The Unified Modeling Language

by

Peter DePasquale

8/30/99
UML Background

• **What is the UML?** -- “The UML is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system”

• **Why model?** --
  - helps enforce communication among teams
  - assures architectural soundness
Goals of the UML

• Be independent of particular programming language and development methodology

• Support higher-level development concepts such as collaborations, frameworks, patterns and components

• Provide extensibility and specialization mechanisms to extend the core concepts
Brief history of UML

9/97  UML 1.1
  /    /
1/97  UML 1.0
     /
6/96  UML 0.9

Unified Method 0.8

Booch '91  OMT-2

Booch '93  Object Modeling Technique-1

Object Oriented Software Engineering (OOSE)

OOPSLA '95
UML’s Goal

- Represent an Object-Oriented software system throughout its development lifecycle
“Use Case” Diagrams

• Objectives
  – Specifies the behavior of the system (or part of it) and is description of a set of sequences of actions that a system performs to yield an observable result of value to an actor

  – Shows the relationship between the system and its environment
Use Case Diagrams

• Notation:
  – *Actors*:
    • represent role played by person or thing interacting with system
    • represented as stick figures
    • can be specialized/generalized

```
    +---------------------+   +---------------------+   +---------------------+
    |                     |   |                     |   |                     |
    |                      |   |                      |   |                      |
    |                      |   |                      |   |                      |
    |                      |   |                      |   |                      |
    |                      |   |                      |   |                      |
    |                      |   |                      |   |                      |
    +---------------------+   +---------------------+   +---------------------+
          Student
          /   /   /
  Graduate Student  Student  Undergraduate Student
```
Use Case Diagrams

• Notation:
  – Use Cases:
    • captures intended behavior of system
    • behavior specified by describing flow of events (informal structured text, pseudocode, formal structured text)
    • graphically represented as oval
    • can be specialized/generalized/included

![Diagram showing Use Case Diagrams]

Place Order <<include>>

Validate User

Check Password

Retinal scan
Use Case Diagram example

- Sample ATM banking system

[Diagram showing interactions between Customer, Individual Customer, Corporate Customer, Retail institution, and Sponsoring financial institution with processes like Perform card transaction, Process customer bill, Reconcile transactions, and Manage customer account within the Credit Card Validation System.]
Class Diagrams

• Objective:
  – To represent a set of classes, interfaces and collaborations and their relationships

• Notation:
  – Class representation: rectangle with 3 regions: name, attributes and operations

<table>
<thead>
<tr>
<th>FraudAgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount: Float</td>
</tr>
<tr>
<td>warning: Integer</td>
</tr>
<tr>
<td>new()</td>
</tr>
<tr>
<td>new(p:Policy)</td>
</tr>
<tr>
<td>process(o:Order)</td>
</tr>
<tr>
<td>isSuspect(o:Order)</td>
</tr>
<tr>
<td>isFraudulent(o:Order)</td>
</tr>
</tbody>
</table>
Sequence Diagrams

• Objective:
  – Represent object interactions over time

• Notation:
  – Objects - named rectangle
    – Lifelines - dashed vertical lines showing life of object
    – Events (within system domain) - directional arrows with labels
Class Diagram Example

Person
- name: String
- ID: Integer
- getContactInfo()
- getPersonalRecords()

ContactInfo
- Spouse's Name: String
- Phone: String

PersonnelRecord
- taxId
- salary
- address
Sequence Diagram example

- Simple description of interactions of use case objects

```
Time

u: ATM User

<table>
<thead>
<tr>
<th>Initiate withdrawal</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: ATM</td>
</tr>
<tr>
<td>&lt;&lt;create&gt;&gt;</td>
</tr>
<tr>
<td>b: Banking Mainframe</td>
</tr>
<tr>
<td>Validate transaction</td>
</tr>
<tr>
<td>Permit transaction</td>
</tr>
<tr>
<td>Dispense cash</td>
</tr>
</tbody>
</table>
```

- 13 -
Component Diagrams

• Objective: - Model the static implementation view of a system:
  – source code
  – executable releases
  – physical databases
  – adaptable systems

• Use to visualize static aspects of system’s physical and their relationships/specific their details for construction
Component Diagrams

– *Processes*

![Component Diagram](image)

– *Subsystems*

![Subsystem Diagram](image)
Calling Dependency Diagram example

- Showing component calling dependencies

![Dependency Diagram](image-url)
Compilation Dependency Diagram example

• Showing dependencies during compilation
Component Dependency Diagram example
Partial UML Architecture

Analysis

Use Case Diagrams

Design

Sequence Diagrams

Implementation

Component Diagrams
Summary

- UML supports representation of system throughout its development (analysis, design and implementation)

- Use Case diagrams - describe system’s behavior during requirements analysis and design

- Component diagrams - shows relationships between software components during design and implementation

- Sequence diagrams - represents object interaction from a temporal standpoint