204362 – Object-Oriented Design

Object Interaction – Sequence Diagrams

Adapted for 204362
by Areerat Trongratsameethong

Bennett, McRobb and Farmer, *Object Oriented Systems Analysis and Design Using UML*
4th Edition, McGraw Hill, 2010

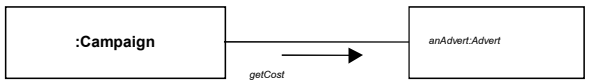
In This Lecture You Will Learn:

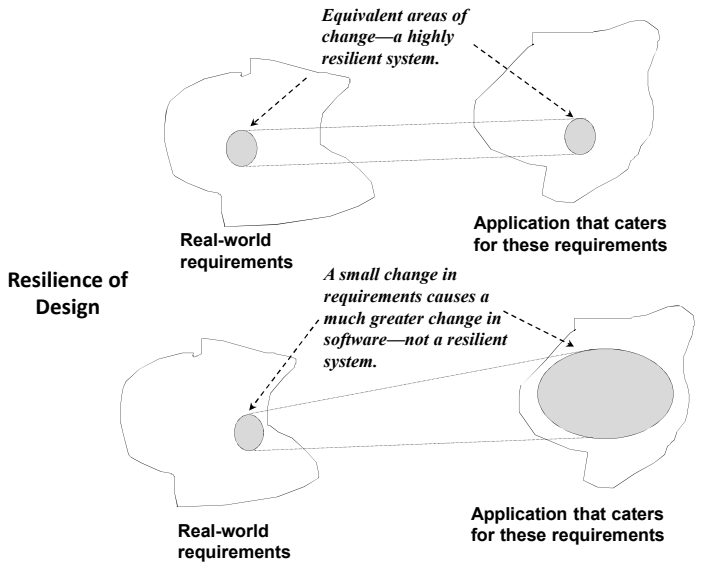
- how to develop object interaction from use cases;
- how to model object interaction using an interaction sequence diagram;
- how to cross-check between interaction diagrams and a class diagram.

Object Messaging

Objects communicate by sending messages. Sending the message getCost() to an Advert object, might use the following syntax.

currentadvertCost = anAdvert.getCost()





Interaction & Collaboration

- A collaboration is a group of objects or classes that work together to provide an element of functionality or behaviour.
- An interaction defines the message passing between lifelines (e.g. objects) within the context of a collaboration to achieve a particular behaviour.

Modelling Interactions

- Interactions can be modelled using various notations
  - Interaction sequence diagrams
  - Communication diagrams
  - Interaction overview diagrams
  - Timing diagrams

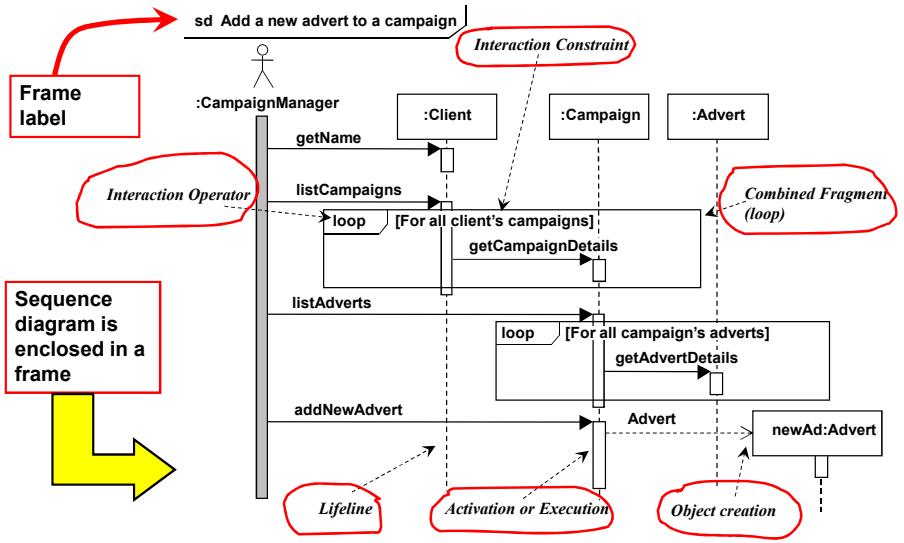
Sequence Diagrams

- Shows an interaction between lifelines (e.g. objects) arranged in a time sequence.
- Can be drawn at different levels of detail and to meet different purposes at several stages in the development life cycle.
- Typically used to represent the detailed object interaction that occurs for one use case or for one operation.

