Sequence Diagrams

Topic # 10

Chapter 15 – UML Interaction Diagrams

Chapter 10 – System Sequence Diagram

Craig Larman

Interaction Diagrams

- A type of behavior diagram
- A series of diagrams describing the *dynamic behavior* of an object-oriented system.
  - A set of messages exchanged among a set of objects within a context to accomplish a purpose.
- Often used to model the way a use case is realized through a sequence of messages between objects.
- The purpose of Interaction diagrams is to:
  - Model interactions between objects
  - Assist in understanding how a system (a use case) actually works
  - Verify that a use case description can be supported by the existing classes
  - Identify responsibilities/operations and assign them to classes

Interaction Diagrams

- Two types of interaction diagrams
  - Collaboration Diagrams
    - Emphasizes structural relations between objects
  - Sequence Diagram
    - Emphasizes how objects interact with order of time

What is Sequence Diagram?

- A sequence diagram shows how objects interact in a specific situation. Sequence diagrams provide an **approximation of time** and the general sequence of these interactions by reading the diagram from top to bottom.
- They are also called event diagrams.
- Illustrates how objects interacts with each other.
- Emphasizes time ordering of messages.
- Can model simple sequential flow, branching, iteration, recursion and concurrency.

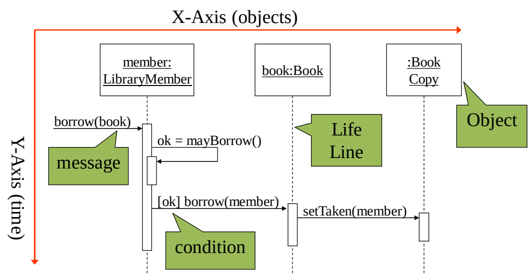
Sequence Diagrams

- They focus on **message sequences**, that is, how messages are sent and received between a number of objects.
- Sequence diagrams have two axes:
  - the vertical axis shows time and
  - the horizontal axis shows a set of objects.
- A sequence diagram also reveals the **interaction for a specific scenario**—a specific interaction between the objects that happens at some point in time during the system's execution (for example, when a specific function is used).

Sequence Diagrams

- A sequence diagram is enclosed by a rectangular frame with the name of the diagram shown in a pentagram in the upper-left corner prefixed by *sd.*
- *On the* horizontal axis are the objects involved in the sequence. Each is represented by an object rectangle with the object and/or class name underlined.
- The rectangle along with the vertical dashed line, called the object's lifeline, indicates the object's execution during the sequence (that is, messages sent or received and the activation of the object).
- Communication between the objects is represented as horizontal message lines between the objects' lifelines.
- To read the sequence diagram, start at the top of the diagram and read down to view the exchange of messages taking place as time passes.

A Sequence Diagram



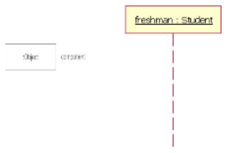
Generic and Instance Form

- Sequence diagrams can be used in two forms:
  - the generic form and
  - the instance form.
- The **instance form** describes a specific scenario in detail; it documents one possible interaction. The instance form does not have any conditions, branches, or loops; it shows the interaction for just the chosen scenario.(e.g. Successful opening of an account)
- The **generic form** describes all possible alternatives in a scenario; therefore branches, conditions, and loops may be included .(e.g. Opening an account)

Basic Notations

Class Roles or Participants

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



An example of the Student class whose instance name is freshman

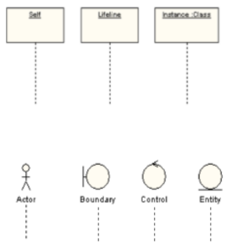
Instance Name : Class Name

Actor Symbol

- Represented by a stick figure, actors are entities that are both interactive with and external to the system.

Lifelines

- An activated object is either executing its own code or is waiting for the return of another object to which it has sent a message.
- The lifeline represents the existence of an object at a particular time; it is drawn as an object icon with a dashed line extending down to the point at which the object stops existing.
- Lifelines indicate the object's presence over time.



Activation or Execution Occurrence

- Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.

Activation or Execution Occurrence

Message

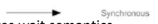
- A message is a communication between objects that conveys information with the expectation that action will be taken.
- Messages can be signals, operation invocations, or something similar (for example, remote procedure calls)
- In the sequence diagram, communication between the objects can be shown with distinct message types.

Messages

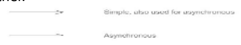
- A message is represented by an arrow between the life lines of two objects.
  - The time required by the receiver object to process the message is denoted by an *activation-box.*
- A message is labeled at minimum with the message name.
  - Arguments and control information (conditions, iteration) may be included.

