GECO: Automatic Generator-Composition for (Aspect-oriented) DSLs

Doctoral Symposium

Reiner Jung

Christian-Albrechts-Universität zu Kiel
Institut für Informatik

26.02.2014
Domain-specific Language (DSL)

- Created for a specific domain or aspect of a software system
- Properties: formal, informal, executable etc.
- Text-based or graphical languages
  - Text-based: grammars and meta-models
  - Graphical: visualization pattern and meta-models
DSL Construction Process

Motivation

Decision Analysis Design Implementation Deployment

[Mernik et al., 2005]
Automatic or semi-automatic tasks

- Meta-model generation
- Editor generation

Manual tasks

- Grammar specification
- Validator construction
- Generator construction
Modeling and Generator Construction

Motivation

Source Meta-Models / DSLs
- ADL
- Data
- Behavior
- UI
- Monitoring
- Protocols

Generator

Target Meta-Models / Technologies & General Purpose Languages (GPL)
- JPA
- J2EE
- Kieker
- JSF
- Javascript
- AspectJ
- Java

Meta-model changes result in generator code degradation

Automatic Generator-Composition

26.02.2014
Modeling and Generator Construction

Motivation

Source Meta-Models / DSLs
- ADL
- Data
- Behavior
- UI
- Monitoring
- Protocols

Target Meta-Models / Technologies & General Purpose Languages (GPL)
- JPA
- J2EE
- Kieker
- JSF
- Javascript
- AspectJ
- Java

Generator

Analysis

New or altered requirements

Meta-model changes result in generator code degradation

Automatic Generator-Composition

26.02.2014
Modeling and Generator Construction

Motivation

Source Meta-Models / DSLs
- ADL
- Data
- Behavior
- UI
- Monitoring
- Protocols

Target Meta-Models / Technologies & General Purpose Languages (GPL)
- JPA
- J2EE
- Kieker
- JSF
- Javascript
- AspectJ
- Java

Generator

Analysis

New or altered requirements

Meta-model changes result in generator code degradation
Solution Overview

The GECO Approach

Source Meta-Models / DSLs

Target Meta-Models / Technologies & General Purpose Languages (GPL)

Key problem: How to combine partial generators?

Automatic Generator-Composition

26.02.2014
Solution Overview

The GECO Approach

Source Meta-Models / DSLs
- ADL
- Data
- Behavior
- UI
- Monitoring
- Workflow

Target Meta-Models / Technologies & General Purpose Languages (GPL)
- JPA
- J2EE
- Kieker
- JSF
- Javascript
- AspectJ
- Java

Analysis

New or altered requirements

Key problem: How to combine partial generators?
Solution Overview

The GECO Approach

Source Meta-Models / DSLs

- ADL
- Data
- Behavior
- UI
- Monitoring
- Workflow

Target Meta-Models / Technologies & General Purpose Languages (GPL)

- JPA
- J2EE
- Kieker
- JSF
- Javascript
- AspectJ
- Java

Analysis

New or altered requirements

Key problem: How to combine partial generators?
Solution Overview

The GECO Approach

Source Meta-Models / DSLs
- ADL
- Data
- Behavior
- UI
- Monitoring
- Workflow

Target Meta-Models / Technologies & General Purpose Languages (GPL)
- JPA
- J2EE
- Kieker
- JSF
- Javascript
- AspectJ
- Java

Analysis

New or altered requirements
Solution Overview

The GECO Approach

Key problem How to combine partial generators?
### Different Generation Pattern

#### The GECO Approach

<table>
<thead>
<tr>
<th>SBM</th>
<th>Source Base Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAM</td>
<td>Source Aspect Model</td>
</tr>
<tr>
<td>TBM</td>
<td>Target Base Model</td>
</tr>
<tr>
<td>TAM</td>
<td>Target Aspect Model</td>
</tr>
<tr>
<td>TM</td>
<td>Target Model</td>
</tr>
</tbody>
</table>

