CSE 403: Software Engineering, Spring 2015
courses.cs.washington.edu/courses/cse403/15sp/

UML Sequence Diagrams

Emina Torlak
emina@cs.washington.edu
Outline

• Overview of sequence diagrams
• Syntax and semantics
• Examples
an overview of sequence diagrams
What is a UML sequence diagram?
What is a UML sequence diagram?

• **Sequence diagram**: an “interaction diagram” that models a single scenario executing in a system
  - 2nd most used UML diagram (behind class diagram)
  - Shows what messages are sent and when
What is a UML sequence diagram?

- **Sequence diagram**: an “interaction diagram” that models a single scenario executing in a system
  - 2nd most used UML diagram (behind class diagram)
  - Shows what messages are sent and when

- Relating UML diagrams to other design artifacts:
  - CRC cards  →  class diagrams
  - Use cases  →  sequence diagrams
Key parts of a sequence diagram

Client

Server

checkEmail

sendUnsentEmail

newEmail

response

[newEmail] get

deleteOldEmail

[5]
Key parts of a sequence diagram

- Participant: an object or an entity; the sequence diagram actor
  - sequence diagram starts with an unattached "found message" arrow
Key parts of a sequence diagram

- **Participant**: an object or an entity; the sequence diagram actor
  - sequence diagram starts with an unattached "found message" arrow
- **Message**: communication between objects
Key parts of a sequence diagram

• **Participant**: an object or an entity; the sequence diagram actor
  • sequence diagram starts with an unattached "found message" arrow

• **Message**: communication between objects

• **Axes in a sequence diagram**:
  • **horizontal**: which participant is acting
  • **vertical**: time (↓ forward in time)
Key parts of a sequence diagram

- **Participant**: an object or an entity; the sequence diagram actor
  - sequence diagram starts with an unattached "found message" arrow
- **Message**: communication between objects
- **Axes in a sequence diagram**:
  - **horizontal**: which participant is acting
  - **vertical**: time (↓ forward in time)
Sequence diagram from a use case

1. The user presses the “check email” button.
2. The client first sends all unsent email to the server.
3. After receiving an acknowledgement, the client asks the server if there is any new email.
4. If so, it downloads the new email.
5. Next, it deletes old thrashed email from the server.
sequence diagrams: syntax and semantics
Representing objects

objectname:classname

Named object
Smith:Patient

Anonymous object
:Patient
Smith

Object of unknown class
Representing objects

objectname:classname

• An object: a box with an underlined label that specifies the object type, and optionally the object name.
  • Write the object's name if it clarifies the diagram.
Representing objects

- An **object**: a box with an **underlined** label that specifies the object type, and optionally the object name.
  - Write the object's name if it clarifies the diagram.

- An object's "life line" is represented by a dashed vertical line.
  - Represents the life span of the object during the scenario being modeled.

**objectname:classname**

- **Named object**
  - Smith:Patient
- **Anonymous object**
  - :Patient
  - Smith
- **Object of unknown class**
- **Lifeline**
Representing messages between objects

• A **message** (method call): **horizontal arrow** to the receiving object.
  • Write message name and arguments above the arrow.

```
:Hospital

admit(patientID, roomType)
```
Different types of messages

• Type of arrow indicates types of messages:
  • Synchronous message: solid arrow with a solid head.
  • Asynchronous message: solid arrow with a stick head.
  • Return message: dashed arrow with stick head.
**Indicating method execution**

- **Activation**: thick box over object's life line, drawn when an object's method is on the stack
  - Either that object is running its code, or it is on the stack waiting for another object's method to finish
- Nest activations to indicate an object calling itself.
Lifetime of objects

- **Object creation**: an arrow with *new* written above it
  - An object created after the start of the scenario appears lower than the others.

- **Object deletion**: X at the bottom of object's lifeline
  - Java doesn't explicitly delete objects; they fall out of scope and are garbage collected.
Alternatives, options, and loops

- **Frame**: a box around part of a sequence diagram
  - if \( \rightarrow \) (opt) [condition]
  - if/else \( \rightarrow \) (alt) [condition], separated by horizontal dashed line
  - loop \( \rightarrow \) (loop) [condition or items to loop over]
Linking sequence diagrams

- If one sequence diagram is too large or refers to another diagram:
  - An unfinished arrow and comment.
  - A ref frame that names the other diagram.

The flow is continued in Diagram 2.
Example sequence diagram

:StoreFront  :Cart  :Inventory

loop

AddItem → ReserveItem

Checkout → ConfirmOrder

ProcessOrder → PlaceItemInOrder
Forms of system control

What can you say about the control flow of each of these systems?

- Is it centralized?
- Is it distributed?
Why use sequence diagrams? Why not code it?
Why use sequence diagrams? Why not code it?

- A good sequence diagram is still above the level of the real code (not all code is drawn on diagram)
Why use sequence diagrams? Why not code it?

• A good sequence diagram is still above the level of the real code (not all code is drawn on diagram)
• Sequence diagrams are language-agnostic (can be implemented in many different languages)
Why use sequence diagrams? Why not code it?

- A good sequence diagram is still above the level of the real code (not all code is drawn on diagram)
- Sequence diagrams are language-agnostic (can be implemented in many different languages)
- Non-coders can read and write sequence diagrams.
Why use sequence diagrams? Why not code it?

• A good sequence diagram is still above the level of the real code (not all code is drawn on diagram)

• Sequence diagrams are language-agnostic (can be implemented in many different languages)

• Non-coders can read and write sequence diagrams.

• Easier to do sequence diagrams as a team.
Why use sequence diagrams? Why not code it?

• A good sequence diagram is still above the level of the real code (not all code is drawn on diagram)

• Sequence diagrams are language-agnostic (can be implemented in many different languages)

• Non-coders can read and write sequence diagrams.

• Easier to do sequence diagrams as a team.

• Can see many objects/classes at a time on same page (visual bandwidth).
sequence diagrams: examples
Flawed sequence diagram I

What's wrong with this sequence diagram?
Flawed sequence diagram 2

What's wrong with this sequence diagram?
Online bookstore example

1. The customer begins the interaction by searching for a book by title.
2. The system will return all books with that title.
3. The customer can look at the book description.
4. The customer can place a book in the shopping cart.
5. The customer can repeat the interaction as many times as desired.
6. The customer can purchase the items in the cart by checking out.
Online bookstore sequence diagram
Summary

• A sequence diagram models a single scenario executing in the system.

• Key components include participants and messages.

• Sequence diagrams provide a high-level view of control flow patterns through the system.