Collaboration Diagrams

Massimo Felici
Room 1402, JCMB, KB
0131 650 5899
mfelici@inf.ed.ac.uk
Interaction Diagrams

- UML Interaction Diagrams refine the kind of activity undertaken in checking with CRC cards.

- There are two different kinds of interaction diagrams:
  - Collaboration Diagrams
  - Sequence Diagrams

- There is some redundancy between Collaboration and Sequence Diagrams:
  - They differently show how elements interact over time.
  - They document in detail how classes realize user cases.
  - Collaboration Diagrams show relationship between objects.
  - Sequence Diagrams focus on the time in which events occur.
Collaboration Diagrams’ Rationale

- Model collaborations between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the passed messages between objects and roles within the collaboration
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes) that participate in use cases
Collaboration Diagrams: An Example

- Each message in a collaboration diagram has a sequence number.
- The top-level message is numbered 1. Messages sent during the same call have the same decimal prefix but suffixes of 1, 2, etc. according to when they occur.
Collaboration Diagrams

- **Specification level** shows generic cases of collaborations
  - *Generic form* captures a collaboration among class roles and association roles and their interactions

- **Instance level** shows a specific instance of an interaction taking place and involving specific object instances
  - *Instance form* captures a scenario among objects conforming to class roles and links conforming to association roles
What is a Collaboration?

- A **Collaboration** is a collection of named **objects** and **actors** with **links** connecting them. They collaborate in performing some task.

- A **Collaboration** defines a set of participants and relationships that are meaningful for a given set of purposes.

- A **Collaboration** between objects working together provides emergent desirable **functionalities** in **Object-Oriented systems**
  - Each object (responsibility) partially supports emergent functionalities
  - Objects are able to produce (usable) high-level functionalities by working together

- Objects collaborate by **communicating** (passing messages) with one another in order to work together.
Collaborations

- **Actors**
  - Each Actor is named and has a role
  - One actor will be the *initiator* of the use case

- **Objects**
  - Each object in the collaboration is named and has its class specified
  - Not all classes need to appear
  - There may be more than one object of a class

- **Links**
  - Links connect objects and actors and are instances of associations
  - Each link corresponds to an association in the class diagram
Interactions

- Use cases and Class Diagrams constrain interactions

- **Associations and Links** in a **Collaboration Diagram** show the paths along which **messages** can be sent from one instance to another

- A **message** is the specification of a **stimulus**

- A **stimulus** represents a specific instance of sending the **message**, with particular arguments
Messages

- **Message Signature**
  - return-value, message-name and argument-list

- **Message Flows**
  - **Procedural** or **Synchronous**: A message is sent by one object to another and the first object waits until the resulting action has completed.
  - **Asynchronous**: A message is sent by one object to another, but the first object does not wait until the resulting action has completed.
  - **Flat**: Each arrow shows a progression from one step to the next in a sequence. Normally the message is asynchronous.
  - **Return**: the explicit return of control from the object to which the message was sent.

- **Further information on Messages**
  - Sequence-expression, Predecessor, Guard-condition
Where should messages go?

- The message is directed from **sender** to **receiver**
- The receiver must understand the message
- The association must be **navigable** in that direction

**Law of Demeter**

- Dealing with a message \( m \) an Object \( O \) can send messages to:
  - Itself
  - Objects sent as argument in the message \( m \)
  - Objects \( O \) creates in responding to \( m \)
  - Objects that are directly accessible from \( O \), using attribute values
Activations: Flow of Control

- **Procedural interactions**
  - At most one object is computing at any time

- **Activation**
  - An object has a live activation from when it receives a message until it responds to the message

- **Waiting for response**
  - Synchronous messages on sending a message to another object, an object will wait until it receives a response

- **Activation task**
  - Activations are stacked and the top activation has control. When the top action responds the next to top regains control and so on...
Constructing Collaboration Diagrams

1. Identify **behavior** whose realization and implementation is specified

2. Identify the **structural elements** (class roles, objects, subsystems) necessary to carry out the functionality of the collaboration
   - Decide on the context of interaction: system, subsystem, use case and operation

3. Model **structural relationships** between those elements to produce a diagram showing the context of the interaction

4. Consider the **alternative scenarios** that may be required
   - Draw instance level collaboration diagrams, if required.
   - Optionally draw a specification level collaboration diagram to summarize the alternative scenarios in the instance level sequence diagrams
Reading/Activity

- Read Chapter 8, Collaboration Diagrams, of the Schaum’s Outlines UML book
- Read Chapter 9, Interaction Sequence Diagrams, of the Schaum’s Outlines UML book
Summary

- **Interaction Diagrams**
  - Collaboration Diagrams
  - Sequence Diagrams

- **Collaboration Diagrams’ Rationale**

- **Collaboration Diagrams**
  - Collaborations
  - Interactions
  - Messages

- **Constructing Collaboration Diagrams**