Customizing UML for Component Design

Marc Born
GMD Fokus
born@fokus.gmd.de

Olaf Kath
Humboldt-Universität zu Berlin
kath@informatik.hu-berlin.de

www.dot-profile.de
Motivation

- Definition of a design method for component-based distributed (telecommunication-) systems
  - System realized by interworking of autonomous distributed entities
  - Interactions and behavior of entities supported by different middleware platforms
  - Support for all relevant concepts of application domain within design method, e.g.
    - different interaction kinds, e.g. continuous media interaction
    - design support for non-functional object interaction aspects, e.g. secure or transactional binding
Motivation & Goals (ctnd.)

- Provision of tool support for design method
  - (graphical) notation support
  - Generation of middleware platform specific code, e.g. interface definitions and implementation code
  - Simulation of systems behavior prior to implementation (wherever possible)

- Enabling design reuse
  - Enable import of IDL 2.3.1 specs
  - Allow for migration of other existing designs (e.g. TINA specifications, Z.130) to emerging platforms
General Approach

Concept Space
System entities and relations

Middleware Platforms
Mapping of Concepts (Code Generation)

Representation of Concepts

Notations

UML Workshop - Palm Springs, CA, November 2000
Concept Space Definitions

- Foundation: Discerning evaluation of current approaches
  - RM-ODP, X.901 - 904
  - Rational Unified Process
  - TINA Computational Modeling Concepts
  - CORBA Component Model
  - Enterprise Java Beans
  - ITU ODL, Z.130
Concept Space Definitions

- Basic concepts
  - Structural aspects
    - Computational object, interfaces, types, interaction kinds
  - Configuration issues
    - Single ports (like CCM), multiple ports (to acquire references dynamically)
  - Interaction rules
    - Binding and binding rules
  - Implementation structure
    - Artifacts, implementation elements
- Deployment aspects
  - Component, assembly, deployment „constraints
Concept Space Definitions

- Basic concepts
- Structured into views

Diagram showing:
- Interaction view
- Instance view
- Structural view
- Implementation view
- Deployment view
Notation

- More then one concrete notation possible
- Candidate Unified Modeling Language:
  - Object-oriented approach
  - Graphical notation
  - Metamodel based
  - Built in extension mechanism
  - Wide spread + existing tool support
- Approach: Definition of UML Profile
  - Extension of metamodel for defined concepts
  - Introduction of stereotypes, tagged values, constraints
  - Realization of profile with existing UML tool
Example

- Hypothetical Task: Develop an *Interactive TV Service* component
  - provide a number of combined audio/video channels to clients
  - shall be able to receive input from its clients in form of joystick and mouse events.
Structural Specifications

<<Signal>> Mouse
(from InteractiveGame)

<<consume>>
consumeMouse

<<Signal>> Joystick
(from InteractiveGame)

consumeJoystick
<<consume>>

<<Signal>> ThreeDJoystick
(from InteractiveGame)

consume3dJoystick

<<MediaSet>> Television
(from Logical View)

<<source>> MyTV

<<ODLInterface>> Service
(from InteractiveGame)

charge_info()

<<CO>> ServiceComponent

<<CO>> ClientComponent

s
<<supports>>

r
<<requires>>
Configuration Specification

- ClientComponent
  - Service
    - channel : r
  - Service
  - Service
    - channels : s
    - initial_channel : s
    - advertisment : s
Interaction Rules

: ClientComponent
  : Predicate

channel : r

: Service
  : ServiceComponent
  : Service

channels : s

{context offered_qos: self.sec >= level_0}
{context OperationalQoS: self.sec = None}
{context OperationalQoS: self.sec <= level_1}

: ClientComponent
  : Predicate

channel : r

intranet

GMD FOKUS

UML Workshop - Palm Springs, CA, November 2000
Deployment Specification (Components)

<<Capsule>>
InteractiveServerImpl

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Logical Package</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceComponent</td>
<td>ServiceComponent</td>
<td>DOT</td>
</tr>
<tr>
<td>SecurityDimension</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>MouseEvent</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>Mouse</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>VideoControl</td>
<td>ServiceComponent</td>
<td>DOT</td>
</tr>
<tr>
<td>ClientComponent</td>
<td>ServiceComponent</td>
<td>DOT</td>
</tr>
<tr>
<td>ThreeDJoystick</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>Audio</td>
<td>Logical View</td>
<td>DOT</td>
</tr>
<tr>
<td>statedescr</td>
<td>ServiceComponent</td>
<td>DOT</td>
</tr>
<tr>
<td>ThreeDJoystick</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>JoyEvent</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>InteractionEvent</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
<tr>
<td>Video</td>
<td>Logical View</td>
<td>DOT</td>
</tr>
<tr>
<td>Television</td>
<td>Logical View</td>
<td>DOT</td>
</tr>
<tr>
<td>Inustick</td>
<td>InteractiveGame</td>
<td>DOT</td>
</tr>
</tbody>
</table>
Deployment Specification (Assemblies)

<<Assembly>>
A1

<<CO>>
ClientComponent
(from ServiceComponents)

<<CO>>
ServiceComponent
(from ServiceComponents)

C1 : ClientComponent
channel : r

S1 : ServiceComponent
initial_channel : s
Code Generation Aspects

Concept Space
System entities and relations

CORBA 2.3 platform + continuous media extensions
• Similar to CORBA Component Model approach
• IDL generation for operational part
• Implied IDL for signal/continuous media communication
• C++ classes for implementation support
• Makefiles for compilation
• Interceptor code for binding rules and QoS negotiation
• Deployment descriptors for automated deployment

SDL 2000 Generation
• Package for interface part
  Data definition
  Signal definition
  Interface definition
• Package for implementation
  Agent definition
  Gate definition

Planned Environments
• CORBA Components Platform
• Quality Aware Middleware
  (QAM, KPN Research)
• ...

UML Workshop - Palm Springs, CA, November 2000
Demonstration

- Realization of UML profile + code generators with Rational ROSE
  - Minimal dependency on UML tool
  - Concept space and code generators realized as separate libraries

- Approach
  - Model Structure, Binding Rules and Ports, Implementation
  - Generation of platform specific IDL, C++ implementation code and makefile as well as SDL code for simulation
  - Embed Service in TINA Platform Environment through import of relevant platform specification