COIS 2240
Software Design & Modelling

Lecture 6

UML Modelling II - Modelling Interactions and Behavior

Taher Ghaleb

TRENT UNIVERSITY

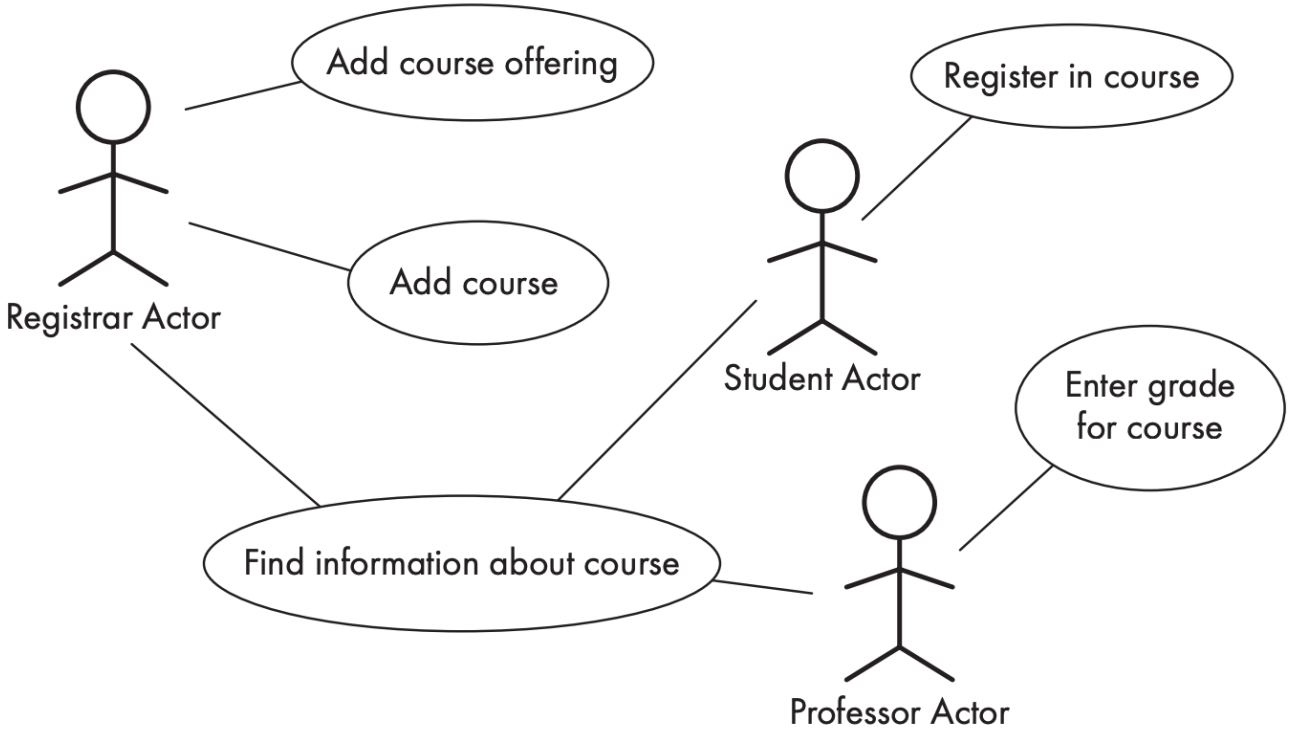
Parts of the content on the slides were sourced from / inspired by the textbook slides

Use Cases

A use case is a typical set of actions that a user performs to complete a given task

- **Objective:** to model the system from the point of view of
  … how users (actors) interact with this system
  … when trying to achieve their objectives (actions)
- It is one of the key activities in ***requirements*** analysis
- A use case model consists of:
  – a set of use cases, written in textual format
  – an *optional* description or diagram indicating how they are related

