The Generator Composition Approach for Aspect-Oriented Domain-Specific Languages

Disputation

Reiner Jung

1st July 2016
Introduction

Motivation

Software-System development and evolution
- Domain, technology and environment changes
- Addition and changes to requirements

Continuously growing complexity

Model-driven software development
- Provides: Specific views and models of software systems
- Requires: Model editors, evaluation tools, and code generators
Common Component Modeling Example

(Rausch et al. 2011; Heinrich et al. 2015)
Introduction

Common Component Modeling Example

(Rausch et al. 2011; Heinrich et al. 2015)
Introduction

Common Component Modeling Example

(Rausch et al. 2011; Heinrich et al. 2015)
Introduction

Common Component Modeling Example

(Rausch et al. 2011; Heinrich et al. 2015)
Introduction

Modeling CoCoME

Source Model

Target Code
Introduction

Modeling CoCoME

Source Model

Target Code

Reiner Jung
Introduction

Modeling CoCoME

System

Repository

Data Types

package data

data class TradingEnterprise {
    long id
    string name
    Product[] productSuppliers
}

data class ProductSupplier {
    long id
    string name
    Product[] offers
}

Source Model

Target Code
Introduction
Modeling CoCoME

System

Repository

Data Types

Behavior

Source Model

Target Code

Reiner Jung
Introduction

Modeling CoCoME

Source Model

System

Repository

Target Code

Data Types

Behavior

Kieker - Monitoring
Introduction
Modeling CoCoME

Source Model

Target Code

AspectJ

JavaEE

JPA - Entity Beans

Reiner Jung
Introduction

Modeling CoCoME

Source Model

System

Repository

Data Types

Behavior

Kieker - Monitoring

AspectJ

JavaEE

JPA - Entity Beans

ProtoCom++
## Introduction

### Modeling CoCoME

![Diagram of System, Repository, Data Types, and Behavior](image.png)

- **Source Model**
  - System
  - Repository
  - Data Types
  - Behavior

- **Target Code**
  - AspectJ
  - JavaEE
  - JPA - Entity Beans

- **ProtoCom++**

- **Kieker - Monitoring**
  - Use PCM on cocome "irl-examples/src/cocome.repository"
  - Advice TraceLogger
  - Pointcut point : EntryLogger

---

**Source Model**

- **System**
- **Repository**
- **Data Types**
- **Behavior**

**Target Code**

- **AspectJ**
- **JavaEE**
- **JPA - Entity Beans**

---

**Reiner Jung**
Introduction

Modeling CoCoME

Source Model

Repository

Data Types

Behavior

System

Repository

Data Types

Behavior

Source Model

Target Model

JavaEE

JPA - Entity Beans

AspectJ

Kieker - Monitoring

ProtoCom++
Introduction
Modeling CoCoME

Source Model

Repository

Data Types

Behavior

Behavior

Kieker - Monitoring

JPA - Entity Beans

JavaEE

AspectJ

Reiner Jung
Key challenges in generator development

- Domain and technology evolution
- Increasing complexity of generators
- Reusability of metamodels and generators
Key challenges in generator development

- Domain and technology evolution
- Increasing complexity of generators
- Reusability of metamodels and generators

Experts Generator and DSL reuse are not applied by industry
Introduction

Contributions

**GECO Approach**
- Metamodel modularization (Jung et al. 2014)
- Generator megamodel patterns (Jung et al. 2016)
- Generator fragment design

**GECO Evaluation**
- Instrumentation aspect and record languages (Jung et al. 2013)
- Generator composition language
- Software architecture evaluation (Jung et al. 2015)
The GECO Approach
The GECO Approach

Metamodels (EMOF)

\[ TG = (T, I, A, C, OE) \]

\[ T = (NT, ET, sT, tT) \]

\[ contains TG = t(n, m) | DC \]

\[ \text{contains} \]

\[ x \text{ contains} TG^y \]

\[ y \text{ contains} TG^z \]

\[ u^y \] (Biermann et al. 2012)
The GECO Approach

Metamodels (EMOF)

\[ TG = (T, \phantom{1234567890}) \]
\[ T = (N_T, E_T, s_T, t_T) \]

(Biermann et al. 2012)
The GECO Approach

Metamodels (EMOF)

\[
TG = (T, T)
\]

\[
T = (N_T, E_T, s_T, t_T)
\]

