A Generator Composition Approach for Aspect-Oriented DSLs

ICMT 2016

Reiner Jung  Robert Heinrich
Wilhelm Hasselbring

5th July 2016
Software 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

➤ Evolution provoke generator alterations
Common Component Modeling Example

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

System

WebFrontend::UseCases

WebService::CashDesk

TradingSystem::Inventory

WebService::Inventory

ServiceAdapter

IEnterpriseReporting

IStoreManager

IEnterpriseManager

ICashDeskModel

ICashDesk

ICardReader

IUserDisplay

IPrinter

IBarcodeScanner

ICashBox

IStoreInventory

AccountSaleEvent

IReporting

IStoreInventoryManager

ServiceAdapter

Reiner Jung
Modeling CoCoME

Source Model

Target Code
Introduction

Modeling CoCoME

System

Repository

Data Types

Reiner Jung
Introduction

Modeling CoCoME

Source Model

Target Code

Repository

Data Types

Behavior

Reiner Jung
Introduction

Modeling CoCoME

Source Model

Target Code

Repository

Data Types

Behavior

Kieker - Monitoring

Reiner Jung
**Introduction**

**Modeling CoCoME**

---

**Source Model**

**Target Code**

---

**AspectJ**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<aspectj>
  <weaver options=""/>
  <aspects>
    <aspect name="AbstractEntryLoggerAdvice0"/>
    <aspect name="AbstractEntryLoggerAdvice1"/>
    <concrete−aspect extends="AbstractEntryLoggerAdvice" name="EntryLoggerAdvice0">
      <pointcut expression="TradingSystem.Inventory.Data.Persistence" name="point"/>
    </concrete−aspect>
  </aspects>
</aspectj>
```

---

**JavaEE**

```java
package org.cocome.tradingsystem.inventory.data.enterprise;
...
public final class EnterpriseQueryProvider implements EnterpriseQuery {
  @Override
  public TradingEnterprise queryEnterpriseById(final long enterpriseId, final IPersistenceContext pctx) {
    final EntityManager em = __getEntityManager(pctx);
    return em.createQuery("SELECT t FROM TradingEnterprise t WHERE t.id = :enterpriseId", TradingEnterprise.class).getResultList();
  }
}
```

---

**WebFrontend::UseCases**

```
package data;
```

---

**Kieker - Monitoring**

```java
use pcm on cocome "irl-examples/src/cocome.repository" advice TraceLogger () {
  before OperationBeforeEvent(time, signature, classname, signature)
  after OperationAfterEvent(time, signature, classname)
}
```

---

**JPA - Entity Beans**

```java
@Persistent
public class TradingEnterprise {
  private long id;
  private String name;
  private List<Store> stores;
  private List<ProductSupplier> productSuppliers;
  private @Transient boolean selected;
  public long getId() {
    return this.id;
  }
  public void setId(final long id) {
    this.id = id;
  }
}
```

---

**Data Types**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```

---

**System**

**Repository**

**Behavior**

**Repository**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```

---

**WebService::Inventory**

```
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
```

---

**Behavior**

```
package cocome
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
```

---

**Source Model**

**Target Code**

---

**Repository**

```
package data;

public interface EnterpriseQueryIf {
  Operation queryEnterpriseById {
    return query TradingEnterprise
  }
}
```

---

**Data Types**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```

---

**System**

**Repository**

**Behavior**

```
package cocome
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
```

---

**Source Model**

**Target Code**

---

**Repository**

```
package data;

public interface EnterpriseQueryIf {
  Operation queryEnterpriseById {
    return query TradingEnterprise
  }
}
```

---

**JavaEE**

```
package org.cocome.tradingsystem.inventory.data.enterprise;
...
public final class EnterpriseQueryProvider implements EnterpriseQuery {
  @Override
  public TradingEnterprise queryEnterpriseById(final long enterpriseId, final IPersistenceContext pctx) {
    final EntityManager em = __getEntityManager(pctx);
    return em.createQuery("SELECT t FROM TradingEnterprise t WHERE t.id = :enterpriseId", TradingEnterprise.class).getResultList();
  }
}
```

---

**Kieker - Monitoring**

