A Generator Composition Approach for Aspect-Oriented DSLs

ICMT 2016

Reiner Jung Robert Heinrich
Wilhelm Hasselbring

5st July 2016

Motivation

Software development and evolution

- Domain, technology and environment changes
- Addition and changes to requirements

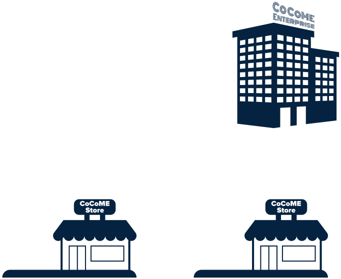
➡ Continuously growing complexity

Model-driven software development

- Provides: Specific views and models of software systems
- Requires: Model editors, evaluation tools, and code generators

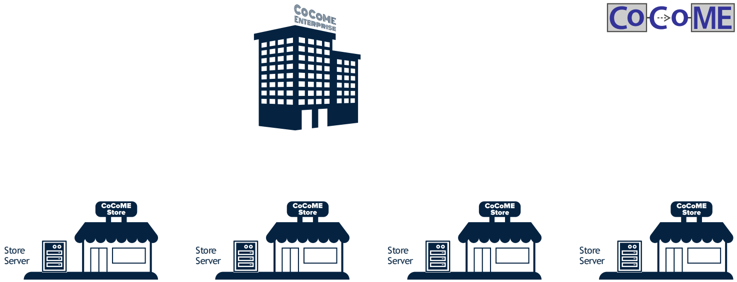
➡ Evolution provoke generator alterations

Common Component Modeling Example



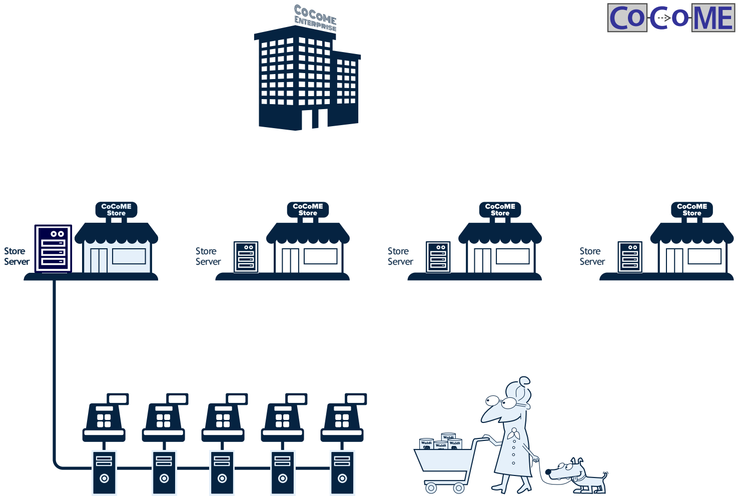
(Rausch et al. 2011; Heinrich et al. 2015)

Common Component Modeling Example



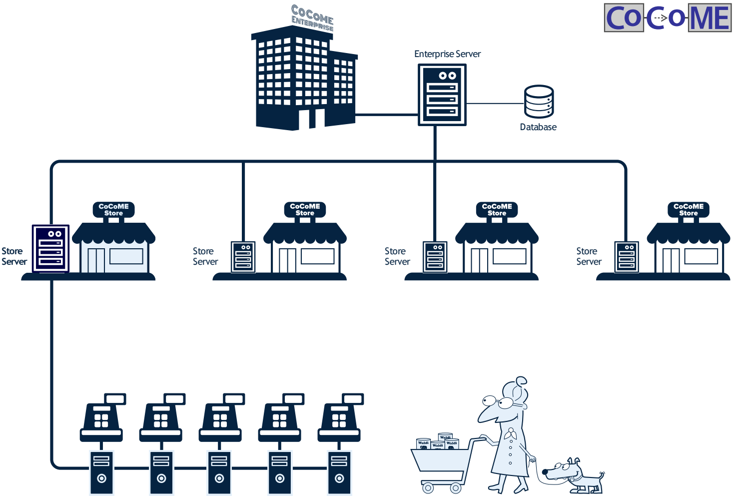
(Rausch et al. 2011; Heinrich et al. 2015)

Common Component Modeling Example



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

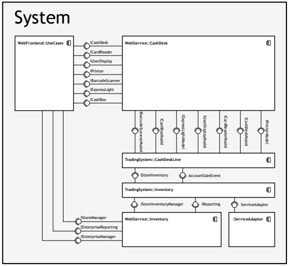
Common Component Modeling Example



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

Modeling CoCoME

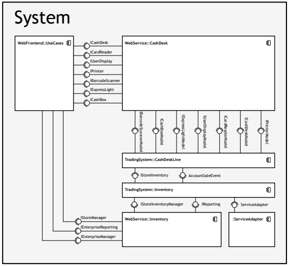
Source Model

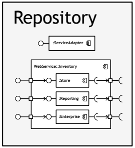


Target Code

Modeling CoCoME

Source Model

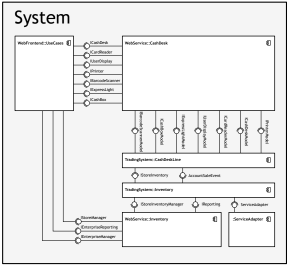


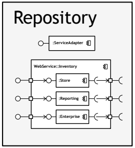


Target Code

Modeling CoCoME

Source Model





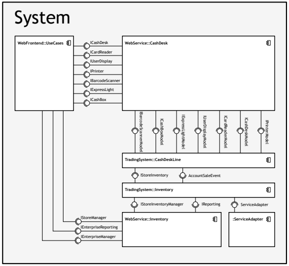
Data Types

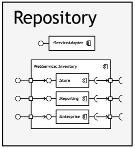
```
package data

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

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

Target Code

Modeling CoCoME

Source Model





Data Types

```
package data

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

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

Behavior

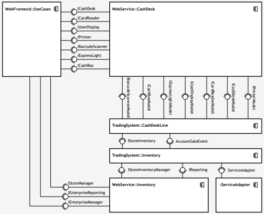
```
package cocome

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
        }
}
```

Target Code

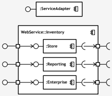
Modeling CoCoME

Source Model

System



Repository



Data Types

```
package data

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

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
        }
}
```

Kieker - Monitoring

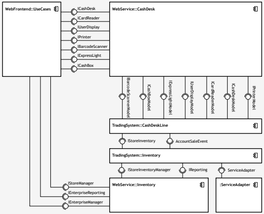
```
use pcm on cocome "irl-examples/src/cocome.repository"

advice TraceLogger () {
    before OperationBeforeEvent(time, signature, classname, signature)
    after OperationAfterEvent(time, signature, classname)
}

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

aspect point : EntryLogger
```

Target Code

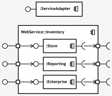
Modeling CoCoME

Source Model

System



Repository



Data Types

```
package data

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

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
        }
}
```

Kieker - Monitoring

```
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
```

Target Code

AspectJ

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<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>
```

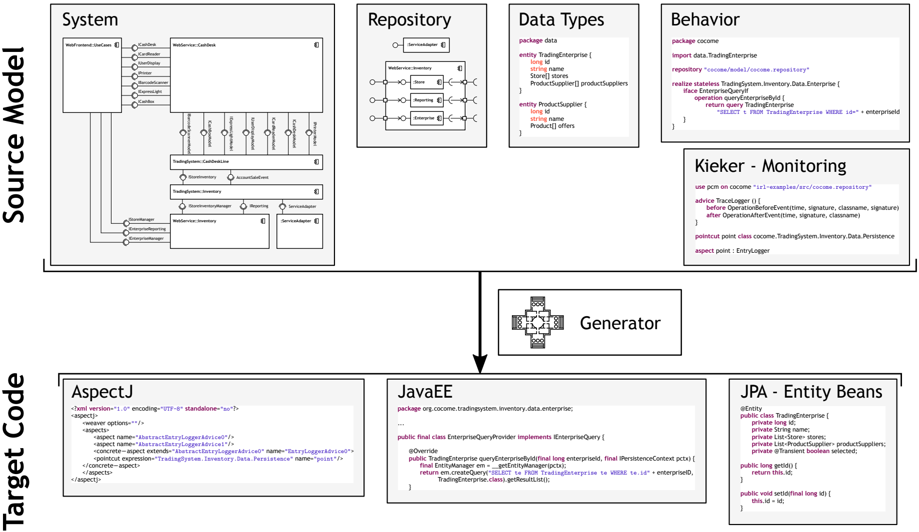
JavaEE

```
package org.cocome.tradingsystem.inventory.data.enterprise;

...

public final class EnterpriseQueryProvider implements IEnterpriseQuery {

    @Override
    public TradingEnterprise 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();
    }
```

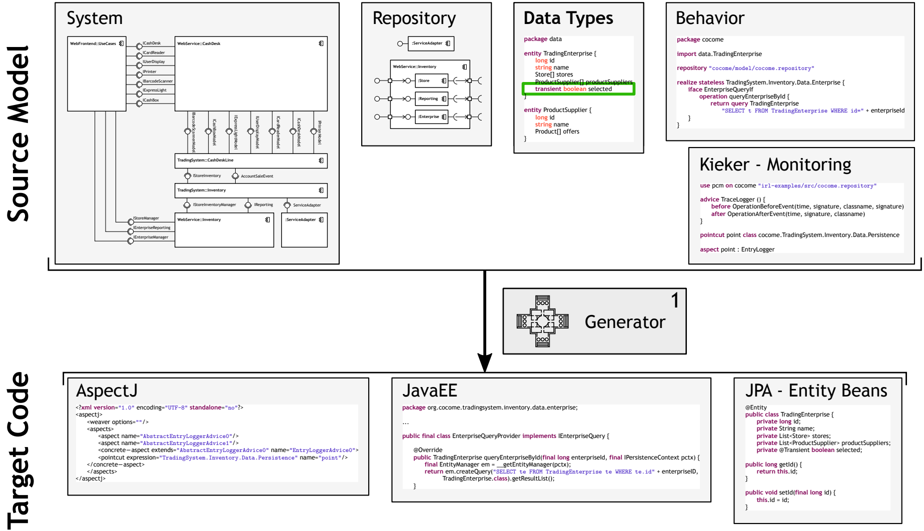
JPA - Entity Beans