Types of Messages

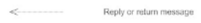
- **Synchronous Message**



  - A synchronous message indicates wait semantics.
  - A synchronous message requires a response before the interaction can continue. It's usually drawn using a line with a solid arrowhead pointing from one object to another.

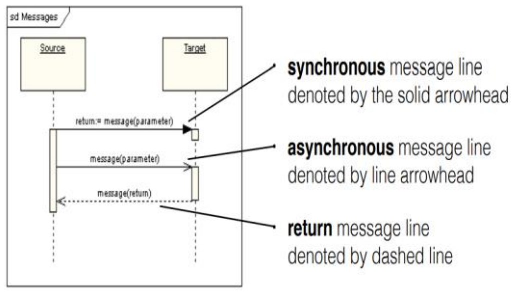


- **Asynchronous message**
  - An asynchronous message reveals that the sending object does not wait, but continues to execute immediately after having sent the message (any result is typically sent back as an asynchronous message as well).



- **Reply or Return Message**
  - A reply message is drawn with a dotted line and an open arrowhead pointing back to the original lifeline.

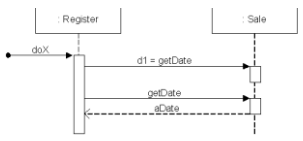
Example


sd Messages
Source
Target
return:= message(parameter)
message(parameter)
message(return)
synchronous message line denoted by the solid arrowhead
asynchronous message line denoted by line arrowhead
return message line denoted by dashed line

Reply or Return

There are two ways to show return result from a message:

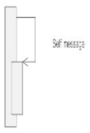
- Using the message syntax returnVar=message(parameter)
- Using a reply(or return) message line at the end of an activation bar.

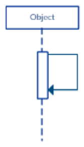


Types of Messages

- **Self Message/*Reflexive message***

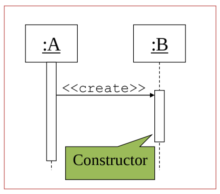
- A message an object sends to itself, usually shown as a U shaped arrow pointing back to itself

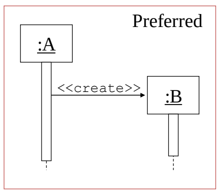




Types of Messages

- Create Message
- An object may create another object via a **<<create>>** message.

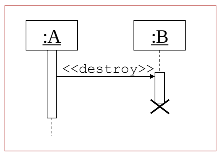




Types of Messages

- An object may destroy another object via a **<<destroy>>** message.
  - An object may destroy itself.
  - Avoid modeling object destruction unless memory management is critical.

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence



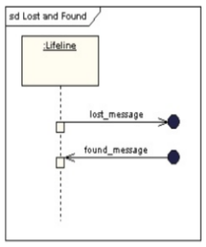
Types of Messages

Found Message

A message sent from an unknown recipient or from a sender not shown on the current diagram, shown by an arrow from an endpoint to a lifeline.

- **Lost Message**

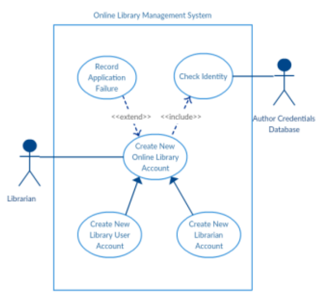
A message sent to an unknown recipient that is not shown on the diagram. It's shown by an arrow going from a lifeline to an endpoint, a filled circle.



Example: Balance Lookup Use Case

- Customer asks the bank for his account balance.
- Bank verifies his account number from the account ledger.
- Bank then sends message to checkaccount for getting the balance.
- The bank informs the customer about balance.

Example: Sequence diagram from use cases


Online Library Management System
Record Application Failure
Check Identity
Author Credentials Database
<<extend>>
<<include>>
Create New Online Library Account
Librarian
Create New Library User Account
Create New Librarian Account

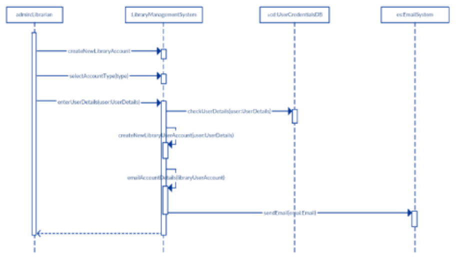
Use Case Description

- Here are the steps that occur in the use case named ‘Create New Library User Account’.
  - The librarian request the system to create a new online library account
  - The librarian then selects the library user account type
  - The librarian enters the user’s details
  - The user’s details are checked using the user Credentials Database
  - The new library user account is created
  - A summary of the of the new account’s details are then emailed to the user

