Software Engineering I CSC-382

Lecture 13

Software Engineering I CS-382

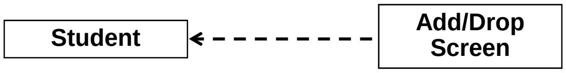
- Lecture 13
- What we will cover: (Details of Analysis Modeling)
  - Chapter 8 Sections 8.7, and 8.8 in Pressman
  - Goal is to continue to develop the methods and tools available from Object Oriented Analysis for analysis modeling

Dependencies

- Both Associations and Aggregations are persistent relationships
- We also need to define and represent the **transitory** relationships as well
  - We do this through **Dependencies**

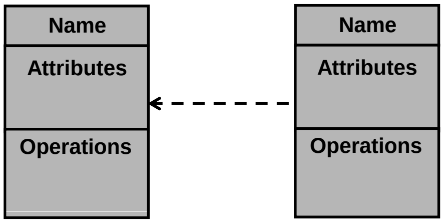
Dependencies II

- An example is the client-server relationship that exists between two classes.
  - The relationships exists only until the responsibility is satisfied and then it ends
  - Often one of the participants is a transitory object as well



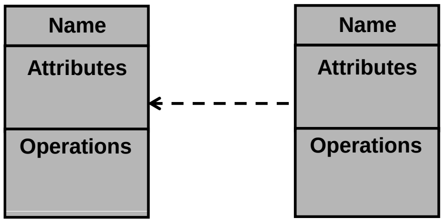
In Class Dependencies Diagram

Can we think of any possible dependencies in our SafeHome ?

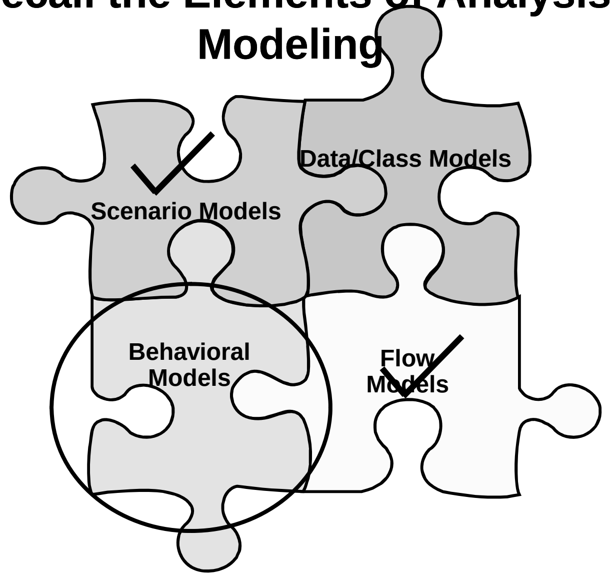


In Class Dependencies Diagram II

Can we think of any possible dependencies in our Registrar System ?



Recall the Elements of Analysis Modeling



Static Modeling

- The modeling up to this point has been **static**
  - We were able to show all the collaborations between objects using our aggregation, association, and dependency diagrams
  - Note these can be combined onto a single **Class Model** diagram
  - Recall collaborations occur via passing of messages

Dynamic Modeling

- Now we want to build a **dynamic** model
  - This dynamic model will provide us details into the actual messages that are being transferred between the objects
  - Initially we will have rough definitions of these messages, but as we move thru to the design of the software they will be detailed further
- The behavioral model indicates how software will respond to external events or stimuli.

Steps to Creating the Behavioral (Dynamic) Model

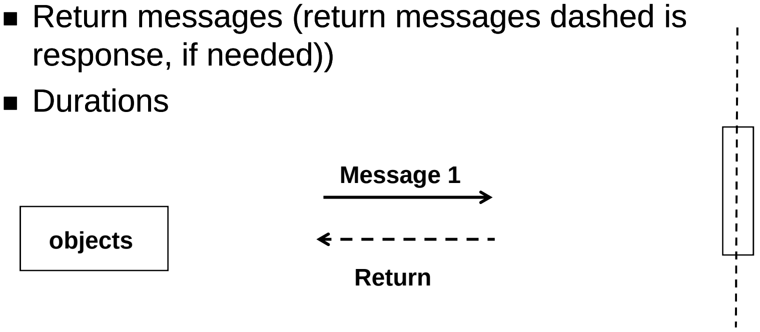
1. Evaluate all use-cases to fully understand the sequence of interaction within the system.
2. Identify events that drive the interaction sequence and understand how these events relate to specific objects.
3. Create a **sequence diagram** for each use-case.
4. Build **state diagrams** as needed to model the behavior within any of the objects that must generate a message.
5. Review the behavioral model to verify accuracy and consistency.