Use Case Diagrams



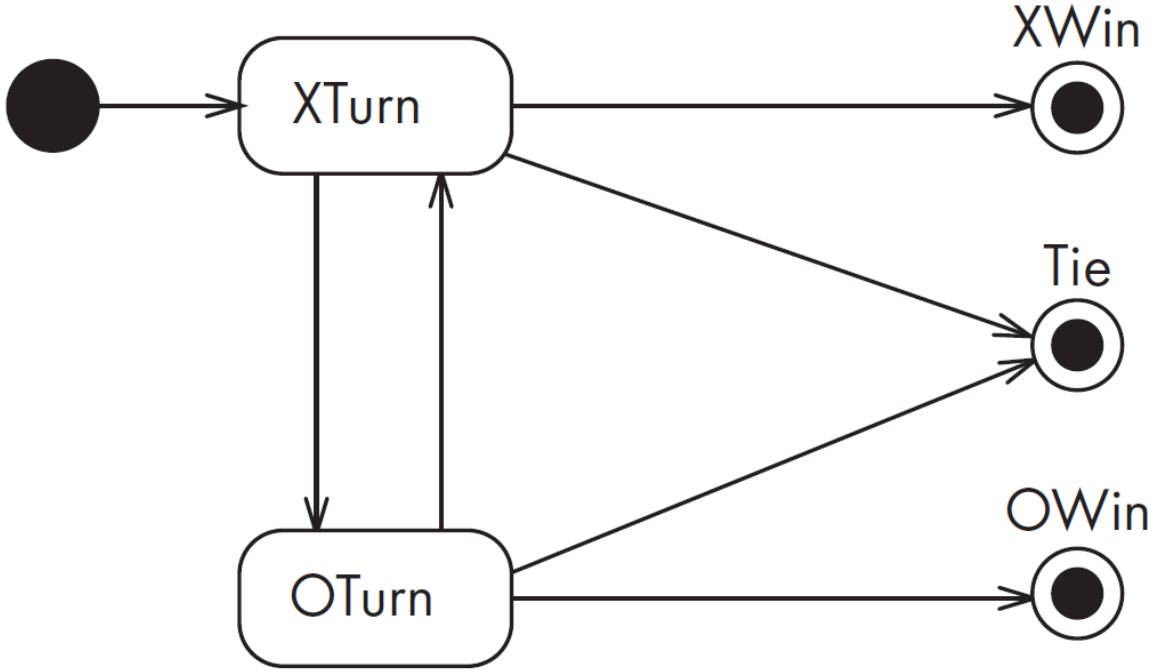
State Diagrams

A state diagram describes the behavior of a system, some part of a system, or an individual object

- At any given point in time, the system or object is in a certain state
  - Being in a state means that it will behave in a specific way in response to any events that occur
- Some events will cause the system to change state
  - In the new state, the system will behave in a different way to events
- A state diagram is a directed graph where the nodes are states, and the arcs are transitions

State diagrams – an example

- Tic-Tac-Toe game (X / O)



States & Transitions

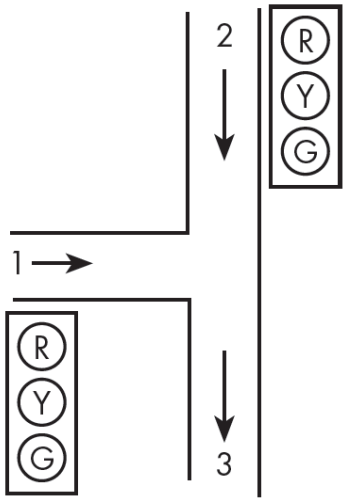
States

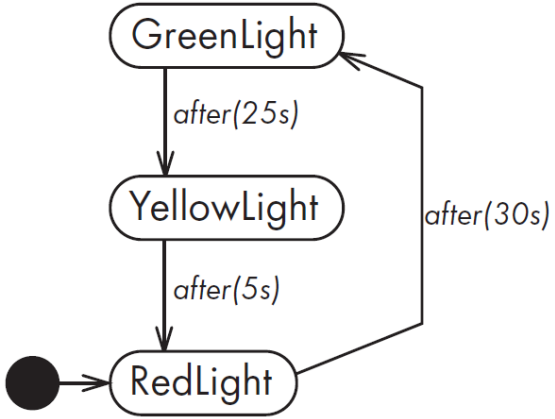
- At any given point in time, the system is in one state
- It will remain in this state until an event occurs that causes it to change state
- A state is represented by a rounded rectangle with the name of the state
- Special states:
  - A black circle represents the start state
  - A circle with a ring around it represents an end state