Sequence Diagrams

- Vertical dimension shows time.
- Objects (or subsystems or other connectable objects) involved in interaction appear horizontally across the page and are represented by lifelines.
- Messages are shown by a solid horizontal arrow.
- The execution or activation of an operation is shown by a rectangle on the relevant lifeline.

Sequence diagram



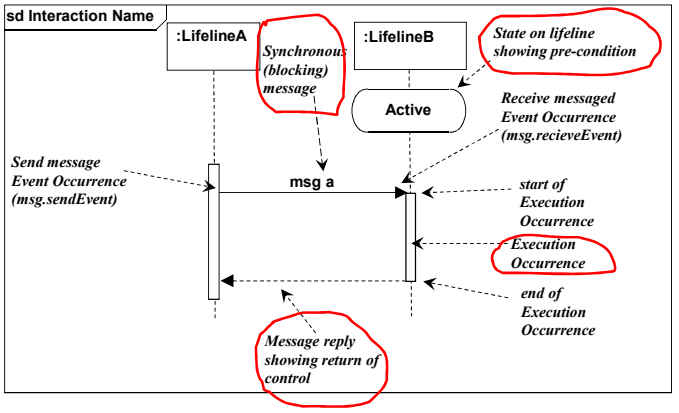
Sequence Diagram

- Iteration is represented by *combined fragment* rectangle with the *interaction operator* 'loop'.
- The loop combined fragment only executes if the guard condition in the interaction constraint evaluates as true.
- Object creation is shown with the construction arrow (dashed) going to the object symbol for the Advert lifeline.

Synchronous Message

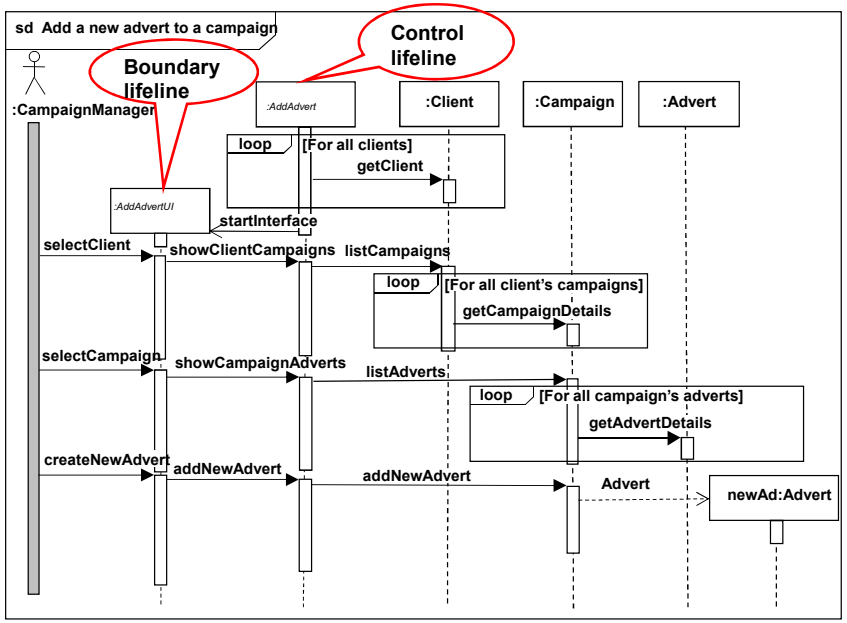
- A *synchronous message* or *procedural call* is shown with a full arrowhead, causes the invoking operation to suspend execution until the focus of control has been returned to it.