```
use pcm on cocome "irl-examples/src/cocome.repository" advice TraceLogger () {
  before OperationBeforeEvent(time, signature, classname, signature)
  after OperationAfterEvent(time, signature, classname)
}
```

---

**JPA - Entity Beans**

```
@Persistent
public class TradingEnterprise {
  private long id;
  private String name;
  private List<Store> stores;
  private List<ProductSupplier> productSuppliers;
  private @Transient boolean selected;
  public long getId() {
    return this.id;
  }
  public void setId(final long id) {
    this.id = id;
  }
}
```

---

**Data Types**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```

---

**System**

**Repository**

**Behavior**

```
package cocome
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
```

---

**Source Model**

**Target Code**

---

**Repository**

```
package data;

public interface EnterpriseQueryIf {
  Operation queryEnterpriseById {
    return query TradingEnterprise
  }
}
```

---

**JavaEE**

```
package org.cocome.tradingsystem.inventory.data.enterprise;
...
public final class EnterpriseQueryProvider implements EnterpriseQuery {
  @Override
  public TradingEnterprise queryEnterpriseById(final long enterpriseId, final IPersistenceContext pctx) {
    final EntityManager em = __getEntityManager(pctx);
    return em.createQuery("SELECT t FROM TradingEnterprise t WHERE t.id = :enterpriseId", TradingEnterprise.class).getResultList();
  }
}
```

---

**Kieker - Monitoring**

```
use pcm on cocome "irl-examples/src/cocome.repository" advice TraceLogger () {
  before OperationBeforeEvent(time, signature, classname, signature)
  after OperationAfterEvent(time, signature, classname)
}
```

---

**JPA - Entity Beans**

```
@Persistent
public class TradingEnterprise {
  private long id;
  private String name;
  private List<Store> stores;
  private List<ProductSupplier> productSuppliers;
  private @Transient boolean selected;
  public long getId() {
    return this.id;
  }
  public void setId(final long id) {
    this.id = id;
  }
}
```

---

**Data Types**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```

---

**System**

**Repository**

**Behavior**

```
package cocome
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
```

---

**Source Model**

**Target Code**

---

**Repository**

```
package data;

public interface EnterpriseQueryIf {
  Operation queryEnterpriseById {
    return query TradingEnterprise
  }
}
```

---

**JavaEE**

```
package org.cocome.tradingsystem.inventory.data.enterprise;
...
public final class EnterpriseQueryProvider implements EnterpriseQuery {
  @Override
  public TradingEnterprise queryEnterpriseById(final long enterpriseId, final IPersistenceContext pctx) {
    final EntityManager em = __getEntityManager(pctx);
    return em.createQuery("SELECT t FROM TradingEnterprise t WHERE t.id = :enterpriseId", TradingEnterprise.class).getResultList();
  }
}
```

---

**Kieker - Monitoring**

```
use pcm on cocome "irl-examples/src/cocome.repository" advice TraceLogger () {
  before OperationBeforeEvent(time, signature, classname, signature)
  after OperationAfterEvent(time, signature, classname)
}
```

---

**JPA - Entity Beans**

```
@Persistent
public class TradingEnterprise {
  private long id;
  private String name;
  private List<Store> stores;
  private List<ProductSupplier> productSuppliers;
  private @Transient boolean selected;
  public long getId() {
    return this.id;
  }
  public void setId(final long id) {
    this.id = id;
  }
}
```

---

**Data Types**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```

---

**System**

**Repository**

**Behavior**

```
package cocome
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
```

---

**Source Model**

**Target Code**

---

**Repository**

```
package data;

public interface EnterpriseQueryIf {
  Operation queryEnterpriseById {
    return query TradingEnterprise
  }
}
```

---

**JavaEE**

