CS 370

REVIEW: UML Diagrams

DR. MICHAEL J. REALE

FALL 2015
Introduction

- **UML**
  - Unified Modeling Language
  - Very well recognized specification for modeling architectures, use cases, etc.
Two basic kinds of UML Diagrams:

- **Behavior diagrams** → dynamic behavior
  - Use case
  - Activity
  - State machine
  - *(Unofficial)* Information flow
  - *Interaction* → emphasize object interactions
    - Sequence
- **Structure diagrams** → static structure
  - Class
  - Object
UML Use Case Diagram

- **Use case diagram**
  - Describe...
    - a set of actions (**use cases**)...
    - that some system or systems (**subject**) should or can perform...
    - in collaboration with one or more external users of the system (**actors**).
UML Use Case Diagram: Notation

- **Use cases**
  - Action or sequence of actions

- **Actors**
  - Person, organization, or external system that plays a role in one or more interactions with your system

- **Associations**
  - Actor – Use Case
    - Invocation (NOT data flow)
  - Actor – Actor
    - One actor is a specific kind of actor
**Extending Use Cases**
- Adds optional behavior

**Including Use Cases**
- Means a given use case is part of another use case
UML Use Case Diagram Example

http://www.agilemodeling.com/artifacts/useCaseDiagram.htm
UML Activity Diagram

- **Activity Diagram**
  - High-level business processes or complex logic
    - Does NOT identify who does what

UML Activity Diagram: Notation

- **Initial node**
- **Activity final node**
- **Activity**
  - Single step of activity
- **Flow/edge**
  - Shows flow of control from one action to the next
- **Flow final**
  - Flow stops ONLY on this path (does not affect other flows)

http://www.sparxsystems.com/resources/uml2_tutorial/uml2_activitydiagram.html
**Condition**
- Must be true to traverse node
- Text on a flow/edge

**Decision**
- Diamond with one flow in and several out

**Merge**
- Merges ALTERNATE flows (NOT parallel flows)
- Diamond with several flows entering and one leaving

**Can actually combine decision and merge into one diamond**
UML Activity Diagram: Notation

- **Fork**
  - Beginning of PARALLEL flows

- **Join**
  - All flows MUST COMPLETE before processing may continue out of join

- Can combine fork and join
You can also have flows that trigger on or send events:

- **Send signal action**
  - Trigger some event

- **Accept signal action**
  - Asynchronous event occurred; flow starts there

- **Wait time action**
  - Event occurs at certain time or on certain schedule
UML Activity Diagram: Example

Online Shopping

[search]

Search Items

[found]

View Item

[made decision]

Add to Shopping Cart

[proceed]

[view cart]

Update Shopping Cart

[update needed]

[done with shopping]

[more shopping]

View Shopping Cart

Check Shopping Cart

A

Proceed to Checkout

B

Checkout

C

Shopping cart can be checked at any time
UML Activity Diagram: Example
State machine diagram
- Describes the states of object/interaction and transitions
  - Generally only for very complex classes
UML State Machine Diagram: Notation

- **Simple state**
  - No substates or regions
  - Optionally has *internal activities*
    - Actions performed while in this state
    - *Special keywords:*
      - **entry** \(\rightarrow\) behavior performed upon entry to the state
      - **do** \(\rightarrow\) ongoing behavior; performed as long as the element is in the state
      - **exit** \(\rightarrow\) behavior performed upon exit from the state
    - Can also add your own conditions:
      - **condition / operation**

[Image of UML State Machine Diagram]
**Composite state**
- Has nested states

- Can also use shorthand notation (and define sub-states elsewhere)
You can also have initial nodes, end nodes, conditions, decisions, forks, and joins like in activity diagrams.
UML State Machine Diagram: Water Example
State vs. Activity Diagram

- **State diagram**
  - Nodes are *states*
  - Transitions are events

- **Activity diagram**
  - Nodes are *activities*
  - Transitions are when activities complete (and conditions are met)
Example: Cheese Matcher In-Game State Diagram

Cleanup Game → go to In-Menu

Ask if Restart

Yes

Initialize Game → new game started

game ready

Cleanup Game

Won Game → [game won]