The Sequence Diagram

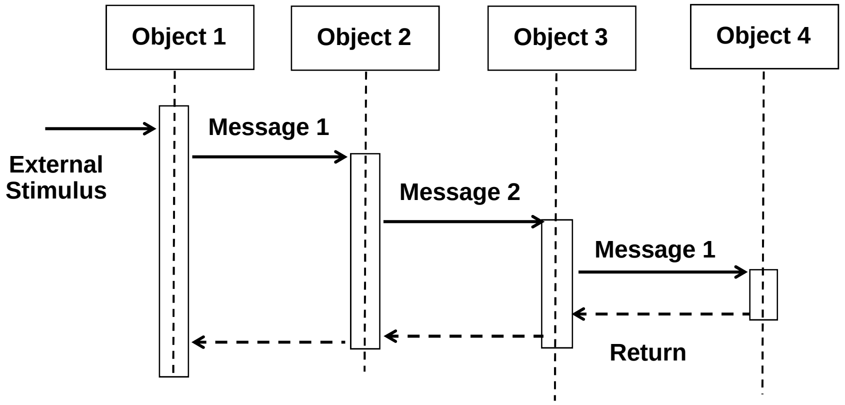
- The sequence diagram adds the dynamic information missing in the Class Diagrams shown earlier
- The textbook defines a relatively complex Sequence Diagram
  - Again I think it is better to simplify things a little and only show the messages on this diagram.
  - Show the state diagrams that generate the output messages from input messages on separate diagrams (to follow)
- Therefore our Sequence Diagram will have:
  - Objects
  - Messages (solid arrow is message)
  - Returns (return messages dashed is response, if needed))

The Sequence Diagram II

- Our Sequence Diagram will have:
  - Objects
  - Messages (solid arrow is message)
  - Return messages (return messages dashed is response, if needed))
  - Durations



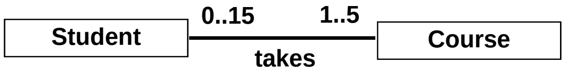
Overview of the Sequence Diagram



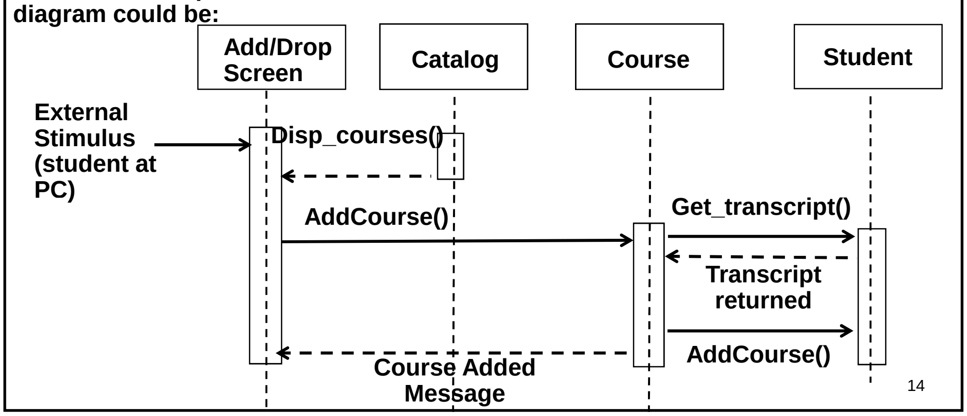
First draw the objects and the order and the message arrows, and then fill in the actual possible message names.

Example of a Sequence Diagram

Recall the student course association:



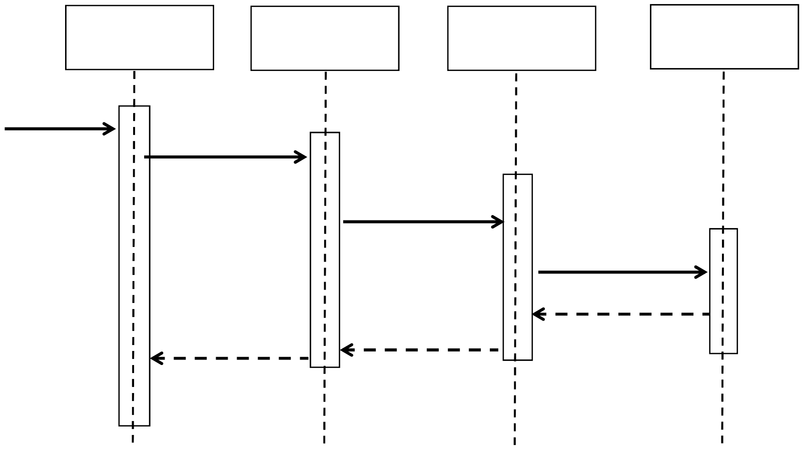
For a student to add a course the sequence diagram could be:



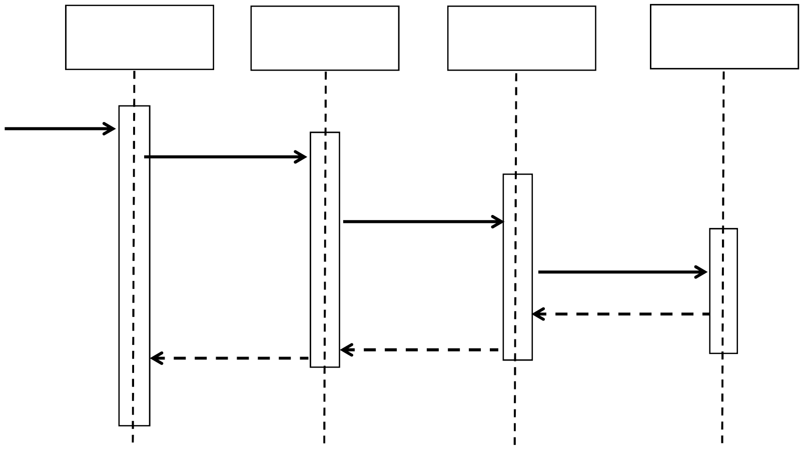
More on the Sequence Diagram

- Internally to the course there should be some logic to decide if student can take class
  - e.g. meets pre-reqs, section is not full, etc.
- We model this with State Diagrams (next)
  - They will provide a means for modeling the decision logic that takes place within the methods of the various classes we are designing

In-Class Sequence Diagram for SafeHome



In-Class Sequence Diagram for Student Loan



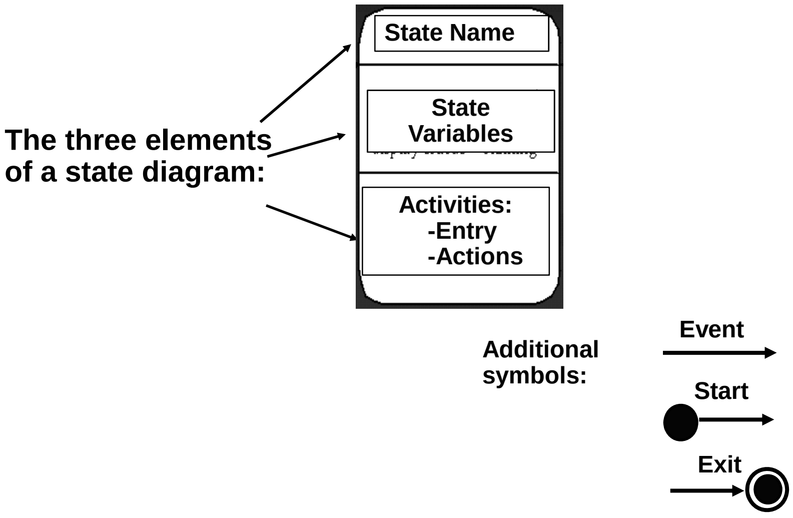
State Representations

- In the context of behavioral modeling, two different characterizations of states must be considered:
  - the state of each class as the system performs its function and
  - the state of the system as observed from the outside as the system performs its function

State Representations II

- The state of a class takes on both passive and active characteristics [CHA93].
  - A **passive state** is simply the current status of all of an object’s attributes.
  - The **active state** of an object indicates the current status of the object as it undergoes a continuing transformation or processing.

The Elements of the State Diagram



The three elements of a state diagram:

State Name

State Variables

Activities:
-Entry
-Actions

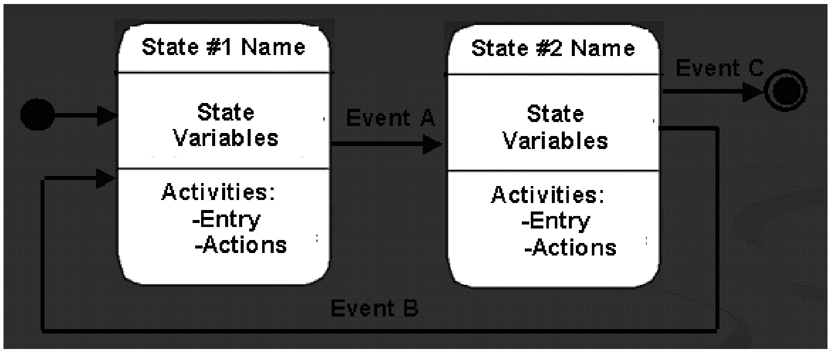
Additional symbols:

Event

Start

Exit

The Elements of the State Diagram



In-Class State Diagram for SafeHome

For Next Class

- Begin Chapter 9 - Design