```
package org.cocome.tradingsystem.inventory.data.enterprise;
...
public final class EnterpriseQueryProvider implements EnterpriseQuery {
  @Override
  public TradingEnterprise queryEnterpriseById(final long enterpriseId, final IPersistenceContext pctx) {
    final EntityManager em = __getEntityManager(pctx);
    return em.createQuery("SELECT t FROM TradingEnterprise t WHERE t.id = :enterpriseId", TradingEnterprise.class).getResultList();
  }
}
```

---

**Kieker - Monitoring**

```
use pcm on cocome "irl-examples/src/cocome.repository" advice TraceLogger () {
  before OperationBeforeEvent(time, signature, classname, signature)
  after OperationAfterEvent(time, signature, classname)
}
```

---

**JPA - Entity Beans**

```
@Persistent
public class TradingEnterprise {
  private long id;
  private String name;
  private List<Store> stores;
  private List<ProductSupplier> productSuppliers;
  private @Transient boolean selected;
  public long getId() {
    return this.id;
  }
  public void setId(final long id) {
    this.id = id;
  }
}
```

---

**Data Types**

```
package data;

public class TradingEnterprise {
  long id
  string name
  ProductSupplier[] productSuppliers
}
```
Introduction
Modeling CoCoME

Source Model

Repository

Data Types

Behavior

System

Repository

Data Types

Behavior

Source Model

Repository

Data Types

Behavior
Introduction
Modeling CoCoME

Source Model

Repository

Data Types

Behavior

Kieker - Monitoring

AspectJ

JavaEE

JPA - Entity Beans

Reiner Jung
### Introduction

**Modeling CoCoME**

---

**Source Model**

![System Diagram](source_system.png)

**Repository**

![Repository Diagram](source_repository.png)

**Data Types**

```
package data

entity TradingEnterprise {
    long id
    string name
    Store[] stores
    ProductSupplier[] productSuppliers
    transient boolean selected
}

entity ProductSupplier {
    long id
    string name
    Product[] offers
}
```

**Behavior**

```
package cocome
import data.TradingEnterprise
repository "cocome/model/cocome.repository"
realize stateless TradingSystem.Inventory.Data.Enterprise {
    iface EnterpriseQueryIf
    operation queryEnterpriseById {
        return query TradingEnterprise
        "SELECT t FROM TradingEnterprise WHERE id=\"+ enterpriseId +\""
    }
}
```

---

**Generator**

![Generator Diagram](generator_diagram.png)

**AspectJ**

```xml
<aspectj>
    <weaver options=""/>
    <aspects>
        <aspect name="AbstractEntryLoggerAdvice0"/>
        <aspect name="AbstractEntryLoggerAdvice1"/>
        <concrete−aspect extends="AbstractEntryLoggerAdvice0" name="EntryLoggerAdvice0">
            <pointcut expression="TradingSystem.Inventory.Data.Persistence" name="point"/>
        </concrete−aspect>
    </aspects>
</aspectj>
```

---

**JavaEE**

```java
package org.cocome.tradingsystem.inventory.data.enterprise;
...
public final class EnterpriseQueryProvider implements EnterpriseQuery {
    @Override
    public final TracingEnterprise queryEnterpriseById(final long enterpriseId, final IPersistenceContext pctx) {
        final EntityManager em = __getEntityManager(pctx);
        return em.createQuery("SELECT te FROM TradingEnterprise te WHERE te.id=?" + enterpriseId, TradingEnterprise.class).getResultList();
    }
}
```

---

**Kieker - Monitoring**

```java
use pcm on cocome "irl-examples/src/cocome.repository"
advice TraceLogger () {
    before OperationBeforeEvent(time, signature, classname, signature)
    after OperationAfterEvent(time, signature, classname)
}
pointcut point: class cocome.TradingSystem.Inventory.Data.Persistence
aspect point: EntryLogger
```

---

**JPA - Entity Beans**