Further Notation

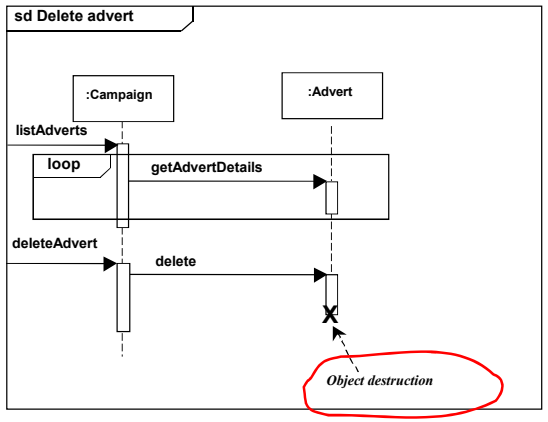


Boundary & Control Classes

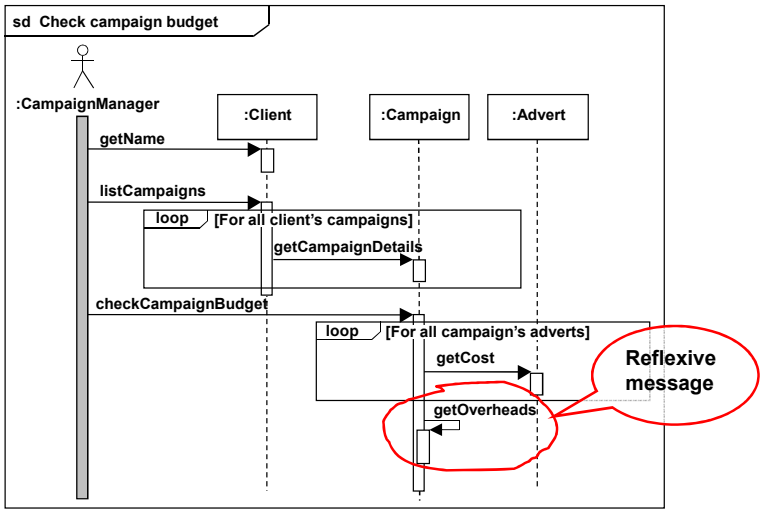
- Most use cases imply at least one boundary object that manages the dialogue between the actor and the system – in the next sequence diagram it is represented by the lifeline `:AddAdvertUI`
- The control object is represented by the lifeline `:AddAdvert` and this manages the overall object communication.



Object Destruction



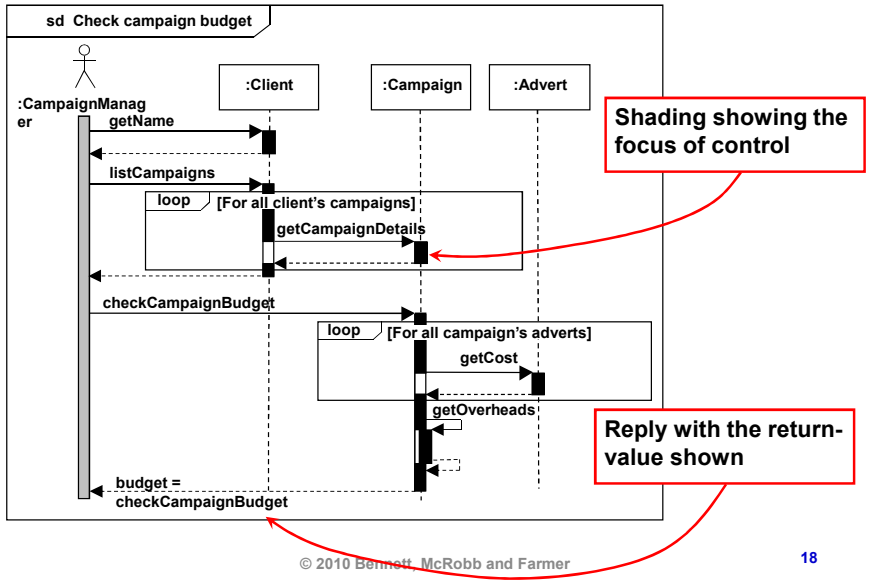
Reflexive Messages



Focus of Control

- Indicates times during an activation when processing is taking place within that object.
- Parts of an activation that are not within the focus of control represent periods when, for example, an operation is waiting for a return from another object.
- May be shown by shading those parts of the activation rectangle that correspond to active processing by an operation.

