UML and MBASE

Dan Port
University of Southern California
CSE Technology Week
February 1999

Why Use The UML With MBASE?

- Quickly becoming the notation standard for visual Object-Oriented modeling
  - OMG effort
  - Many major contributors such as Rational, Inc.
- Much (and growing) tool support
  - Rose98
  - Visio
  - Object Architect
- Rich and extensible modeling language
Challenges

- Not clear on how to support multiple levels of abstraction
  - Different audiences for MBASE models
  - UML "Design" focused, MBASE DD, A, D, and I
- UML Models are not explicitly integrated
  - Between and within diagrams
  - Between and within abstractions
- Model clashes
  - Examples:
    - COTS and custom objects
    - Use-Case Based System Specification, Requirements Driven
- UML terminology not entirely consistent or appropriate for all MBASE models (Success, Process, Product, Property)

Our Objectives

- Clear communication to intended audience
  - simple, straightforward, focused
- encapsulated views
  - separation of concerns
  - complexity management
- evolution of information
  - natural integration of views through refinement
  - promotes conceptual integrity, simple traceability
- Consistent with UML and related items
  - reduced dependency on custom documentation
  - take advantage of tool support
- Expressiveness flexibility: adapt to project size
Approach

- Slightly extend UML via "stereotypes"
  - added Entity, Generic Component, Behavior Class
  - added Entity, Context Diagram, Logical Component View, System Layer View models

- Modified use of some Existing UML related models
  - Use-Cases specialized to:
    - Organization Activities
    - System Behaviors
    - Requirement Scenarios
    - Test Cases

Approach Continued...

- Modified use of some Existing UML related models (continued...)
  - UML Class Diagrams
    - E-R Diagrams - "cloud" stereotype, simple, general
    - Component Models - Packages or "Generic Component"
      - Components are "compositions" and "composable"
    - Enterprise Behavior Classification - "behavior" stereotype
    - Enterprise Class Model - inheritance, optional aggregation
    - Object Interactions - explicit reference (outlet), inheritance discouraged
    - Logical Component View - Block Diagram + Components
    - System Layers - C2 style "light"
    - Object Class - inheritance, optional aggregation, implementation stereotypes and independent hierarchies
OCD 2.4 Entity Model

OCD 2.5 Activity Model
SSAD 2.1 Component Model

SSAD 2.2 Behavior Model
SSAD 2.4 Enterprise Model

Components

SSAD 2.4 Enterprise Model

Behaviors
SSAD 3.2.1 Detailed Components

- HOME
- M MENU
- Access
- M MENU Access
- User Controls

SSAD 3.2.2 Object Specifications
### SSAD 3.3.1 Operations Model

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Data</td>
<td>Transmit data from the source to the target</td>
</tr>
<tr>
<td>Receive Data</td>
<td>Receive data from the target</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>Acknowledge receipt of data</td>
</tr>
<tr>
<td>Error</td>
<td>Report a communication error</td>
</tr>
</tbody>
</table>

### Diagram

The diagram illustrates the interaction between the source and target nodes, showing the flow of data and the acknowledgment process.