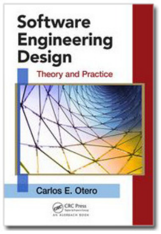
Chapter 2: Software Design with the Unified Modeling Language

Session III: UML Behavioral Modeling

Software Engineering Design: Theory and Practice
by Carlos E. Otero



SESSION'S AGENDA

- ➢ UML behavioral modeling
  - ✓ What is behavioral modeling?
  - ✓ Why is it important?

- ➢ Use case diagrams
  - ✓ Actors
  - ✓ System boundary
  - ✓ Common relationships

- ➢ Interaction diagrams
  - ✓ Communication diagrams
  - ✓ Sequence diagrams
  - ✓ Concurrency modeling

- ➢ Summary and conclusion of UML coverage

UML BEHAVIORAL MODELING

- In the previous session, we presented structural modeling and made a case for how it is essential to evaluate, characterize, and visualize the structural design of software systems from various perspective. Specifically, we presented:
    - ✓ Logical structural designs
        - At different levels of abstraction
    - ✓ Physical structural designs

- We also presented (very vaguely) the concept of **Quality** and discussed how structural modeling can be used to evaluate and access quality in terms of some quality goals, such as *reusability* and *maintainability*.

- Although structural designs work well for evaluating some quality attributes of systems, they are inadequate for others, such as *performance*.
    - ✓ Structural designs also provide poor techniques for evaluating the dynamic aspects and interactions of systems.

- UML provides several diagrams to model and reason about the dynamic aspects and of systems.
    - ✓ These can be used to model almost any behavioral aspect of modern software systems
    - ✓ They are used in many practical development projects.

UML Behavioral Modeling

- From previous sessions, we learned that the most common UML *behavioral diagrams* include:

Behavioral Diagram	Description
Use Case Diagram	Used to capture, specify, and visualize required system behavior .
Sequence Diagram	Used to capture, specify, and visualize system interactions with emphasis on the time-order sequence of messages exchanged.
Communication Diagram	Used to capture, specify, and visualize system interactions with emphasis on the structural order of entities participating in the message exchange.
State Machine Diagram	Used to capture, specify, and visualize system behavior as a set of discrete states and the transitions between them.
Activity Diagram	Used to capture, specify, and visualize system behavior; provide mechanisms for modeling that includes conditional statements, repetition, concurrency, and parallel execution and thus can be used at many different levels of abstraction, from modeling business work flows to code.

- Other behavioral diagrams include:
  - ✓ Timing diagram
  - ✓ Interaction overview diagram

UML Behavioral Modeling – Use Case Diagram

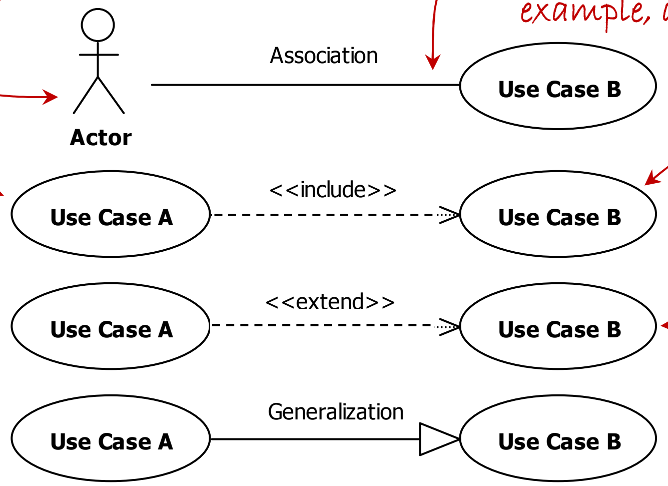
- Use case diagram
  - ✓ Behavioral diagram used to capture, specify, and visualize required system behavior.
    - Required system behavior are just requirements!
  - ✓ The main elements of use case diagrams are *actors*, *use cases*, and the *relationships* connecting them together.

