UML2.0 & Components

Sébastien Gérard, François Terrier

CEA/Saclay, DTSI/SLA/LLSP
F-91191 Gif sur Yvette Cedex France
Phone: +33 1 69 08 58 24 - 62 59 ; Fax: +33 1 69 08 83 95
{ sebastien.gerard, francois.terrier }@cea.fr

www-drt.cea.fr/acclist.htm

www.carroll-research.org sancy.ensieta.fr/mda
Four RFPs for a new UML standard ➔ UML2

- **UML 2.0 Infrastructure RFP**
  - Improve UML alignment with other OMG modeling standards
  - E.g. MOF and XMI
  - Make the UML easier to understand, implement and extend
  - Improve extensibility mechanisms of the UML

- **UML 2.0 Superstructure RFP**
  - Support Component-Based Software Engineering
  - Clarify the semantics of the generalization, dependency, and association relationships
  - Support encapsulation and scalability in behavioral modeling
    - E.g. for state machines and interactions
  - Remove restrictions on activity graph modeling

- **UML 2.0 OCL RFP**
  - It solicits proposals for defining an OCL metamodel consistent with the UML

- **UML 2.0 Diagram Interchange RFP**
  - It focuses on the problem of UML diagram interchange
OMG process: ~2 years process

- UML2 infrastructure & superstructure: \( \rightarrow \) Voted

... Finalization Task Force started (*CEA member*)

- Will trigger revisions on SPT, SPEM, AS, EDOC, QoS...
Content of the UML2 superstructure specification

- **Part I: Structure**
  - Classes
  - Components
  - Composite Structures
  - Deployments

- **Part II: Behavior**
  - Actions
  - Activities
  - Common Behaviors
  - Interactions
  - State Machines
  - Use Cases

- **Part III: Supplement**
  - Auxiliary Constructs (Information flows, Models, Primitive types and Templates)
  - Profiles

- **Appendices**
  - Appendix A. Diagrams
  - Appendix B. Standard Stereotypes
  - Appendix C. Component Profile Examples
  - Appendix D. Tabular Notations
  - Appendix E. Classifiers Taxonomy
  - Appendix F. XMI Serialization and Schema
  - Appendix G. Glossary
Overview of the component package

- Model software systems of arbitrary size & complexity

- Adress Component-Based Software Engineering
  
  ➤ "A component is modeled throughout the development life cycle and successively refined into deployment and run-time."

  ➤ From implementation level to model level

- Define two categories of component
  
  ➤ Physical components (e.g. EJB or CORBA components)
    ➤ BasicComponents

  ➤ Logical components (e.g. business components)
    ➤ PackagingComponents
The UML 2 component model

- Propose structure diagrams to model component-based application

- Main concepts
  - Component and Connector

- Two components models
  - Interfaces: from programming models ex. EJB, CCM
    Components communicating via messages through the interfaces themselves (like usual component model, e.g. CCM... )
  - Capsules and Ports (typed by interfaces): from ROOM
    Components communicate together via messages going through their ports (like processes in SDL or capsules of "ROOM" model)
Details of the Component meta-class

- Modular replaceable part of a system encapsulating its contents

- Can be replaced at design-time or run-time

- Contain provided & required interfaces specifying:
  - Structural features (attribute, association, …)
  - Behavioral features (operation, reception, statemachine, …)

- Provide a two level view
  - Has an external view (or “black-box” view)
    - through its interfaces that may also specify contracts
  - Has an internal view (or “white-box” view)
    - Private properties and realizing classifiers
Chapter 2. Components

Notation for external view ("black-box" view)

Figure 1: condensed notation

Figure 2: notation with explicit interfaces

Figure 3: intermediate notation
Chapter 2. Interface: declaration

➔ public, partial and abstract view of a Classifier
(without implementation: no instances of interfaces)

Figure 2: notation with detailed interfaces

Figure 3: intermediate notation

➔ Defined as types, apply to classifiers, component type
Chapter 2. Interface: realization

- Interface: is realized by classifiers, possibly several...
- Similarly a classifier can realize several interfaces

```
<table>
<thead>
<tr>
<th>« Interface »</th>
<th>Starter</th>
</tr>
</thead>
<tbody>
<tr>
<td>start()</td>
<td></td>
</tr>
<tr>
<td>stop()</td>
<td></td>
</tr>
<tr>
<td>« reception »</td>
<td>OnOff</td>
</tr>
<tr>
<td>maxSp: float</td>
<td></td>
</tr>
</tbody>
</table>

TorqueManager

- start()
- stop()
- calcTorque()
- « reception »
- OnOff

+ maxSp: float
- targetSp: float

<table>
<thead>
<tr>
<th>« Interface »</th>
<th>Alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop()</td>
<td></td>
</tr>
<tr>
<td>« reception »</td>
<td>OnOff</td>
</tr>
</tbody>
</table>
```

