Automated Transformation from Descartes Modeling Language to Palladio Component Model

Jürgen Walter, Simon Eismann, Adrian Hildebrandt

Dept. of Computer Science, University of Würzburg

Symposium on Software Performance, Nov 6th 2015, Munich, Germany
Differences PCM and DML?
Motivation

Benefits Transformation
- Tooling can be reused
- Improves understanding of differences
- Flexibility to change

Benefits Automation
- Automated transformation is faster, less error-prone and less expensive compared to manual extraction
Project Overview

Problem
- Tooling has to be developed for each formalism
- Comparison challenging
- Manual transformation error-prone and time consuming

Idea
- Model transformation of DML to PCM

Benefit
- Reuse of existing PCM tooling for DML
- Improved understanding of differences and similarities
- Simplified change of formalism

Action
- Implement a model-to-model transformation
Architectural Performance Modeling

Adaptation Process

Adaptation Points

System Architecture QoS Model

Application Architecture

Resource Landscape

Component Specifications (Interfaces + RDSEFFs)

Assembly Model (System)

Deployment Model (Allocation)

User Behaviour Model (Usage Model)

PCM Instance

<<Component Developer>>

<<Software Architect>>

<<System Deployer>>

<<Domain Expert>>

Jürgen Walter: Automated Transformation of Descartes Modeling Language to Palladio Component Model
Transformation Overview

Jürgen Walter: Automated Transformation of Descartes Modeling Language to Palladio Component Model
Hierarchical resource landscape

Information loss
Resource Landscape
Specification of Processing Resources

platform:/resource/MediaStore/default.resourcelandscape
  - Distributed Data Center distributedDataCenter
    - Data Center smallDataCenter
    - Computing Infrastructure AppServer
      - Runtime Environment
        - Active Resource Specification HDD
          - Processing Resource Specification HDD
        - Active Resource Specification CPU
        - Storage Infrastructure DBServer
        - Networking Infrastructure aName

HYPervisor

HYPervisor
OPERATING SYSTEM
OS VM
PROCESS VM
MIDDLEWARE
OTHER
Processing Resources

Hardware Layer
ComputingInfrastructure
Software Layer

“Server”

Resource Container

Information loss
Service Behavior

Example:

DML Abstraction Layers
- Fine-grained behavior
- Coarse-grained behavior
- Blackbox behavior

PCM Abstraction Layer
- Fine-grained behavior/SEFF
Fine-Grained Behavior Mapping

- FineGrainedBehavior
  - Component InternalBehavior
    - Actions
    - Abstract Action
  - Signature

- Resource DemandingSEFF
  - Steps
    - Start Action
    - Stop Action
    - Signature
Coarse-Grained Behavior Mapping

[Diagram showing CoarseGrained Behavior mapping to Resource Demanding SEFF steps, including Loop and Branch Actions, and Internal Action.]
Blackbox Behavior Mapping

- BlackBoxBehavior
  - ResponseTime
    - RandomVariable

- ResourceDemandingSEFF
  - Steps
    - InternalAction
      - ParametricResourceDemand
        - ProcessingResourceType: DELAY
          - PCMRandomVariable

- Start Action
- Stop Action
Technical Transformation Alternatives

- Direct-Manipulation
  - Multi-purpose programming language
  - e.g. JAVA

- Relational/Declarative
  - “Constraint solving”
  - e.g. QVT-R

- Operational/Imperative
  - e.g. Xtend

- Hybrid
  - Declarative and imperative
  - e.g. ATL, ETL
# Evaluation

## 26 Unit Tests

<table>
<thead>
<tr>
<th>Components</th>
<th>Behavior</th>
<th>Interface</th>
<th>Resource Landscape</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasicComponent</td>
<td>BlackBoxBehavior</td>
<td>Interface</td>
<td>LinkingResource</td>
</tr>
<tr>
<td>ComposedStructure</td>
<td>FineGrainedBehavior</td>
<td>Signature</td>
<td>ComputingInfrastructure</td>
</tr>
<tr>
<td>AssemblyConnector</td>
<td>ReleaseAction</td>
<td>Parameter</td>
<td></td>
</tr>
<tr>
<td>ProvidingDelegationConnector</td>
<td>AquireAction</td>
<td>PrimitiveDataType</td>
<td></td>
</tr>
<tr>
<td>RequiringDelegationConnector</td>
<td>ExternalCallAction</td>
<td>ComposedDataType</td>
<td></td>
</tr>
<tr>
<td>AssemblyContext</td>
<td>InternalCallAction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### StoEx

- BoolPMF, DoublePMF, EnumPMF, BoxedPDF, variableExpression, Term, Multiplication
Case-study
Summary

Problem

• Tooling has to be developed for each formalism
• Comparison challenging and time consuming
• Manual transformation error-prone

Idea

• Model transformation of DML to PCM

Benefit

• Reuse of existing PCM tooling for DML
• Improved understanding of differences and similarities
• Simplified change of formalism

Action

• Implement a model-to-model transformation
THANK YOU

https://se3.informatik.uni-wuerzburg.de/descartes/dml2pcm