- Actors are entities used to model users or other systems that interact with the system being modeled (i.e., the subject). Examples include:
  - ✓ Operators
  - ✓ Sensors
  - ✓ Client computers

- Use cases are entities used in use case diagrams to specify the required behavior of a system.
  - ✓ They provide the means to capture, model, and visualize the systems' required behavior.
  - ✓ They do this without any knowledge of programming technology, so that different stakeholders with different backgrounds can reason about the system.

UML Behavioral Modeling – Use Case Diagram

- Common UML relationships applied in use case diagrams.



Named actor

Actor and use case communicate, for example, actor executes use case.

Use case A requires the behavior from use case B

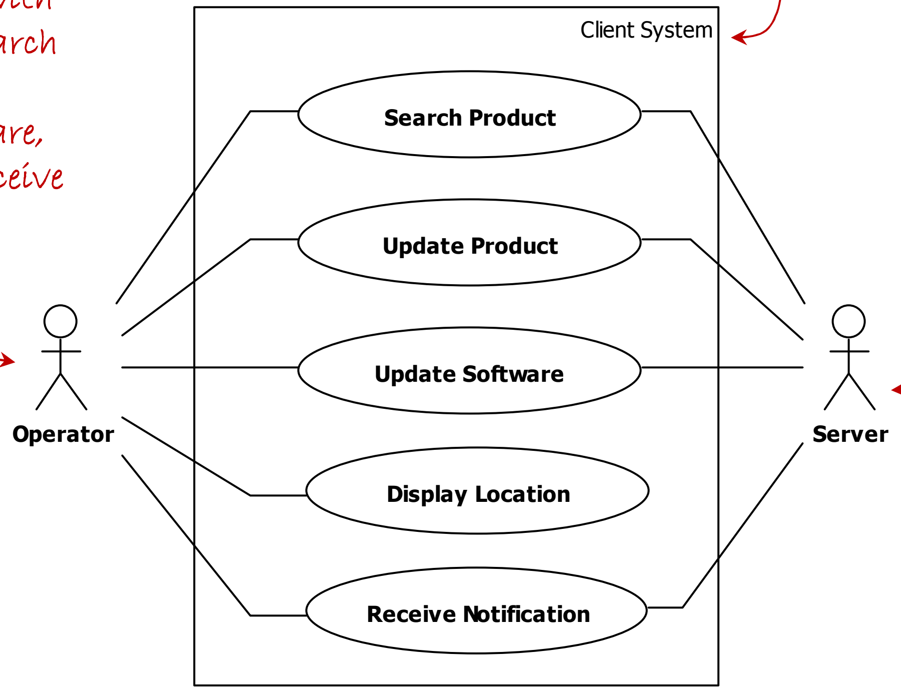
Use case B MUST be executed when use case A is executed!

Use case A extends the behavior of use case B. Use case B may be executed when use case A is executed, but it is not mandatory.

Use case A is a specialization of use case B

<u>Important</u>:
include relationship specifies a mandatory inclusion
Extend relationship specifies an optional inclusion

