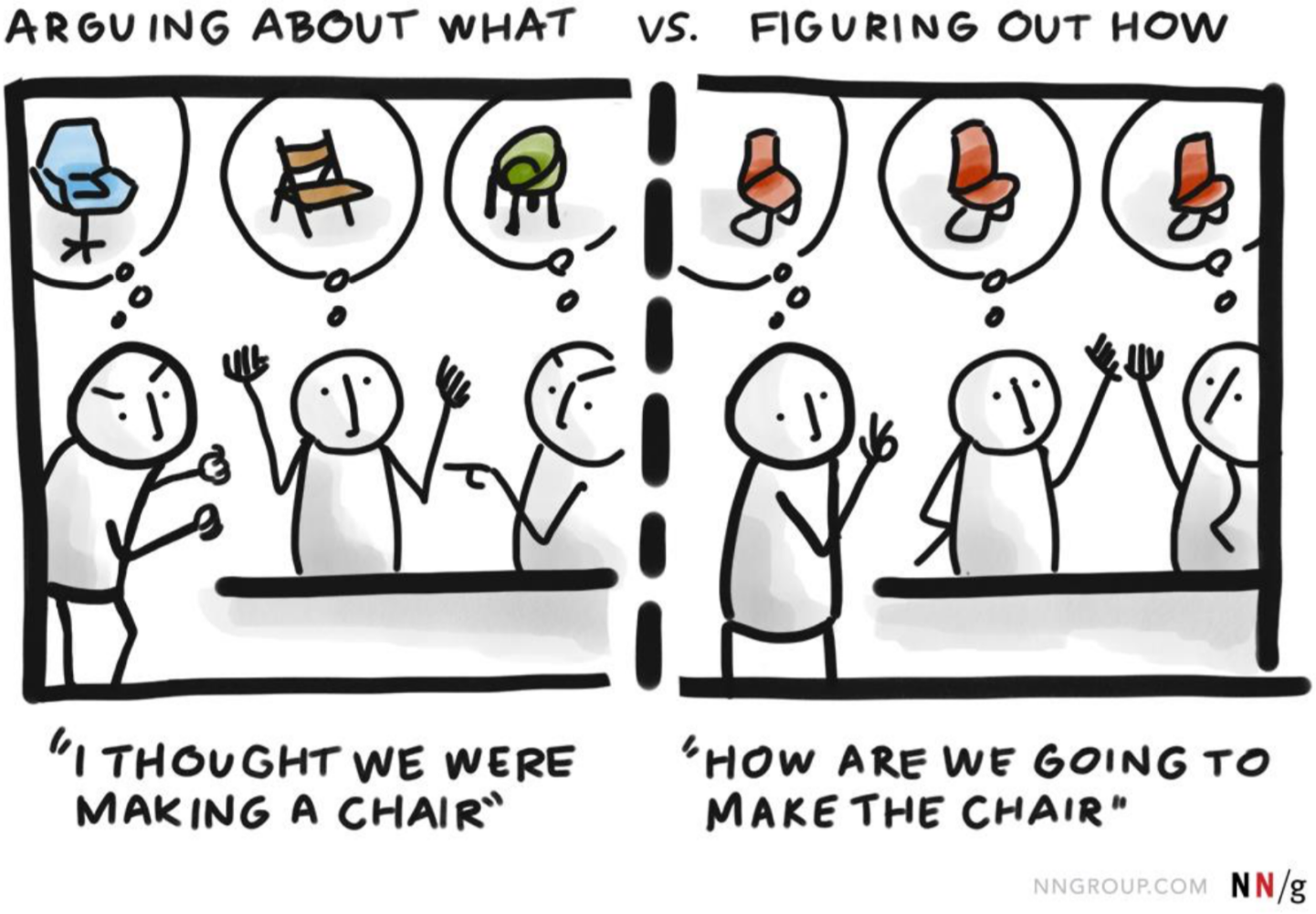


Figure 3: From https://www.nngroup.com/articles/design-thinking-team-building/

Some Effective Teamwork Strategies

- Get to know each other
- Exchange contact information
- Figure out how you prefer to communicate with each other
- Identify a common goal
- Define roles and responsibilities
- Address problems quickly
- Communicate, communicate, communicate

Qualities of a Well Functioning Team

- Roles are well defined
- Consensus can be reached
  - **Consensus**: moving forward with a decision regardless of agreement; "every team member can move forward without bad mouthing the decision"
- Respectful environment
- Organized

Qualities of a Good Team Member

- Is respectful
- Supportive of the team goals
- Understands his/her role in the team
- Contributes to the team according to his/her role