Transitions

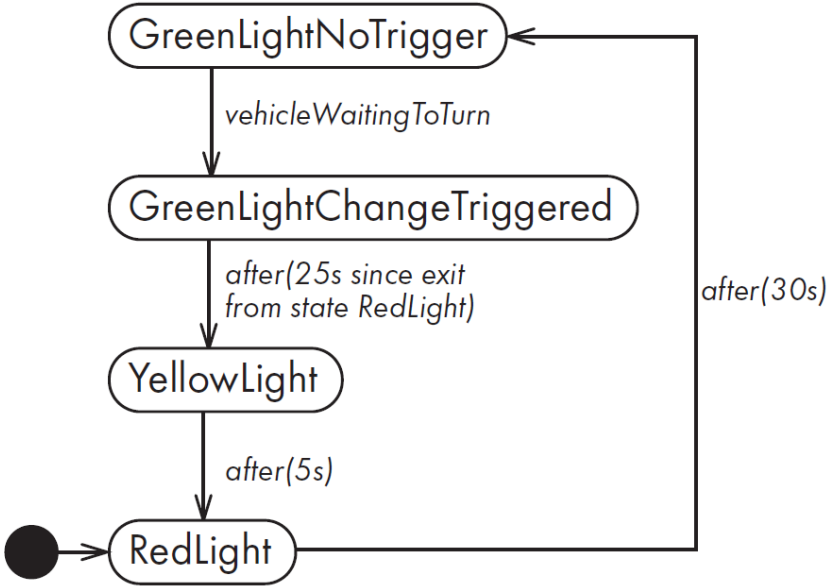
- A transition represents a change of state in response to an event
  - It is considered to occur instantaneously
- The label on each transition is the event that causes the change of state

Example of transitions with time-outs and conditions



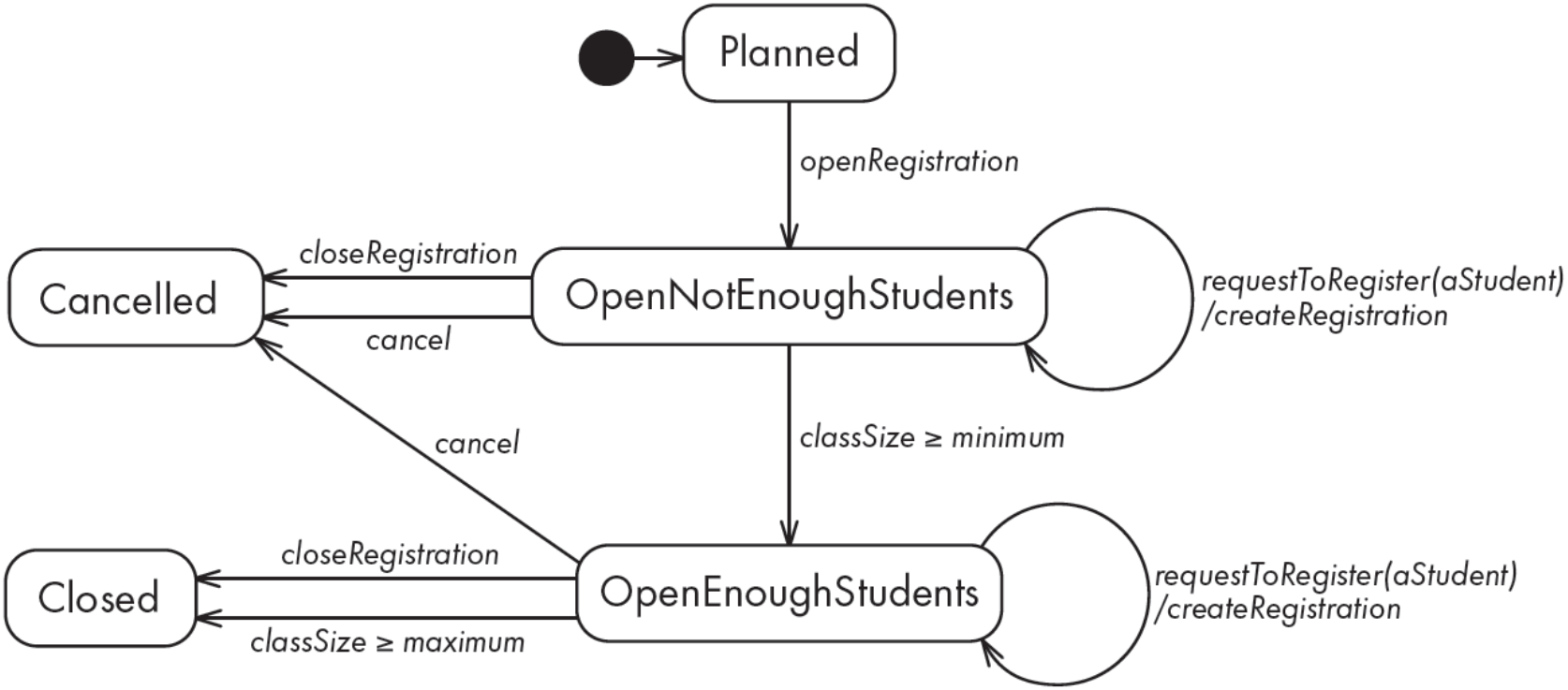


(a)