```
@Entity
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;
}
```

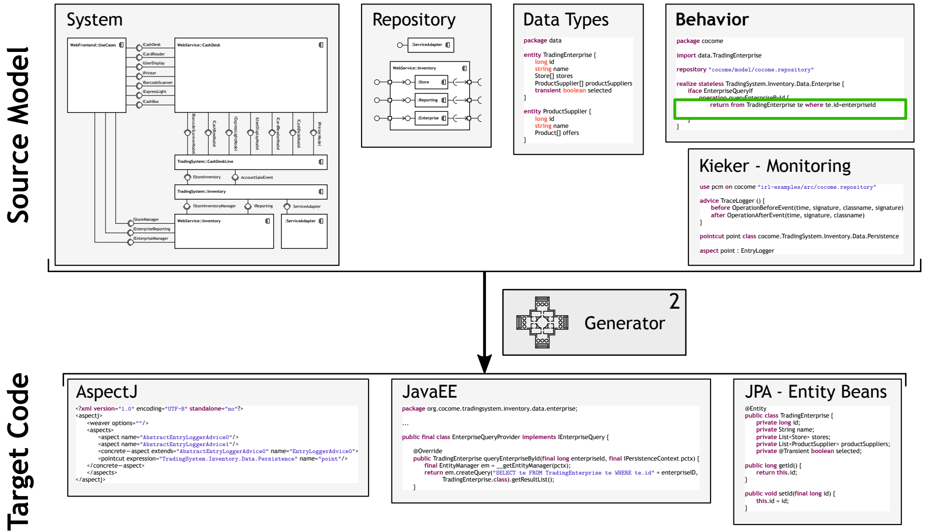
Modeling CoCoME



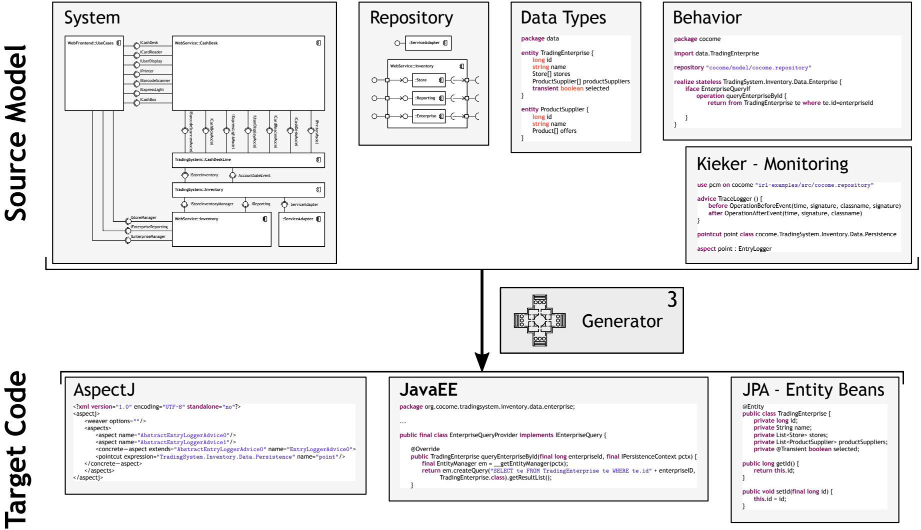
Modeling CoCoME



Modeling CoCoME



Modeling CoCoME



Challenges

Key challenges in generator development

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

Challenges

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

Contributions

GECO Approach

- Generator megamodel patterns (Jung et al. 2016)
- Generator fragment design

GECO Artifacts

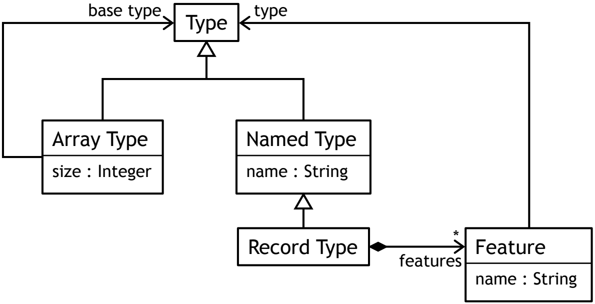
- Instrumentation aspect and record languages (Jung et al. 2013)
- Generator composition language
- Software architecture evaluation (Jung et al. 2015)

Foundations

Metamodel Partitions

Metamodel Structures

- Structure and typing

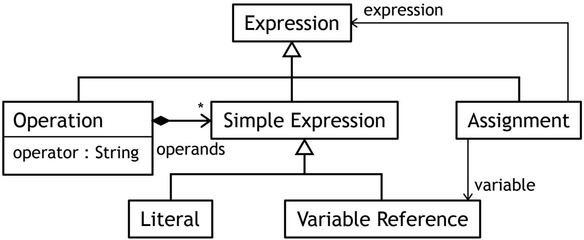


(Jung et al. 2014)

Metamodel Partitions

Metamodel Structures

- Structure and typing
- Expressions

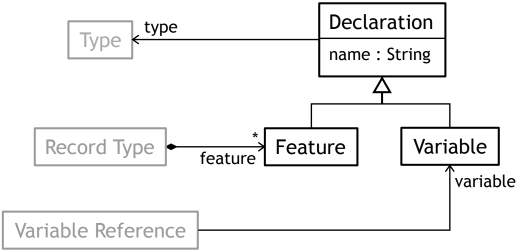


(Jung et al. 2014)

Metamodel Partitions

Metamodel Structures

- Structure and typing
- Expressions
- Declaration

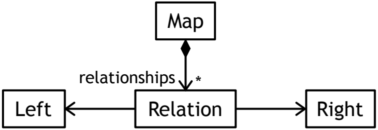


(Jung et al. 2014)

Metamodel Partitions

Metamodel Structures

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



(Jung et al. 2014)

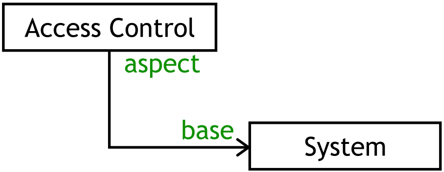
Metamodel Roles

Aspect-Oriented Modeling

(Jung et al. 2014)

Metamodel Roles

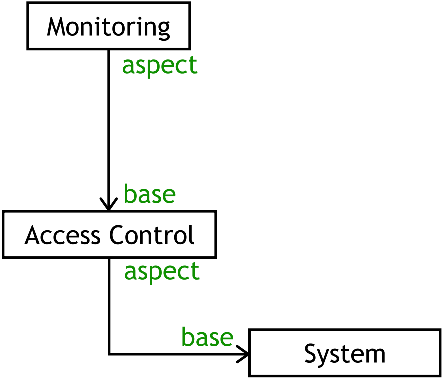
Aspect-Oriented Modeling



(Jung et al. 2014)

Metamodel Roles

Aspect-Oriented Modeling

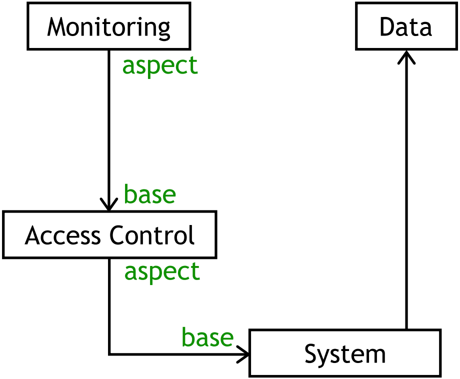


(Jung et al. 2014)

Metamodel Roles

Aspect-Oriented Modeling

View-Based Modeling

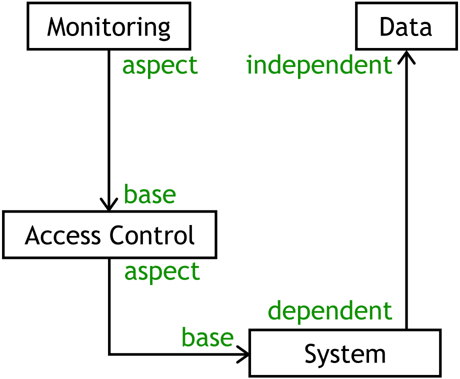


(Jung et al. 2014)

Metamodel Roles

Aspect-Oriented Modeling

View-Based Modeling

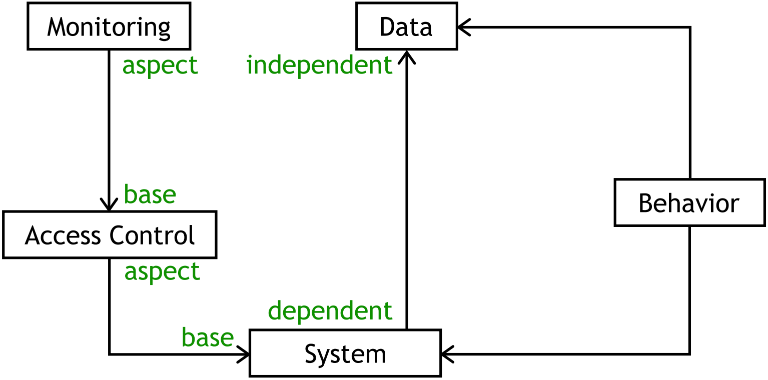


(Jung et al. 2014)

Metamodel Roles

Aspect-Oriented Modeling

View-Based Modeling

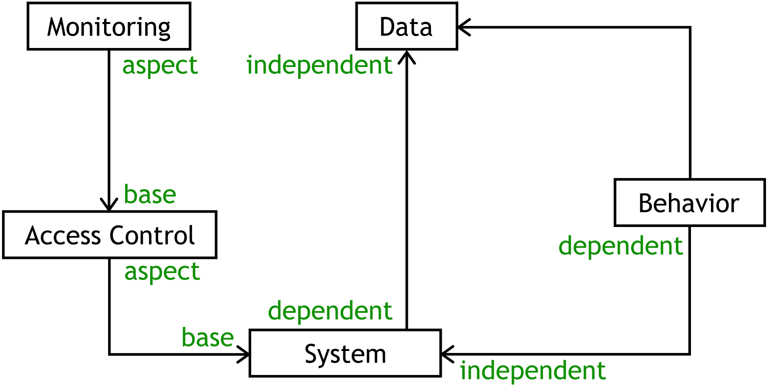


(Jung et al. 2014)

Metamodel Roles

Aspect-Oriented Modeling

View-Based Modeling



(Jung et al. 2014)

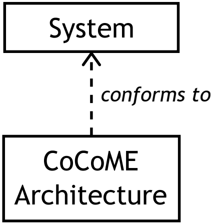
Megamodel Notation

Example Megamodel based on CoCoME Szenario

(Bézivin et al. 2004; Favre 2004)

Megamodel Notation

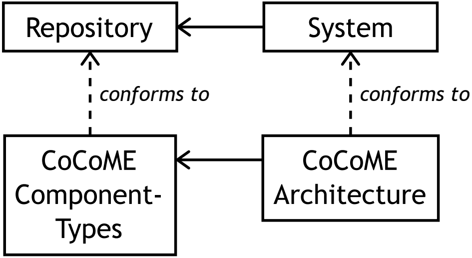
Example Megamodel based on CoCoME Szenario



(Bézivin et al. 2004; Favre 2004)

Megamodel Notation

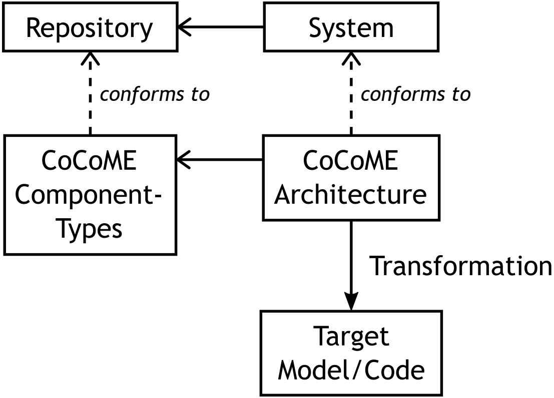
Example Megamodel based on CoCoME Szenario



(Bézivin et al. 2004; Favre 2004)

Megamodel Notation

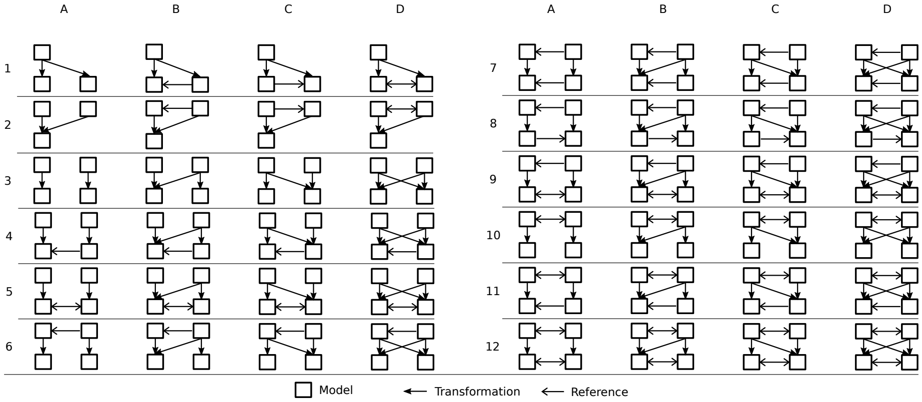
Example Megamodel based on CoCoME Szenario



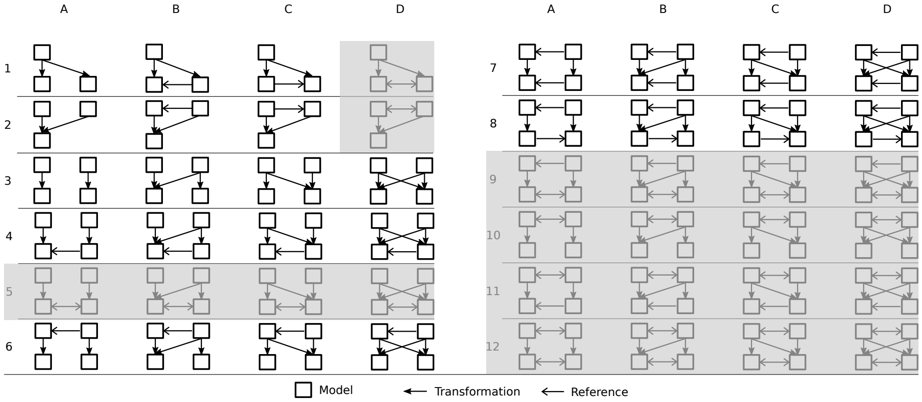
(Bézivin et al. 2004; Favre 2004)

The GECO Approach

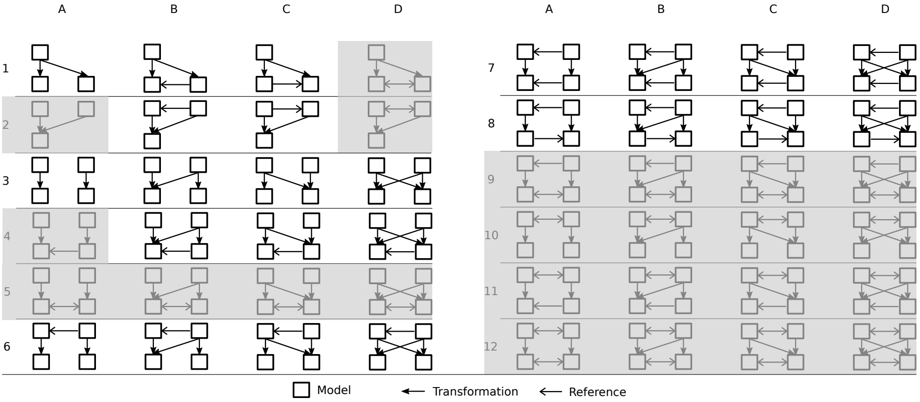
Generator Composition Pattern Candiates



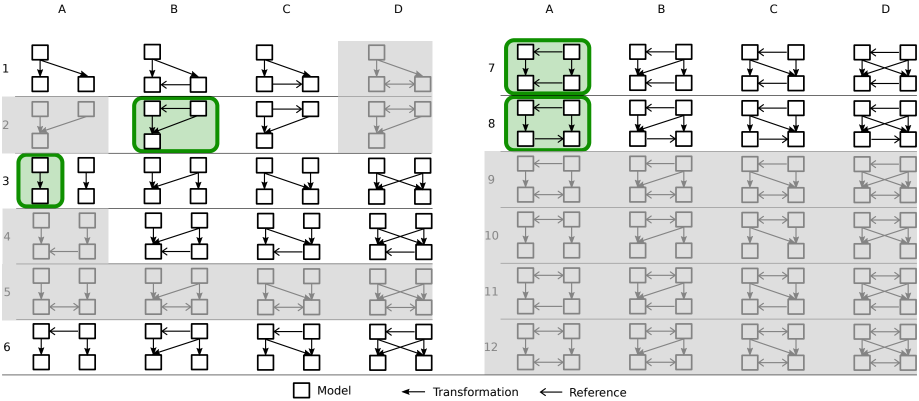
Generator Composition Pattern Candiates



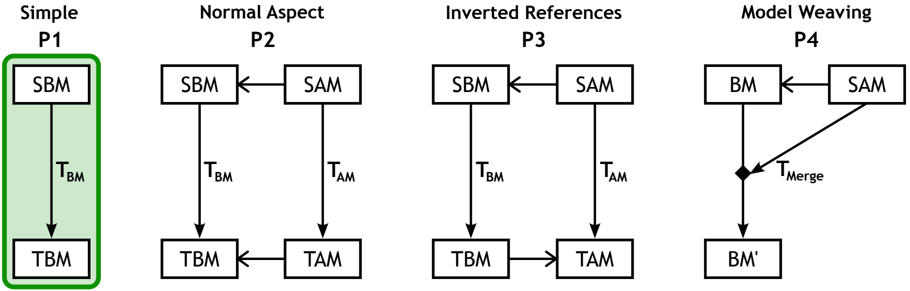
Generator Composition Pattern Candiates



Generator Composition Pattern Candiates

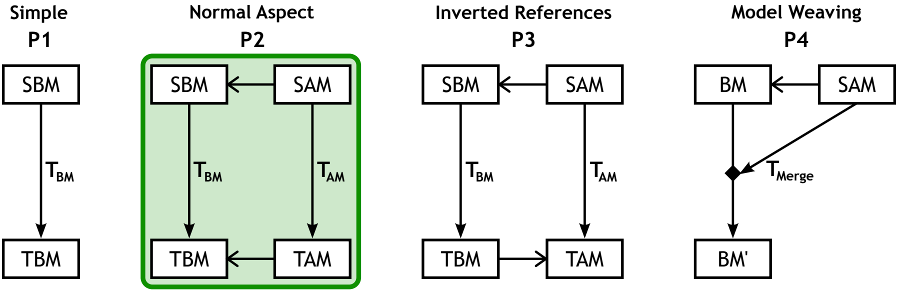


Generator Composition Megamodel Patterns



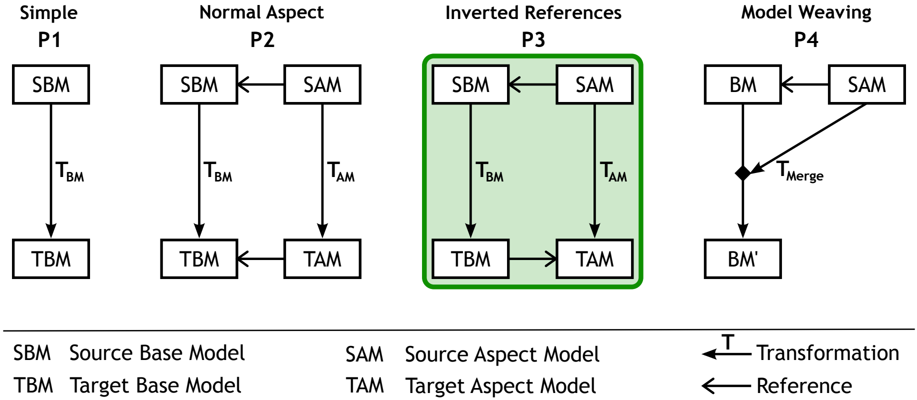
SBM	Source Base Model	SAM	Source Aspect Model	←T—	Transformation
TBM	Target Base Model	TAM	Target Aspect Model	←	Reference

Generator Composition Megamodel Patterns

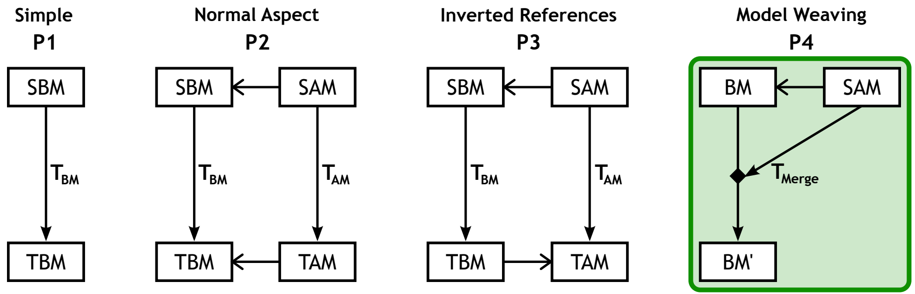


SBM	Source Base Model	SAM	Source Aspect Model	←T—	Transformation
TBM	Target Base Model	TAM	Target Aspect Model	←—	Reference

