GECO: Automatic **Ge**nerator-**Co**mposition for (Aspect-oriented) DSLs

Doctoral Symposium - MODELS 2014

Reiner Jung

Christian-Albrechts-University
Kiel, Germany

30.09.2014

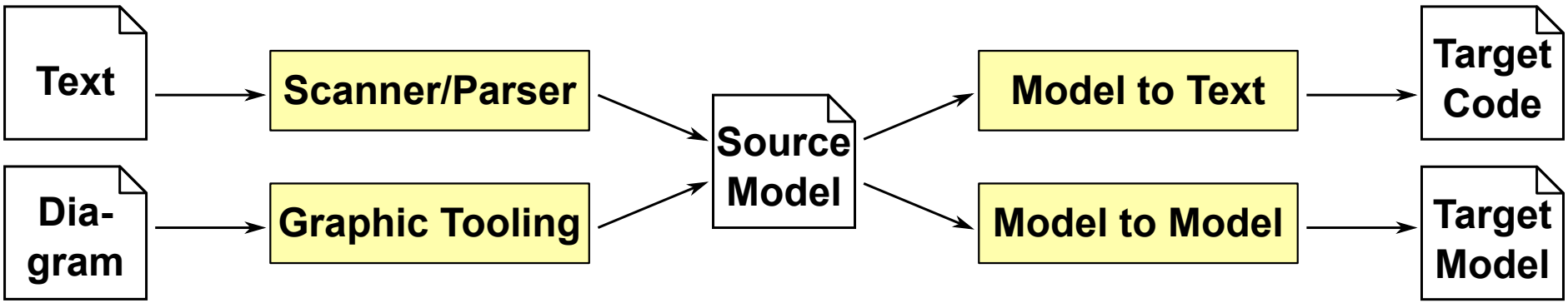
MENGES
KoSSE

Design For
FUTURE

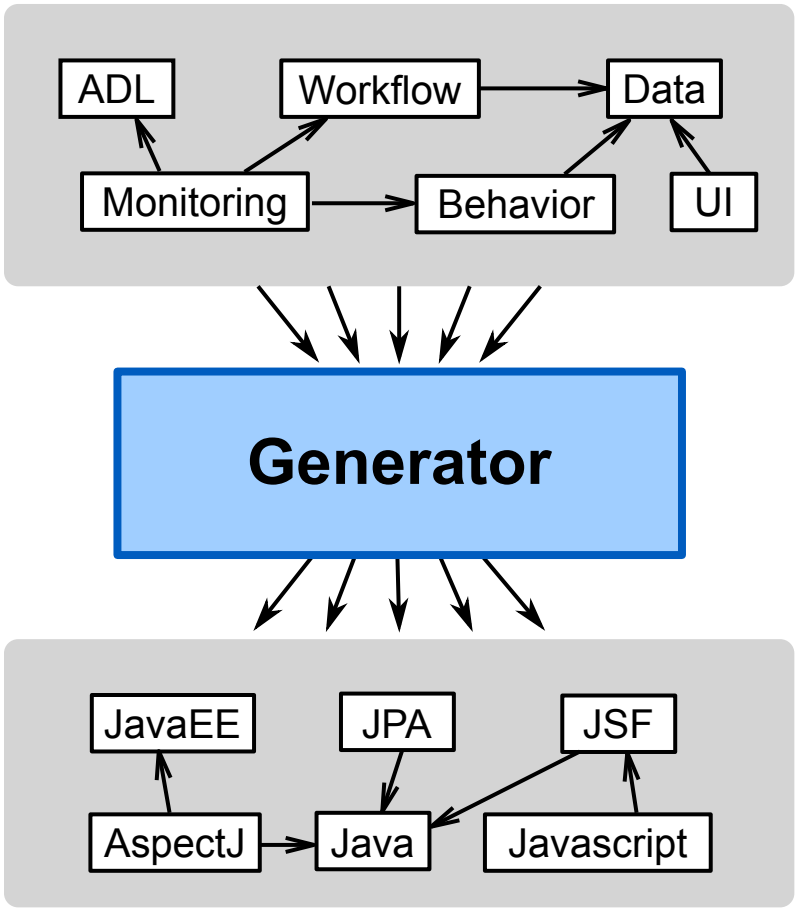
iObserve

Domain-specific Language (DSL)