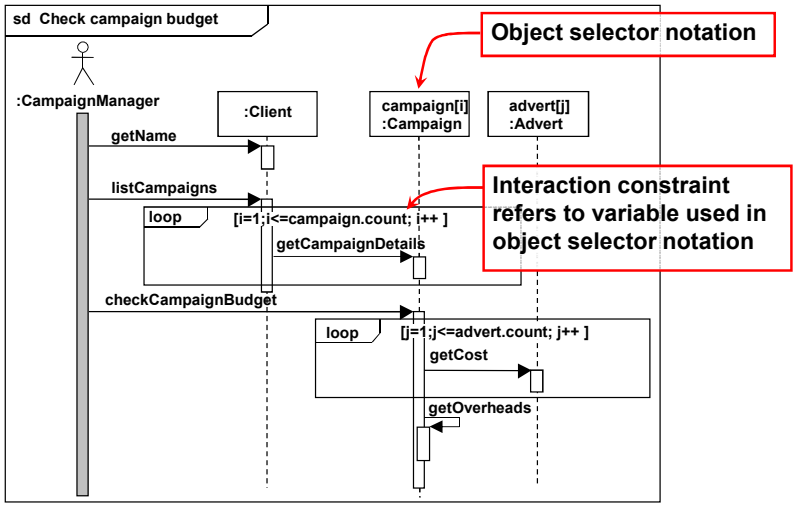
Focus of Control



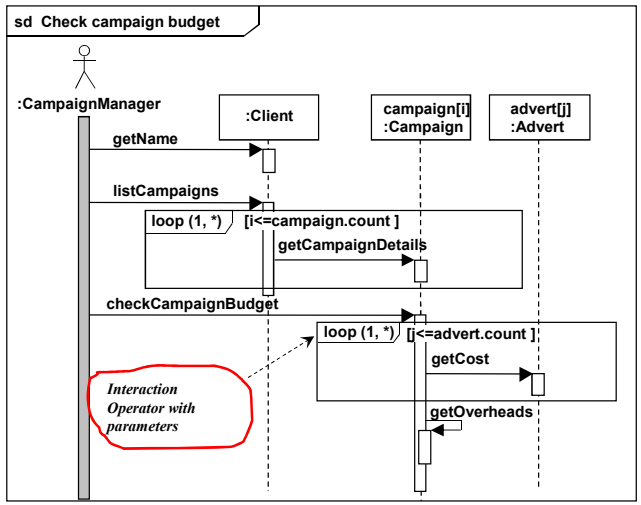
Reply Message

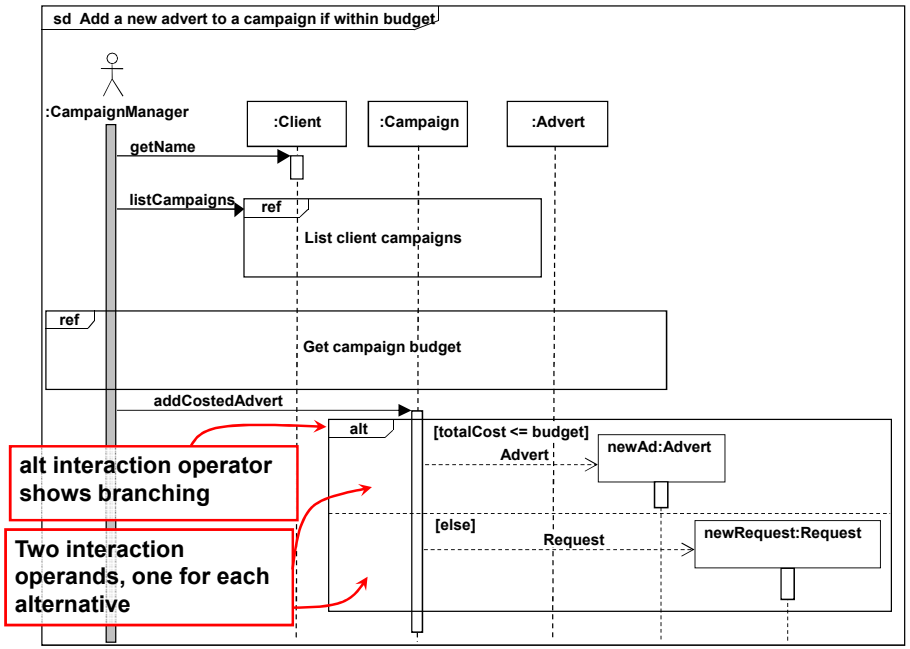
- A *reply message* returns the control to the object that originated the message that began the activation.
- Reply messages are shown with a dashed arrow, but it is optional to show them at all since it can be assumed that control is returned to the originating object at the end of the activation.

