Lecture 19: Intro. to Design (Part 2)

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2002

# Credit where Credit is Due

- Some material presented in this lecture is taken from section 6 of Maciaszek's "Requirements Analysis and System Design". © Addison Wesley, 2000



# Goals for this Lecture

- Cover the material presented in Section 6.2 of the textbook
  - Introduce
    - Detailed Design
      - take analysis models and elaborate them with technical details
    - Collaboration
      - a term the UML uses to refer to sets of objects ccollaborating to perform a task



# Last Lecture

- Covered
  - Architectural Design
    - Choose a software architectural style
    - Databases and BCED
  - Reuse Strategy
    - Buy vs. Build
    - Granularity
      - toolkits, frameworks, patterns
  - Components
    - Comparison vs. package, class, interface
  - Nodes
    - Deployment Diagrams

# Detailed Design

- In OO A&D, detailed design is a direct continuation from analysis
  - Our objective is to transform analysis models into design models that can be implemented by developers
- Architectural design impacts detailed design by selecting a target hardware/software platform (or platforms) and by selecting the components that will deploy our implemented design into the "real world"



# Detailed Design, continued

- In analysis, we simplify models by abstracting away (or deferring) technical details that would either
  - get in the way of understanding our application domain, or…
  - lead us down an implementation path too early and hence constrain our choices later in development
- In detailed design, we do the opposite
  - we start with analysis models and add technical details, or…
  - "drill down" a layer of abstraction on a particular analysis model and start creating a "design time" model from scratch

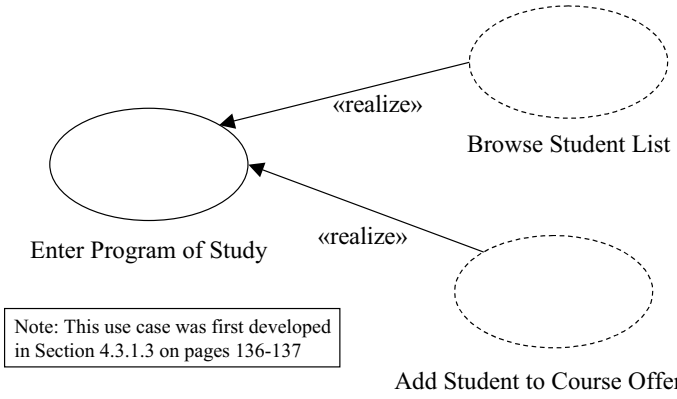


# Collaboration

- The UML uses the term "collaboration" to refer to sets of objects collaborating to perform a task
  - In particular, collaborations are used to specify the *realization* of use cases and operations
- Collaborations are notated as ellipses with dashed borders (see next slide)



# Collaboration Example




Note: This use case was first developed in Section 4.3.1.3 on pages 136-137

# Comments on Example

- Each collaboration needs an associated model that displays the details of the collaboration
  - Think "Interaction Diagram"
  - In particular, a collaboration diagram
    - Similar to a sequence diagram (indeed one can be converted into the other) but emphasizing different aspects of the collaboration
      - sequence diagrams emphasize the order of messages between objects
      - collaboration diagrams emphasize the associations between objects and the messages that flow over these associations





# Collaboration Diagrams

- Collaboration diagrams consist of objects
  - object names are interpreted as "role : class"
  - collections are shown as "stacks of objects"
- Associations between objects are shown; messages travel across the association in the direction indicated
  - Messages can be numbered to show the exact order in which messages are generated
  - As with other UML diagrams, messages sent to collections can be prefixed with an asterick ("*") to indicate that the same message is sent to each member of the collection
- The collaboration diagram in Fig. 6.14 (pg. 210) corresponds to the sequence diagram in Fig. 2.33 (pg. 66)





# Messages

- Figure 6.15 on page 211 shows three alternative approaches to rendering message information
  - I prefer the second approach
- In summary, messages can make use of
  - in and out parameters
  - explicit return value indicated
  - data tokens (think of these as OIDs being passed across the association with the message)





# Misc. topics on Messages

- Types of Messages (Section 6.2.2.3)
  - Read Messages
  - Update Messages
  - Collaborative Messages
  - This section is not all that useful, except as perhaps a brainstorming tool for adding operations to classes
- Overriding versus overloading (Section 6.2.2.4)
  - A method can be overridden as discussed in chapter 5
  - A method can also be overloaded
    - Same method name, different set of parameters
  - Again, this section is not all that useful; Maciaszek seems to be using it as an opportunity to say that developing message signatures is an important task for detailed design

# Implementation Note

- Figures 6.16 and 6.17 provide an excellent example of how associations (and hence collaborations) are implemented between classes
- Associations between a class and a collection, will result in an attribute in the class with a collection-based type (List, Set, HashTable, …) parameterized with the particular type of object held in the collection



# More on Messages

- Self messages
  - Collaboration diagrams can indicate self messages…in which an object sends a message to itself
    - This is shown in Figure 6.18 on page 215
- Asynchronous messages
  - Asynchronous messages are notated using a half arrow notation (See Fig. 6.19)
    - With asynchronous messages, senders do not block on a target object's response; they simply send the message and move on
- Callbacks
  - Callbacks allow target objects to send messages back to a previous message sender (see figure 6.20); in order to work, the sender's OID must be passed to the target object



# Realization of Use Cases

- Collaborations are to design what use cases are to analysis
  - They help "drive" their respective stages
  - However, due to differences in abstraction, typically multiple collaborations are needed to realize a single use case
- Collaborations consist of a structural part and a behavioral part
  - The structural part is the subset of the class diagram that covers each of the objects participating in the collaboration
    - developing a collaboration during design will lead to the original class diagram being updated with new operations along with operation signatures
  - The behavioral part is an interaction that defines the specific interaction of the collaboration's objects



# Realization of Operations

- Collaborations can also be used to model realizations of complex operations
  - however, activity diagrams will often suffice
  - alternative approach for simpler operations
    - attach pseudocode to operation using a UML note
- Examples of Realization: pages 217-221