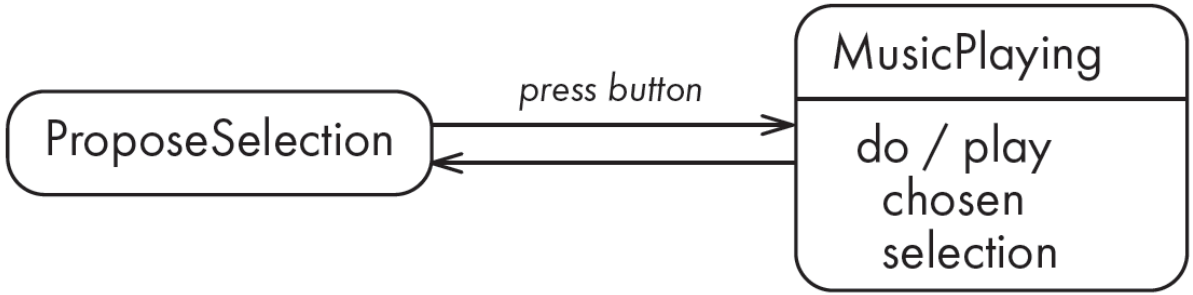
(b)

Example with conditional transitions



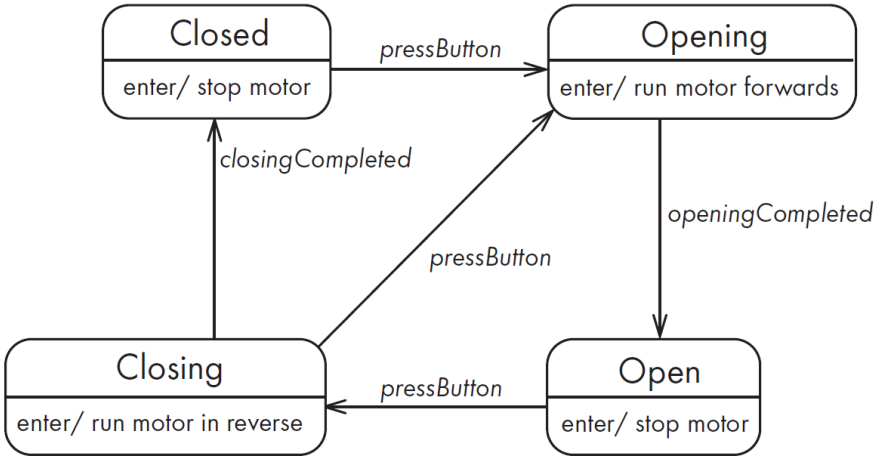
Activities in state diagrams

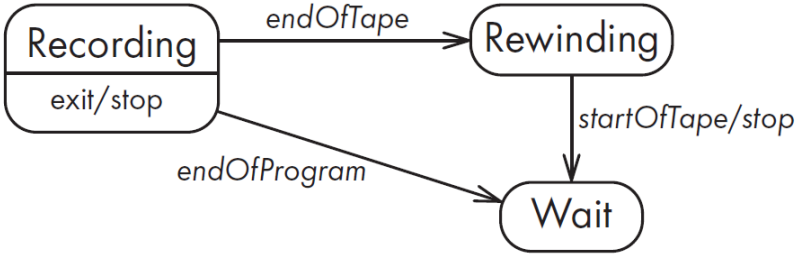
- An activity is something that takes place while the system is in a state
  - It takes a period of time
  - The system may take a transition out of the state in response to completion of the activity
  - Some other outgoing transition may result in:
    - The interruption of the activity
    - An early exit from the state