Object Selector Notation



Interaction Operators

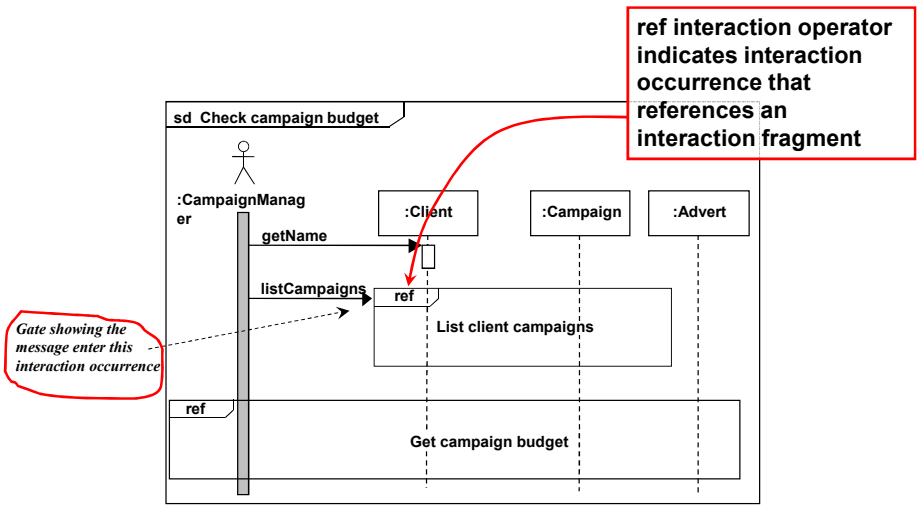




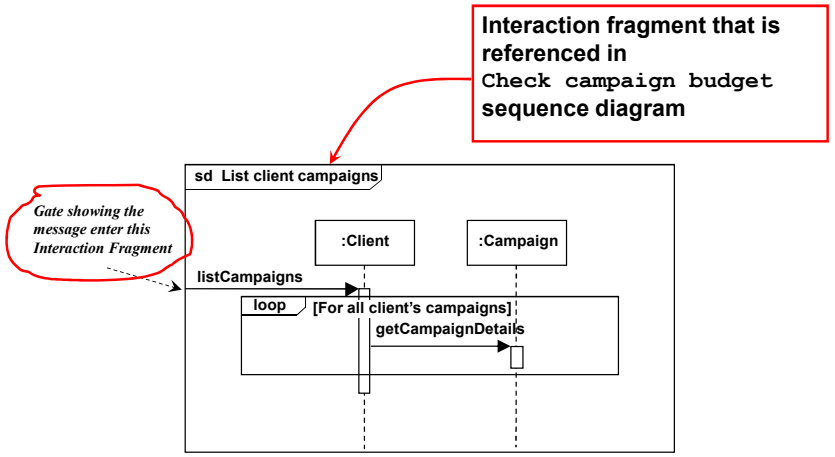
Handling Complexity

- Complex interactions can be modelled using various different techniques
  - Interaction fragments
  - Lifelines for subsystems or groups of objects
  - Continuations
  - Interaction Overview Diagrams (see later lecture)