```java
@Entity
public class TradingEnterprise {
    private long id;
    private String name;
    private List<Store> stores;
    private List<ProductSupplier> productSuppliers;
    private @Transient boolean selected;
    // Constructor, getters, and setters
}
```

---

**Reiner Jung**
Introduction

Modeling CoCoME
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
GECO Approach
- Generator megamodel patterns
- Generator fragment design

GECO Artifacts
- Instrumentation aspect and record languages
- Generator composition language
- Software architecture evaluation

(Jung et al. 2016)

(Jung et al. 2013)

(Jung et al. 2015)
Foundations
**Metamodel Structures**

- Structure and typing

(Jung et al. 2014)
Metamodel Structures

- Structure and typing
- Expressions

(Jung et al. 2014)
Metamodel Structures

- Structure and typing
- Expressions
- Declaration

(Jung et al. 2014)
Metamodel Structures

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

(Jung et al. 2014)
Aspect-Oriented Modeling

(Jung et al. 2014)
Aspect-Oriented Modeling

(Jung et al. 2014)
Aspect-Oriented Modeling

System
  └── Access Control
      ├── Monitoring
      │    └── System
      │    └── base
      │        └── aspect
      │            └── base
      └── base

(Jung et al. 2014)
Foundations

Metamodel Roles

Aspect-Oriented Modeling

- Monitoring
  - aspect

Access Control
  - base
  - aspect

System
  - base

View-Based Modeling

- Data

(Jung et al. 2014)
Foundations

Metamodel Roles

Aspect-Oriented Modeling

1. Monitoring
   - base
   - aspect

2. Access Control
   - base
   - aspect

3. System
   - dependent
   - base

View-Based Modeling

4. Data
   - independent

(Jung et al. 2014)
Foundations

Metamodel Roles

Aspect-Oriented Modeling

- Monitoring
  - aspect
  - base

Access Control
  - aspect
  - base

System
  - dependent

View-Based Modeling

- Data
  - independent

Behavior

(Jung et al. 2014)
Foundations

Metamodel Roles

Aspect-Oriented Modeling

- Monitoring
- Access Control
- System

View-Based Modeling

- Data
- Behavior

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

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

System

conforms to

CoCoME Architecture

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

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

Generator Composition Pattern Candidates

Model  Transformation  Reference

Reiner Jung
The GECO Approach
Generator Composition Pattern Candidates

Reiner Jung
The GECO Approach

Generator Composition Pattern Candidates

Reiner Jung
The GECO Approach

Generator Composition Pattern Candidates

1. Model
2. Transformation
3. Reference

Reiner Jung
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
SAM  Source Aspect Model
TAM  Target Aspect Model

Transformations:
T  Transformation
Reference
The GECO Approach

Generator Composition Megamodell Patterns

Simple
P1

Simple
P2

Normal Aspect

Inverted References

Model Weaving

P3

P4

TBM Target Base Model
SBM Source Base Model
TAM Target Aspect Model
SAM Source Aspect Model
BM Source Base Model
BM’ Source Aspect Model

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

Transformation

Reference

Reiner Jung 13 / 33
The GECO Approach

Generator Composition Megamodel Patterns

**Simple P1**
- **SBM** (Source Base Model) → **TBM** (Target Base Model) via transformation $T_{BM}$

**Normal Aspect P2**
- **SBM** → **TBM** via transformation $T_{BM}$
- **SAM** (Source Aspect Model) → **TAM** (Target Aspect Model) via transformation $T_{AM}$

**Inverted References P3**
- **SBM** → **TBM** via transformation $T_{BM}$
- **SAM** → **TAM** via transformation $T_{AM}$

**Model Weaving P4**
- **BM** (Base Model) → **SAM** (Source Aspect Model) via transformation $T_{Merge}$

---

<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>

$T$ Transformation

$\leftarrow$ Reference
The GECO Approach

Generator Composition Megamodel Patterns

Simple
P1

Simple
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>
</tbody>
</table>

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

T Transformation

Reiner Jung 13 / 33
The GECO Approach

Generator Composition Megamodel Patterns

Simple P1