Actions in state diagrams

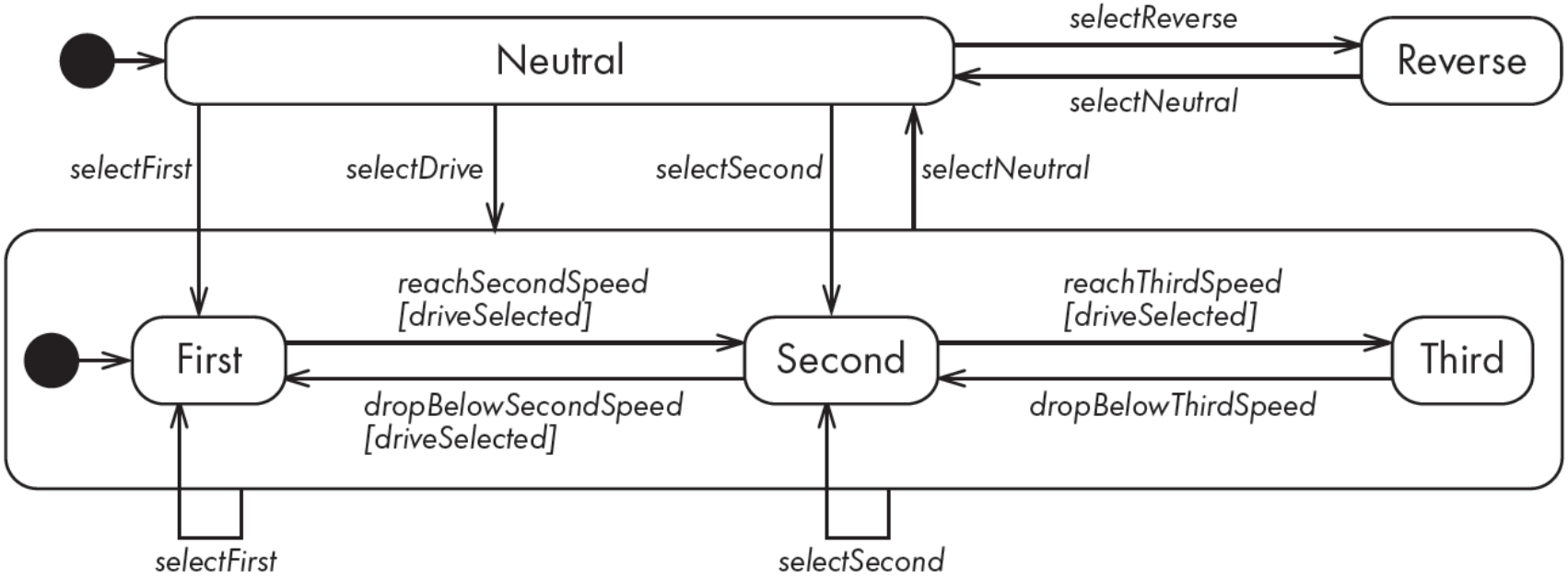
- An action is something that takes place effectively instantaneously
  - When a particular transition is taken,
  - Upon entry into a particular state, or
  - Upon exit from a particular state
- An action should consume non-noticeable amount of time



