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 & Goals

- 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
      - Operational, signal and continuous media interaction
    - design support for non-functional object interaction aspects
      - explicit and implicit binding, Binding policies
Motivation & Goals

- Definition of a design method for component-based distributed (telecommunication-) systems
- 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 (whereever possible)
- Design Reuse
  - Enable import of IDL 2.3.1 specs
  - Allow for migration of other existing designs (e.g. TINA specifications) to emerging platforms
Motivation & Goals

- Definition of a design method for component-based distributed (telecommunication-) systems
- Provision of tool support for design method
- Method and middleware platform applied at European and Japanese PNOs
  - Lessons learned from early application phases
    - need more arguments to apply UML!
    - need better tools!
    - finding the best fitting UML construct for identified concepts is not trivial!
General Approach

Concept Space
System entities and relations

Representation of Concepts
-> Model

Mapping of Concepts
-> Components

Middleware Platforms

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

- Foundation: Discerning evaluation of current approaches
- Basic concepts
  - Structural aspects
    - Computational object, interfaces, types, interaction kinds
  - Configuration issues - Ports
    - Static configurations -> single ports (like CCM)
    - Dynamic configuration -> 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

- Foundation: Discerning evaluation of current approaches
- Basic concepts
- Structured into views
Notation

- More than one concrete notations possible
  - Many have been developed already
  - Don’t invent a new one!
Notation

- More than one concrete notations possible
- Unified Modeling Language:
  - Object-oriented approach
  - Graphical notation
  - Metamodel based
  - Built in extension mechanism
  - Wide spread + existing tool support
  - But – still some resistance in the telecoms arena
Notation

- More than one concrete notation possible
- Candidate Unified Modeling Language:
  - 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.
Configuration Specification

: ClientComponent
  channel : r

  : Service

: ServiceComponent
  channels : s
  initial_channel : s
  advertisement : s

  : Service
Implementation Specification

```
<<CO>>
ServiceComponent

<<implements>>
<<Artifact>>
Signal Processing

<<implements>>
<<Artifact>>
Charging

<<implements>>
<<Artifact>>
VideoControl

<<supply>>
Service
(from InteractiveGame)

<<consume>>
consumeMouse
charge_info()

<<consume>>
consumeJoystick

<<consume>>
consume3dJoystick

<<Signal>>
Mouse
(from InteractiveGame)

<<Signal>>
Joystick
(from InteractiveGame)

<<Signal>>
ThreeDJoystick
(from InteractiveGame)

Notation for „realized“ in the applied tool

UML Workshop - Palm Springs, CA, November 2000
```
Interaction Rules

ClientComponent

ServiceComponent

Service

channels : s

Intranet

Predicate

Predicate

ClientComponent

ClientComponent

ServiceComponent

ServiceComponent

{context OperationalQoS: self.sec = None}

{context offered_qos: self.sec >= level_0}

{context OperationalQoS: self.sec <= level_1}
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
Code Generation Aspects

Concept Space
System entities and relations

CORBA 2.3 platform + continuous media extensions

SDL 2000 Generation
- Package for interface part
  - Data definition
  - Signal definition
  - Interface definition
- Package for implementation
  - Agent definition
  - Gate definition
Code Generation Aspects

Concept Space
System entities and relations

CORBA 2.3 platform + continuous media extensions

SDL 2000 Generation

Planned Environments
- CORBA Components Platform – a subset of our proprietary platform, could lead to a proposal for C++ language mapping for CIDL
- Quality Aware Middleware (QAM, KPN Research)
- ...
Implementation Issues

- Realization of UML profile + code generators + RE tools
  - Based on Rational ROSE Extensibility API
  - Minimal dependency on UML tool
  - Concept space and code generators realized as separate libraries
Conclusions

- To base the concept space definitions on RM-ODP turned out senseful
  - Defined views refine ODP viewpoint languages
- With the profiling mechanism, all concepts can be expressed with UML
  - Do we need to standardise a component design profile?
- If restricted to the concepts of CORBA Components, the introduced profile covers some important aspects
  - The code generation for the implementation view could be starting point for a CIDL language mapping