Generator Composition Megamodel Patterns

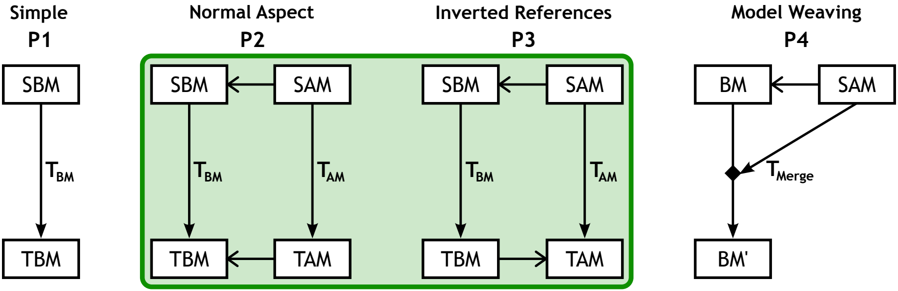


Generator Composition Megamodel Patterns



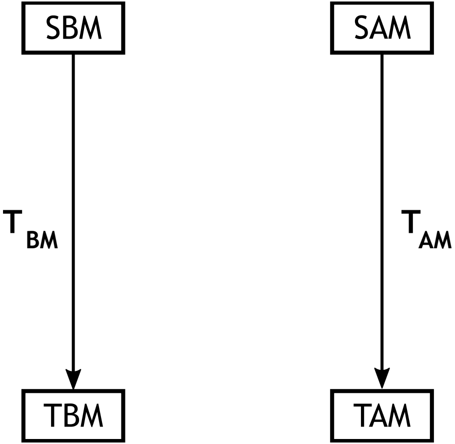
SBM	Source Base Model	SAM	Source Aspect Model	←T	Transformation
TBM	Target Base Model	TAM	Target Aspect Model	←	Reference

Generator Composition Megamodel Patterns

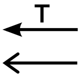


SBM	Source Base Model	SAM	Source Aspect Model	← T	Transformation
TBM	Target Base Model	TAM	Target Aspect Model	←	Reference

Pattern P2 - Normal Aspect

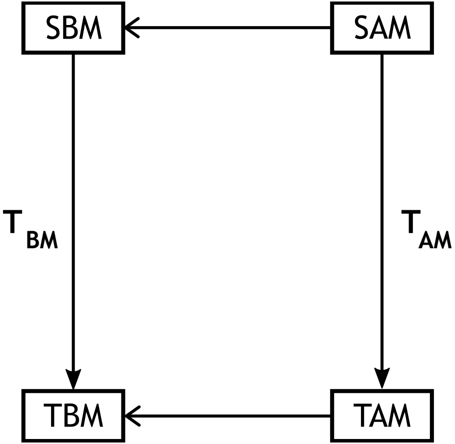


SBM	Source Base Model
SAM	Source Aspect Model
TBM	Target Base Model
TAM	Target Aspect Model
TRM	Trace Model

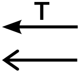


Transformation
Reference

Pattern P2 - Normal Aspect

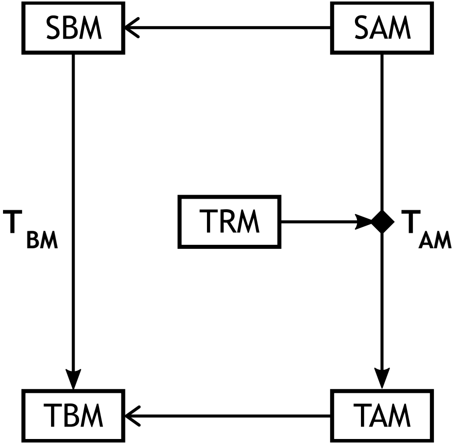


SBM	Source Base Model
SAM	Source Aspect Model
TBM	Target Base Model
TAM	Target Aspect Model
TRM	Trace Model

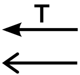


Transformation
Reference

Pattern P2 - Normal Aspect

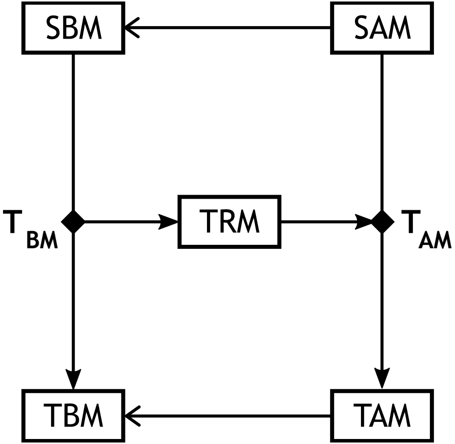


SBM	Source Base Model
SAM	Source Aspect Model
TBM	Target Base Model
TAM	Target Aspect Model
TRM	Trace Model

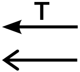


Transformation
Reference

Pattern P2 - Normal Aspect

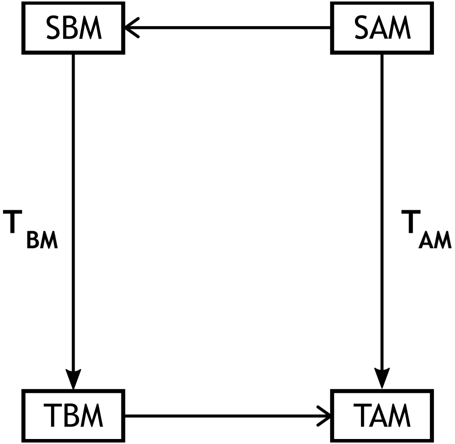


SBM	Source Base Model
SAM	Source Aspect Model
TBM	Target Base Model
TAM	Target Aspect Model
TRM	Trace Model



Transformation
Reference

Pattern P3 - Inverted References

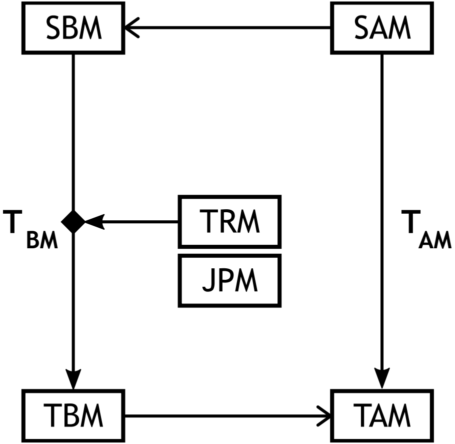


SBM	Source Base Model
SAM	Source Aspect Model
TBM	Target Base Model
TAM	Target Aspect Model
TRM	Trace Model
JPM	Join Point Model

T ← Transformation

← Reference

Pattern P3 - Inverted References

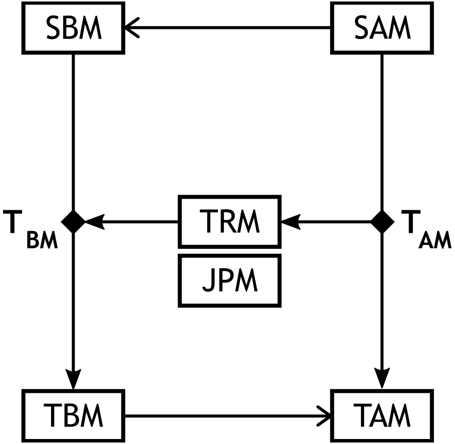


SBM Source Base Model
SAM Source Aspect Model
TBM Target Base Model
TAM Target Aspect Model
TRM Trace Model
JPM Join Point Model

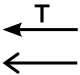
Transformation


Reference

Pattern P3 - Inverted References

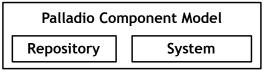


SBM	Source Base Model
SAM	Source Aspect Model
TBM	Target Base Model
TAM	Target Aspect Model
TRM	Trace Model
JPM	Join Point Model

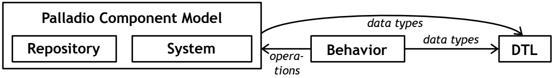


Transformation
Reference

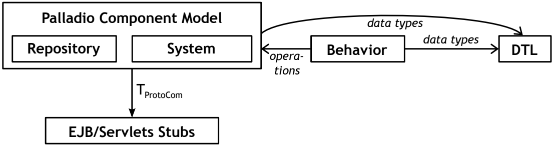
CoCoME Case Study - Generator Megamodel



CoCoME Case Study - Generator Megamodel

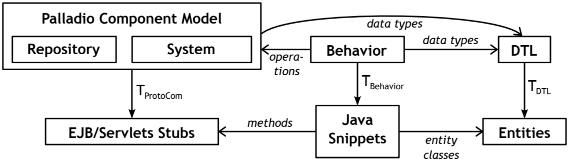


CoCoME Case Study - Generator Megamodel



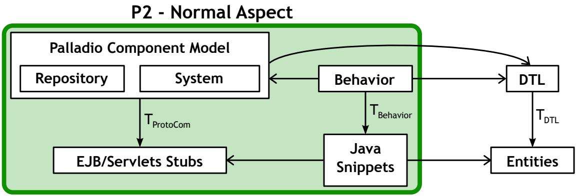
iObserve CoCoME

CoCoME Case Study - Generator Megamodel

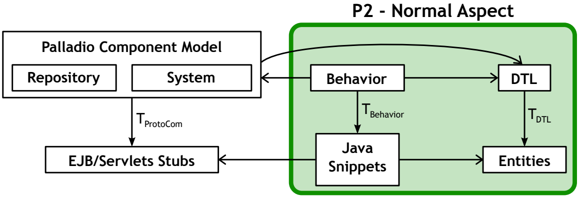


iObserve CoCoME

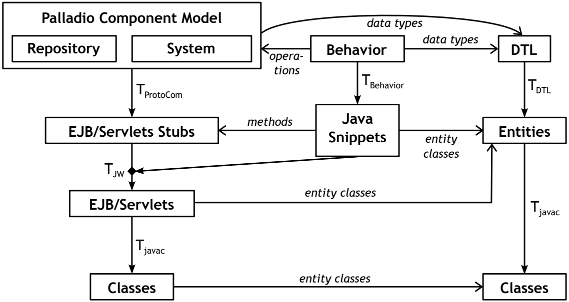
CoCoME Case Study - Generator Megamodel



CoCoME Case Study - Generator Megamodel

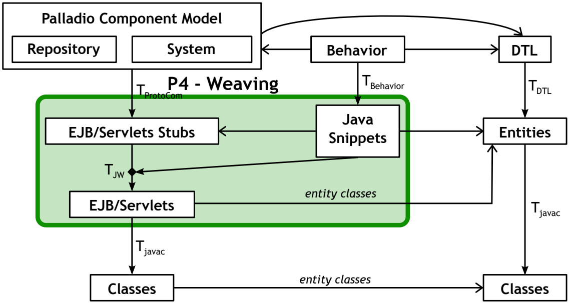


CoCoME Case Study - Generator Megamodel

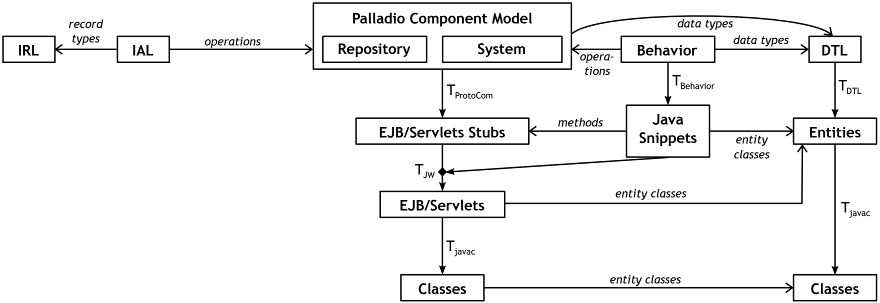


iObserve CoCoME

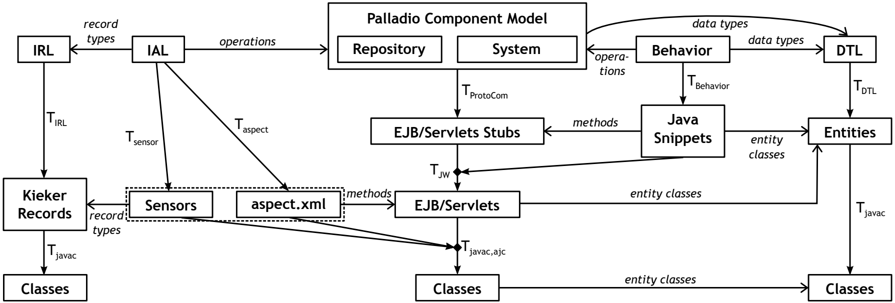
CoCoME Case Study - Generator Megamodel



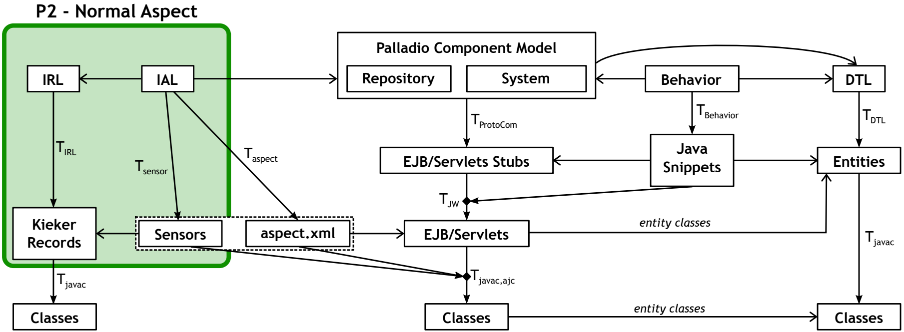
CoCoME Case Study - Generator Megamodel



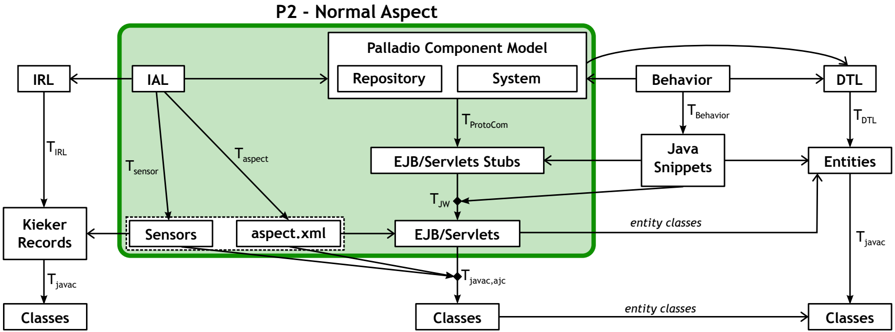
CoCoME Case Study - Generator Megamodel



CoCoME Case Study - Generator Megamodel

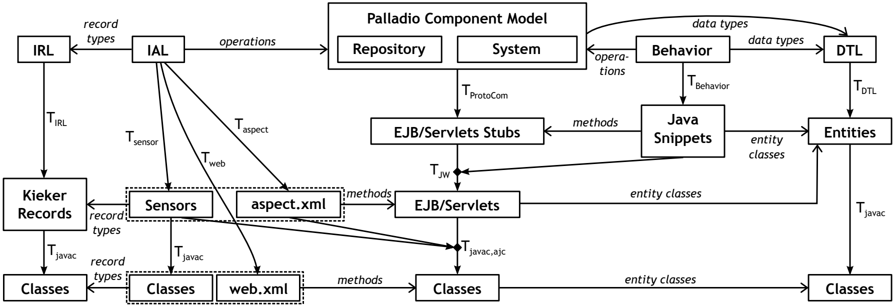


CoCoME Case Study - Generator Megamodel



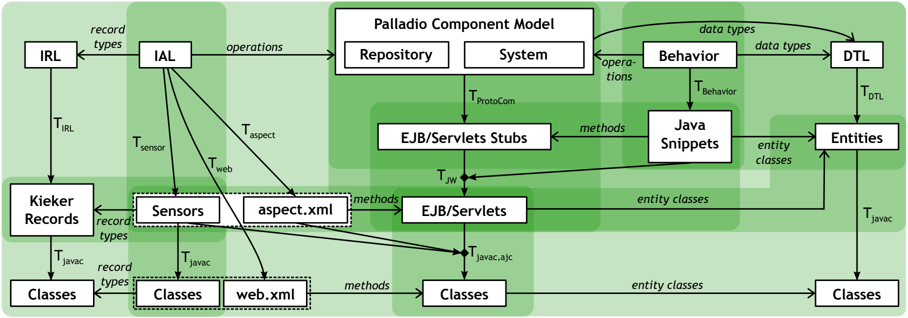
iObserve CoCoME

CoCoME Case Study - Generator Megamodel



iObserve CoCoME

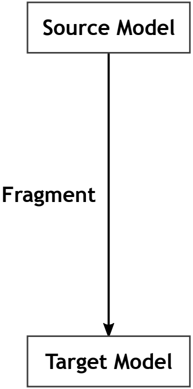
CoCoME Case Study - Generator Megamodel