- Text-based: meta-models and grammars
- Graphical: meta-models and visualization pattern

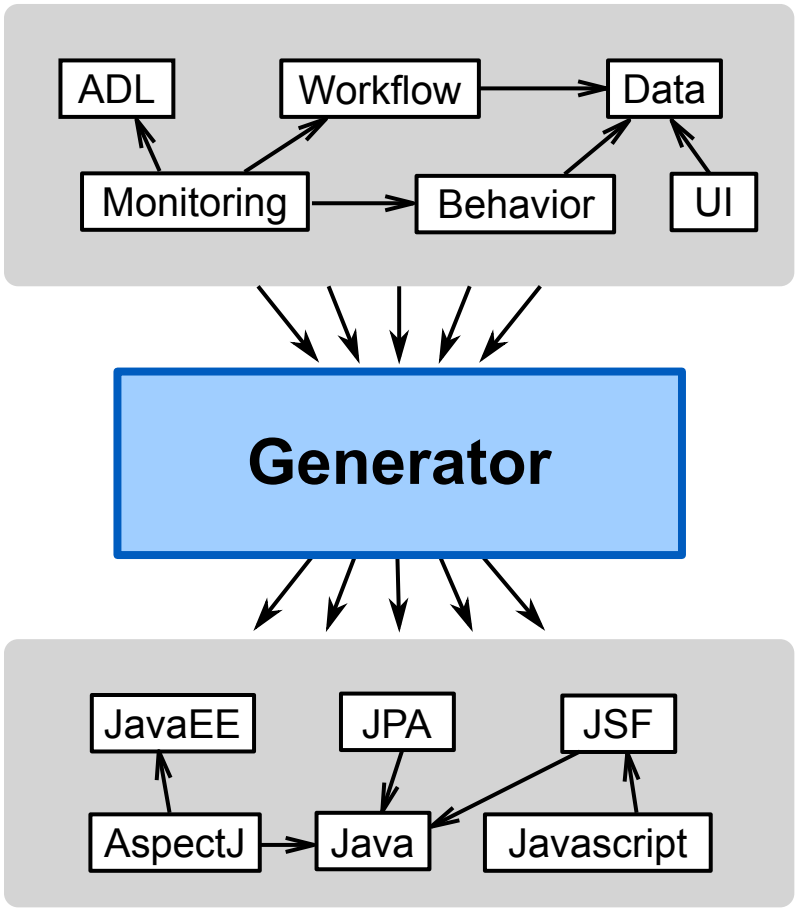


Source Meta-Models



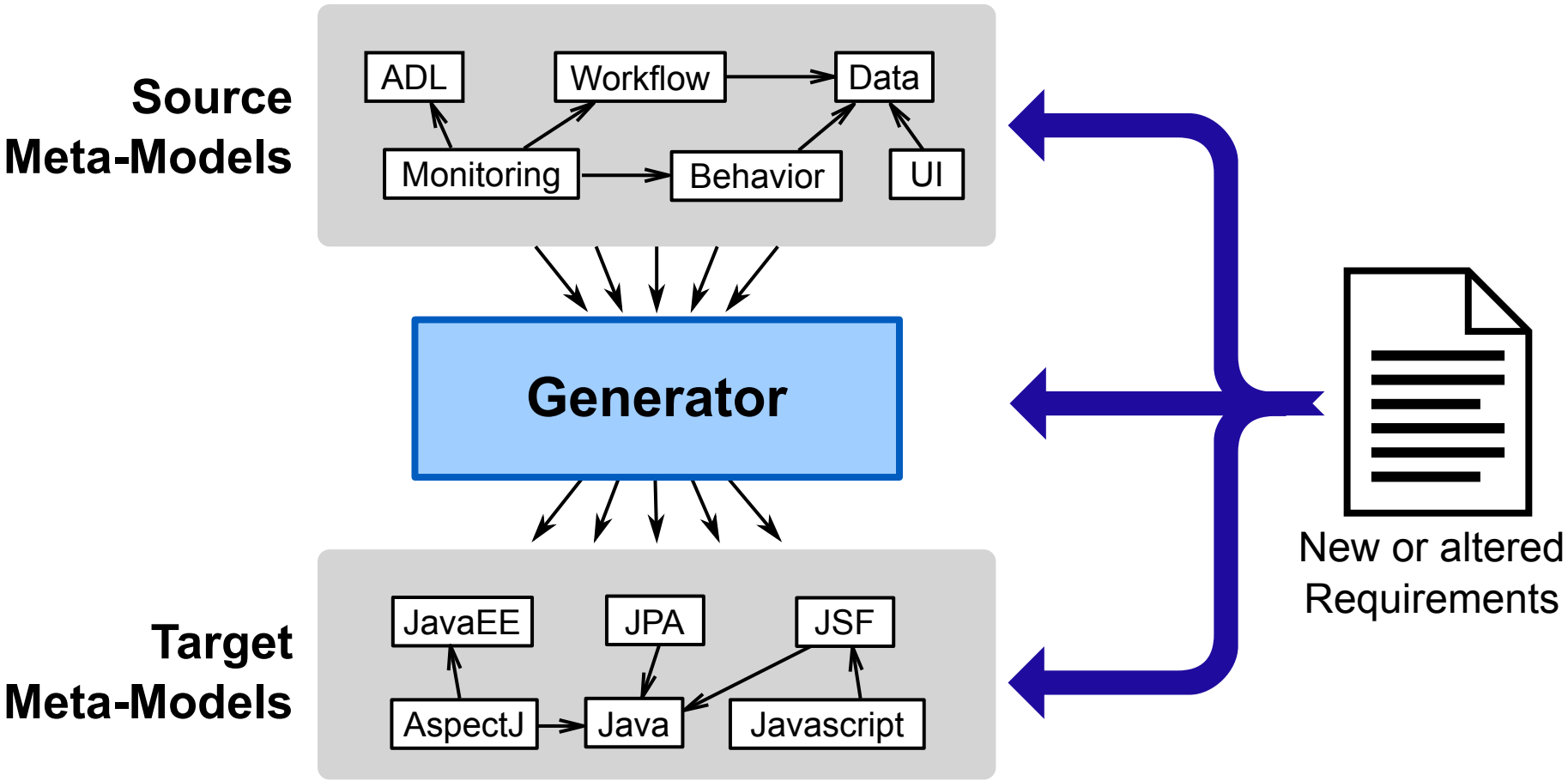
Target Meta-Models

Source Meta-Models



Target Meta-Models

New or altered Requirements



- Every meta-model change cause a generator code change
- Multiple changes convery code degradation

Related Work

Approaches

- Formal Design Analysis Framework [Bennett et al., 2010]
- Theme/UML [Clarke and Baniassad, 2005]
- Reusable Aspect Models [Klein and Kienzle, 2007]

Characteristics

- UML centric
- Use of model weaving [Morin et al., 2008, Didonet et al., 2006]
- Focus on modelling, not on model generation

Aspect Modeling Domain [Mehmood and Jawawi, 2013]
Limited work on generator composition and evolution

Aspect Modeling Domain [Mehmood and Jawawi, 2013]
Limited work on generator composition and evolution

Generator for product lines [Kapova et al., 2010]

- Activation of generator fragments based on features
- Goal: Support of target model variations

Aspect Modeling Domain [Mehmood and Jawawi, 2013]
Limited work on generator composition and evolution

Generator for product lines [Kapova et al., 2010]

