Realizing Use cases in the Design Model

Use case Model

Requirements
1.
2.
3.
4.
...

Use Case

Design Model

Collaboration

Participating Classes
Use-case driven design is a key theme in a variety of software processes based on the UML.

UML supports specific modelling constructs that realize use cases in the implementation.

Collaborations (Communications) enhance the systematic and aggregate behavioural aspects of the system.

Collaborations support traceability from requirements expressed in use cases into the design.
Communication Diagrams

- 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
Communication Diagrams

- The communication is implicit in a Sequence Diagram, rather than explicitly represented as in a Communication Diagram

- There is some redundancy between Communication and Sequence Diagrams
  - They differently show how elements interact over time
  - They document in detail how classes realize user cases
  - Communication Diagrams show relationship between objects
  - Sequence Diagrams focus on the time in which events occur
Slide 3: Communication Diagrams

• Communication Diagrams, formerly called Collaboration Diagrams.

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

Sequence and Communication Diagrams

Massimo Felici
What is a Collaboration?

Communication Diagrams

• A Collaboration is a collection of named objects and actors with links connecting them

• 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

• Objects collaborate by communicating (passing messages) with one another in order to work together
Slide 5: Communication Diagrams

- Objects and actors collaborate in performing some task. Each object (responsibility) partially supports emergent functionalities.

- Objects are able to produce (usable) high-level functionalities by working 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
Communication Diagrams

• Specification level shows generic cases of collaborations (communications)
  
  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
Lifelines and Links

• Participants on a collaboration diagram are represented by a rectangle

• The syntax for the name of a lifeline

  [connectable-element-name][‘[‘selector’]’][:class-name][decomposition]

• A communication link is shown with a single line that connects two participants
Slide 9: Lifelines and Links

- In UML 2.0 lifeline names are no longer underlined.
- UML 2.0 introduces rectangular frames around communication diagrams.
Communication Diagrams

Diagram 10
Messages

- A message on a communication diagram is shown using an arrow from the message sender to the message receiver.

- Message Signature: return-value, message-name, argument-list

- 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.
Messages

<table>
<thead>
<tr>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation</td>
<td>Reply</td>
</tr>
</tbody>
</table>
Messages

• 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.
Messages

• Messages occurring at the same time: Adding a number-and-letter notation to indicate that a message happens at the same time as another message

• Invoking a message multiple times: Looping constraint, e.g., *[i=0..9]*

• Sending a message based on a condition: A guardian condition is made up of a logical boolean statement, e.g., [condition=true]

• When a participant sends a message to itself
Messages

• 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
Slide 15: Messages

Suggested Readings

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...
Creation and Deletion

• In Sequence Diagrams, it is possible to use the lifelines
  – New objects have their icon inserted when they are created
  – Destroyed objects have their lifeline terminated with $\times$

• In Communication Diagrams the objects are labelled:
  – New for objects created in the collaboration
  – Destroyed for objects destroyed during the collaboration
Slide 17: Example

1: n := getName()

2: new DirectorOfStudies (n)

3: destroy()

: Lector
: UTO

: DirectorOfStudies

{new}

{destroyed}
# Communication vs. Sequence Diagrams

<table>
<thead>
<tr>
<th></th>
<th>Communication Diagrams</th>
<th>Sequence Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Links</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Message Signature</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parallel Messages</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Asynchronous messages</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Message Ordering</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Create &amp; Maintain</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Slide 18: Communication vs. Sequence Diagrams

- Shows participants effectively: Both Communication and Sequence diagrams show participants effectively
- Showing the links between participants: Communication diagrams explicitly and clearly show the links between participants
- Showing message signatures: Both Communication and Sequence diagrams show messages effectively
- Support parallel messages: Both Communication and Sequence diagrams show parallel messages effectively
- Support asynchronous messages: Sequence diagrams explicitly and clearly show the links between participants
- Easy to read message ordering: Sequence diagrams explicitly and clearly show message ordering
- Easy to create and maintain: Communication diagrams do have the edge on the ease-of-maintenance
Constructing Communication Diagrams

1. Identify behaviour

2. Identify the structural elements

3. Model structural relationships

4. Consider the alternative scenarios
Slide 19: Constructing Communication Diagrams

1. Identify behaviour 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 summarise the alternative scenarios in the instance level sequence diagrams
Readings

Required Readings

• UML course textbook, Chapter 10 on More on Interaction Diagrams

Suggested Readings

Summary

• Interaction Diagrams
  – Sequence Diagrams
  – Communication Diagrams

• Communication Diagrams Rationale

• Communication Diagrams
  – Collaborations
  – Interactions
  – Messages

• Constructing Communication Diagrams