References

Transformation

References
Different Generation Pattern

The GECO Approach

- SBM (Source Base Model)
- SAM (Source Aspect Model)
- TBM (Target Base Model)
- TAM (Target Aspect Model)
- TM (Target Model)

Transformation:
- \( T_{BM} \)
- \( T_{AM} \)

References:
Different Generation Pattern

The GECO Approach

SBM  Source Base Model  TBM  Target Base Model  \( \xrightarrow{T} \) Transformation
SAM  Source Aspect Model  TAM  Target Aspect Model  \( \xleftarrow{- -} \) References
TM   Target Model

Target Aspect Model
Target Base Model
Target Model
Different Generation Pattern

The GECO Approach

SBM Source Base Model
SAM Source Aspect Model
TBM Target Base Model
TAM Target Aspect Model
TM Target Model

T Transformation

References
Different Generation Pattern

The GECO Approach

SBM  TBM  TAM  SAM

TBM  TAM

SBM  TBM  TAM  SAM

TBM  TAM

SBM  TBM  TAM  SAM

TBM  TAM

SBM  TBM  TAM  SAM

TBM  TAM

SBM  Source Base Model
SAM  Source Aspect Model
TBM  Target Base Model
TAM  Target Aspect Model
TM   Target Model

Transformation

References
Potential Approaches

The GECO Approach

Approach 1 Weaving models
- Base and aspect model generators produce target models
- Reference resolver for join points (source to target level)
- Weaver combines target aspect and target base model
- Various target level technologies (AspectJ, Kermeta, ...)

Approach 2 Weaving partial generators
- Base and aspect generator are woven
- Integration based on source level meta-model references
- Requires strict coding guidelines and insertion pattern
- Can be based on higher order transformations [Tisi et al., 2009]
Approach 1

The GECO Approach

Base Model (BM)  Aspect Model (AM)
Approach 1

The GECO Approach

Base Meta-Model ← Base Model (BM) → Aspect Model (AM) → Aspect Meta-Model
Approach 1

The GECO Approach

Base Meta-Model \rightarrow Base Model (BM) \rightarrow Aspect Model (AM) \rightarrow Aspect Meta-Model

Base Meta-Model

Base Model (BM)

Aspect Model (AM)

T_{BM}

T_{AM}
Approach 1

The GECO Approach

Base Meta-Model ← Base Model (BM)

Target Base Meta-Model ← Target Base Model

Target Aspect Model Fragm.

Target Aspect Meta-Model

T_BM

T_AM

Weaver

Merged Target Meta-Model ← Merged Target Model

conforms to
Approach 1

The GECO Approach

Base Meta-Model

Base Model (BM)

Aspect Model (AM)

Aspect Meta-Model

Target Base Meta-Model

Target Base Model

Target Aspect Model Fragm.

Target Aspect Meta-Model

T_{BM}

T_{AM}

C, Java, EMF, JVM byte code

Merged Target Meta-Model

Merged Target Model

conforms to

AspectJ

AspectC++

Kermeta Weaver

Atlas Weaver

Weaver

Automatic Generator-Composition

26.02.2014 9 / 21
Approach 1

The GECO Approach

Base Meta-Model \rightarrow Base Model (BM) \rightarrow Aspect Model (AM) \rightarrow Aspect Meta-Model

Target Base Meta-Model \rightarrow Target Base Model \rightarrow Target Aspect Model Fragm. \rightarrow Target Aspect Meta-Model

Weaver

Merged Target Meta-Model \rightarrow Merged Target Model

conforms to
Approach 1

The GECO Approach

Base Meta-Model \( \xrightarrow{T_{BM}} \) Base Model (BM) \( \xrightarrow{} \) Aspect Model (AM) \( \xrightarrow{} \) Aspect Meta-Model

Target Base Meta-Model

Target Base Model

Target Aspect Model Fragm.