7 **P2 - Normal Aspect**
2 **P4 - Weaving**

Internal Structure of Generator Fragments

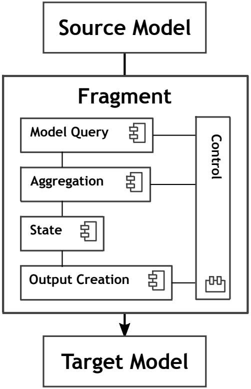
Technical Dimension



Semantic Dimension

Internal Structure of Generator Fragments

Technical Dimension



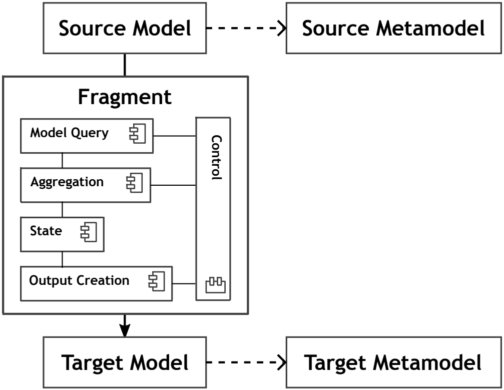
(Mens et al. 2006; Biehl 2010)

Semantic Dimension

Internal Structure of Generator Fragments

Technical Dimension

Semantic Dimension

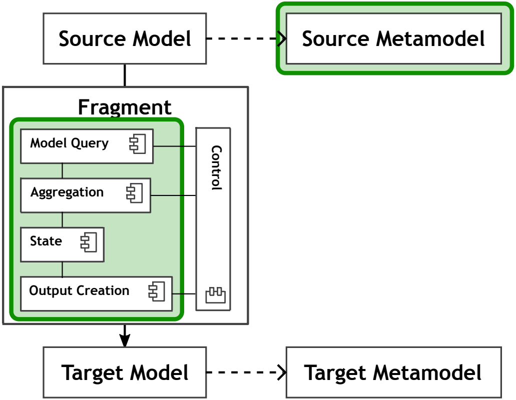


(Mens et al. 2006; Biehl 2010)

Internal Structure of Generator Fragments

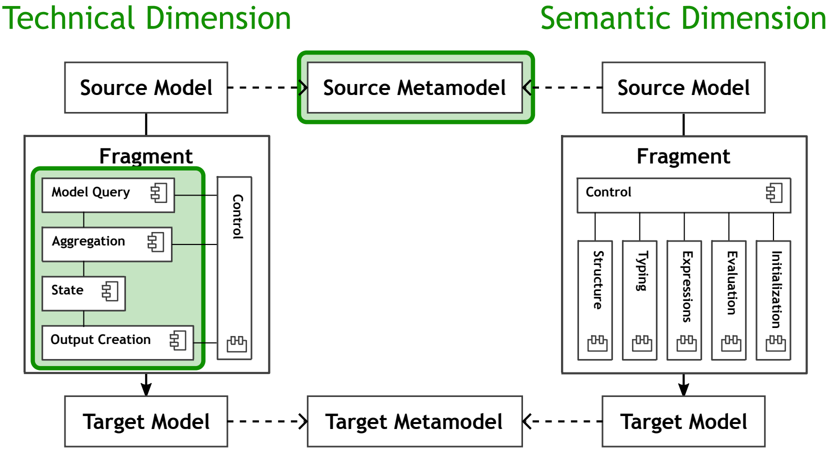
Technical Dimension

Semantic Dimension



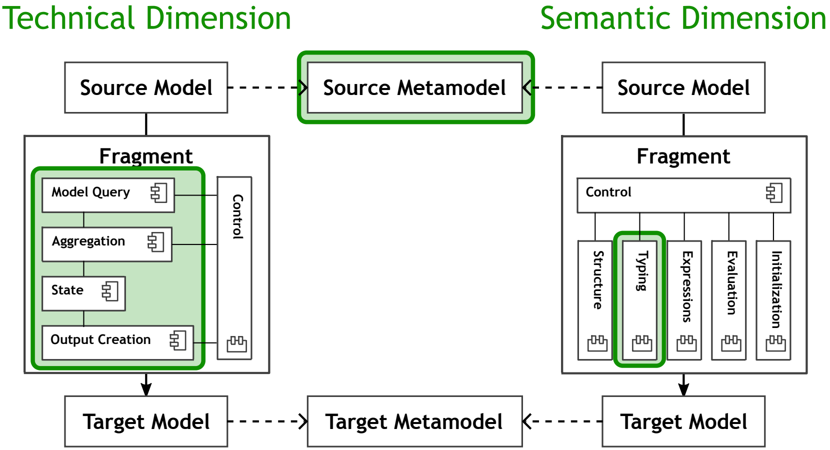
(Mens et al. 2006; Biehl 2010)

Internal Structure of Generator Fragments



(Mens et al. 2006; Biehl 2010)

Internal Structure of Generator Fragments



(Mens et al. 2006; Biehl 2010)

Evaluation

Evaluation Design

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

Questions

Utility and program quality (ISO91; ISO11)

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

Evolvability (Rowe et al. 1998; Koziolek 2011)

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

Reusability (ISO91; ISO11)

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

Questions

Utility and program quality (ISO91; ISO11)

- **modularity** of different generator implementations

Evolvability (Rowe et al. 1998; Koziolek 2011)

- Change in **modularity** during the evolution

Reusability (ISO91; ISO11)

- **modularity** of the generator implementations

Metrics

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

CoCoME Case Study

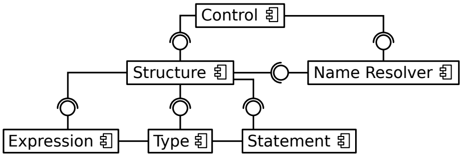
Common Component Modeling Example (Heinrich et al. 2015)

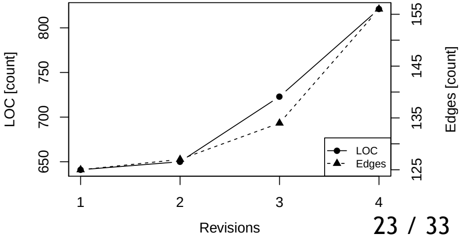
- **Domain**: Information system
- **Source**: PCM, data type, behavior and monitoring DSLs
- **Target**: Java EE and AspectJ
- **Evaluation**: Combination of existing and new generators
  - ProtoCom (Giacinto et al. 2013)
  - Data types, behavior and monitoring (Jung et al. 2013)
- **Evolution steps** ($F_{Behavior}$): 4

➨ Test **GECO**'s feasibility for generator construction

CoCoME Case Study – Behavior Evolution

Behavior Generator

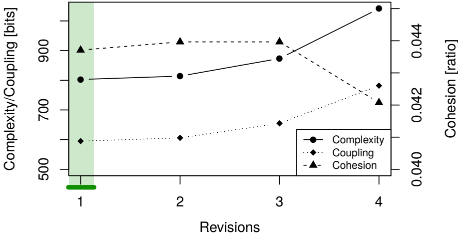


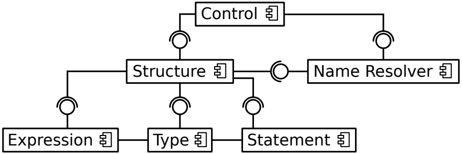


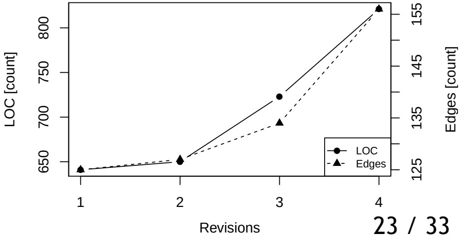
CoCoME Case Study – Behavior Evolution

Behavior Generator

1. Basic functionality



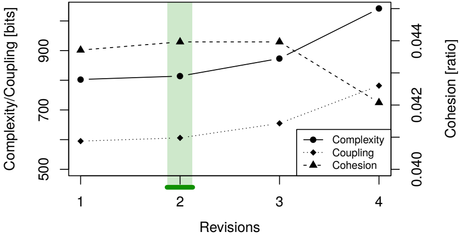


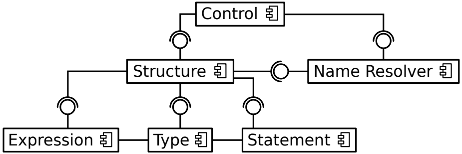


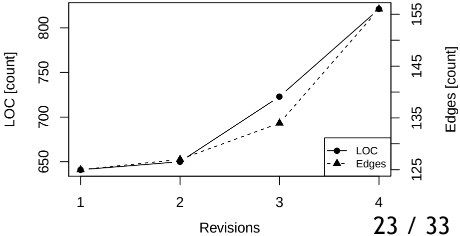
CoCoME Case Study – Behavior Evolution

Behavior Generator

1. Basic functionality
2. Stateless/-full components



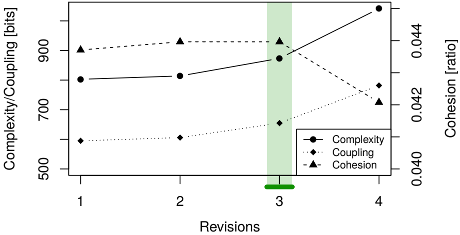


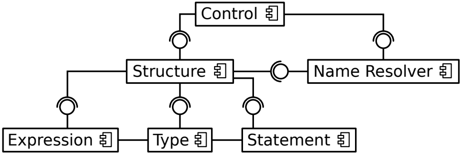


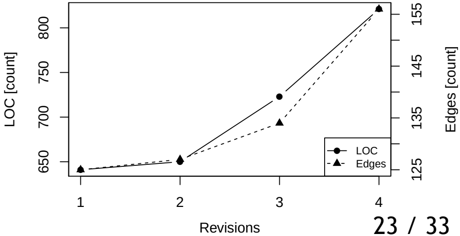
CoCoME Case Study – Behavior Evolution

Behavior Generator

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



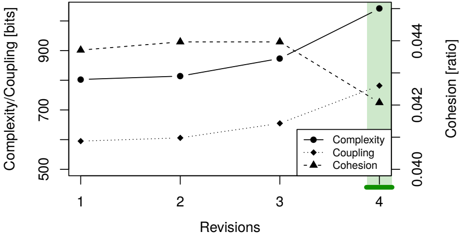


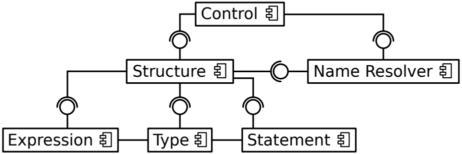


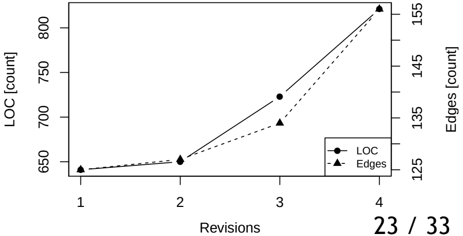
CoCoME Case Study – Behavior Evolution

Behavior Generator

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







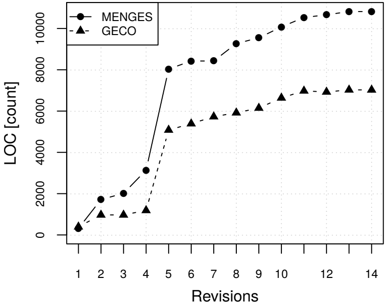
MENGES Case Study

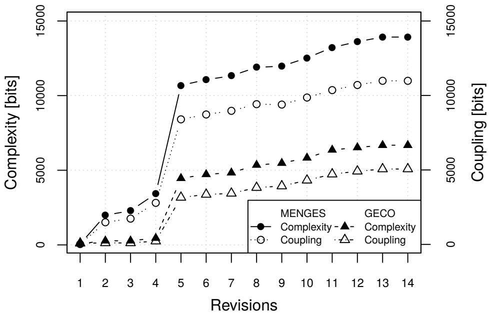
New Generator for MENGES (Goerigk et al. 2012)

- **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**

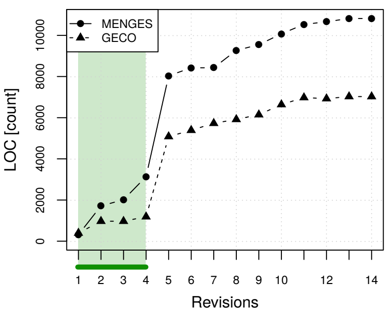
MENGES Case Study – Generator Comparison

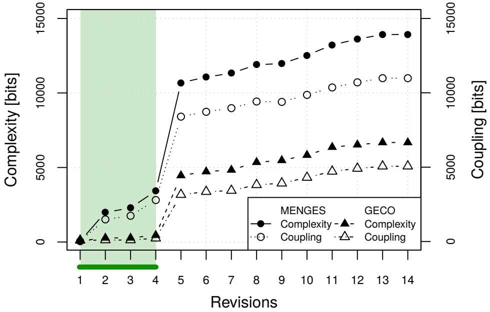