(Biermann et al. 2012)
The GECO Approach

Metamodels (EMOF)

\[ TG = (T, I, \ldots) \]
\[ T = (N_T, E_T, s_T, t_T) \]

(Biermann et al. 2012)
The GECO Approach

Metamodels (EMOF)

\[ TG = (T, I, A, C, ) \]
\[ T = (N_T, E_T, s_T, t_T) \]

(Biermann et al. 2012)
The GECO Approach

Metamodels (EMOF)

\[
TG = (T, I, A, C, OE)
\]

\[
T = (N_T, E_T, s_T, t_T)
\]

\[
\text{contains}_{TG} = \{(n, m) | \exists c \in C : n \preceq s_T(c) \land m \preceq t_T(c)\} \cup \{(x, y) | \exists y \in N_T : (x \text{ contains}_{TG} y \land y \text{ contains}_{TG} z)\}
\]

(Biermann et al. 2012)
The GECO Approach

Metamodels (EMOF)

\[
TG = (T, I, A, C, OE)
\]
\[
T = (N_T, E_T, s_T, t_T)
\]
\[
contains_{TG} = \{(n, m) | \exists c \in C : n < s_T(c) \land m < t_T(c)\} \cup \\
\{(x, y) | \exists y \in N_T : (x \ contains_{TG} y \land y \ contains_{TG} z)\}
\]

(Biermann et al. 2012)
Identifying Metamodel Partitions

1. Find all root classes
   \[ R = T \cup N T \cup P N T \cup e T P e T \]

2. Form parts for all \( r_i, P r_i \):
   \[ P_i = \text{contains TG}(r_i) \]

3. Detect overlapping parts \( O \) with
   \[ n = |R| \]
   \[ O = \{ P_i \cap X P_j \mid i, j \in [1, \ldots, n] \}^{i \neq j} \]

4. Remove the overlapping parts \( O \), with
   \[ m = |O| \]
   \[ P_i \cap X (m \neq j) = 0 \]

5. Remove identified partitions \( P_i \) from graph

6. Reiterate process with remaining graph
Identifying Metamodel Partitions

1. Find all root classes $R \subseteq N_T$

$$R = \{ \forall n_t \in N_T | \forall e_T \in E_T, ((e_t, n_T) \in s_T \land (e_T, n_T) \in t_T) \lor (e_T, n_T) \notin t_T \}$$
Identifying Metamodel Partitions
1. Find all root classes $R \subseteq N_T$

$$R = \{ \forall n_t \in N_T | \forall e_T \in E_T, ((e_t, n_T) \in s_T \land (e_T, n_T) \in t_T) \lor (e_T, n_T) \notin t_T \}$$

2. Form parts for all $r_i \in R$: $P_i = \text{contains}_{T_G}(r_i) \cup \{r_i\}$
Identifying Metamodel Partitions

1. Find all root classes \( R \subseteq N_T \)

\[
R = \{ \forall n_t \in N_T | \forall e_T \in E_T, ((e_t, n_T) \in s_T \land (e_T, n_T) \in t_T) \lor (e_T, n_T) \notin t_T \}
\]

2. Form parts for all \( r_i \in R \): \( P_i = \text{contains}_{TG}(r_i) \cup \{ r_i \} \)

3. Detect overlapping parts \( O_k \) with \( n = |R| \)

\[
O = \{ P_i \cap P_j | \forall i, j \in [1 \ldots n] \land i \neq j \land P_i \cap P_j \neq \emptyset \}
\]
Identifying Metamodel Partitions

1. Find all root classes $R \subseteq N_T$

   \[ R = \{ \forall n_t \in N_T | \forall e_T \in E_T, ((e_t, n_T) \in s_T \land (e_T, n_T) \in t_T) \lor (e_T, n_T) \notin t_T \} \]

2. Form parts for all $r_i \in R$: $P_i = contains_{TG}(r_i) \cup \{r_i\}$

3. Detect overlapping parts $O_k$ with \( n = |R| \)

   \[ O = \{ P_i \cap P_j | \forall i, j \in [1 \ldots n] \land i \neq j \land P_i \cap P_j \neq \emptyset \} \]

4. Remove the overlapping parts $O_k$, with \( m = |O| \)

   \[ \forall i \in [1 \ldots n] \quad P_i' = P_i \cap (\bigcup_{j=0}^{m} O_k) \]
Identifying Metamodel Partitions