- association means conforming realization association
Chapter 2. Interface: behavior

- can have constraints on use such as
- pre- or post-cond or protocol state machine

**Operation stop ()**

\[ \text{Pre} \ C_1 = \text{curState}.\text{has}(\text{running}) \text{ and } C_1 \]

\[ \text{Pre: Pre} \ C_1 \]

\[ \text{Post: if initCond} = \text{Pre} \ C_1 \]
\[ \text{then curState} = \text{waiting and } C_2 = \text{true} \]

**Operation start ()**

\[ \text{Pre} \ C_1 = \text{curState}.\text{has}(\text{waiting}) \]

\[ \text{Pre: Pre} \ C_1 \]

\[ \text{Post: if initCond} = \text{Pre} \ C_1 \]
\[ \text{then curState} = \text{running} \]

**Signal OnOff**

\[ \text{Pre} \ C_1 = \text{curState}.\text{has}(\text{waiting}) \]

\[ \text{Pre} \ C_2 = \text{curState}.\text{has}(\text{running}) \]

\[ \text{Pre: Pre} \ C_1 \text{ or Pre} \ C_2 \]

\[ \text{Post: if initCond} = \text{Pre} \ C_1 \]
\[ \text{then curState} = \text{running} \]
\[ \text{else if initCond} = \text{Pre} \ C_2 \]
\[ \text{then curState} = \text{waiting} \]
Chapter 2. Interface behavior conformance

- Interface / realization conformance protocol state machine
  ➤ State invariant, pre- and post-conditions of interface protocol apply on realization state machine

**TorqueManager**

+ start()
+ stop()
- calcTorque()
« reception »
+ OnOff
+ maxSp: float
- targetSp: float

**Starter**

start()
stop()
« reception »
OnOff
maxSp: float

➤ New states, transitions, operations, receptions and attributes are allowed

Possible formal interpretation:
- Real. state Inv. ⇒ Interf. state Inv.
- For each common operation
  ➤ Interf. Pre ⇒ Real. Pre
  ➤ Real. Post ⇒ Interf. Pre
Chapter 2. Interface: behavior

... A same realization conforming with two interfaces

- TorqueManager
  - start()
  - stop()
  - calcTorque()
  - « reception » +OnOff

- Starter
  - start()
  - stop()
  - « reception » +OnOff
  - maxSp: float

- « Interface »
- Alarm
  - stop()
  - « reception » +OnOff

FERIA-ADL – 02/12/2003 “UML2&Components"
Notation for internal view ("white-box" view)

« component » SpeedRegulator

« Provided interfaces »
- Starter
  - start()
  - stop()
« Required interfaces »
- Display

« realizations »
- TorqueManager
- SpeedSensorManager

Realization specification using general dependencies

SpeedRegulator

- TorqueManager
  - mySSM 0..1

Realization specification using nested classifiers

SpeedSensorManager

Display

Starter
Details of the **Connector** meta-class

- **Delegation connector** links interfaces of a component with contained parts
  - Used to model behavior hierarchy in nested components

- **Assembly connector** links required interface to a provided interface
  - Used to model assembly of components

- **Assume conformance of interfaces required and offered** on declaration and behavior
Embedded Components

- Delegation connector links interfaces/ports of a component with contained parts
  - Used to model behavior hierarchy in nested components
Chapter 2. Interface versus Ports

- Ports are instances typed by Interfaces...