MENGES Case Study – Generator Comparison

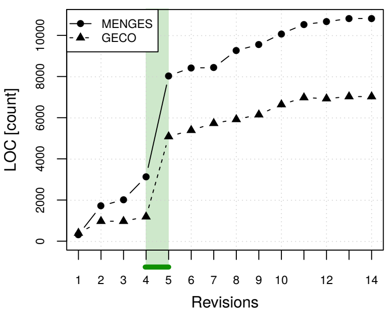
Structure and Typing

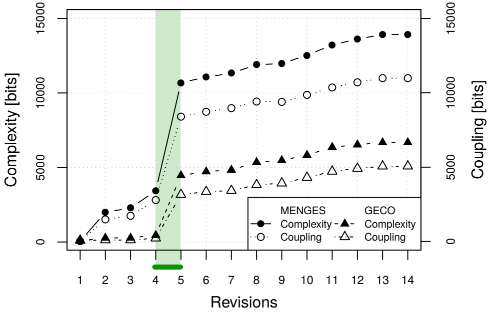




MENGES Case Study – Generator Comparison

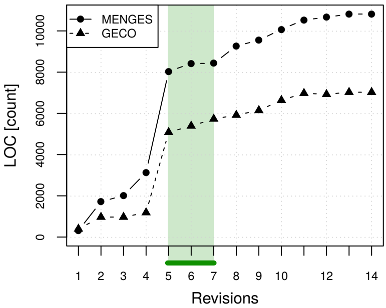
Expressions and Statements

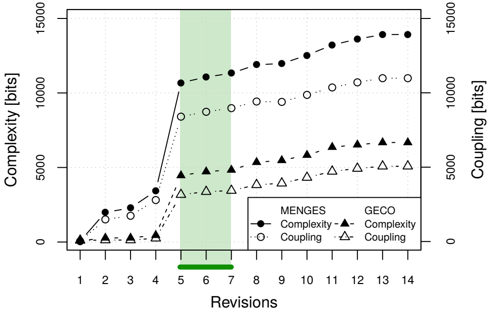




MENGES Case Study – Generator Comparison

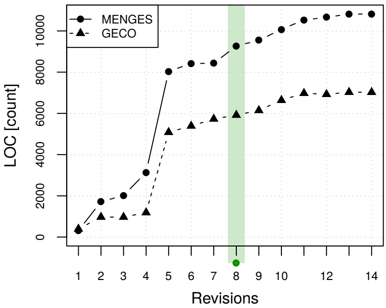
Refactoring and Communications

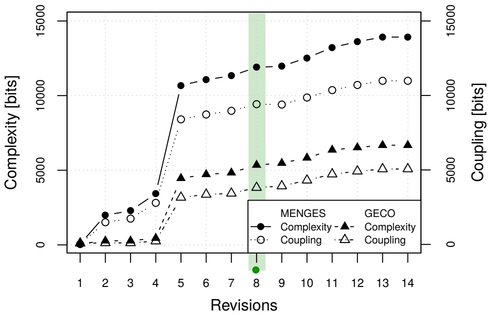




MENGES Case Study – Generator Comparison

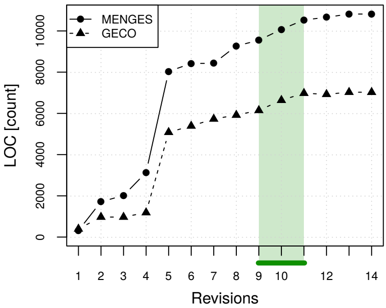
Improvment of Polymorphism

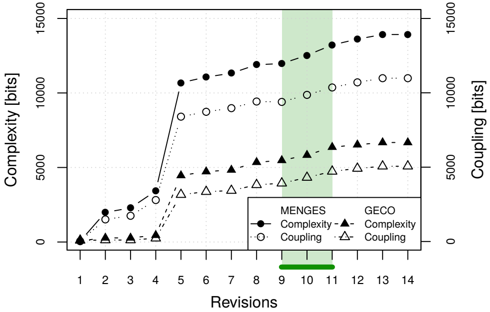




MENGES Case Study – Generator Comparison

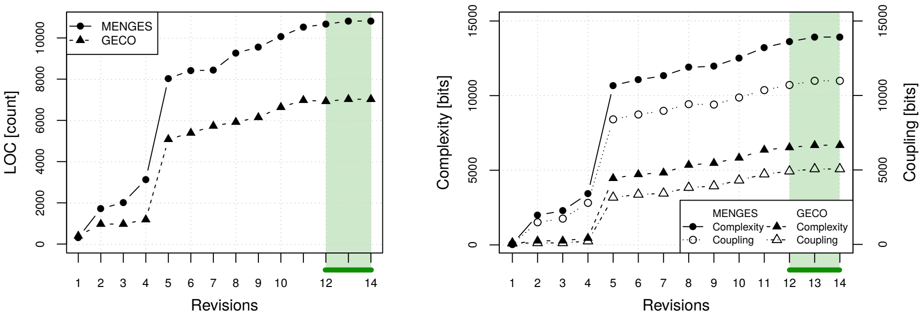
Timers and Template Improvements





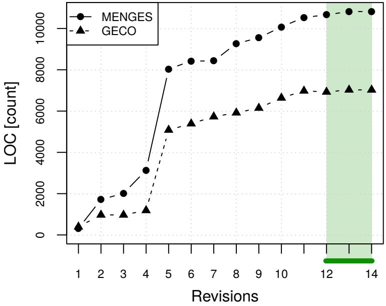
MENGES Case Study – Generator Comparison

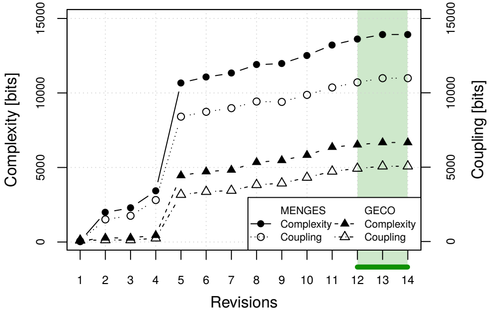
Maintenance



MENGES Case Study – Generator Comparison

Maintenance





	MENGES	GECO	Ratio
Lines of code	10816	7025	1.5396
Complexity	13921.88	6675.88	2.0854
Coupling	10983.81	5060.83	2.1704

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 (Mehmood et al. 2013)

- Theme/UML (Clarke et al. 2005)
- FDAF (Bennett et al. 2010)
- RAM-based (Kienzle et al. 2009; Kramer et al. 2011)

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)

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

Contributions

Approach

- Generator composition megamodel patterns
- Internal modularization of generator fragments

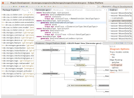
Replication Package

- 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

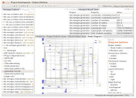
Generator framework and composition tooling

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



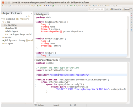
Architecture analysis tool

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



Generators used in CoCoME case study

https://github.com/research-iobserve/



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

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

Bibliography I

Aizenbud-Reshef, Neta et al. (2005). “Operational Semantics for Traceability.” In: *ECMDA Traceability Workshop*. Nuremberg, Germany, 7–14.

Allen, Edward B. (2002). “Measuring graph abstractions of software: an information-theory approach.” In: *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, pp. 182–193. DOI: 10.1109/METRIC.2002.1011337.

Allen, Edward B. et al. (2007). “Measuring size, complexity, and coupling of hypergraph abstractions of software: An information-theory approach.” English. In: *Software Quality Journal* 15.2, pp. 179–212. DOI: 10.1007/s11219-006-9010-3.

Antoniol, Giuliano et al. (2002). “Recovering Traceability Links Between Code and Documentation.” In: *IEEE Transactions on Software Engineering* 28.10, pp. 970–983. DOI: 10.1109/TSE.2002.1041053.

Bibliography II

Atkinson, Colin et al. (2010). "Orthographic Software Modeling: A Practical Approach to View-Based Development." In: *Evaluation of Novel Approaches to Software Engineering*. Ed. by Leszek A. Maciaszek et al. Vol. 69. Communications in Computer and Information Science. Springer, pp. 206–219. DOI: 10.1007/978-3-642-14819-4_15.

Bennett, Jeannette et al. (2010). "Aspect-oriented model-driven skeleton code generation: A graph-based transformation approach." In: *Science of Computer Programming* 75.8. Designing high quality system/software architectures, pp. 689–725. DOI: 10.1016/j.scico.2009.05.005.