1. Find all root classes $R \subseteq N_T$

   \[ R = \{ \forall n_t \in N_T | \forall e_T \in E_T, ((e_t, n_T) \in s_T \land (e_T, n_T) \in t_T) \lor (e_T, n_T) \notin t_T \} \]

2. Form parts for all $r_i \in R$: $P_i = \text{contains}_{TG}(r_i) \cup \{r_i\}$

3. Detect overlapping parts $O_k$ with $n = |R|

   \[ O = \{ P_i \cap P_j | \forall i, j \in [1 \ldots n] \land i \neq j \land P_i \cap P_j \neq \emptyset \} \]

4. Remove the overlapping parts $O_k$, with $m = |O|

   \[ \forall i \in [1 \ldots n] \quad P'_i = P_i \cap (\bigcup_{j=0}^{m} O_k) \]

5. Remove identified partitions $P'_i$ from graph
The GECO Approach

Metamodel Modularization

Identifying Metamodel Partitions

1. Find all root classes \( R \subseteq N_T \)

\[
R = \{ \forall n_t \in N_T | \forall e_T \in E_T, ((e_t, n_T) \in s_T \land (e_T, n_T) \in t_T) \lor (e_T, n_T) \notin t_T \}
\]

2. Form parts for all \( r_i \in R \): \( P_i = contains_{TG}(r_i) \cup \{r_i\} \)

3. Detect overlapping parts \( O_k \) with \( n = |R| \)

\[
O = \{ P_i \cap P_j | \forall i, j \in [1 \ldots n] \land i \neq j \land P_i \cap P_j \neq \emptyset \}
\]

4. Remove the overlapping parts \( O_k \), with \( m = |O| \)

\[
\forall i \in [1 \ldots n] \quad P'_i = P_i \cap \bigcup_{j=0}^{m} O_k
\]

5. Remove identified partitions \( P'_i \) from graph

6. Reiterate process with remaining graph
Metamodel Structures

- Structure and typing

(Jung et al. 2014)
The GECO Approach

Metamodel Modularization

Metamodel Structures

- Structure and typing
- Expressions

(Jung et al. 2014)
The GECO Approach

Metamodel Modularization

Metamodel Structures

- Structure and typing
- Expressions
- Declaration

(Jung et al. 2014)
The GECO Approach
Metamodel Modularization

Metamodel Structures

- Structure and typing
- Expressions
- Declaration
- Maps, e.g., Traces, Pointcuts

(Jung et al. 2014)
Aspect-Oriented Modeling

(Jung et al. 2014)
The GECO Approach

Metamodel Roles

Aspect-Oriented Modeling

(Jung et al. 2014)
The GECO Approach

Metamodel Roles

Aspect-Oriented Modeling

```
Monitoring
  ↓ aspect
Access Control
  ↓ base
System
```

(Jung et al. 2014)
The GECO Approach

Metamodel Roles

Aspect-Oriented Modeling

- Monitoring
  - aspect
- Access Control
  - base
  - aspect

View-Based Modeling

- Data
- System
  - base

(Jung et al. 2014)
The GECO Approach

Metamodel Roles

Aspect-Oriented Modeling

- Monitoring
  - base
  - aspect

Access Control
  - base
  - aspect

View-Based Modeling

- Data
  - independent

System
  - dependent
  - base

(Jung et al. 2014)
The GECO Approach

Metamodel Roles

Aspect-Oriented Modeling

- Monitoring
  - aspect
- Access Control
  - aspect
- System
  - base

View-Based Modeling

- Data
  - independent
- Behavior
  - dependent

(Jung et al. 2014)
The GECO Approach

Metamodel Roles

Aspect-Oriented Modeling

View-Based Modeling

(Jung et al. 2014)
Example Megamodel based on CoCoME Scenario

CoCoME Architecture

(Bézivin et al. 2004; Favre 2004)
Example Megamodel based on CoCoME Scenario

(Bézivin et al. 2004; Favre 2004)
Example Megamodel based on CoCoME Szenario

(Bézivin et al. 2004; Favre 2004)
Example Megamodel based on CoCoME Scenario

(Bézivin et al. 2004; Favre 2004)
The GECO Approach

Composition Megamodel Pattern Candidates

Model Transformation Reference

Reiner Jung
The GECO Approach

Generator Composition Megamodel Patterns

Simple
P1

Normal Aspect
P2

Inverted References
P3

Model Weaving
P4

<table>
<thead>
<tr>
<th>SBM</th>
<th>Source Base Model</th>
<th>SAM</th>
<th>Source Aspect Model</th>
<th>F</th>
<th>Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBM</td>
<td>Target Base Model</td>
<td>TAM</td>
<td>Target Aspect Model</td>
<td></td>
<td>Reference</td>
</tr>
</tbody>
</table>
The GECO Approach