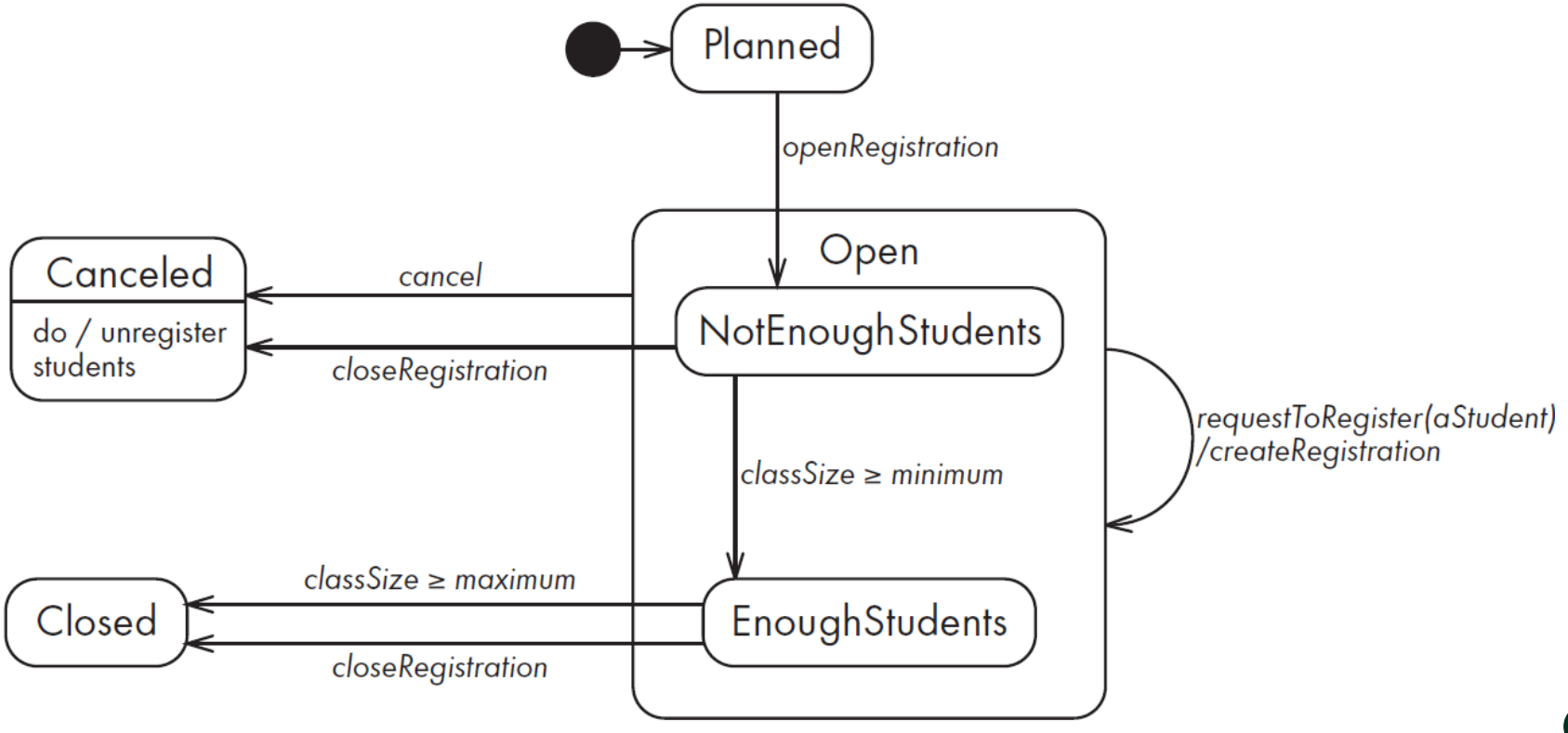


Nested substates and guard conditions

- A state diagram can be nested inside a state.
  - The states of the inner diagram are called substates



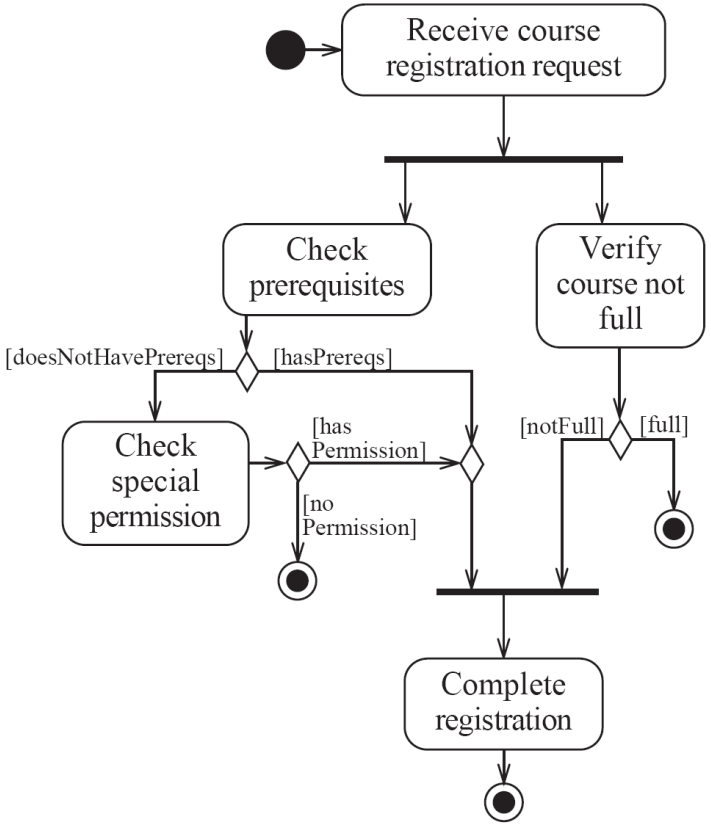
State diagram – an example with substates



