Introduction to Software Engineering: Unified Modeling Language Diagrams

John T. Bell

Department of Computer Science
University of Illinois, Chicago

Based on materials from multiple sources . . .

Acknowledgements

Images for this presentation were extracted from:

• Gamma, Erich et. al. “Design Patterns”.
In a UML class diagram, which of the following arrow types indicates specialization/generalization, a.k.a. inheritance?

---

**BRACE YOURSELVES**

**UML DIAGRAMS ARE COMING**
Overview

• The Unified Modeling Language, UML, is a standard means of communicating (object-oriented) software designs, mostly via diagrams.
• UML communicates software concepts in much the same way that blueprints communicate architecture and building specifications.
• Just as architects use different diagram types for different concepts, UML has different diagram types as well.

Part I – Product Description and Requirements Development

• Use case diagrams, a.k.a. scenario diagrams.
• Notes
• “Definition” of scenarios vs. use cases for use in CS 440.
• Use cases and how they are used.
Use case / scenario diagrams list examples of who uses the system how.

- System boundary is clearly defined.
- Each oval is a separate use case.
- "Actors" are any entity external to the system.
- Relationships between use cases can be indicated.

More Use Case / Scenario Diagram Examples

Relationships between use cases may be indicated with «includes», «extends», inheritance, or other labels.
“Notes” can be attached to any diagram.

A Problem with the term “Use Cases”

“Jacobson’s definition of a use case must have left some ambiguity; other authors have written about use cases and very few of them have the same idea of what it is. There are about 40 published definitions of “use case” with almost none of them agreeing. This chaos is unfortunate.”

- Robertson & Robertson, Chapter 4

Use Cases vs. Scenarios
For the purposes of CS 440:

• **Scenarios** are informal *stories*, written in a narrative fashion. They serve to convey general understanding in classical development and as acceptance tests in agile development.

• **Use Cases** are more formal, following a rigid template with specific contents, used as a first step in classical requirements development.

• “Use case” diagrams may also be used to document relationships between scenarios.

One Potential Use Case Template

<table>
<thead>
<tr>
<th>Use case ID:</th>
<th>Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre-conditions:</td>
<td></td>
</tr>
<tr>
<td>post-conditions:</td>
<td></td>
</tr>
<tr>
<td>initiated by:</td>
<td></td>
</tr>
<tr>
<td>Triggerring Event:</td>
<td></td>
</tr>
<tr>
<td>Additional Actors:</td>
<td></td>
</tr>
</tbody>
</table>

**Sequence of Events:**
1. Initiating event or action should be step 1, taken by initiating actor.
2. System response follows, indented right.
3. All external actions are aligned with step 1. (*"algorithm" style*)
4. All system responses are indented right, aligned with step 2. (*"response" style*)
5. All steps should be expressed in the active voice, clearly indicating *who* performs each action
6. The sequence of events should show a back-and-forth stimulus-response relationship.

**Alternatives:** These would be normal and expected variations from the base case.
**Exceptions:** These would be unusual variations from the base case, often caused by problems.

• List as NA if not applicable.
• The following **may be** added:
  • related use cases or scenarios
  • associated tests, systems, classes, etc.
  • revision history
  • references to other documents
  • author(s) / originator(s)
  • notes
• Alternatives and exceptions may be listed either as separate use cases or as notes to a base case.
• For regularly occurring periodic events, “time” can be listed as the initiating actor.
Use cases play an important role in requirements development.

- Detailed use cases document the specific actions that must be taken by the system during the course of a use case.
- These actions can be directly translated into functional requirements. (What it must **do**.)
  - Sample action: “The system displays a form for the user to enter their personal data.”
  - Sample requirement: “The system must display a form in which the user can enter personal data.”

Review

What is shown in a UML use-case diagram?

A. A list of different use-cases, and which actors are involved in each one.
B. Swim lanes, indicating exactly who performs each step of a use-case.
C. The different classes involved in a use-case, and the relationships between them.
D. The inheritance structure of the use-case tree, showing which use-cases are derived from which other use-cases.
E. The steps taken by the system during the course of a use-case.
Part II – Documenting Design

- Sequence Diagrams
- Collaboration / Communication Diagrams
- Class Diagrams
- Component Diagrams
- Package Diagrams
- Deployment Diagrams
- Activity Diagrams
- Finite State Diagrams

Sequence diagrams document the steps taken during a single use case.

Sequence diagrams are often the first step towards identifying the classes needed for a new application.

Underline indicates an object, as opposed to a class.
Sample Sequence Diagram for a Two-Button Watch

Communication Diagrams convey (almost) the same info as Sequence Ds

More compact, but a little less information

a.k.a. Collaboration diagrams.
Class Diagrams are the most widely used and widely known. Aggregates exist independently of their container; Composites do not.

Package diagrams group classes into subsystems or other groupings.

Associations can be made between or inside of packages.

Figure 2-17 Example of packages. The FieldOffice and EmergencyReport classes are located in the FieldStation package, and the Dispatcher and Incident classes are located in the DispatcherStation package.

Figure 2-18 Example of packages are used to group the related classes into packages.
Deployment diagrams show the allocation of components to platforms.

Deployment platforms may be (combos of) hardware or software.

Figure 7-2 A UML deployment diagram representing the allocation of components to different nodes. Web browsers on PCs and Macs can access a WebServer that provides information from a Database.

Figure 7-4 Allocation of MyTrip subsystems to devices and execution environments (UML deployment diagram). RoutingSubsystem runs on the OnBoardComputer; PlanningSubsystem runs on an Apache server.
Detailed Class Diagrams are Used for Object Design

Review

What type of diagram is nearly equivalent to a sequence diagram?

A. Class diagram
B. Communication diagram
C. Deployment diagram
D. Package diagram
E. Series diagram
Activity Diagrams are dynamic, showing activities & results

Swimlanes on activity diagrams indicate who does what
Finite State Diagrams show possible states of a system & transitions

Review

What type of diagram is shown here?

A. Class diagram
B. Package diagram
C. Sequence diagram
D. Use-Case diagram
E. Watch diagram