Generator Composition Megamodel Patterns

Simple

P1

SBM

F_{BM}

TBM

Normal Aspect

P2

SBM

F_{BM}

F_{AM}

TBM

TAM

Inverted References

P3

SBM

F_{BM}

F_{AM}

TBM

TAM

SAM

Model Weaving

P4

BM

F_{FMerge}

SAM

F_{BM}

F_{AM}

TBM

TAM

BM'

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

F → Fragment

F ← Reference

Reiner Jung
The GECO Approach

Generator Composition Megamodel Patterns

Simple
P1

Normal Aspect
P2

Inverted References
P3

Model Weaving
P4

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

F: Fragment
Reference
The GECO Approach

Generator Composition Megamodel Patterns

Simple P1
- SBM → TBM
  - \( F_{BM} \)

Normal Aspect P2
- SBM → SAM
  - \( F_{BM} \)
  - \( F_{AM} \)
- TBM ← TAM

Inverted References P3
- SBM ← SAM
  - \( F_{BM} \)
  - \( F_{AM} \)
- TBM → TAM

Model Weaving P4
- BM ← SAM
  - \( F_{BM} \)
  - \( F_{AM} \)
  - \( F_{Merge} \)
- BM’

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

\( F \) = Fragment
\( \leftarrow \) = Reference
The GECO Approach
Generator Composition Megamodel Patterns

Simple
P1

Normal Aspect
P2

Inverted References
P3

Model Weaving
P4

SBM Source Base Model
TBM Target Base Model
SBM Source Base Model
TBM Target Base Model

References

SBM Source Base Model
TBM Target Base Model
SBM Source Base Model
TBM Target Base Model

F Fragment

Reiner Jung
The GECO Approach

Pattern P2 - Normal Aspect

SBM \rightarrow TBM

F_{BM}

SAM \rightarrow TAM

F_{AM}

SBM  Source Base Model
SAM  Source Aspect Model
TBM  Target Base Model
TAM  Target Aspect Model
TRM  Trace Model

F  Fragment
Reference
The GECO Approach

Pattern P2 - Normal Aspect

SBM  Source Base Model
SAM  Source Aspect Model
TBM  Target Base Model
TAM  Target Aspect Model
TRM  Trace Model

F  Fragment
F  Reference

SBM

TBM

SAM

TAM

F_{BM}  
F_{AM}
The GECO Approach

Pattern P2 - Normal Aspect

- **SBM**: Source Base Model
- **SAM**: Source Aspect Model
- **TBM**: Target Base Model
- **TAM**: Target Aspect Model
- **TRM**: Trace Model
- **FBM**: Fragment Reference

Diagram:

- SBM → SAM
- TBM → TRM
- TRM → FAM
- FBM
- TBM ← TAM
The GECO Approach

Pattern P2 - Normal Aspect

SBM  Source Base Model
SAM  Source Aspect Model
TBM  Target Base Model
TAM  Target Aspect Model
TRM  Trace Model

F  Fragment

Reference

Reiner Jung
The GECO Approach

Pattern P3 - Inverted References

SBM Source Base Model
SAM Source Aspect Model
TBM Target Base Model
TAM Target Aspect Model
TRM Trace Model
JPM Join Point Model

F Fragment

Reference

Reiner Jung 16 / 34
The GECO Approach

Pattern P3 - Inverted References

SBM Source Base Model
SAM Source Aspect Model
TBM Target Base Model
TAM Target Aspect Model
TRM Trace Model
JPM Join Point Model

Fragment Reference
The GECO Approach

Pattern P3 - Inverted References

SBM  Source Base Model
SAM  Source Aspect Model
TBM  Target Base Model
TAM  Target Aspect Model
TRM  Trace Model
JPM  Join Point Model

