Need to Represent Runtime Interactions

- Cannot model how a system is actually achieve its functionality, with use cases and classes alone.

- Sequence diagrams
  - Model runtime interactions between system parts, together with communication diagrams and timing diagrams
  - Capture the order of interactions between system parts

Components of a Sequence Diagram

- Participants
  - System parts, interacting each other
  - Horizontally placed, without being overlapped
  - Each has a corresponding lifeline running down the page

- Standard format for participant names
  - name [selector] : class_name ref decomposition
Time, Events, Messages

- **Time**
  - Flows down
  - All about ordering, not duration
- **Events**
  - *A point where an interaction occurs*
  - Building blocks for signals or **messages**

Message Signatures

- **Format**
  - `attribute = message_name (arguments) : return_type`
  - Arguments are in the format of "<name> : <class>"

Activation Bars and Nested Messages

- **Activation Bars**
  - Optional
  - Means that objects are active – working on tasks.
- **Nested Messages**
  - A message triggers another message

Message Arrows (Types)

- **Important to understand the behavior between objects**
- **Synchronous messages**
  - Caller waits for the receiver to return from the invocation
Asynchronous message
- Caller initiate interaction(s) and not wait for return
  - e.g., editing and printing function in a word processor software
- In Java, threads are used.

In Java, threads are used.

Return message
- Optional
- Can clutter diagram

Participant creation and deletion message
- Participants could be alive for limited time in an interaction
  - i.e., they are created and destructed.
- Can use stereotyped messages:
  <<create>>
  <<destroy>>

Destroy messages are often not needed.
Example Sequence Diagram for a Use Case

Main flow from Use case description

Main Flow | Step | Action
---|---|---
1 | | The Administrator asks the system to create a new blog account.
2 | | The Administrator selects the regular blog account type.
3 | | The Administrator enters the author's details.
4 | | The author's details are checked using the Author Credentials Database.
5 | | The new regular blog account is created.
6 | | A summary of the new blog account's details are emailed to the author.

A Refined Sequence Diagram

Work on sequence diagram invariably goes on throughout the life of system modeling.

Top-Level Sequence Diagram Example

Applying Participant Creation
Managing Complex Interactions

- Sequence diagram can easily become huge and messy, contains too much details.
  - Difficult to understand and maintain
  - No built-in solution for loop and alternative flows

- Sequence Fragments (in UML 2.0)
  - Represented as a box enclosing a portion of the interactions within a sequence diagram
  - Assign fragment type
Sequence Fragment Example: ref type

- Represent "referenced" fragment
- Help diagram maintenance

sd means sequence diagram

Sequence Fragment Types

- ref
  - Sequence fragment reuse
- assert
  - Transaction - every step in an interaction must occur successfully
- loop
  - Parameters: min, max, guard condition
  - Loops interactions in a fragment
- alt
  - Parameters: multiple guard conditions
  - Models alternative paths

Opt
- single guard condition
- Single path conditionally executed

Neg
- Show some interactions that cannot (or should not) be executed. – although they are not yet deleted from the diagram

Par
- Parallel execution of the interactions in a fragment

Region
- Interactions in a critical region. (lock)
**Focus on the Links between Participants**

- Model which links are needed between participants to pass messages
  - The links are implied in sequence diagram
  - Interaction order is not the first priority in communication diagram.
    - However, the sequence diagram and communication diagram can often be converted each other.

- Participants, Links, and Messages
  - Order by numbering messages

**Nested Message and Concurrent Messages**

- Numbering scheme

**Concurrent Messages**

**Describing Messages**

- Invoking a Message Multiple Times
  - Example: *[i = 0..9]*

- Conditional invocation

**Describing Messages (cont.)**

- When a participant sends a message to itself
  - Case: an object calls its own method
Creating Communication Diagram

- Recall the sequence diagram from the previous slide.

Example Communication Diagram

- It is not necessary to have a sequence diagram before creating a communication diagram.
- The first step is to identify participants.

Example Communication Diagram (cont.)

- Add links for communication

Example Communication Diagram (cont.)

- Add messages
Example Communication Diagram (cont.)

Communication vs. Sequence Diagram
- Which one to use?
  - Communication diagram
    - Show the links between participants
  - Sequence diagram
    - Easy to read message ordering
    - Show asynchronous messages
  - Either one would be okay.
    - Show participants effectively
    - Showing message signatures
    - Support parallel messages
    - Easy to create and maintain diagrams

Modeling Detailed Timing Information
- Timing examples
  - An interaction that has to be completed within 10 seconds
  - Time to return after message must be 1 millisecond.
- Commonly associated with real-time or embedded systems
- Associate event with timing information:
  - Event invocation time
  - Message delivery time
  - Execution duration
- Look similar to a plot on a logic analyzer
Comparison with a logic analyzer
- Events ↔ Signals
- States represent participant states

Timing Diagram: An Example

Creating Timing Diagram
- Recall the sequence diagram from the previous slide.

Timing Constraints Examples

Requirement A.2
The content management system shall allow an administrator to create a new regular blog account, provided the personal details of the author are verified using the Author Credentials Database.

Requirement A.2 (Updated)
The content management system shall allow an administrator to create a new regular blog account within five seconds of the information being entered, provided the personal details of the author are verified using the Author Credentials Database.

Creating Timing Diagram
- Add participants involved in an interaction
- Omitted :AuthorDetails and :RegularBlogAccount because they are not involved in state changes
Creating Timing Diagram (cont.)

- Add states of each participant

Creating Timing Diagram (cont.)

- Representing timing
  - Exact time measurement
  - Relative time indicators
    - $t$ represents a point in time that is of interest

Creating Timing Diagram (cont.)

- Participant’s state-line
Timing Diagram

- For the example
  - With messages

Event on a Timing Diagram

- Distinction between events and messages are not very interested in timing diagram
- Events are represented as arrow from one participant’s state-line to another participant's state-line

Timing Constraints

- Can be associated with an event or a state
- May or may not be accompanied by boundary arrows
Timing Diagram

- For the example
  - With messages
  - With timing constraints

Organizing Participants in Timing Diagram

- For better readability

Alternate Notation

- Timing diagram can be easily messy
  - Especially for modeling complex interactions
  - Does not scale well

- Good property of alternate notation
  - Use single line for each participant
  - State-line is not needed
High-Level View of Interaction Relations

- Sequence, communication, and timing diagrams focus on specific details on interactions.
- Interaction overview diagram glues separate interactions together into a single picture for achieving a system concern.
  - Looks similar to activity diagram
  - Each action is an interaction

Interaction Overview Diagram

- Shows control flow passing through interactions
  - Begins with an initial node and ends with a final node
  - Can represent decisions, parallel actions, and loops
Interaction Overview Diagram
Example for the "Create a New Regular Blog" use case.