Bézivin, Jean et al. (2004). "On the Need for Megamodels." In: *Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications,(2004)*. Vancouver, Canada. URL: https://hal.archives-ouvertes.fr/hal-01222947.

Biehl, Matthias (2010). *Literature Study on Model Transformations*. Technical Report ISRN/KTH/MMK/R-10/07-SE. Stockholm, Sweden: Royal Institute of Technology.

Bibliography III

Clarke, Siobhàn et al. (2005). *Aspect-Oriented Analysis and Design*. Addison-Wesley Professional. ISBN: 0321246748.

Etien, Anne et al. (2015). "Localized model transformations for building large-scale transformations." English. In: *Software & Systems Modeling* 14.3, pp. 1189–1213. DOI: 10.1007/s10270-013-0379-8.

Favre, Jean-Marie (2004). "Foundations of Model (Driven) (Reverse) Engineering – Episode I: Story of The Fidus Papyrus and the Solarus." In: *Post-Proceedings of Dagstuhl seminar on model driven reverse engineering*.

Galvão, I. et al. (2007). "Survey of Traceability Approaches in Model-Driven Engineering." In: *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pp. 313–313. DOI: 10.1109/EDOC.2007.42.

Giacinto, Daria et al. (2013). "Towards Integrating Java EE into ProtoCom." In: *KPDAYS*, pp. 69–78.

Bibliography IV

Goerigk, Wolfgang et al. (2012). “Entwurf einer domänenspezifischen Sprache für elektronische Stellwerke.” In: *Software Engineering*. Ed. by Stefan Jähnichen et al. Vol. 198. LNI. GI, pp. 119–130. ISBN: 978-3-88579-292-5.

Grammel, Birgit et al. (2010). “A generic traceability framework for facet-based traceability data extraction in model-driven software development.” In: *Proceedings of the 6th ECMFA Traceability Workshop*. ECMFA-TW ’10. Paris, France: ACM, pp. 7–14. DOI: 10.1145/1814392.1814394.

Heinrich, Robert et al. (2015). “A Platform for Empirical Research on Information System Evolution.” In: *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015*. KSI Research Inc. and Knowledge Systems Institute Graduate School, pp. 415–420. DOI: 10.18293/SEKE2015-66.

IEC03 (2003). *IEC EN 61131-3*. Standard.

Bibliography V

ISO91 (1991). *International Standard ISO/IEC 9126. Information technology: Software product evaluation: Quality characteristics and guidelines for their use*. Standard. International Standards Organisation.

ISO11 (2011). *ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. Standard. International Standards Organisation.

Jörges, Sven (2013). *Construction and Evolution of Code Generators - A Model-Driven and Service-Oriented Approach*. Vol. 7747. Lecture Notes in Computer Science. Springer, pp. 3–221. ISBN: 978-3-642-36126-5.

Jouault, Frédéric (2005). "Loosely Coupled Traceability for ATL." In: *In Proceedings of the European Conference on Model Driven Architecture (ECMDA) workshop on traceability*, pp. 29–37.

Jung, Reiner et al. (2013). "Model-driven Instrumentation with Kieker and Palladio to Forecast Dynamic Applications." In: *KPDAYS*, pp. 99–108.

Bibliography VI

Jung, Reiner et al. (2014). "A Method for Aspect-oriented Meta-Model Evolution." In: *Proceedings of the 2Nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. VAO '14. York, UK: ACM, 19:19–19:22. DOI: 10.1145/2631675.2631681.

Jung, Reiner et al. (2015). *A Tool for Entropy-based Analysis of Design-time and Runtime Models*.

– (2016). "GECO: A Generator Composition Approach for Aspect-Oriented DSLs." In: *9th International Conference on Model Transformation*. (in prep).