UML Behavioral Modeling – Use Case Diagram



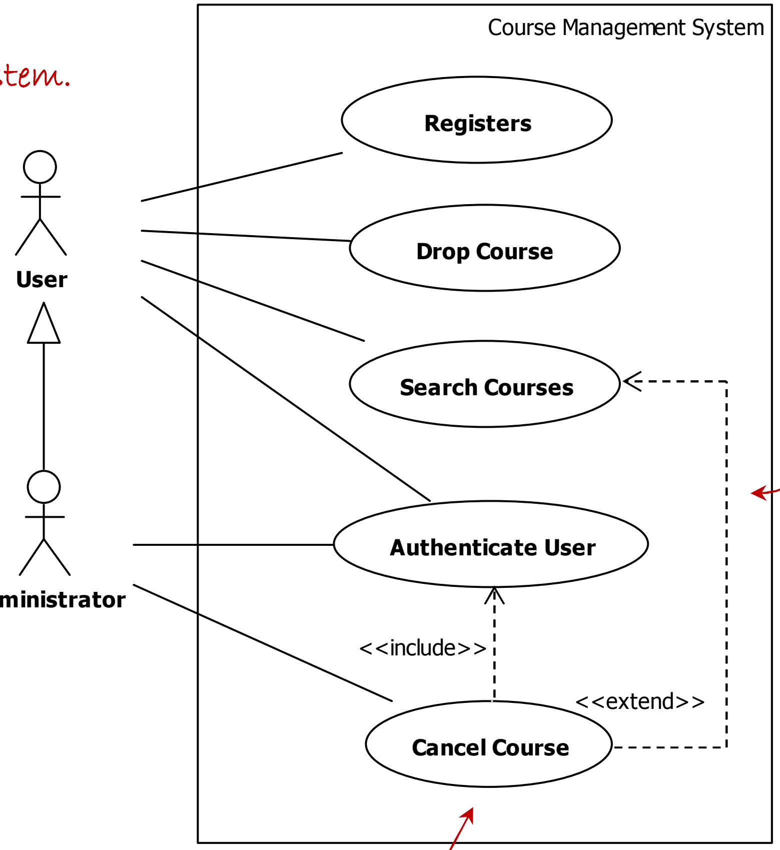
Optional System Boundary to model the Subject

The Operator interacts with the Client System to search for products, update products, update software, display location, and receive notifications.

The Server also interacts with the Client System

This models the behavior of a client system. An Operator and Server both interact with the Client System, denoted by the System Boundary.

UML Behavioral Modeling – Use Case Diagram



Optional System Boundary to model the Subject

Actor interacts with the system.

A specialized user

To cancel a course, you may want to search for the course

Alternatively, you may want to enter the course number directly without searching for it.

You cannot cancel a course unless your identity has been authenticated!

UML Behavioral Modeling – Interaction Diagrams

- Interaction diagrams can be used to model complex interactions between design units together with the messages exchanges and the type of the exchange.

- Interaction diagrams can be used at the architectural level.
  - ✓ For example, they allow us to model interaction among software components.

- Interaction diagrams can be used at the detailed design level.
  - ✓ For example, they allow us to model interactions among objects at run-time

- Interaction diagrams can be used at the construction design level.
  - ✓ For example, they allow us to model conditional and repetition structures.

- In many situations, interaction diagrams reveal many important issues related to the quality of the system.

- Two types of interaction diagrams are:
  - ✓ Communication diagrams
  - ✓ Sequence diagrams

UML BEHAVIORAL MODELING – COMMUNICATION DIAGRAM

- Communication diagrams
  - ✓ Behavioral diagrams used to capture, specify, evaluate, and visualize system interactions with emphasis on the structural order of entities participating in the message exchange.

- When using communication diagrams, entities can be modeled as
  - ✓ Objects, representing instances of classes and components.
  - ✓ Roles, representing a prototypical instance

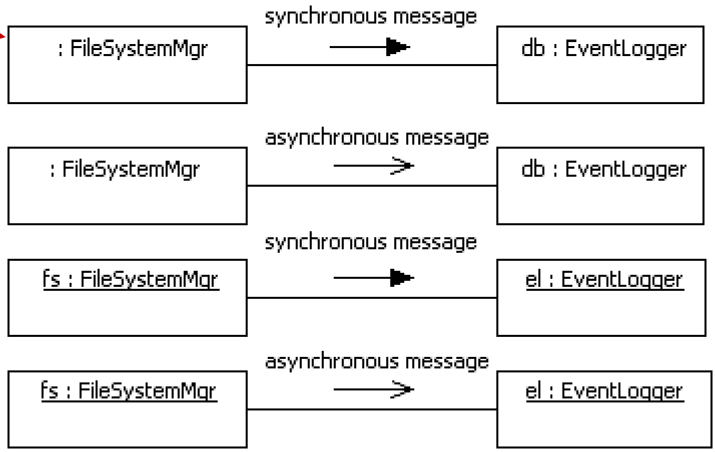
- Both objects and roles can be connected to model the exchange of messages using:
  - ✓ Links to connect objects
  - ✓ Connectors to connect roles
  - ✓ Both links and connectors look exactly alike, as a solid line. They only differ semantically.