F Fragment
Reference
The GECO Approach

CoCoME Case Study - Generator Megamodel

Palladio Component Model

- Repository
- System

Reiner Jung
CoCoME Case Study - Generator Megamodel

The GECO Approach

Palladio Component Model
- Repository
- System

Behavior

DTL

data types

operations

Reiner Jung
The GECO Approach
CoCoME Case Study - Generator Megamodel

Palladio Component Model

Repository
System

Behavior

EJB/Servlets Stubs

DTL

\[ F_{\text{ProtoCom}} \]
The GECO Approach

CoCoME Case Study - Generator Megamodel
The GECO Approach

CoCoME Case Study - Generator Megamodel

P2 - Normal Aspect

Palladio Component Model
- Repository
- System

Behavior

EJB/Servlets Stubs

DTL

Java Snippets

Entities

F_{ProtoCom}

F_{Behavior}

F_{DTL}

Reiner Jung 17 / 34
The GECO Approach
CoCoME Case Study - Generator Megamodel

Palladio Component Model
Repository System

Behavior
data types
operations

EJB/Servlets Stubs
F_{ProtoCom}
F_{JW}

DTL

Java Snippets
entity classes
methods

EJB/Servlets

Classes
entity classes

Entities

Classes
entity classes

F_{javac}

F_{javac}

F_{Behavior}

F_{Behavior}

F_{DTL}

F_{DTL}

Reiner Jung
The GECO Approach

CoCoME Case Study - Generator Megamodel

Palladio Component Model
Repository
System

Behavior

DTL

EJB/Servlets Stubs

Java Snippets

Entities

EJB/Servlets

Classes

entity classes

Reiner Jung

17 / 34
The GECO Approach
CoCoME Case Study - Generator Megamodel
The GECO Approach

CoCoME Case Study - Generator Megamodel

![Diagram of the GECO Approach]

- **IRL** to **IAL** with operations
- **Palladio Component Model** with Repository and System
  - **Behavior** with data types and operations
  - **DTL**
  - **EJB/Servlets Stubs** with methods
  - **Java Snippets**
  - **Entities**
  - **Classes**

**Kieker Records**

**Classes**
The GECO Approach

CoCoME Case Study - Generator Megamodell
The GECO Approach

CoCoME Case Study - Generator Megamodel

P2 - Normal Aspect

Repository
System
Palladio Component Model
DTL Behavior
EJB/Servlets
EJB/Servlets Stubs
Java Snippets
Sensors
Aspect
aspect.xml
Fjavac
Fjavac,ajc
classes
Fsensor
Faspect
IRL IAL
Kieker
Records
Classes

iObserve CoCoME
The GECO Approach

CoCoME Case Study - Generator Megamodel

- Repository
- Palladio Component Model
  - System
  - Behavior
  - DTL
- EJB/Servlets Stubs
- Java Snippets
- Entities

- Kieker Records
- Sensors
- aspect.xml

- Classes
- web.xml

- iObserve CoCoME
The GECO Approach

CoCoME Case Study - Generator Megamodel

7 P2 - Normal Aspect
2 P4 - Weaving
The GECO Approach

Internal Structure of Generator Fragments

Technical Dimension  Semantic Dimension

(Mens et al. 2006; Biehl 2010)
The GECO Approach

Internal Structure of Generator Fragments

Technical Dimension

Source Model

Fragment

Target Model

Semantic Dimension

(Mens et al. 2006; Biehl 2010)

Reiner Jung
The GECO Approach

Internal Structure of Generator Fragments

Technical Dimension

Source Model

Fragment

Model Query
Aggregation
State
Output Creation

Control

Target Model

Semantic Dimension

(Mens et al. 2006; Biehl 2010)

Reiner Jung
The GECO Approach

Internal Structure of Generator Fragments

Technical Dimension

Source Model

Fragment

Model Query

Aggregation

State

Output Creation

Control

Target Model

Semantic Dimension

Source Model

Fragment

Target Model

(Mens et al. 2006; Biehl 2010)

Reiner Jung
The GECO Approach

Internal Structure of Generator Fragments

Technical Dimension

Source Model

Fragment

Model Query

Aggregation

State

Output Creation

Target Model

Semantic Dimension

Source Model

Fragment

Control

Expressions

Structure

Typing

Evaluation

Initialization

Target Model

(Mens et al. 2006; Biehl 2010)