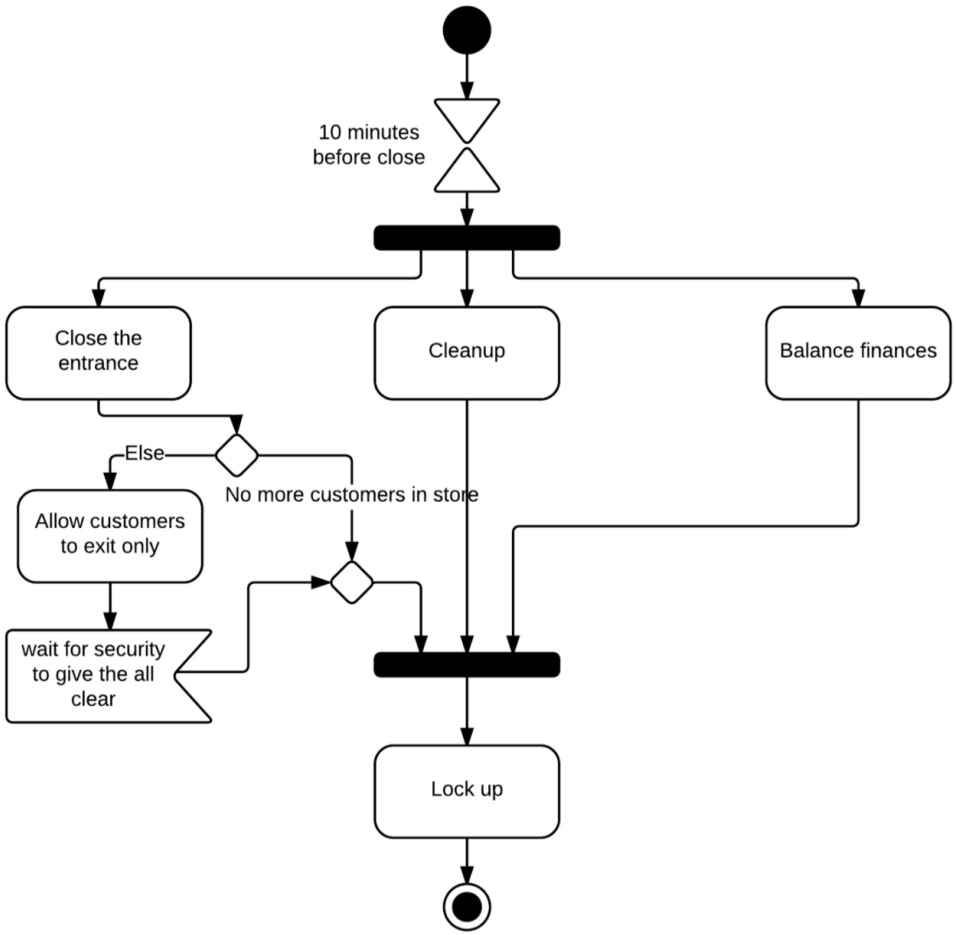
Communication Skills for All Team Members

- Can explain your own ideas clearly
- Can express your feelings in an open but non-threatening way
- **Listen** carefully to others
- Ask questions to clarify others' ideas & emotions
- Can sense how others feel based on their nonverbal communication
- Initiate conversations about group or process if you sense tensions brewing
- Reflect on activities & interactions of your group