Kapova, Lucia et al. (2010). "Domain-specific templates for refinement transformations." In: *MDI '10: First International Workshop on Model-Drive Interoperability*. Oslo, Norway: ACM, pp. 69–78. ISBN: 978-1-4503-0292-0. DOI: 10.1145/1866272.1866282.

Kienzle, Jörg et al. (2009). "Aspect-oriented multi-view modeling." In: *AOSD*. Ed. by Kevin J. Sullivan et al., pp. 87–98. ISBN: 978-1-60558-442-3.

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

Bibliography VII

Koziolek, Heiko (2011). "Sustainability Evaluation of Software Architectures: A Systematic Review." In: *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*. QoSA-ISARCS '11. Boulder, Colorado, USA: ACM, pp. 3–12. DOI: 10.1145/2000259.2000263.

Kramer, Max E. et al. (2011). "Mapping aspect-oriented models to aspect-oriented code." In: *Proceedings of the 2010 international conference on Models in software engineering*. MODELS'10. Oslo, Norway: Springer, pp. 125–139. DOI: 10.1007/978-3-642-21210-9_12.

Laitinen, Kari (1996). "Estimating Understandability of Software Documents." In: *SIGSOFT Software Engineering Notes* 21.4, pp. 81–92. ISSN: 0163-5948. DOI: 10.1145/232069.232092.

Bibliography VIII

Mehmood, Abid et al. (2013). "Aspect-oriented model-driven code generation: A systematic mapping study." In: *Information and Software Technology* 55.2. Special Section: Component-Based Software Engineering (CBSE), 2011, pp. 395–411. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2012.09.003.

Mens, Tom et al. (2006). "A Taxonomy of Model Transformation." In: *Proc. of the Int'l WS on Graph and Model Transformation*. Vol. 152. Elsevier, pp. 125–142.

Rausch, Andreas et al., eds. (2011). *The Common Component Modelling Example (CoCoME)*. Vol. 5153. Lecture Notes in Computer Science. Springer.

Rowe, David et al. (1998). "Defining Systems Evolvability - A Taxonomy of Change." In: *International Conference and Workshop: Engineering of Computer-Based Systems*. Maale Hachamisha, Israel: IEEE Computer Society, pp. 45+.

Bibliography IX

Saada, H. et al. (2013). "Recovering model transformation traces using multi-objective optimization." In: *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pp. 688–693. DOI: 10.1109/ASE.2013.6693134.

Sánchez Cuadrado, Jesús et al. (2008). "Approaches for Model Transformation Reuse: Factorization and Composition." In: *Proceedings of the 1st International Conference on Theory and Practice of Model Transformations*. ICMT '08. Zurich, Switzerland: Springer, pp. 168–182. ISBN: 978-3-540-69926-2. DOI: 10.1007/978-3-540-69927-9_12.

Vanhooff, Bert et al. (2006). "Towards a Transformation Chain Modeling Language." English. In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by Stamatis Vassiliadis et al. Vol. 4017. Lecture Notes in Computer Science. Springer, pp. 39–48. DOI: 10.1007/11796435_6.

Wimmer, Manuel et al. (2011). "Reusing Model Transformations across Heterogeneous Metamodels." In: *ECEASST* 50.

Foundations

Metamodel Partitioning

Identifying Metamodel Partitions

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

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

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

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 $n = |R|$

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

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

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 $n = |R|$

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

