A Method for Aspect-Oriented Meta-Model Evolution

VAO 2014 – York

Reiner Jung    Robert Heinrich    Eric Schmieders
Misha Strittmatter    Wilhelm Hasselbring

22nd July 2014

This work was partially supported by the DFG (German Research Foundation) under the Priority Program SPP1593: Design For Future – Managed Software Evolution (grant HA 2038/4-1, RE 1674/6-1, PO 607/3-1) and the Helmholtz Association of German Research Centers
Driver of Meta-Model Evolution

1. Long-living software systems
2. Convergence of design-time and run-time models
e.g. MAPE scenarios [Ghezzi, 2011]

Aspect-Oriented Modeling (Concern-Driven Development)

- Separate concerns
- Reduce meta-model complexity
- Ease meta-model evolution and reuse
Palladio

- **Domain** Performance prediction
- **Meta-Model** 188 classes (PCM)
- **Concerns** Components, deployment, workloads, …

Scenario

- Reuse PCM as run-time meta-model for forecasting

Extensions

- Cloud
- Monitoring
Related Meta-Modeling Approaches

Related Work

- Meta-Model Extension by Subclassing [Steinberg et al., 2009]
- Decorator Pattern Approach [Langer et al., 2012]
- EMF Profiles
Architecture Meta-Model

Model \( \rightarrow \star \rightarrow \) Component
Meta-Model Extension by Subclassing
[Steinberg et al., 2009]

Related Work

Architecture Meta-Model

Model <- * Component

Probed Component

Monitoring MM
Meta-Model Extension by Subclassing

[Steinberg et al., 2009]

Related Work

Architecture Meta-Model

Model

Component

* Probed Component

Monitoring MM
Meta-Model Extension by Subclassing
[Steinberg et al., 2009]

Related Work

Architecture Meta-Model

- Model
- Component
- Cloud Component
- Probed Component
- Cloud MM
- Monitoring MM

Jung, Heinrich, Schmieder, Strittmatter, Hasselbring
Meta-Model Extension by Subclassing
[Steinberg et al., 2009]

Related Work

Architecture Meta-Model

- Model
- Component
- Cloud Component
- Probed Component
- Cloud MM
- Monitoring MM
Solution Requirements

cf. [Langer et al., 2012]

Method

1. **Non-invasive** meta-model annotation mechanism
2. **Separation of concerns** from different modeling domains
3. Honor different **types of relationships and roles** of meta-models
4. Should support a **formal specification** for extensions/annotations
5. Must support **existing technology** to ease integration
6. Easy application and **compatible with existing tooling**
Overview

Method

Central ideas

- Aspect-oriented meta-models
- Semantic of references
- Contextual meta-model patterns
Aspect-oriented Modeling Roles

Architecture Meta-Model

Model  *  Component
Aspect-oriented Modeling Roles

Architecture Meta-Model

- Model
- Component
- *

Deployment Meta-Model

- Model
- Computing Node
- *

Monitoring MM

- Probe
- *

Method
Aspect-oriented Modeling Roles

Method

Architecture Meta-Model

- Model
  - Component
    - * aspect

Deployment Meta-Model

- Model
  - Computing Node
    - * base

Monitoring MM

- Probe
  - *
Aspect-oriented Modeling Roles

Method

Architecture Meta-Model

- Model
- Component
- *

Monitoring MM

- Probe
- *

Deployment Meta-Model

- Model
- Computing Node
- *

Cloud Meta-Model

- Node Configuration
- base
- aspect

Jung, Heinrich, Schmieder, Strittmatter, Hasselbring

Meta-Model Evolution

22-07-2014 8 / 12
Semantic of References

Syntactical View on References

Semantical View on References

A -> B

A <-> B

A <-> B
Semantic of References

Syntactical View on References

Semantical View on References

Monitoring

Architecture

Probe

Component

Model Roles

Aspect/Base

Jung, Heinrich, Schmieder, Strittmatter, Hasselbring

Meta-Model Evolution
Semantic of References

Method

Syntactical View on References

A → B

Semantical View on References

Monitoring

Probe → Component

Architecture

Property ❯ Type

Method Body ❯ Method Declaration

Queue ❯ Trace

Derived Node

Component

Original Node

Aspect/Base Model Roles
Typing & Declarations
Derived Models

Jung, Heinrich, Schmieder, Strittmatter, Hasselbring
Meta-Model Evolution

22-07-2014
Use Cases
- Models@Runtime
- Generators
- Editors
- Simulation and Evaluation

Pattern
- Tracability
- Navigation
- Queries
- Execution / Behavior
- Data
- State
Use Case: Run-Time Model

Palladio Component Model

<<component>>
A
<<component>>
B
Use Case: Run-Time Model

- Observation Data Model
- Monitoring Trace Model
- Observation Update Transformation
- Palladio Component Model
  - <<component>> A
  - <<component>> B

Method
Use Case: Run-Time Model

Method

Observation Data Model

Monitoring Trace Model

Observation Update Transformation

Palladio Component Model

<<component>>

A

<<component>>

B

PCM2LQN Transformation

λ

μ
Use Case: Run-Time Model

Method

Observation Data Model

Observation Update Transformation

Palladio Component Model

<<component>>
A
<<component>>
B

PCM2LQN Transformation

LQN2PCM Trace Model

λ
μ

Evaluation

Deployment Adaptation

Result Model

Jung, Heinrich, Schmieder, Strittmatter, Hasselbring

Meta-Model Evolution

22-07-2014
Summary

- **Non-invasive** annotation mechanism through AOM
- **Separate concerns** represented by separate meta-models
- Better understanding of **pattern and semantics** in meta-models
- Use EMF/Ecore as **formal notation** for meta-models
- No new meta-meta-model required
- **Compatible** with existing frameworks

Outlook

- Detailed definition of meta-model pattern
  - Implications on tooling
- **Evaluation** of our method in a PCM modernization effort
  - Solid core PCM
  - Wide palette of extensions
Appendix
Conclusions

Model

Abstract Component

Component

Component Decorator

Cloud Component

Reliable Component

Cloud MM

Reliability MM

Base Meta-Model

Model

*
EMF Profiles
[Langer et al., 2012]

Conclusions

Features

- Provide a profile mechanism for EMF
- Multiple profiles can be annotated to one class
- Generic reusable profiles for reuse
- Comes with its own (meta-)meta-model

Downside

No tool-integration

- Xtext DSL framework [XText, 2011]
- KLighD diagram and auto-layout framework [Schneider, Spönemann, & von Hanxleden, 2013]
- Genmodel facilities