Activity Diagram

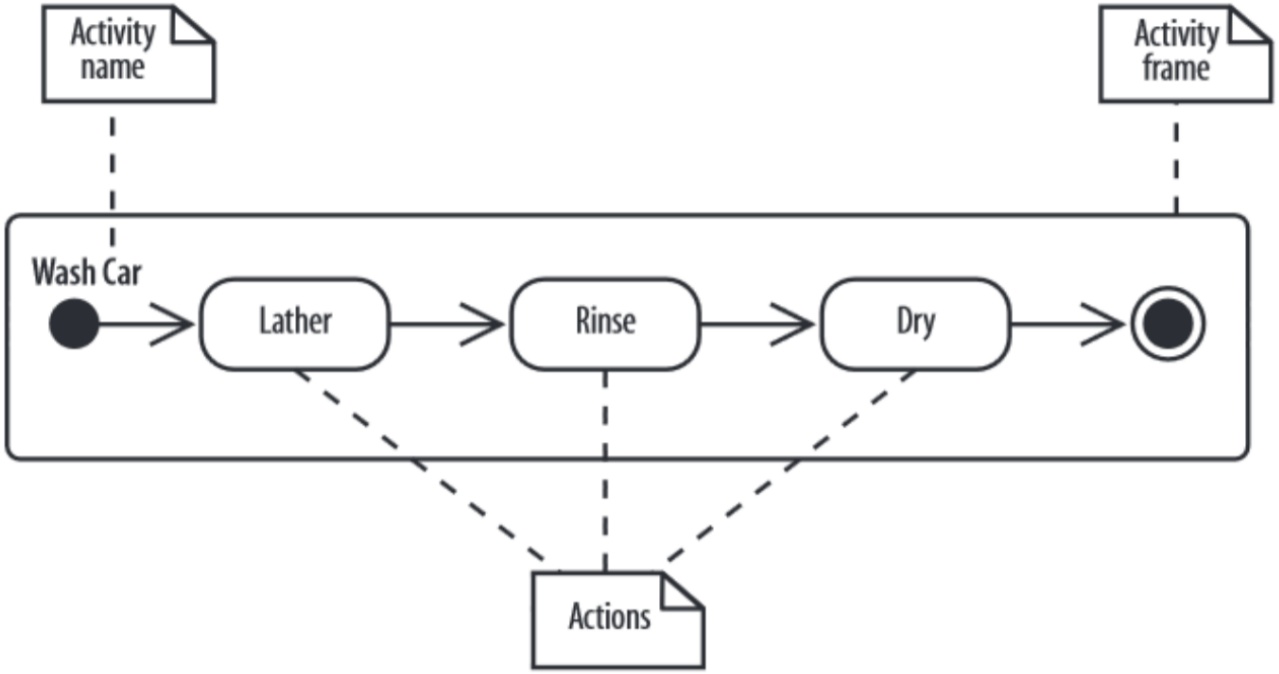
Activity Diagram

An **activity diagram** is used to describe procedural logic, business processes, or work flow.

