Model-Driven Development of Recursive CORBA Component Assemblies

Gernot Schmoelzer, Egon Teiniker, Stefan Mitterdorfer, Christian Kreiner, Zsolt Kovacs, Reinhold Weiss

Institute for Technical Informatics
Graz University of Technology
AUSTRIA
Overview

- Motivation
- Related Work
- Component Modeling
- Component Development Process
- CCM Metamodel Extension
- Transformation Process Example
- Implementation
- Conclusion and Future Work
Motivation

• Requirements for CBD
  – Granularity of Components
  – Encapsulation and Abstraction
  – MDA Approach
  – Extensibility Mechanism
  – Use of standards
Motivation

• Requirements for CBD
  – Granularity of Components
  – Encapsulation and Abstraction
  – MDA Approach
  – Extensibility Mechanism

• Solution
  – Hierarchical Component Composition
  – Modeling Process according to MDA
  – CCM Implementation
Related Work - MDA

MDA Layers

- Business model
- Platform-independent Components
- Platform-specific Components

PIM - Cross-model - PSM

Refinement
Related Work – Component Modeling

UML 1.4

• Component Diagram
• Deployment Diagram

• Possible:
  – components, dependencies, compositions, etc.

• Difficult:
  – provided vs. required interfaces
  – external/internal parts and relationships
  – component attribute definition
Related Work – Component Modeling

UML 2.0

• Composite Structure D.
• Component Diagram

• Possible:
  – external and internal structure of classifier
  – external and internal relationships
Component Development Process

Model Transformations:

- t1: UML 2 $\rightarrow$ ext. CCM
- t2: ext. CCM $\rightarrow$ UML 1.4
- t3: ext. CCM $\rightarrow$ IDL / XML
CCM Metamodel Extensions

Interface Connector:
- connecting one receptacle with one or more facets

Delegation Connector:
- connecting either multiple facets or multiple receptacles
Model Transformation: UML 2 → ext. CCM
Transformation Process Example

UML 2.0

UML 1.4 – CCM Profile

<<CORBAInterface>>
<<CCMComponent>>
<<CCMProvides>>
<<CCMUses>>
<<CCMDelegate>>
Implementation

Component IDL Definition

Component C1 {
    provides I1 f1;
    provides I4 f2;
    provides I5 f3;
    uses I2 r1
}

Component C2 {
    provides I2 f1;
    provides I6 f2;
    provides I5 f3;
    uses I3 r1;
    uses I2 r1
}

Component C3 {
    provides I1 f1;
    provides I6 f2;
    provides I3 f3; // private port
    uses I3 r1;
    uses I1 r2;    // private port
    uses I6 r3;    // private port
}
Implementation

Model

Component Assembly
Descriptor (XML)

```
<componentassembly>
  ...
  <connections>
    <connectinterface>
      <usesport>
        <usesidentifier>r1</usesidentifier>
        <componentinstantiationref idref="C1"/>
      </usesport>
      
      <providesport>
        <providesidentifier>f1</providesidentifier>
        <componentinstantiationref idref="C2"/>
      </providesport>
    </connectinterface>
    ...
  </connections>
  ...
</componentassembly>
```
Component Assembly Implementation

UML 2.0 Model

Assembly Configuration

Private ports

Public ports

Assembly Manager

C3 Home
C1 Home
C2 Home

:C3_AssyMgr
Component Assembly Implementation

Sequence Diagram

Client  C3  Assembly  C1  C2

1: create
2: build
5: provide_I3
9: connect_I1
7: provide_I6
6: provide_I1
10: connect_I6
11: connect_I2
12: connect_I3
3: create
4: create
6: provide_I1
7: provide_I2
8: provide_I6
11: connect_I2
12: connect_I3
Results

- **Reusability and extension mechanism**
  - wrapping reusable components
  - adding additional functionality

- **Continuous test-concept**
  - reusing test cases (mirror components)
  - adding additional functionality
Results

- Reduce of complexity
  - hiding complex component interconnections
  - automatic assembling
  - different roles in the development process require different abstraction levels

---

Institute for Technical Informatics
http://www.iti.tu-graz.ac.at

Graz University of Technology
Conclusion and Future Work

• Conclusion
  – Approach of modeling recursive component compositions in UML 2.0 (PIM)
  – Mapping of UML 2.0 modeling constructs to CCM Profile (PSM)
  – Implementation

• Future Work
  – Use in large business applications
  – Applying the concept to local/remote components
    (http://ccmtools.sourceforge.net)
  – Dynamic assembling
  – Deployment and Configuration
Discussion

Thank You!
Model Transformation: ext. CCM → UML 1.4

CCM_Metamodel

ExtComponentDef
- isInstance : boolean
  * 1 +component
     * +ends

ConnectorEndDef
+ ifaceName : String
+ navigable : boolean
  2..* +ends
  +connector 1

ConnectorDef
+ connectorKind : ConnectorKind

CCM_Profile

<< stereotype >>
CCMConnect

<< stereotype >>
CCMDelegate

<< stereotype >>
CCMComponent

<< BaseElement >>

UML

Association