Game Lost → [game lost]

Check Game State → [no more matching cheeses]

[game still going]

Board Update → [matching cheeses]

Waiting for User Action → cheeses swapped
(Unofficial) UML Information Flow

- Information Flow diagram
  - Shows exchange of information between actors and system entities at very high level
  - Not an official part of UML
UML Sequence Diagram

- **Sequence diagram**
  - Shows *message* exchanges between *lifelines*
    - *Lifeline* = individual participant in the interaction
      - Represents a SINGLE instance
      - Primarily objects and actors
    - *Message* = one specific kind of communication
      - Specifies sender and receiver
      - Either start of execution or sending/receiving a signal
UML Sequence Diagram: Message Notation

- **Synchronous call**
  - Typically operation/function call
  - Wait until response
  - Filled arrow head

- **Asynchronous call/signal**
  - Send message → proceed immediately without waiting for return value
  - Open arrow head
UML Sequence Diagram: Message Notation

- **Create**
  - Creates a new lifeline (often class instance)

- **Delete**
  - Destroys a lifeline
  - Usually an X on the bottom to indicate end

- **Reply**
  - Reply to an operation call
  - Dashed line with arrow
**Execution**

- Period in participant's lifetime when it is:
  - Executing behavior/action
  - Sending signal
  - Waiting for reply
- Two execution occurrences → start and finish
- Thin grey or white rectangle
UML Sequence Diagram: Interaction Fragments

- **Interaction fragment**
  - Most general interaction unit
    - E.g., Execution

- **Combined fragment**
  - Interaction fragment that defines a combination (expression) of interaction fragments
  - Many different kinds:
    - `alt` - alternatives
    - `opt` - option
    - `loop` – iteration
    - `break` - break
    - `par` - parallel
    - `ref` - reference
UML Sequence Diagram: Combined Fragments

- **`alt` – alternatives**
  - At most one of the these will be executed, subject to guard conditions

- **`opt` – option**
  - Either guard condition true and it executes, or nothing happens

- **`loop` – iteration**
  - Loops through execution
    - No guards/limits \( \rightarrow \) loop forever
    - Can set max iterations
    - Can set boolean guard
UML Sequence Diagram: Combined Fragments

- **break - break**
  - If guard is true, break out of loop

- **par - parallel**
  - Execute in any order, possibly in parallel
  - *Shorthand notation when order on ONE lifeline is insignificant:*
**UML Sequence Diagram: Combined Fragments**

- `ref` → *interaction use*
  - Allows you to refer to another interaction
  - Combined fragment with operator `ref`
UML Sequence Diagram: Bookshop Example
UML Sequence Diagram: Facebook Auth. Example
UML Class Diagram

- **Class diagram**
  - *Static* model → classes, types, contents, relationships
    - Inheritance, aggregation, and association
    - Operations and attributes
UML Class Diagram: Notation

- **Class**
  - Represented by box
  - Name, attributes, and methods

- **Association**
  - Regular arrow
  - Can have numbers on either end:
    - 1 \( \rightarrow \) 1 student
    - ...is enrolled in...
    - 1..* \( \rightarrow \) 1 or more courses
UML Class Diagram: Notation

- **Abstract class**
  - Italicize the name

- **Interface**
  - Used the phrase «interface>> above name
UML Class Diagram: Notation

- **Inheritance**

- **Implements**
UML Class Diagram: Notation

- **Uses**

![Diagram](image-url)
**UML Class Diagram: Notation**

- **Aggregation**
  - "Weak aggregation"
    - A has a B, but:
      - B could also be in other aggregations
      - B still may exist if A deleted

- **Composition**
  - "Strong aggregation"
    - whole/part relationship
    - binary association
    - part could be included in at most one composite (whole) at a time
    - if a composite (whole) is deleted, all of its composite parts are "normally" deleted with it.
- **Object diagram**
  - Depicts objects and their relationships at a point in time
Specific instance of class
- Name: Classname

-Marvin: Student
Class vs. Object Diagram

- **Class diagram**
  - Shows *class type* hierarchies, contents, and relationships

- **Object diagram**
  - Shows *instances of classes* and their relationships