- The type of message exchanges include:
  - ✓ Asynchronous
  - ✓ Synchronous

UML Behavioral Modeling – Communication Diagram

FileSystemMgr Role.

In synchronous messaging, sender and receiver work in lock-step to achieve an operation.



: FileSystemMgr

synchronous message

db : EventLogger

: FileSystemMgr

asynchronous message

db : EventLogger

fs : FileSystemMgr

synchronous message

el : EventLogger

fs : FileSystemMgr

asynchronous message

el : EventLogger

In asynchronous messaging, sender continues execution while receiver completes operation independent from the sender

EventLogger Object.

Asynchronous messaging between two run-time object instances

<u>Important</u>:
Synchronous and Asynchronous messaging are are important when reasoning about concurrency issues!

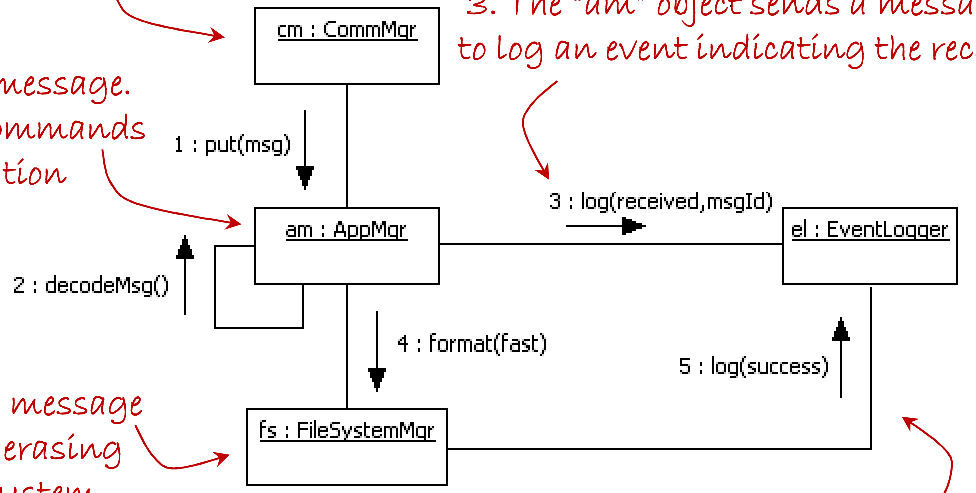
UML Behavioral Modeling – Communication Diagram

Communication diagram of objects at run-time using synchronous messaging

1. The "cm" object of type CommMgr sends a message to another object of type AppMgr.

2. The "am" object decodes the message. If finds out that the message commands a file system format operation

3. The "am" object sends a message to the "el" object to log an event indicating the receipt of the message



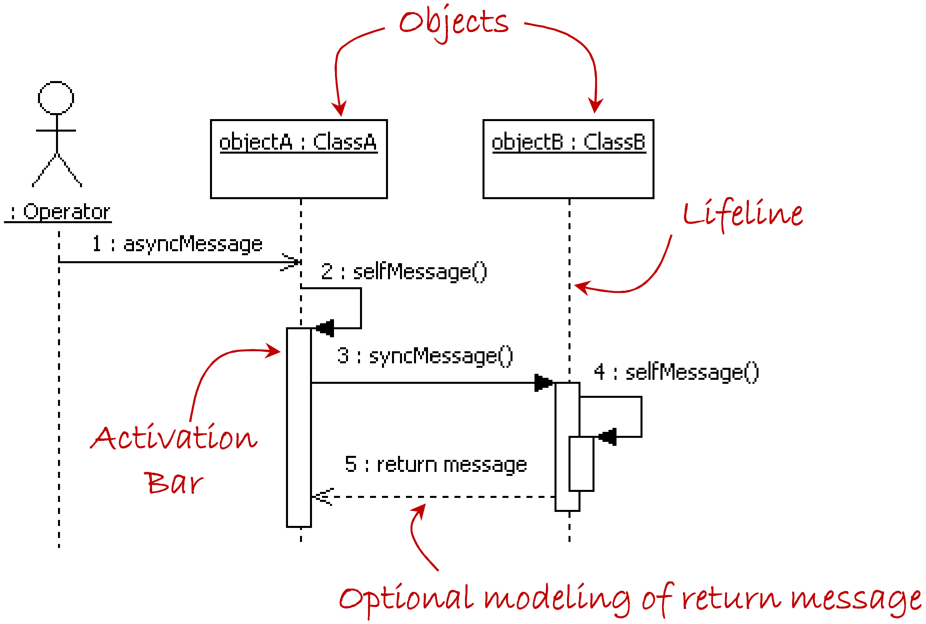
4. The "am" object sends a message to the "fs" object to begin erasing the data from the file system