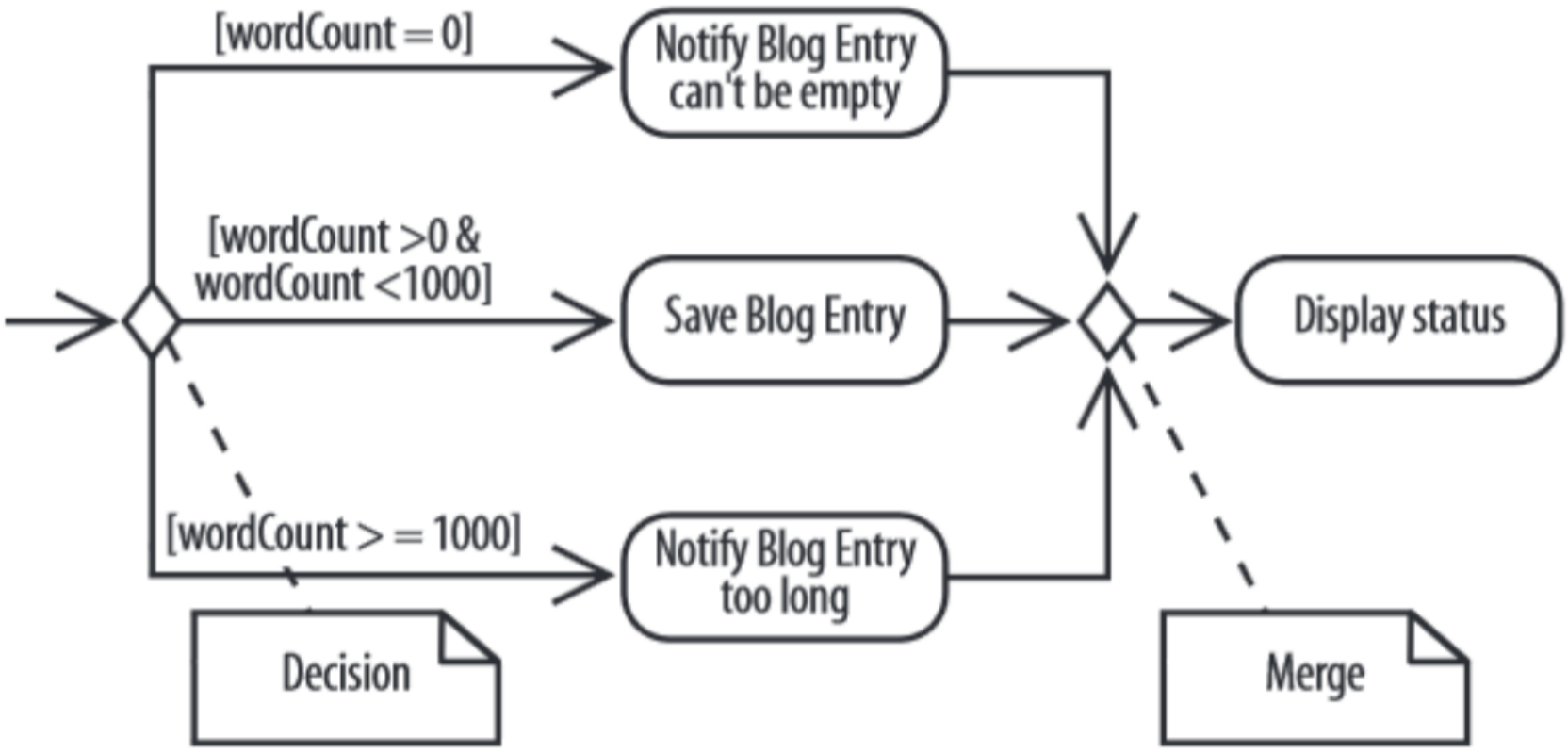


Activity vs Action

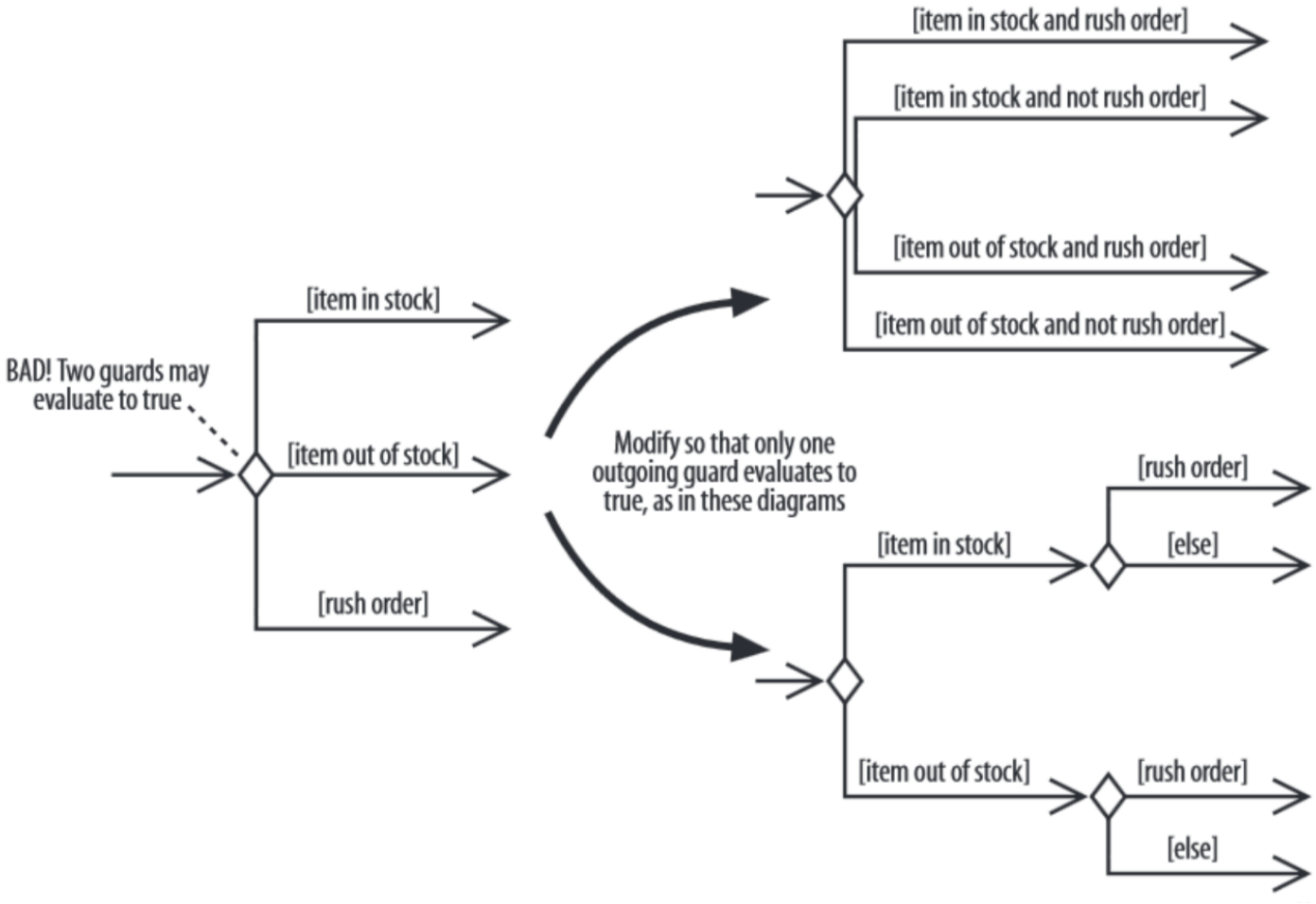
- **Activity**: a sequence of actions
- Nodes in the activity diagram are called **actions**