Reiner Jung
Evaluation
Evaluation

Evaluation Design

**Qualitative Evaluation** based on GQM
Evaluate the effect of **GECO** on

- **Goal G1** the utility and program quality
- **Goal G2** the evolvability
- **Goal G3** the reusability

**Complementing expert interviews**
- Relevance of the goals
- Applicability of **GECO**
Evaluation

Questions

Utility and program quality

- **effort** spent on the development of features
- **modularity** of different generator implementations
- **understandability** of the implementations

Evolvability

- Change in **modularity** during the evolution
- Change in **understandability** during the evolution
- Effects on the **changeability** during evolution
- Change in **stability** during evolution

Reusability

- **modifiability** of the generator implementations
- **modularity** of the generator implementations
- **generality** of the generator implementations
Modularity

- Low complexity of the system
- Low coupling of modules of a system
- High inner module cohesion of a system

(Allen 2002; Allen et al. 2007)
Common Component Modeling Example  

- **Domain:** Information system
- **Source:** PCM, data type, behavior and monitoring DSLs
- **Target:** Java EE and AspectJ
- **Evaluation:** Combination of existing and new generators
  - ProtoCom
  - Data types, behavior and monitoring

**Evolution steps** ($F_{Behavior}$): 4

- Test GECO’s feasibility for generator construction
Evaluation

CoCoME Case Study - Behavior Evolution

Behavior Generator

Reiner Jung
Behavior Generator
1. Basic functionality
Evaluation

CoCoME Case Study - Behavior Evolution

Behavior Generator
1. Basic functionality
2. Stateless/-full components

Behavior Generator
- Control
- Structure
- Name Resolver
- Expression
- Type
- Statement

Complexity/Coupling [bits]
- Complexity
- Coupling
- Cohesion

Revisions
- 1
- 2
- 3
- 4

LOC [count]
- LOC

Edges [count]
- LOC
- Edges

Reiner Jung
Behavior Generator
1. Basic functionality
2. Stateless/-full components
3. Java EE lifecycle functions
Evaluation
CoCoME Case Study - Behavior Evolution

Behavior Generator
1. Basic functionality
2. Stateless/-full components
3. Java EE lifecycle functions
4. Persistence support

Reiner Jung
**New Generator for MENGES**  
(Goerigk et al. 2012)

- **Domain:** Embedded system for railway control centers
- **Source:** Nine DSLs covering different aspects and views
- **Target:** Single output model in PLCOpenXML for IEC61131-3 (IEC03)
- **Evaluation:** Comparison of generator implementations
  - Original MENGES generator
  - GECO-based generator
- **Evolution steps:** 14

➤ Test evolution effects of using **GECO**
Evaluation

MENGES Case Study - Generator Comparison

Reiner Jung 26 / 34
Structure and Typing

![Graph showing LOC, Complexity, and Coupling over revisions for MENGES and GECO. The graphs illustrate the evolution of these metrics with revisions.](image)

**Evaluation**

**MENGES Case Study - Generator Comparison**

- **Lines of Code (LOC) [count]**
  - MENGES: 10816
  - GECO: 7025
  - Ratio: 1.5396

- **Complexity [bits]**
  - MENGES: 13921.88
  - GECO: 6675.88
  - Ratio: 2.0854

- **Coupling [bits]**
  - MENGES: 10983.81
  - GECO: 5060.83
  - Ratio: 2.1704

Reiner Jung 26 / 34
Expressions and Statements

Reiner Jung
Evaluation

MENGES Case Study - Generator Comparison

Refactoring and Communications

![Graph showing LOC, Complexity, and Coupling over Revisions for MENGES and GECO](image)

Reiner Jung 26 / 34
Evaluation

MENGES Case Study - Generator Comparison

Improvment of Polymorphism

- LOC [count]
- Complexity [bits]
- Coupling [bits]

MENGES GECO Ratio

- Lines of code: 10816 7025 1.5396
- Complexity: 13921.88 6675.88 2.0854
- Coupling: 10983.81 5060.83 2.1704
Timers and Template Improvements

- LOC [count]
- Complexity [bits]
- Coupling [bits]

