An Introduction to UML

Presenter: Josh Prowant

Purdue University Fort Wayne Campus
CPET 545 Service Oriented Architecture & Enterprise Applications
September 18, 2008

Outline

• Introduction
• Modeling Language Introduction
• UML Purpose/Benefits
• Fundamentals of UML Diagrams
• Models and Diagrams
• General Examples
• UML Process
Introduction

• Unified Modeling Language (UML)
  ▫ Standard for software and systems development
  ▫ High level of abstraction allows focus on important aspects of system’s design
  ▫ Ensures business functionality is complete and correct, end-user needs met, technical requirements met before expensive changes
  ▫ System-independent

Modeling Language

• Helps describe a system
  ▫ Notation: elements that make up a modeling language
  ▫ Semantics: descriptions of what notation means
• Source code
  ▫ Too detailed and not understood by common stakeholder
• Informal language
  ▫ Room for interpretation
Why UML?

- **Formal language**
  - Every element has a strongly defined meaning
- **Concise**
  - Simple and straightforward notation
- **Comprehensive**
  - Describes all important aspects of a system
- **Scalable**
  - Can handle massive or small-scale projects
- **Built on lessons learned**
  - Culmination of best practices of Object Oriented community
- **The Standard**
  - Transformability and interoperability

Fundamentals of UML Diagrams

- *Notes* for additional comments
- *Stereotypes* for special use or intent (elements role)
  - Typically has an associated icon (actor symbol)
  - Guillemets

Figure 1: UML Note and Stereotype

Figure 2: A Stereotype using guillemets
Models and Diagrams

- UML as a sketch – convey key points
- UML as a blueprint – detailed specification of a system with UML diagrams
- UML as a programming language – UML model to executable code

Models and Diagrams

- Structure Diagrams
  - Class, Object, Component, Composite Structure, Package, Deployment
- Behavior Diagrams
  - Use Case, Activity, State Machine
- Interaction Diagrams
  - Sequence, Communication, Timing, Interaction Overview
Models and Diagrams

Table 1: UML 2.0 Diagrams [1]

<table>
<thead>
<tr>
<th>Diagram type</th>
<th>What can be modeled?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case</td>
<td>Interactions between the system and users or other external systems. Also helpful in mapping requirements to your systems.</td>
</tr>
<tr>
<td>Activity</td>
<td>Sequential and parallel activities within the system.</td>
</tr>
<tr>
<td>Class</td>
<td>Classes, types, interfaces, and the relationships between them.</td>
</tr>
<tr>
<td>Object</td>
<td>Object instances of the classes defined in class diagrams in configurations that are important to the system.</td>
</tr>
<tr>
<td>Sequence</td>
<td>Interactions between objects where the order of the interactions is important.</td>
</tr>
<tr>
<td>Communication</td>
<td>The ways in which objects interact and the connections that are needed to support that interaction.</td>
</tr>
<tr>
<td>Timing</td>
<td>Interactions between objects where timing is an important concern.</td>
</tr>
<tr>
<td>Interaction</td>
<td>Used to collect sequence, communication, and timing diagrams together to capture an important interaction that occurs within the system.</td>
</tr>
<tr>
<td>Overview</td>
<td>The internal organization of groups of classes and components.</td>
</tr>
<tr>
<td>Composite</td>
<td>The internals of a class or component, and can describe class relationships within a given context.</td>
</tr>
<tr>
<td>Structure</td>
<td>Important components within the system and the interfaces they use to interact with each other.</td>
</tr>
<tr>
<td>Component</td>
<td>The hierarchical organization of groups of classes and components.</td>
</tr>
<tr>
<td>Package</td>
<td>How the system is finally deployed in a given real-world situation.</td>
</tr>
</tbody>
</table>

Class Diagram

- UML models consist of objects that interact by sending messages
  - Objects have things they know (attributes) and things they can do (behaviors or operations)
  - Objects are instances of classes

![Class Diagram](image-url)
Object Diagram

• Much like class diagram
• Name of specific instance of class on the left side of a colon; name of class on the right side of the colon
• Helpful for complicated relationships between instances of classes

myWasher : WashingMachine

Figure 4: A UML Object

Package Diagram

• To simplify complex class diagrams, classes can be grouped into packages

Figure 5: Package Diagram
Component and Deployment Diagrams

- Component (code module) diagrams are physical analogs of class diagrams
- Deployment diagrams show the physical configurations of software and hardware (nodes)

Use Case Diagram

- Describe system’s behavior from standpoint of external user
- Actor can be person or another system
State Machine Diagram

- State of an object depends on its current activity or condition (rounded rectangles)
- State machine diagram shows states of an object and transitions that cause a change in state

Activity Diagram

- Similar to a flowchart
- Shows activities involved in a single process (state diagram shows process itself)
Sequence and Communication Diagrams

- Interactions among objects – interaction diagrams
- Sequence Diagram
  - How operations are carried out (messages sent)
  - Organized by time and progresses from top to bottom
- Communication Diagram
  - Focus on object roles instead of the times that messages are sent
  - Sequence number shows order of messages

Sequence Diagram

- Operation of a washing machine
  - Lifeline: time that an object exists
  - Activation bar: duration of the execution message

Figure 10: Sequence Diagram
Communication Diagram

• Messages among timer, water pipe, and drum of washing machine example

![Communication Diagram]

Figure 11: Communication Diagram

UML Process

• Object Management Group
  1. Select a methodology (process used to gather and analyze requirements and design application to meet them)
  2. Select a UML Development Tool (aligned with a methodology)
  3. Get training
References