Decisions & Merges

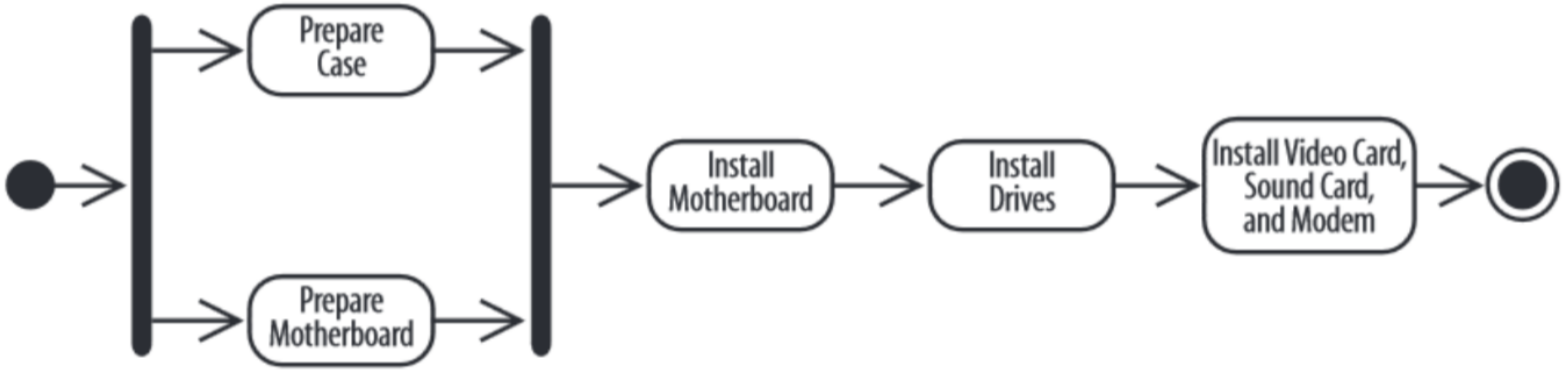


Decisions branches should be mutually exclusive

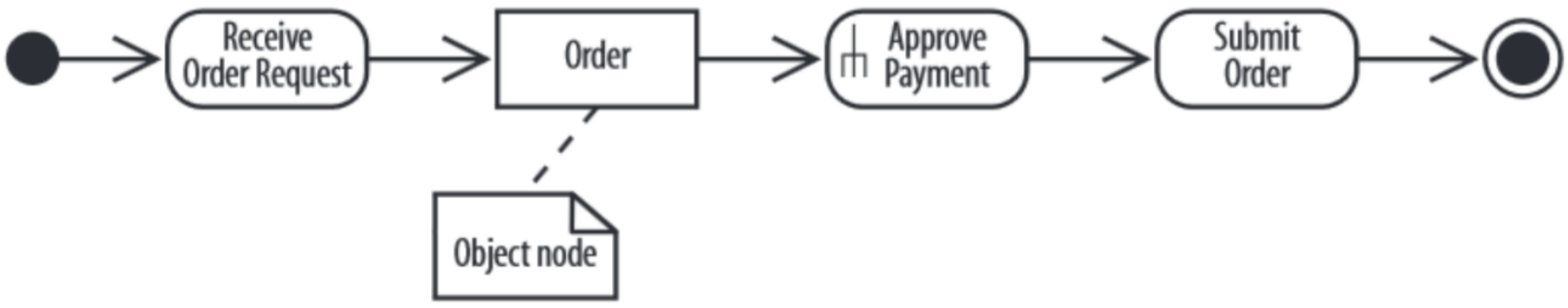


Forks & Joins

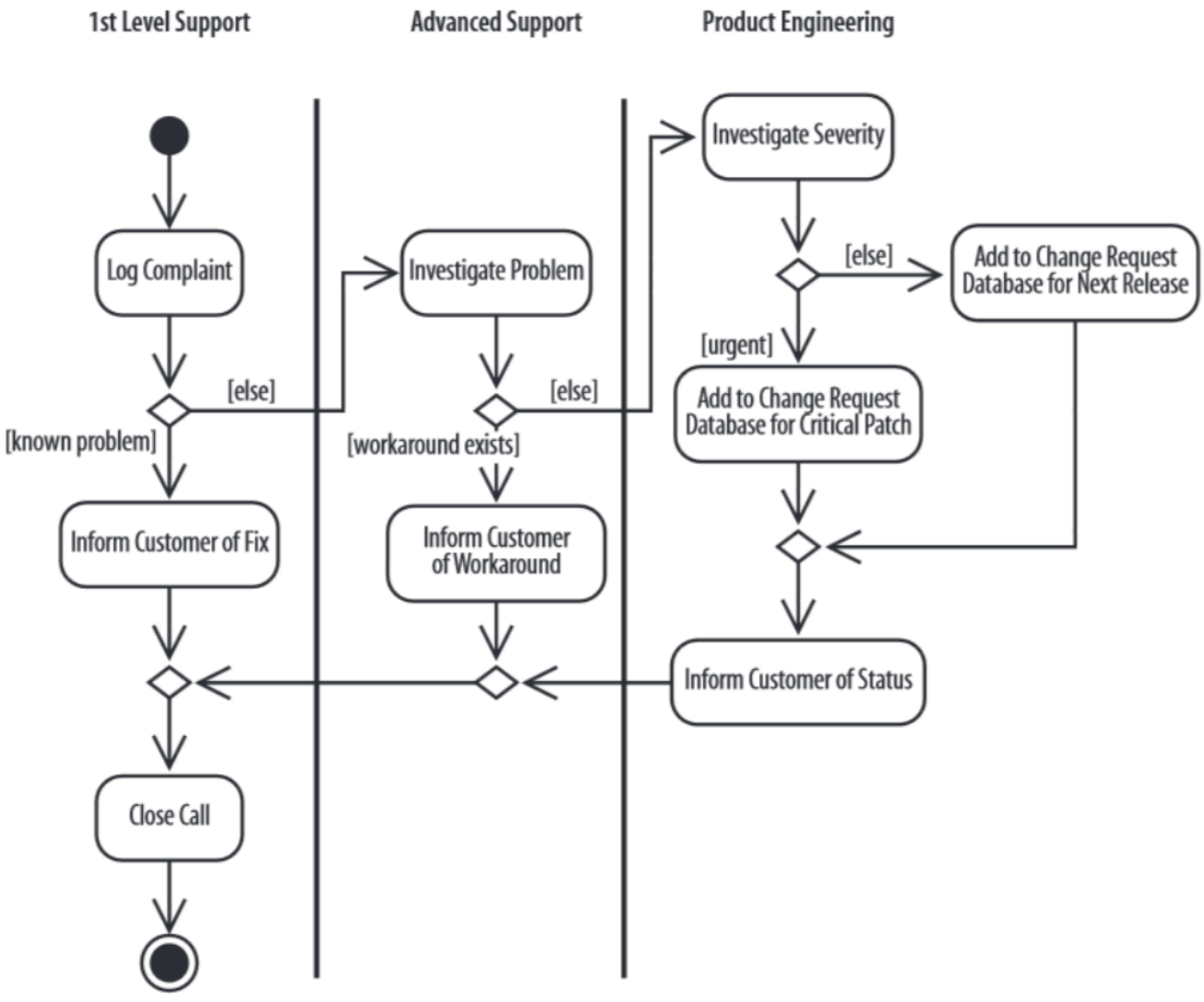
A **fork** represents the start of parallel