MENGES
- Complexity
- Coupling

GECO
- Complexity
- Coupling

MENGES GECO Ratio

Lines of code
- MENGES: 10816
- GECO: 7025

Complexity
- MENGES: 13921.88
- GECO: 6675.88

Coupling
- MENGES: 10983.81
- GECO: 5060.83
Evaluation

MENGES Case Study - Generator Comparison

Maintenance

Reiner Jung
Evaluation

MENGES Case Study – Generator Comparison

Maintenance

<table>
<thead>
<tr>
<th></th>
<th>MENGES</th>
<th>GECO</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
<td>10816</td>
<td>7025</td>
<td>1.5396</td>
</tr>
<tr>
<td>Complexity</td>
<td>13921.88</td>
<td>6675.88</td>
<td>2.0854</td>
</tr>
<tr>
<td>Coupling</td>
<td>10983.81</td>
<td>5060.83</td>
<td>2.1704</td>
</tr>
</tbody>
</table>
Evaluation

Interviews

Results

- Reuse not applied by practitioners
- **GECO** patterns and modularization
  - Supportive for generator development
  - Applicable to own generator development/evolution

Industry

- Interviewees 17
- Experience range first year to senior engineer
- Agile/iterative development
- Evolution induced by customers and framework evolution

Research

- Interviewees 6
- PhD candidates and postdoc researchers
- Agile/iterative development, limited maintenance
- Evolution induced by personal/project needs

Reiner Jung 27 / 34
Related Work
Related Work

Modeling and Code Generation

Aspect-oriented and view-based modeling
- Reusable aspect models (RAM) (Klein et al. 2007)
- Orthographic software modeling (OSM) (Atkinson et al. 2010)

Aspect-oriented code generation
- Theme/UML (Mehmood et al. 2013)
- FDAF (Clarke et al. 2005)
- RAM-based (Bennett et al. 2010)

(Kienzle et al. 2009; Kramer et al. 2011)
Related Work

Transformation Modularization

Reuse and product lines
- Template-based transformations (Kapova et al. 2010)
- Genesys approach (Jörges 2013)

Modularization
- Genericity for model management operations (Wimmer et al. 2011)
- Factorization and composition of transformation (Sánchez Cuadrado et al. 2008)
- Chaining of transformations (Vanhooff et al. 2006)
- Localized transformations (Etien et al. 2015)
Conclusion
Conclusion

Contributions

Approach
- Metamodel modularization and partitioning
- Generator composition megamodel patterns
- Internal modularization of generator fragments

Replication Package
- Sources and datasets
  http://dx.doi.org/10.5281/zenodo.46552
- Software snapshots
  http://dx.doi.org/10.5281/zenodo.47129
- MENGES sources can be accessed via b+m informatik AG
Generator framework and composition tooling

https://github.com/rju/geco-composition-language.git
Architecture analysis tool

https://github.com/rju/architecture-evaluation-tool.git
Generators used in CoCoME case study

https://github.com/research-iobserve/
Generator framework and composition tooling
https://github.com/rju/geco-composition-language.git

Architecture analysis tool
https://github.com/rju/architecture-evaluation-tool.git

Generators used in CoCoME case study
https://github.com/research-iobserve/
Conclusion

Future Work

Evaluation
- Metamodel modularization and partitioning
- GECO used for modernization, e.g., ProtoCom
- Evaluating technology impact on megamodel patterns

Tool Development
- Integration of GECO generators in build systems
- Instrumentation aspect and record language
  - IAL integration in Kieker
  - IRL evolution, e.g., trace support
References


Klein, Jacques et al. (2007). “Reusable Aspect Models.” In: 11th Workshop on Aspect-Oriented Modeling, AOM at Models’07,


Foundations
Model Traces
Model traces are a set of source and target nodes with a relation between them. (Aizenbud-Reshef et al. 2005)

Approaches (Galvão et al. 2007)
- Constructive
  - TraceAddr adds trace model support to ATL (Jouault 2005)
- Reconstructive
  - Heuristic (Grammel et al. 2010; Saada et al. 2013)
  - Probabilistic (Antoniol et al. 2002)
  - Property matching
Approach
Approach

Join-Point Computation

Source
Models

Monitoring Aspect

Shopping Cart
Approach

Join-Point Computation

Source Models

Monitoring Aspect

call(void init (...)) ||
call(void service(...))