- Reify realization classes of a set of interfaces...
User benefits from this new component concept

- **Component concept supported across life cycle**
  - In 1.x implementation heritage

- **Provided and Required Interfaces**
  - Also model the requirements on environment

- **Wiring or Assembly of components**
  - Model the connections between Provided and Required Interfaces (must be type substitutable)
  - Option to tie into Collaborations to model detailed usages (or reuse) of components
  - Option to add contracts (any Behavior) to Connectors

- **Detailed internal structure option**
  - Model the internal structure of complex components by defining the instance level structure
Superstructure presentation agenda

- Part I: Structure
  - Chapter 1. Classes
  - Chapter 2. Components
  - Chapter 4. Deployments

- Part II: Behavior
  - Chapter 5. Actions
  - Chapter 6. Activities
  - Chapter 8. Interactions
  - Chapter 9. State Machines

- Appendix A. Diagrams
Details of the deployments concepts

- Define the execution architecture of systems that represent the assignment of software artifacts to nodes

- Main related concepts
  - Artefact specifies a physical piece of information
    - E.g. model files, source files, binary, …
  - Device
    - Model physical computational resource with processing capability
    - May support artifacts deployment for execution
    - May consist of other devices
  - ExecutionEnvironment implements a standard set of services that Components require at execution
    - E.g. «OS», «workflow engine», «database system»
  - DeploymentSpecification is a general mechanism to parameterize a Deployment relationship
Notations for deployment

- Artifact
  - « artifact » SpeedRegulator.h
  - « artifact » SpeedRegulator.lib
- Component
  - « component » SpeedRegulator
  - « artifacts » SpeedRegulator.lib SpeedRegulator.h
- Device
  - « device » PowerPC 780
  - « OS » Linux
  - ExecutionEnvironment
  - CommunicationPath
- Device
  - « device » RegDisplay
  - « manifest »
  - « manifest »
Notations for deployment (seq.)

« device »
PowerPC 780

« OS »
Linux

« component »
SpeedRegulator

« artifacts »
SpeedRegulator.lib
SpeedRegulator.h

« device »
RegDisplay

« component »
RegulatorScreen
Outlines of an ACCORD/UML component

- Based on the UML2 paradigm of component
  - Same entity refined throughout the dev. Process
- High-level and Low-level views
Component at PrM level

Component based Architecture Model

MyComponent

- External view
  - Header file and associated library

MyComponentHF « HeaderFile »
MyComponentLib « Library »

- Internal view
  - All 3 models

MyComponent « PrM »
UML2 as ADL…

- **Component** *(type vs instance):*
  entity for grouping and encapsulating *(classes vs objects)*
  
  ➢ Hierarchy on structure, but not on dependencies

- **Very (too?) open semantic**…
  ➢ task, memory, headers, binaries, text files, etc.

- **All UML communications**
  ➢ synchronous, asynchronous, point to point, diffusion…

Component (type) view

Component instance view

Classes view

Objects (class instances) view
UML2 as ADL…

- Notion of connectors
  - Types: interfaces
  - Instances: Ports
  - Conformance rules
  
  ... very general, abstract \(\rightarrow\) not reified with concrete protocols

- Notion of behavior
  - Operations, Signals
  - Protocol statemachines
  - + attributes & associations

- Configuration:
  - Component diagrams (type/instances) + collaborations (e.g., seq. diag)
  - Connections and delegations
UML2 as ADL… need of finalization

… Mainly focussed on model level, than on final implementation

- Component and concurrency
  - Task, active objects inside the components…
  - Management of concurrent messages…

- Interface and QoS…

- Interface versus Port…

- Refinement, traceability rules → methodology
UML2 as ADL… ongoing works

- In the FTF, refinement required on:
  - notion of Port…
  - delegation, realization and mapping rules

- In update of QoS concept defined for UML1.x profiles (SPT, QoS&FT)

- In works on executable UML (MDA WG)
  - For component model simulation, analysis, etc.
CEA-List related researchs

- CARROLL: Common program among Thales, CEA & INRIA on system engineering (Model Driven + Middleware)
  - PROTES: UML profile for embedded system
  - ICE: Real Time Embedded components & containers for CORBA

- ACOTRIS: Linking asynchronous UML with synchronous formalisms (SIGNAL, SynDEx) and hardware architectures

- AIT-WOODDES: UML method for automotive systems
  - Embedded Real Time Component pattern

- EAST-AEE: UML profile for Automotive ADL...

- ARTIST NoE on Advanced Real Time: « Modeling & Component »

- Families: UML components and product families

- CLIPS: UML components for distributed industrial automation

... AS-CNRS « Composants et architectures » !
Web sites

- www.carroll-research.org
- www.acotris.c-s.fr
- http://wooddes.intranet.gr
- www.east-eea.net
- www.systemes-critiques.org/ARTIST
- http://sancy.ensieta.fr/mda
FAQ

- Implementation of one interface by several classes implementing each a part of the interface?
  - Pb of protocol statemachine composition and conformity

- Dynamic delegation, routing the link between the interface and its implementation dynamically to various possible implementations...

- ...