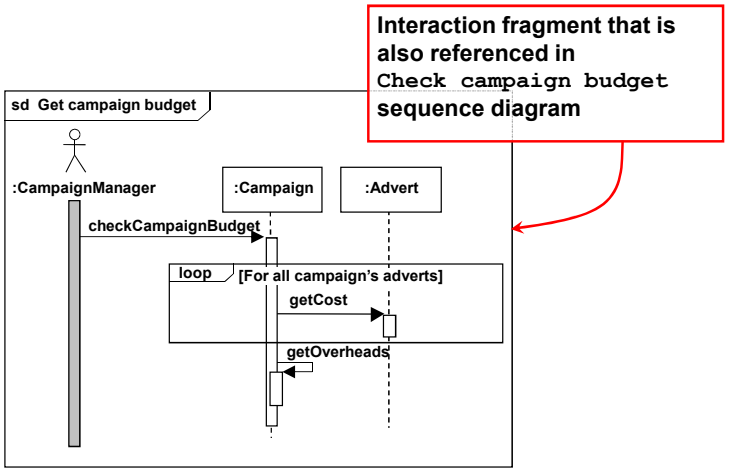
Using Interaction Fragments

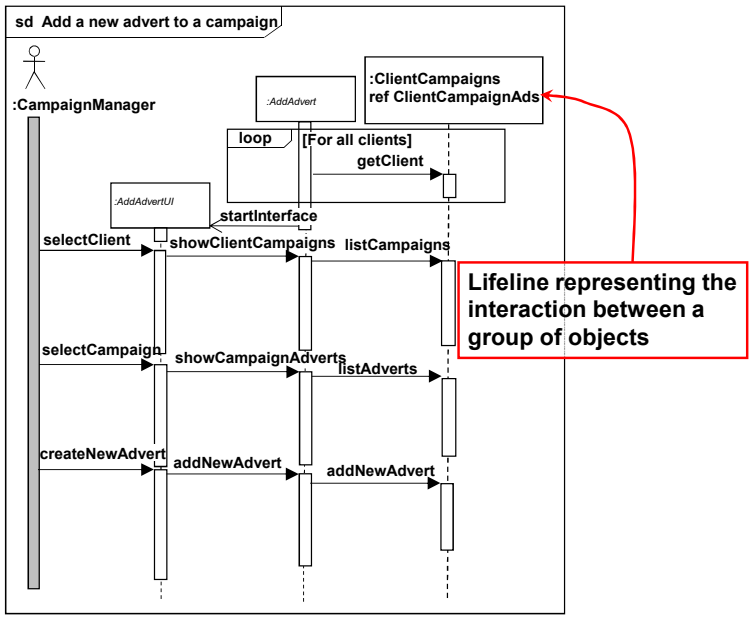


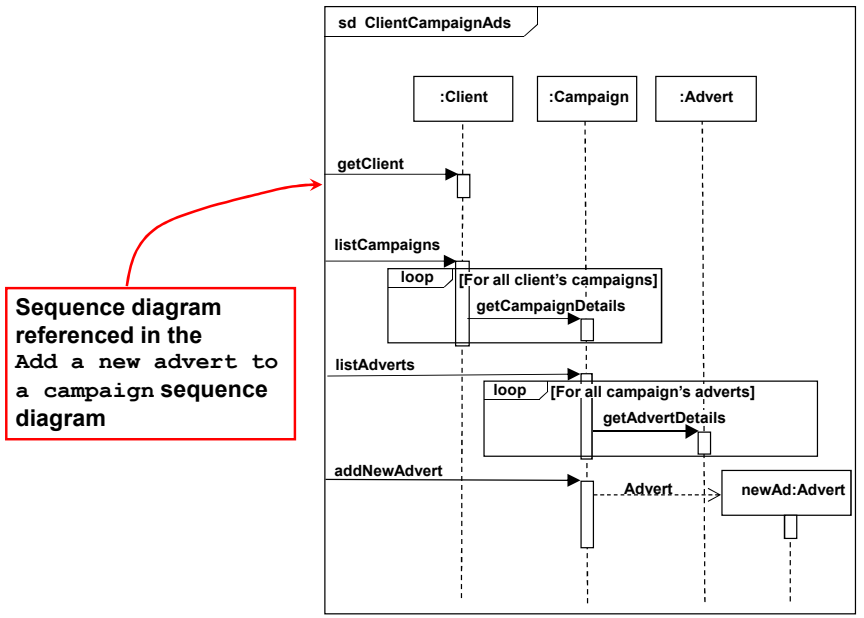
Interaction Fragment



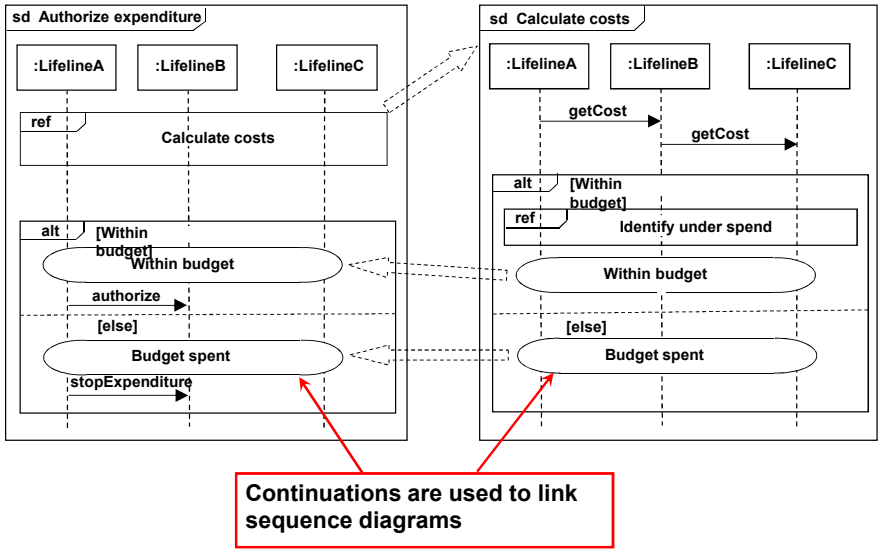
Interaction Fragment







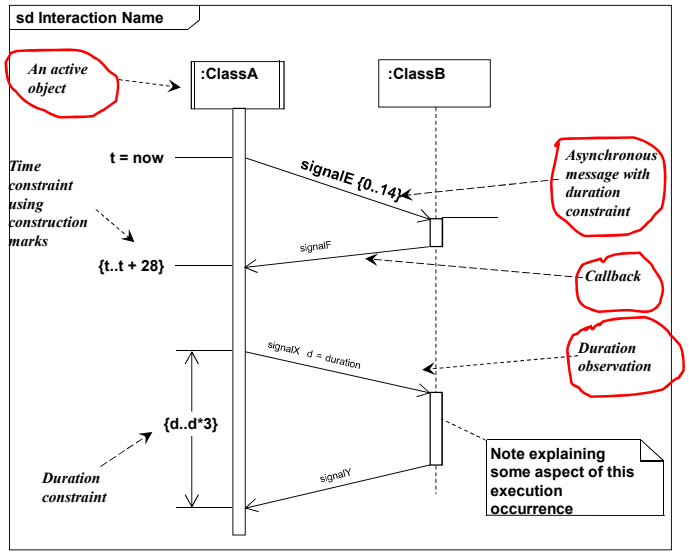
Using Continuations



Asynchronous Message

- An *asynchronous message*, drawn with an open arrowhead, does not cause the invoking operation to halt execution while it awaits a return.

Further Notation



Interaction Operators	Explanation and use
alt	Alternatives represents alternative behaviours, each choice of behaviour being shown in a separate operand. The operand whose interaction constraint is evaluated as true executes.
opt	Option describes a single choice of operand that will only execute if its interaction constraint evaluates as true.
break	Break indicates that the combined fragment is performed instead of the remainder of the enclosing interaction fragment.
par	Parallel indicates that the execution operands in the combined fragment may be merged in any sequence once the event sequence in each operand is preserved.
seq	Weak Sequencing results in the ordering of each operand being maintained but event occurrence from different operands on different lifelines may occur in any order. The order of event occurrences on common operands is the same as the order of the operands.
strict	Strict Sequencing imposes a strict sequence on execution of the operands but does not apply to nested fragments.
neg	Negative describes an operand that is invalid.
critical	Critical Region imposes a constraint on the operand that none of its event occurrences on the lifelines in the region can be interleaved.
ignore	Ignore indicates the message types, specified as parameters, that should be ignored in the interaction.
consider	Consider states which messages should be consider in the interaction. This is equivalent to stating that all others should be ignored.