Target Models

class Cart extends HTTPServlet {
    public void init (...) {
        ...
    }
    public void service (...) {
        ...
    }
}

AspectJ Pointcuts

Java Servlet

Reiner Jung
Approach

Join-Point Computation

Source Models

Monitoring Aspect

call(void init (...)) ||
call(void service(...))

Target Models

class Cart extends HTTPServlet {
  public void init (...) {
    ...
  }
  public void service (...) {
    ...
  }
}

AspectJ Pointcuts

Java Servlet

Reiner Jung
Approach

Join-Point Computation

Source Models

Monitoring Aspect

Shopping Cart

Target Models

aspectJ Pointcuts

class Cart

method init

return type void

parameter . . . .

method service

return type void

parameter . . . .

Java Servlet

Reiner Jung
Approach
Join-Point Computation

Source Models

Monitoring Aspect

Target Models

Shopping Cart

AspectJ Pointcuts

class Cart

method init

return type void

parameter

method service

return type void

parameter

Trace Model

Java Servlet

AspectJ Pointcuts

Reiner Jung
Approach

Join-Point Computation

Source Models

Monitoring Aspect

Target Models

Shopping Cart

Trace Model

AspectJ Pointcuts

Java Servlet

class Cart

method init

return type void

parameter

method service

return type void

parameter

Reiner Jung
Approach

Join-Point Computation

Source Models

Monitoring Aspect

Target Models

AspectJ Pointcuts

Shopping Cart

Trace Model

class Cart

method service

method init

return type void

parameter

return type void

parameter

Java Servlet

Reiner Jung
Evaluation
Evaluation

**Metrics**

**Effort** developer days per feature

**Modularity**

- Low *complexity* of the system
- Low *coupling* of modules of a system
- High inner module *cohesion* of a system

*(Allen 2002; Allen et al. 2007)*

**Understandability** inverse of *complexity*

*(Laitinen 1996)*

**Changeability**

- Low *coupling* of modules of a system
- High inner module *cohesion* of a system

*(ISO11)*

**Stability of the code base**

- Low *coupling* of modules of a system

*(ISO11)*

**Evaluation** Measure properties for each revision

Reiner Jung
Evaluation

Metric: Amount of Information in the System

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Hyperedges</th>
<th>Probability ( \hat{p}_l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>1/5</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
<td>1/5</td>
</tr>
<tr>
<td>3,5</td>
<td>001</td>
<td>2/5</td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>1/5</td>
</tr>
</tbody>
</table>

\[
\text{Size}(S) = \sum_{i=1}^{n} (- \log_2 \hat{p}_{L(i)})
\]

\[
\text{Size}(S) = 3 \times 2.322 + 2 \times 1.322 = 9.610 \text{bit}
\]

Metric by Edward B. Allen Allen et al. 2007
Evaluation

MENGES LOC and Modules

Reiner Jung
Evaluation

MENGES LOC and Modules

Reiner Jung
Evaluation

MENGES Coupling Delta

Reiner Jung
Evaluation

MENGES Coupling Delta

Reiner Jung

53 / 34
Tooling
Java Graph Mapping

- Modules represent classes
- Nodes represent methods
- Edges represent
  - method calls
  - access to class features
- Java interfaces (modules)
- Framework classes (only when used)
- Ignore data type classes

Software Complexity Analysis

https://github.com/rju/architecture-evaluation-tool
package demo

import demo.EntryEvent
import demo.ExtendedEntryEvent
import demo.ExitEvent

use pcm on cocome "irl-examples/src/cocome.repository"

advice EntryLogger () {
    before EntryEvent(time, $signature) ExtendedEntryEvent(time, $signature, $classname)
    after ExitEvent(time, $signature)
}

pointcut point class cocome.TradingSystem.Inventory.Data.Persistence

pointcut complex class cocome.TradingSystem.Inventory {
    Data.**
    exclude Data.Persistence.**
}

aspect point : EntryLogger

aspect complex : EntryLogger
package demo

@author 'Reiner Jung' @since '1.5'
entity ArrayExample {
    int [10] staticArray
    int [] dynamicArray
    int [10][5][][9] mixed
    string [][][][6] stringMixed
}

template Event {
    long timestamp
}

template OperationSignature {
    string signature
}

entity EntryEvent : Event, OperationSignature
entity ExitEvent : Event, OperationSignature

entity ExtendedEntryEvent extends ExitEvent {
    string classSig
}