<table>
<thead>
<tr>
<th>SBM</th>
<th>TBM</th>
</tr>
</thead>
</table>

Normal Aspect P2

<table>
<thead>
<tr>
<th>SBM</th>
<th>SAM</th>
<th>TBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBM</td>
<td>TAM</td>
<td></td>
</tr>
</tbody>
</table>

Inverted References P3

<table>
<thead>
<tr>
<th>SBM</th>
<th>SAM</th>
<th>TBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBM</td>
<td>TAM</td>
<td></td>
</tr>
</tbody>
</table>

Model Weaving P4

<table>
<thead>
<tr>
<th>BM</th>
<th>SAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM'</td>
<td></td>
</tr>
</tbody>
</table>

SBM  Source Base Model  SAM  Source Aspect Model  T  Transformation
TBM  Target Base Model  TAM  Target Aspect Model  Reference

Reiner Jung 13 / 33
The GECO Approach

Pattern P2 - Normal Aspect

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

SBM Source Base Model

Transformation
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

T Transformation
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
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

Transformation
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

T Transformation
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

Transformation
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

Transformation References:
- SBM → SAM
- TBM → TRM
- TAM → JPM

Reiner Jung
The GECO Approach

CoCoME Case Study - Generator Megamodel

Palladio Component Model

- Repository
- System
The GECO Approach

CoCoME Case Study - Generator Megamodel
The GECO Approach

CoCoME Case Study - Generator Megamodel
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

DTL

EJB/Servlets Stubs

Java Snippets

Entities

T_{ProtoCom}

T_{Behavior}

T_{DTL}
The GECO Approach

CoCoME Case Study - Generator Megamodel

Palladio Component Model
- Repository
- System

EJB/Servlets Stubs

Behavior

DTL

Java Snippets

Entities

P2 - Normal Aspect

T_{ProtoCom}

T_{Behavior}

T_{DTL}
The GECO Approach

CoCoME Case Study - Generator Megamodel

Diagram showing the relationships between Palladio Component Model, System, Behavior, DTL, EJB/Servlets Stubs, Java Snippets, Entities, and Classes.
The GECO Approach

CoCoME Case Study - Generator Megamodel

- Repository
- System
- Behavior
- DTL
- EJB/Servlets Stubs
- Java Snippets
- Entities
- Classes

IRL → IAL: record types operations → Palladio Component Model

Repository → System → Behavior → DTL

T_{ProtoCom} → EJB/Servlets Stubs

T_{javac} → EJB/Servlets

T_{javac} → Classes

T_{javac} → Java Snippets

T_{javac} → Entities

Reiner Jung
The GECO Approach

CoCoME Case Study - Generator Megamodel

[Diagram showing the relationships between IRL, IAL, Palladio Component Model, EJB/Servlets Stubs, Java Snippets, DTL, and Classes, with various transformations and operations indicated.]
The GECO Approach

CoCoME Case Study - Generator Megamodel

P2 - Normal Aspect

Repository
System
Behavior
DTL
EJB/Servlets Stubs
Java Snippets
Entities
Classes

Kieker
Records
Sensors
aspect.xml

IRL
IAL

T_{IRL}
T_{sensor}
T_{aspect}

Reiner Jung
The GECO Approach

CoCoME Case Study - Generator Megamodel

P2 - Normal Aspect

Repository
System
DTL
Behavior
EJB/Servlets
TProtoCom
Tjavac
Tjavac,ajc
Classes
entity classes

Sensors
aspect.xml

EJB/Servlets

Behavior
Entities

DTL

Kieker
Records

IAL

IRL

TIRL

Classes

Tjavac

Tjavac,ajc

Tsensor

Taspect

TJW

Java
Snippets

Records

IRL

IAL

Palladio Component Model

Repository
System

TProtoCom

Behavior

DTL

Entities

Classes

entity classes

Tjavac

Tjavac,ajc
The GECO Approach
CoCoME Case Study - Generator Megamodel

Reiner Jung 16 / 33
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**

Source Model

Fragment

Target Model

**Semantic Dimension**

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

Internal Structure of Generator Fragments

Technical Dimension

- Source Model
- Fragment
  - Model Query
  - Aggregation
  - State
  - Output Creation
- Control

Semantic Dimension

- Source Metamodel
- Target Metamodel
- Target Model

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

Internal Structure of Generator Fragments

Technical Dimension

Source Model

Fragment

Model Query
Aggregation
State