assert	Assertion states that the sequence of messaging in the operand is the only valid continuation.
loop	Loop is used to indicate an operand that is repeated a number times until the interaction constraint for the loop is no longer true.

Guidelines for Sequence Diagrams

1. Decide at what level you are modelling the interaction.
2. Identify the main elements involved in the interaction.
3. Consider the alternative scenarios that may be needed.
4. Identify the main elements involved in the interaction.

Guidelines for Sequence Diagrams

5. Draw the outline structure of the diagram.
6. Add the detailed interaction.
7. Check for consistency with linked sequence diagrams and modify as necessary.
8. Check for consistency with other UML diagrams or models.

Model Consistency

- The allocation of operations to objects must be consistent with the class diagram and the message signature must match that of the operation.
  - Can be enforced through CASE tools.
- Every sending object must have the object reference for the destination object.
  - Either an association exists between the classes or another object passes the reference to the sender.
  - This issue is key in determining association design Message pathways should be carefully analysed.

Model Consistency

- All forms of interaction diagrams used should be consistent.
- Messages on interaction diagrams must be consistent with the state machine for the participating objects.
- Implicit state changes in interactions diagrams must be consistent with those explicitly modelled in state machine.

Summary

In this lecture you have learned about:

- how to develop object interaction from use cases;
- how to model object interaction using an interaction sequence diagram;
- how to cross-check between interaction diagrams and a class diagram.

References

- UML Reference Manual (OMG, 2009)
- Bennett, Skelton and Lunn (2005)

(For full bibliographic details, see Bennett, McRobb and Farmer)