actions. A **join** waits for all parallel actions to complete.



Object Nodes

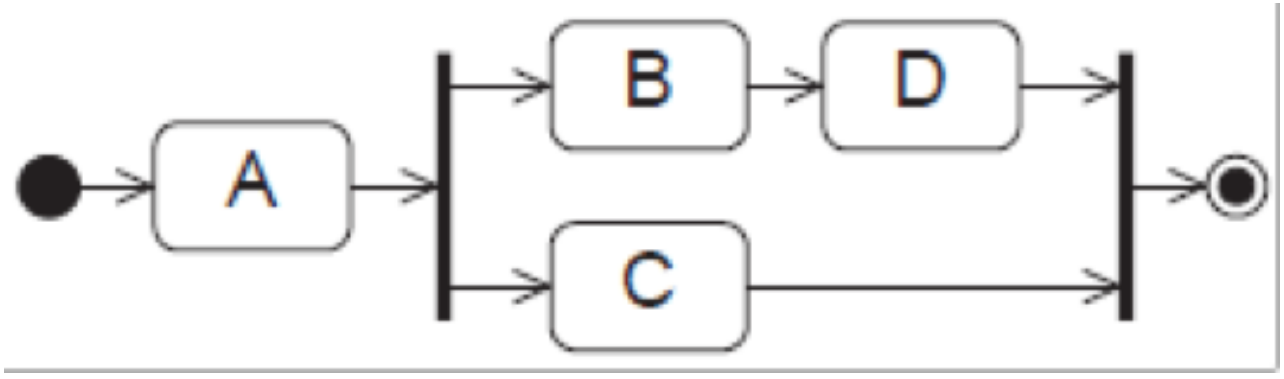


Partitions (Swimlanes)



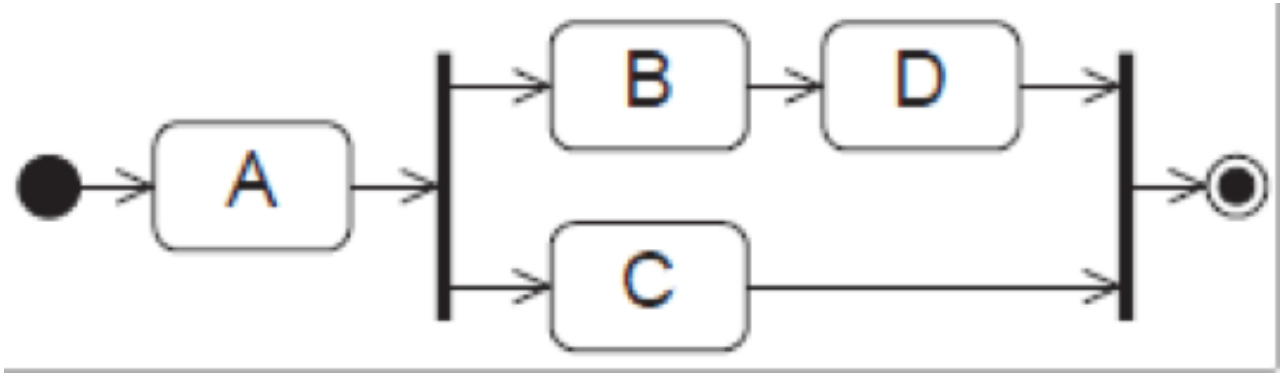
Exercise

List all possible action orders for the following activity.