Output Creation

Source Metammodel

Semantic Dimension

Target Model

Control

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

Internal Structure of Generator Fragments

Technical Dimension

Source Model

Source Metamodel

Target Model

Semantic Dimension

Source Metamodel

Source Model

Target Metamodel

Target Model

Fragment

Model Query

Aggregation

State

Output Creation

Control

Expressions

Structure

Evaluation

Typing

Initialization

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

Internal Structure of Generator Fragments

(Mens et al. 2006; Biehl 2010)
Evaluation
Qualitative Evaluation with two case studies (based on GQM)
Evaluate the effect of GECO on

Goal G1  the utility and program quality
Goal G2  the evolvability
Goal G3  the reusability
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
Evaluation

Questions

Utility and program quality

(IS091; ISO11)

- modularity of different generator implementations

Evolvability

(Rowe et al. 1998; Koziolek 2011)

- Change in modularity during the evolution

Reusability

(IS091; ISO11)

- modularity of the generator implementations
Modularity (Allen 2002; Allen et al. 2007)

- Low complexity of the system
- Low coupling of modules of a system
- High inner module cohesion of a system
Common Component Modeling Example (Heinrich et al. 2015)

- **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
Behavior Generator

1. Basic functionality
2. Stateless/-full components
CoCoME Case Study - Behavior Evolution

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

Reiner Jung
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

![Graphs showing LOC, Complexity, and Coupling for MENGES and GECO across revisions.](image)

- **Lines of Code (LOC)**
  - MENGES: 10,816
  - GECO: 7,025
  - Ratio: 1.5396

- **Complexity**
  - MENGES: 13,921.88
  - GECO: 6,675.88
  - Ratio: 2.0854

- **Coupling**
  - MENGES: 10,983.81
  - GECO: 5,060.83
  - Ratio: 2.1704

Reiner Jung
Evaluation

MENGES Case Study - Generator Comparison

Structure and Typing

Reiner Jung 25 / 33
Expressions and Statements

**Evaluation**

**MENGES Case Study - Generator Comparison**

**Expressions and Statements**

- **LOC [count]**
  - **MENGES**
  - **GECO**

- **Complexity [bits]**
  - **MENGES**
  - **GECO**

- **Coupling [bits]**
  - **MENGES**
  - **GECO**

Reiner Jung 25 / 33
Evaluation

MENGES Case Study - Generator Comparison

Refactoring and Communications

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

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

<table>
<thead>
<tr>
<th>MENGES</th>
<th>GECO</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
<td>10816</td>
<td>7025</td>
</tr>
<tr>
<td>Complexity</td>
<td>13921.88</td>
<td>6675.88</td>
</tr>
<tr>
<td>Coupling</td>
<td>10983.81</td>
<td>5060.83</td>
</tr>
</tbody>
</table>
Evaluation
MENGES Case Study - Generator Comparison

Improvement of Polymorphism

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

- MENGES
- GECO

Lines of code:
- MENGES: 10816
- GECO: 7025
- Ratio: 1.5396

Complexity:
- MENGES: 13921.88
- GECO: 6675.88
- Ratio: 2.0854

Coupling:
- MENGES: 10983.81
- GECO: 5060.83
- Ratio: 2.1704
Timers and Template Improvements

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

MENGES vs. GECO

Reiner Jung 25 / 33
Evaluation

MENGES Case Study - Generator Comparison

Maintenance

- 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
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>
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)
- FDAG (Clarke et al. 2005)
- RAM-based (Kienzle et al. 2009; Kramer et al. 2011)
Related Work

Transformation Modularization

Reuse and product lines
- Template-based transformations
- Genesys approach

(Kapova et al. 2010)
(Jörges 2013)

Modularization
- Genericity for model management operations
- Factorization and composition of transformation
- Chaining of transformations
- Localized transformations

(Wimmer et al. 2011)
(Sánchez Cuadrado et al. 2008)
(Vanhooff et al. 2006)
(Etien et al. 2015)
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

- 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
Contributions