Weaver

Merged Target Meta-Model \( \xleftarrow{}) \) Merged Target Model

Reference Transformation
- Model traces for source model nodes
- Source model references
- Selection of target nodes
- Construction of target model references

conforms to
Evaluation Scenarios

Planned Evaluation

Generator for CoCoME [Rausch et al., 2011]
- Domain: Common Component Modeling Example (CoCoME)
- Users: DFG-SPP *Design for Future* projects
- Target platform: Cloud, J2EE, JSF, Java, AspectJ

Generator for MENGES DSL [Goerigk et al., 2012]
- Domain: Railway control centers (RCC)
- Users: Developers and engineers of RCC
- Target platform: PLCopen/XML, C
Evaluation Overview

Planned Evaluation

Goal A  Technique and method evaluation
Goal B  Approach evaluation in an enterprise scenario
Goal C  Approach evaluation in a legacy scenario with multiple target platforms
Goal A: Technique and Method Evaluation

Planned Evaluation

Objectives

- Feasibility of different model-element reference concepts
- Feasibility of model/code weaving approach
- Practicability of model-element reference concepts
  - With researchers of iObserve projects and students

Scenario

- Partial CoCoME model, based on the Palladio Component Model (PCM) [Becker et al., 2009]
- Instrumentation aspect language (IAL) [Jung et al., 2013a]
- Combination of IAL & ProtoCom [Giacinto and Lehrig, 2013]
Goal B: Enterprise Scenario

Planned Evaluation

Objectives

- Integration test of method and tooling
- Practicability of approach and tooling (J2EE domain)

Scenario

- Complete CoCoME model based on PCM and additional DSLs
- Adaptation of generators driven by CoCoME modification scenarios

Collaboration

- University Stuttgart
- UDE, KIT, CAU
- Other CoCoME collaborators
Goal C: Multiple Target Platforms

Planned Evaluation

Objectives

- Feasible for different target model/language domains (text, XML)
- Practicability of approach and tooling (RCC domain)
- Comparison with existing generator ⇒ cost benefit

Scenario

- Legacy meta-model and type-system
- Creation of generators for different target languages

Collaboration & Expert Interviews

- b+m Informatik, Stefan Zeug
- Scheidt & Bachmann, Hauke Fuhrmann
Current State

Project Status

Published

- Type-Systems for DSLs [Jung et al., 2013b]
- Data type language (DTL) [Jung, 2013]
- Instrumentation aspect language [Jung et al., 2013a]

Unpublished

- Survey on model join point notations
- Designing extensible meta-models (addressing PCM issues)
Work-Packages

WP1 Literature and Technology Research
  ▶ Model weaver & graph transformation
  ▶ Compiler construction

WP2 Communication
  ▶ More publications and presentations

WP3 Approach

WP4 Solution Design and Implementation

WP5 Evaluation
Aspect Oriented-Modeling (AOM)

- Formal Design Analysis Framework (FDAF) [Bennett et al., 2010]
  - Supports only UML class diagrams
  - Stub generation Java and AspectJ
- Theme/UML [Clarke and Baniassad, 2005]
  - Supports UML structural and behavior models
  - No code generation
  - Themes as aspects
- Reusable Aspect Models (RAM) [Klein and Kienzle, 2007]
  - Supports UML class and activity models
  - Weaves in modeling domain
  - Code generation for Java and AspectJ
Related Work

Project Status

Aspect Oriented-Modeling (AOM)

- Formal Design Analysis Framework (FDAF) [Bennett et al., 2010]
  - Supports only UML class diagrams
  - Stub generation Java and AspectJ
- Theme/UML [Clarke and Baniassad, 2005]
  - Supports UML structural and behavior models
  - No code generation
  - Themes as aspects
- Reusable Aspect Models (RAM) [Klein and Kienzle, 2007]
  - Supports UML class and activity models
  - Weaves in modeling domain
  - Code generation for Java and AspectJ

Code Generation

- No work on iterative/agile development [Mehmood and Jawawi, 2013]
Summary

- DSLs are developed iteratively
- Generators can depend on multiple DSLs
  - Resulting in complex generators
- Iterations of each DSL cause
  - Generator architecture degradation

Goal Provide an approach and tooling to improve generator
- construction
- re-use

Next step Realizing the first evaluation scenario

Release date July/August 2015