Exercise

List all possible action orders for the following activity.



- A, B, D, C
- A, B, C, D
- A, C, B, D

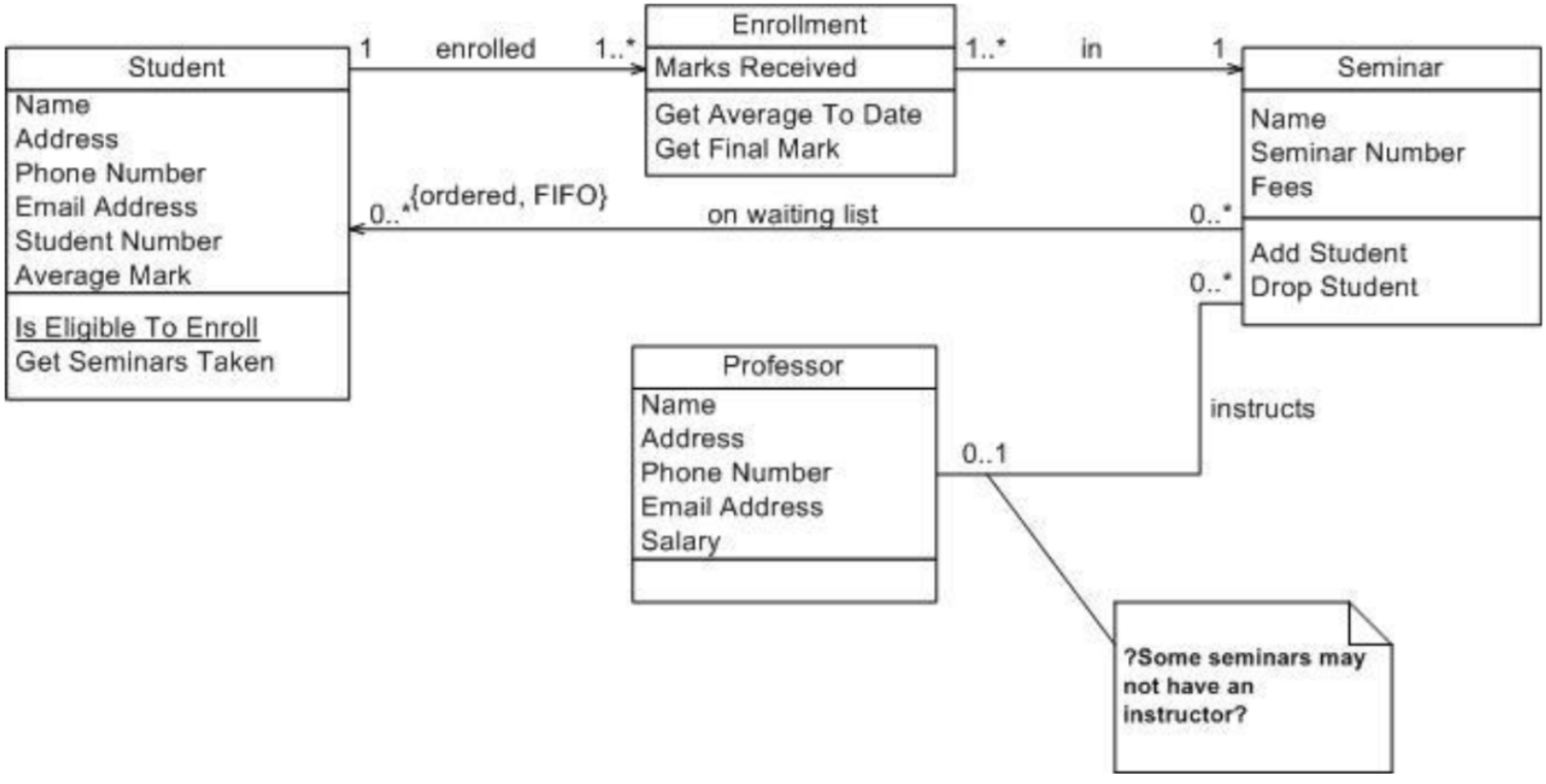
Exercise

Create an activity diagram for submitting a blog post on MarkUs.