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

- 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
Conclusion
Summary

**GECO Approach** http://www.oiloftrop.de/geco-approach/  
- 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*

**Resources** https://github.com/  
- Framework and tool  
  rju/geco-composition-language.git  
- Architecture analysis  
  rju/architecture-evaluation-tool.git  
- CoCoME DSLs  
  research-iobserve  
- Monitoring DSLs  
  kieker-monitoring/instrumentation-languages
References


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


Foundations
Identifying Metamodel Partitions

1. Find all root classes $R = t_n P T R t e T P e T P s T R t T u$

2. Form parts for all $r_i P R$: $P_i = contains TG (r_i) Y r_i u$

3. Detect overlapping parts $O_k with n = |R| O_k = t P i X P j |_{i,j=1..n} ^ {i \neq j} P i X P j \neq H u$

4. Remove the overlapping parts $O_k, with m = |O_k| @ {i=1..n} P 1 i = P i X (m \neq j=0 O_k)$

5. Remove identified partitions $P_1 i$ from graph

6. Reiterate process with remaining graph
Identifying Metamodel Partitions

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

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

Metamodel Partitioning

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 $r_i \in R$: $P_i = contains_{TG}(r_i) \cup \{r_i\}$
Foundations

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

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

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

$$O = \{P_i \cap P_j \mid \forall i, j \in [1 \ldots n] \land i \neq j \land P_i \cap P_j \neq \emptyset\}$$
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 $r_i \in R$: $P_i = \text{contains}_{TG}(r_i) \cup \{r_i\}$

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 $m = |O|$ 

$$\forall i \in [1 \ldots n] \quad P_i' = P_i \cap \left( \bigcup_{j=0}^{m} O_k \right)$$
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 \( m = |O| \)

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

5. Remove identified partitions \( P_i' \) from graph
Identifying Metamodel Partitions

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

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

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

6. Reiterate process with remaining graph
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

Target Models
- Cart
  - init (return type: void, parameter: . . .)
  - service (return type: void, parameter: . . .)

AspectJ Pointcuts

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

AspectJ Pointcuts

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
Approach

Join-Point Computation

Source Models

Monitoring Aspect

Target Models

Shopping Cart

AspectJ Pointcuts Java Servlet
class Cart
method init
return type void
parameter
.  .  .  .  .
method service
return type void
parameter
.  .  .  .  .
Trace Model

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

Reiner Jung
Evaluation
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
Evaluation

Metrics

Effort developer days per feature

Modularity (Allen 2002; Allen et al. 2007)
- Low complexity of the system
- Low coupling of modules of a system
- High inner module cohesion of a system

Understandability inverse of complexity (Laitinen 1996)

Changeability (ISO11)
- Low coupling of modules of a system
- High inner module cohesion of a system

Stability of the code base (ISO11)
- Low coupling of modules of a system

Evaluation Measure properties for each revision
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>

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

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

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

MENGES LOC and Modules

Reiner Jung
Evaluation

MENGES LOC and Modules

Reiner Jung
MENGES Coupling Delta

Evaluation

Reiner Jung

Generators
-● Old
-△ New

Revisions

Coupling [bits Δ]

0 2000 4000

2 3 4 5 6 7 8 9 10 11 12 13 14

Reiner Jung
Evaluation

MENGES Coupling Delta

Reiner Jung
Evaluation

MENGES Complexity Delta

Reiner Jung 55 / 33
Tooling
Tooling

Architecture Analysis Tool

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
}