UML Sequence Diagrams

Eileen Kraemer
CSE 335
Michigan State University

Types of Diagrams

- Structural Diagrams – focus on static aspects of the software system
  - Class, Object, Component, Deployment

- Behavioral Diagrams – focus on dynamic aspects of the software system
  - Use-case, Interaction, State Chart, Activity

Structural Diagrams

- **Class Diagram** – set of classes and their relationships. Describes interface to the class (set of operations describing services)
- **Object Diagram** – set of objects (class instances) and their relationships
- **Component Diagram** – logical groupings of elements and their relationships
- **Deployment Diagram** - set of computational resources (nodes) that host each component.

Behavioral Diagram

- **Use Case Diagram** – high-level behaviors of the system, user goals, external entities: actors
- **Sequence Diagram** – focus on time ordering of messages
- **Collaboration Diagram** – focus on structural organization of objects and messages
- **State Chart Diagram** – event driven state changes of system
- **Activity Diagram** – flow of control between activities

Use Case Diagrams

- Describes a set of sequences.
- Each sequence represents the interactions of things outside the system (*actors*) with the system itself (and key abstractions)
- Use cases represent the functional requirements of the system (non-functional requirements must be given elsewhere)
Use case

- Each use case has a descriptive name
- Describes what a system does but not how it does it.
- Use case names must be unique within a given package
- Examples: withdraw money, process loan

Actor

- Actors have a name
- An actor is a set of roles that users of use cases play when interacting with the system
- They are external entities
- They may be external an system or DB
- Examples: Customer, Loan officer

What is a Use Case

- Use case captures some user-visible functionality
- Granularity of functionality depends on the level of detail in your model
- Each use case achieves a discrete goal for the user
- Use Cases are generated through requirements elicitation

Example

Extend and Include

Example (generalization)
• Modeling Behavior
• Sequence Diagrams

Refining the Object Model
• Typically, only very simplistic object models can be directly derived from use cases.
• A better understanding of the behavior of each use case is necessary (i.e., analysis)
• Use interaction diagrams to specify and detail the behavior of use cases
• This helps to identify and refine key abstractions and relationships
• Operations, attributes, and messages are also identified during this process

Interaction Diagrams
• There is one (or more) Interaction diagram per use case
  – Represent a sequence of interactions
  – Made up of objects, links, and messages
• Sequence diagrams
  – Models flow of control by time ordering
  – Emphasizes passing messages wrt time
  – Shows simple iteration and branching
• Collaboration diagrams
  – Models flow of control by organization
  – Structural relationships among instances in the interaction
  – Shows complex iteration and branching

Sequence Diagrams
• X-axis is objects
  – Object that initiates interaction is left most
  – Object to the right are increasingly more subordinate
• Y-axis is time
  – Messages sent and received are ordered by time
• Object life lines represent the existence over a period of time
• Activation (double line) is the execution of the procedure.

Message Passing
• Send – sends a signal (message) to an object
• Return – returns a value to a caller
• Call – invoke an operation
• Stereotypes
  – <<create>>
  – <<destroy>>

Example UML Sequence Diagram

[Diagram showing sequence of interactions with objects and messages, including stereotypes like <<create>> and <<destroy>>]
Mail System Objects

- Caller, owner, administrator
- Mailbox, extension, password, greeting
- Message, message list
- Mail system
- Input reader/device
Properties of Sequence Diagrams

- Initiator is leftmost object (boundary object)
- Next is typically a control object
- Then comes entity objects

Collaboration Diagrams

- Emphasizes the organization of the objects that participate in an interaction
- Classifier roles
- Association
- Messages, flow, and sequencing

Example Collaboration Diagram

Leave a Message

1: dial
2: checkInput
3: dial
4: ext.getExtension()
6: talk
7: msg.getMessage()

Mailbox

InputReader

sys

ext:=getExtension()
verifyExtension
getMessage
lookup(ext)
saveMessage(mes)
dial(dddd)
hangup
create
talk()
caller
mailbox
mes
recorder
inputReader
ext

Properties of Sequence Diagrams

Collaboration Diagrams
Collaboration vs Sequence

- The two diagrams really show the same information
- Collaboration diagrams show more static structure (however, class diagrams are better at this)
- Sequence diagrams clearly highlight the orderings and very useful for multi-tasking

Summary (Interaction Diagrams)

- Well structured interaction diagrams:
  - Is focused on communicating one aspect of a system’s dynamics
  - Contains only those elements that are essential to understanding
  - Is not so minimalistic that it misinforms the reader about the semantics that are important
- Diagrams should have meaningful names
- Layout diagram to minimize line crossings
- Use branching sparingly (leave for activity dia)

State Diagrams

- Finite state machines (i.e., automata, Mealy/Moore, state transition)
- Used to describe the behavior of one object (or sometimes an operator) for a number of scenarios that affect the object
- They are not good for showing interaction between objects (use interaction diagrams)
- Only use when the behavior of a object is complex and more detail is needed

State Diagram Features

- Event – something that happens at a specific point
  - Alarm goes off
- Condition – something that has a duration
  - Alarm is on
  - Fuel level is low
- State – an abstraction of the attributes and relationships of an object (or system)
  - The fuel tank is in a too low level when the fuel level is below level x for n seconds

Example: on/off Switch

![on/off Switch State Diagram]

Using guards and actions

![Using guards and actions Diagram]