Class Diagrams

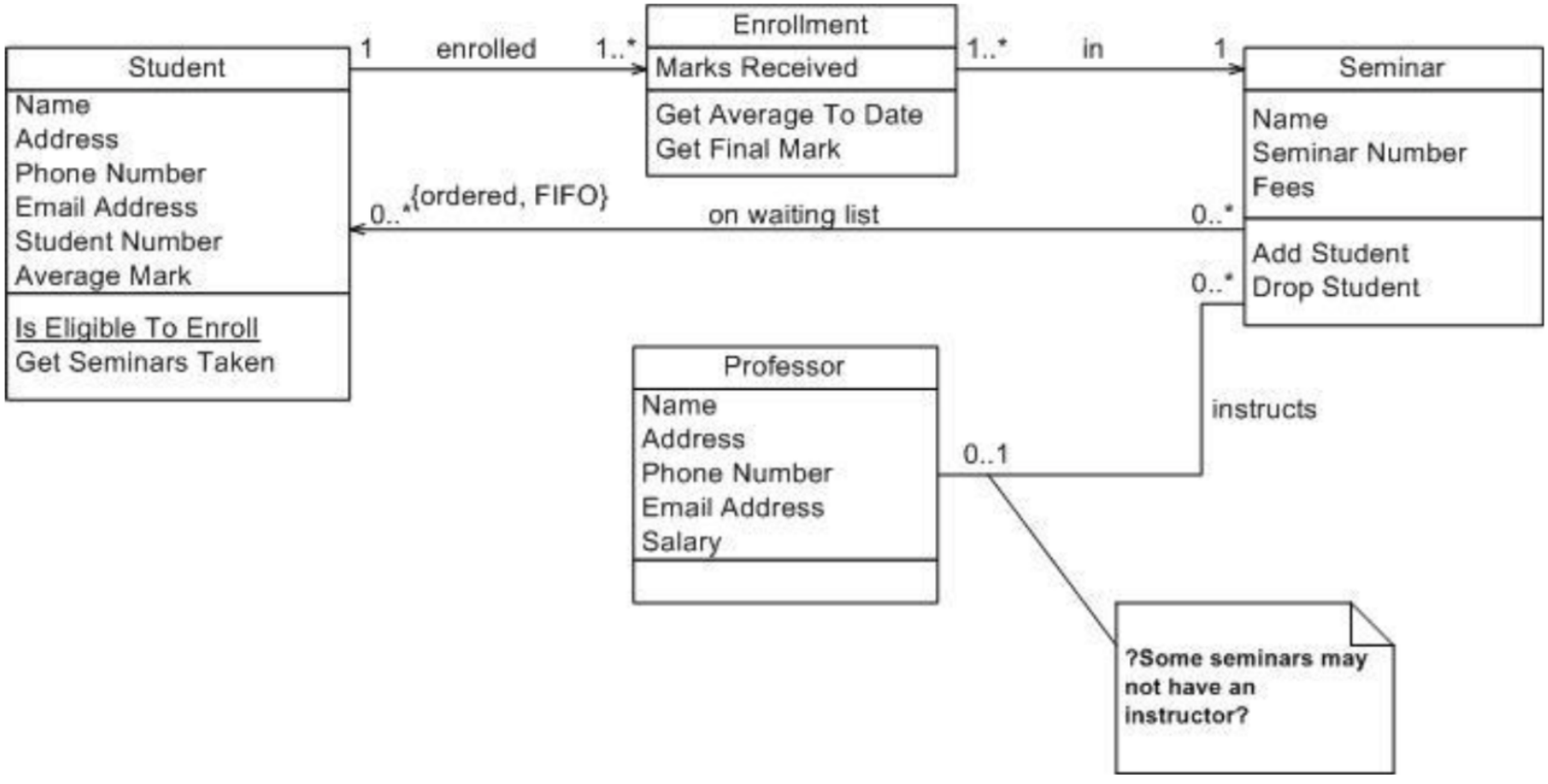
Class Diagram

A **class diagram** illustrates *classes* in a system, shows the *properties* and *operations* of a class, and relationships between classes.



Class Diagram

- What are the **classes** in this system?
- What are some **properties** (attributes)?
- What are some **operations** (methods)?
- What are some **relationships** between the classes?



Learning Outcomes

- Identify, read, and understand use case, activity, and class diagrams.
- Identify when to use use case, activity, and class diagrams.
- Be able to draw use case diagrams for familiar systems.
- Be able to draw activity diagrams for familiar activities.
- Identify issues with use case diagrams:
  - duplicate use case
  - poor use case naming
  - inconsistent use case level,
  - mixing business vs. software systems
- Identify issues with activity diagrams:
  - Missing nodes
  - Improper node type
  - Non-mutually exclusive branches

References

These slides borrow heavily from Nia McCash's CSC290 slides from Fall 2017.