5. Once complete, the "fs" object sends a message to the "el" object to log an event indicating success

Given this model, can "cm" begin other operations before the file system format is complete?

UML Behavioral Modeling – Sequence Diagram

- ➢ Sequence diagrams
  - ➢ Similar to Communication diagrams, but, they put emphasis on the time-order sequence of messages exchanged.



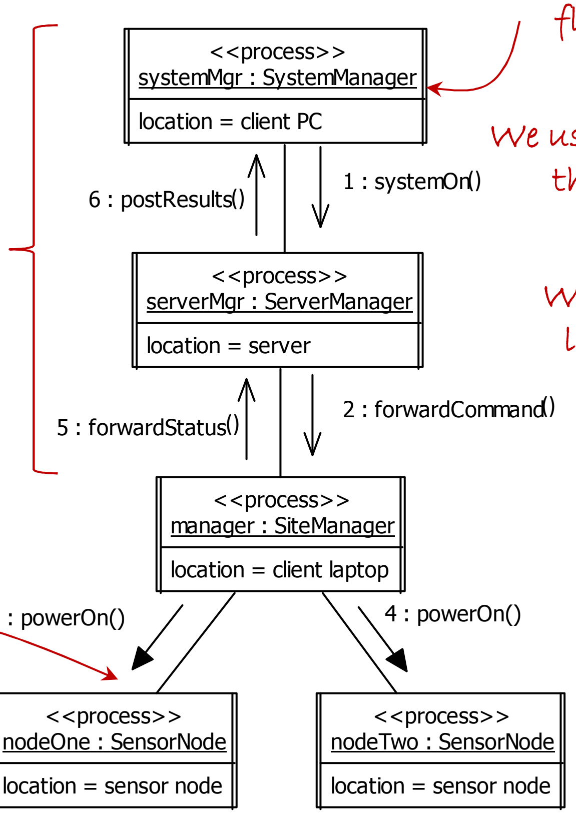
UML Behavioral Modeling

- Finally, a word about concurrency…

SystemManager process, located on client PC does not have to wait until Sensor Nodes are powered on to begin doing other work

Asynchronous messaging

Synchronous messaging



<<process>>
systemMgr : SystemManager
location = client PC

<<process>>
serverMgr : ServerManager
location = server

<<process>>
manager : SiteManager
location = client laptop

<<process>>
nodeOne : SensorNode
location = sensor node

<<process>>
nodeTwo : SensorNode
location = sensor node

1 : systemOn()

2 : forwardCommand()

3 : powerOn()

4 : powerOn()

5 : forwardStatus()

6 : postResults()

We use Active Objects to model independent flows of control, e.g., processes.

We use stereotype <<process>> to specify these are actual run-time processes.

We use Tagged Values to specify the location of the running processes.

SiteManager works in lock-step with nodeOne. Once nodeOn is powered on, then SiteManager initiates power on sequence of nodeTwo

WHAT'S NEXT…

- Now that we are equipped with the necessary UML tools, we can now explore how to used them to design a software architecture.

- In the next session, we will present Software Architecture in more detail, including :
  - ✓ Understanding the role of software architecture within the design phase.
  - ✓ Explore in more detail the architectural tasks and problem-solving during architecture.