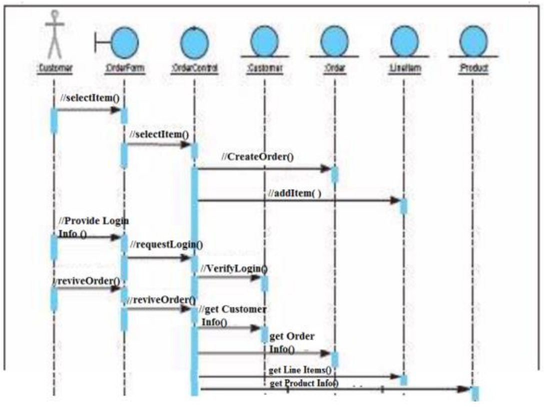
Example: ‘Create New User Account’ --Sequence diagram.

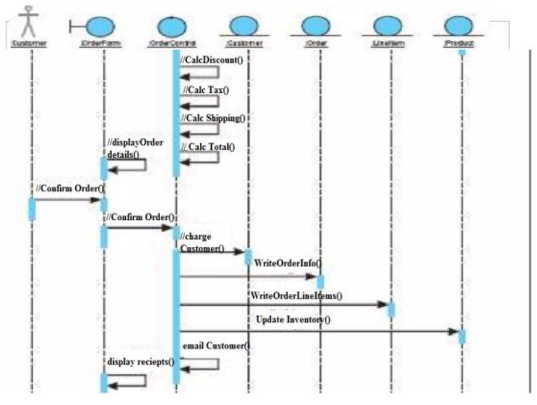
- Before drawing the sequence diagram, it’s necessary to identify the objects or actors that would be involved in creating a new user account. These would be;
- Librarian
- Online Library Management system
- User credentials database
- Email system

Solution



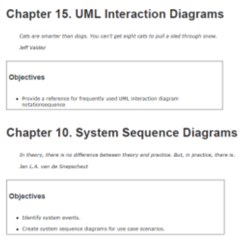
admin:Librarian
:LibraryManagementSystem
ucd:UserCredentialsDB
es:EmailSystem
createNewLibraryAccount
selectAccountType(type)
enterUserDetails(user:UserDetails)
checkUserDetails(user:UserDetails)
createNewLibraryUserAccount(user:UserDetails)
emailAccountDetails(libraryUserAccount)
sendEmail(email:Email)





READING

Chapter 10,15 – Applying UML and Pattern by Craig Larman 3rd Edition


Chapter 15. UML Interaction Diagrams

Cats are smarter than dogs. You can't get eight cats to pull a sled through snow.

Jeff Valdez

Objectives

- Provide a reference for frequently used UML interaction diagram notationsequence

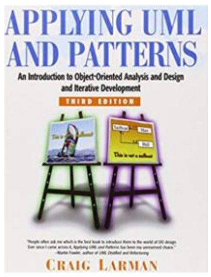
Chapter 10. System Sequence Diagrams

In theory, there is no difference between theory and practice. But, in practice, there is.

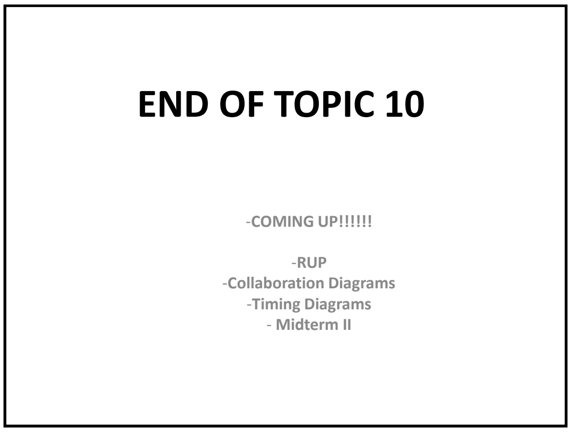
Jan L.A. van de Snepscheut

Objectives

- Identify system events.
- Create system sequence diagrams for use case scenarios.



APPLYING UML AND PATTERNS
An Introduction to Object-Oriented Analysis and Design and Iterative Development
THIRD EDITION
CRAIG LARMAN



END OF TOPIC 10

-COMING UP!!!!!!

-RUP
-Collaboration Diagrams
-Timing Diagrams
- Midterm II