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

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

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

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 $n = |R|$

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

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

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

5. Remove identified partitions $P_i^{'}$ from graph

Metamodel Partitioning

Identifying Metamodel Partitions

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

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

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 $n = |R|$

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

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

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

5. Remove identified partitions $P_i^{'}$ from graph
6. Reiterate process with remaining graph

Model Traceability

Model Traces

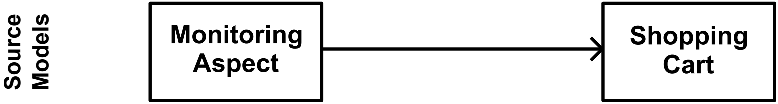
Model traces are a set of source and target nodes with a relation between them. (Aizenbud-Reshef et al. 2005)

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

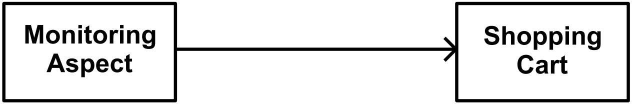
Approach

Join-Point Computation



Join-Point Computation

Source Models



Target Models

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

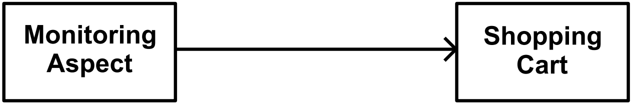
AspectJ Pointcuts

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

Java Servlet

Join-Point Computation

Source Models



Target Models

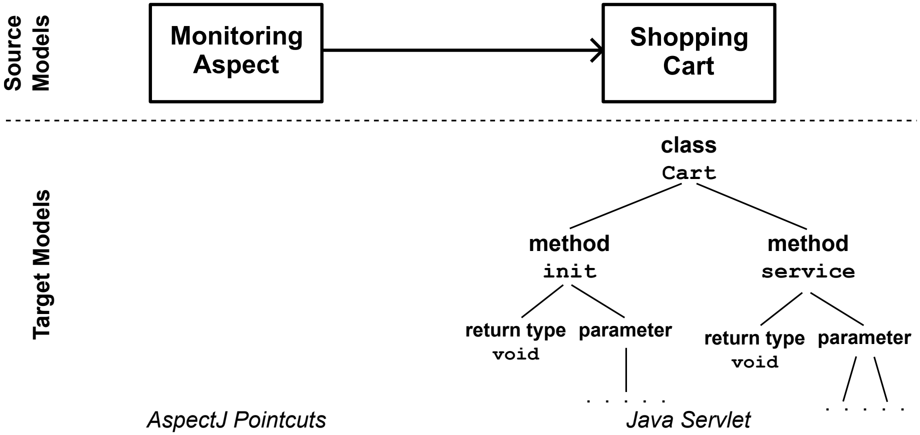
```
call(void init (...)) ||
call(void service(...))
```

AspectJ Pointcuts

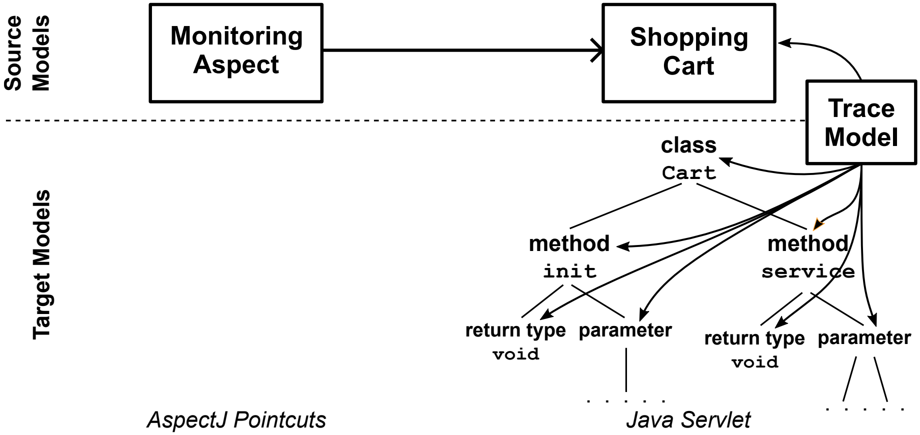
```
class Cart extends HTTPServlet {
      public void init (...) {
            ...
      }
      public void service (...) {
            ...
      }
      ...
}
```

Java Servlet

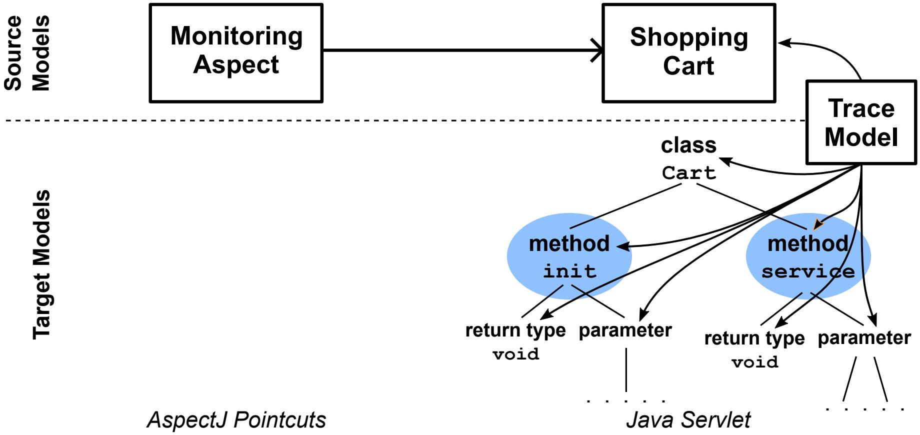
Join-Point Computation



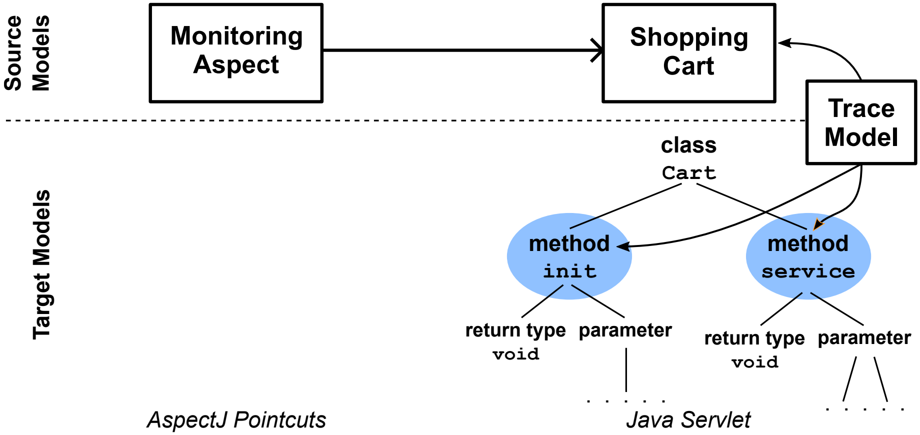
Join-Point Computation



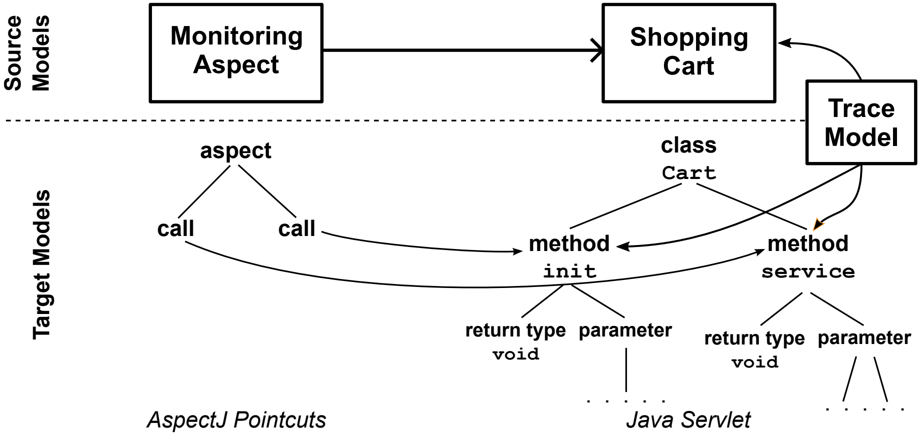
Join-Point Computation



Join-Point Computation



Join-Point Computation



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

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)

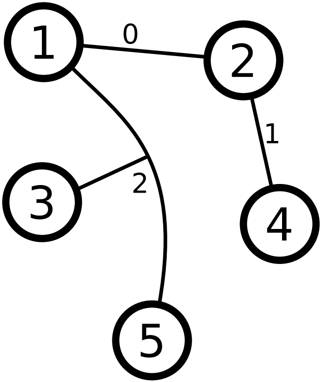
- Low **coupling** of modules of a system
- High inner module **cohesion** of a system

Stability of the code base (ISO11)

- Low **coupling** of modules of a system

Evaluation Measure properties for each revision

Metric: Amount of Information in the System



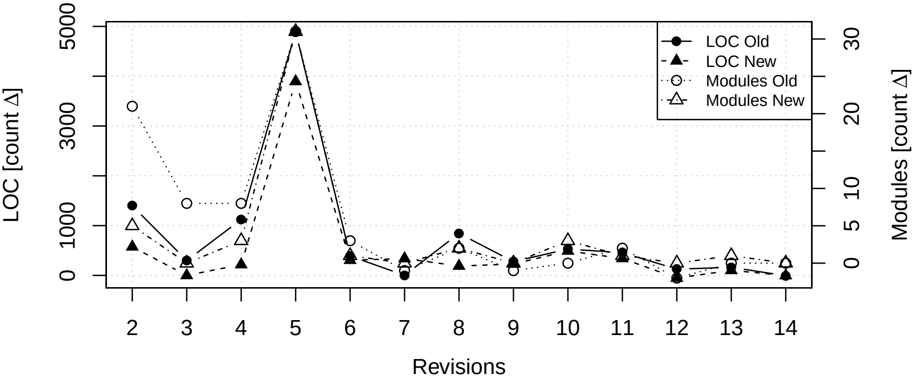
Nodes	Hyperedges	Probability $\hat{p}_l$
1	101	1/5
2	110	1/5
3,5	001	2/5
4	010	1/5

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

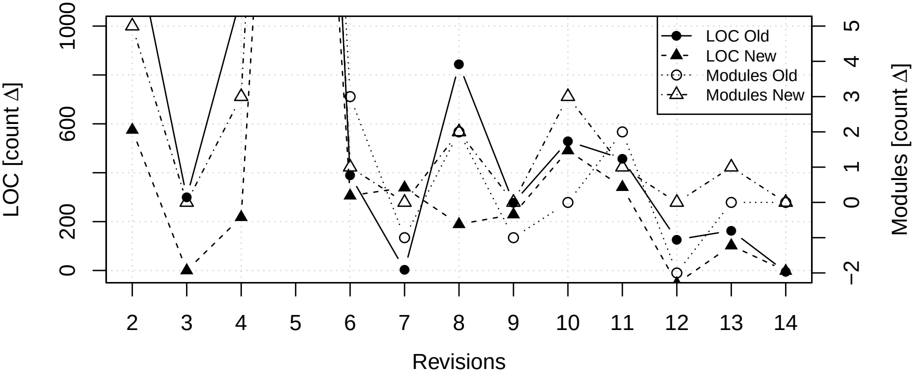
$$Size(S) = 3 * 2.322 + 2 * 1.322 = 9.610 bit$$

Metric by Edward B. Allen Allen et al. 2007

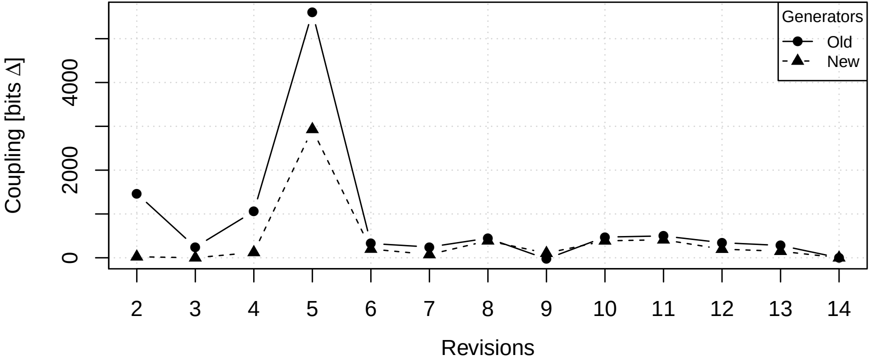
MENGES LOC and Modules



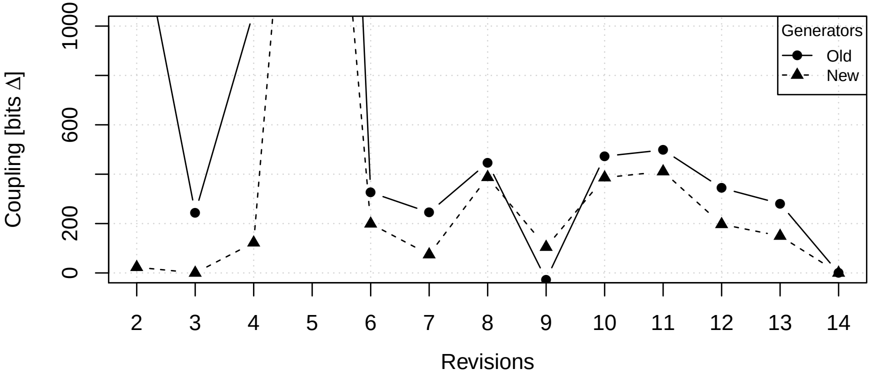
MENGES LOC and Modules



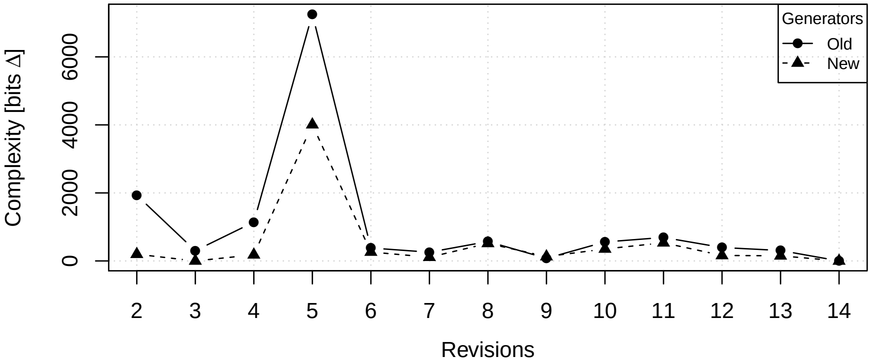
MENGES Coupling Delta



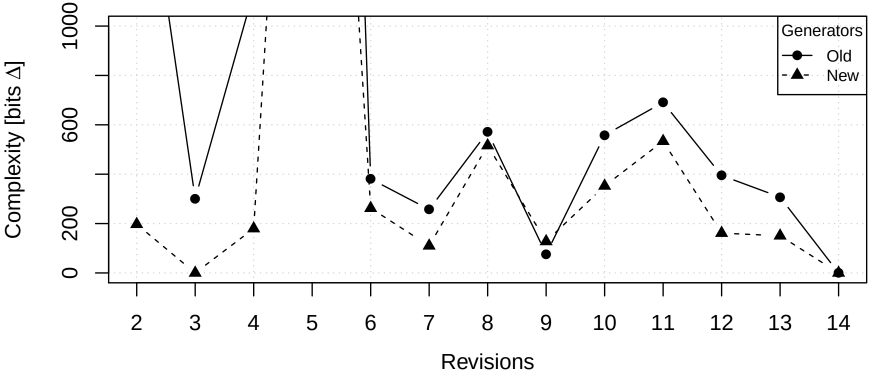
MENGES Coupling Delta



MENGES Complexity Delta



MENGES Complexity Delta



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`

Instrumentation Aspect Language

```
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
```

Instrumentation Record Language

```
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
}
```