Activity Diagrams

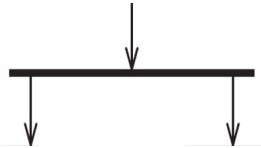
- An activity diagram is similar to a state diagram
  - Except most transitions are caused by internal events, such as the completion of a computation
- An activity diagram:
  - Can be used to understand the flow of work that an object or component performs
  - Can also be used to visualize interrelation and interaction between different use cases
  - Is most often associated with several classes
- One of the strengths of activity diagrams is the representation of concurrent activities

Activity diagrams – an example

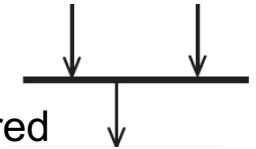


Representing concurrency

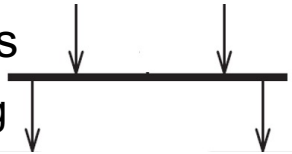
Concurrency is shown using forks, joins and rendezvous



- A *fork* has one incoming transition and multiple outgoing transitions
  - The execution splits into two concurrent threads



- A *join* has multiple incoming transitions and one outgoing transition
  - Outgoing transition is taken only when all incoming transitions have occurred
  - If one incoming transition occurs, a wait occurs until the other transitions occur



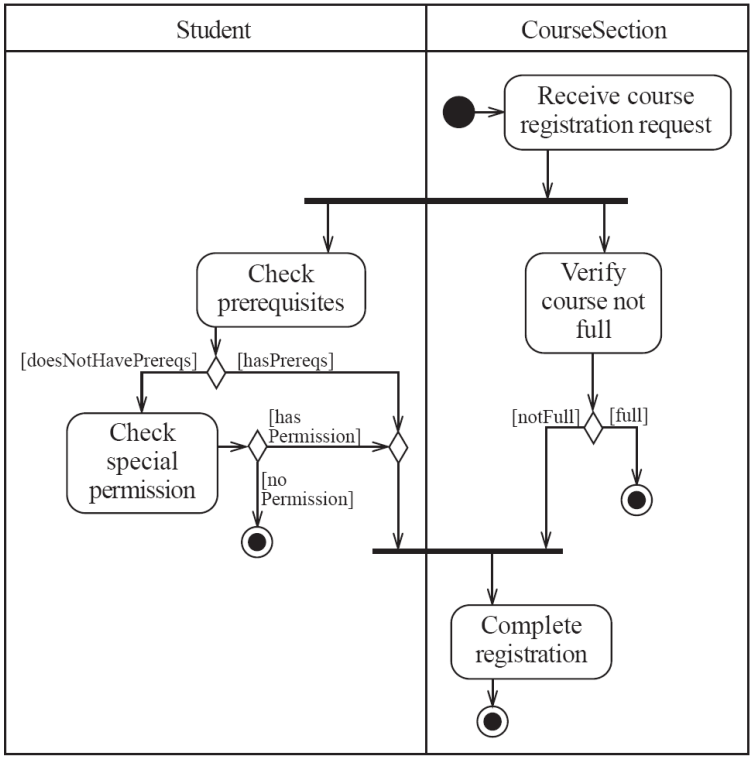
- A *rendezvous* has multiple incoming and multiple outgoing transitions
  - Once all the incoming transitions occur, the system takes all the outgoing transitions, each in a separate thread

Branching:

- *Decision* node: one incoming, multiple outgoing transitions (Boolean)
- *Merge* node: multiple incoming, one outgoing transitions

Swimlanes

- The partition of activities among the existing classes can be explicitly shown using swimlanes



State and Activity Diagrams

- You should use these diagrams for the parts of your system that you find most complex
  - I.e. not for every class
  - It could take too much time
- Activity and state diagrams help you create a correct implementation
- This is particularly true when behavior is distributed across several use cases
  - E.g. a state diagram is useful when different conditions cause instances to respond differently to the same event

Difficulties in Modelling System Behavior

Dynamic modelling is a difficult skill

- Large system have large numbers of possible paths a system can take
- It is hard to choose the classes to which to allocate each behavior:
  - Ensure that skilled developers lead the process, and ensure that all aspects of your models are properly reviewed
  - Work iteratively:
    - Develop *initial* class diagrams, use cases, responsibilities, and state diagrams
    - Then go back and verify that all of these are consistent, modify them as necessary
  - Drawing different diagrams that capture related, but distinct, information will often highlight problems