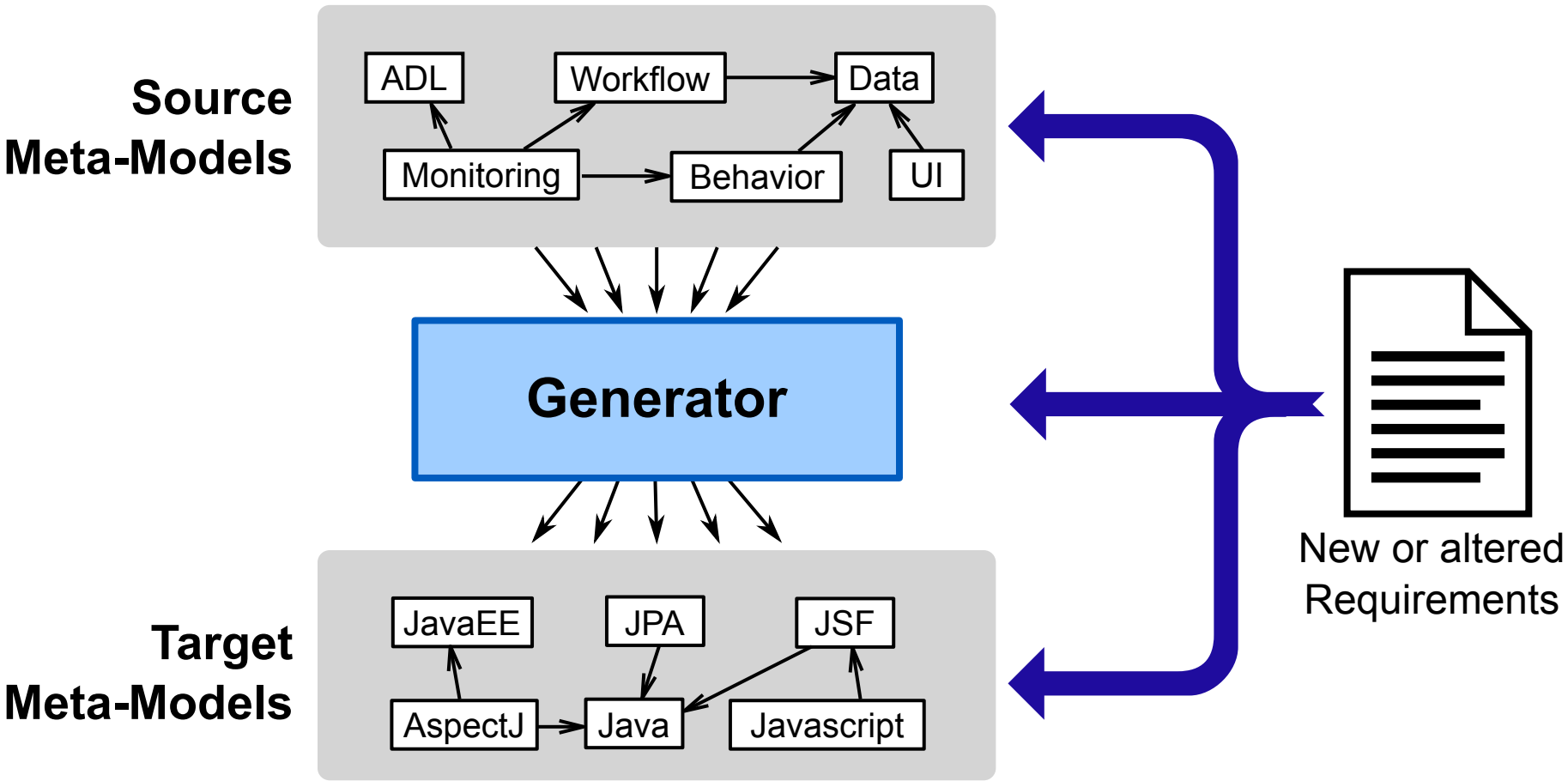
- Activation of generator fragments based on features
- Goal: Support of target model variations

Genesys generator composition [Jörges, 2013]

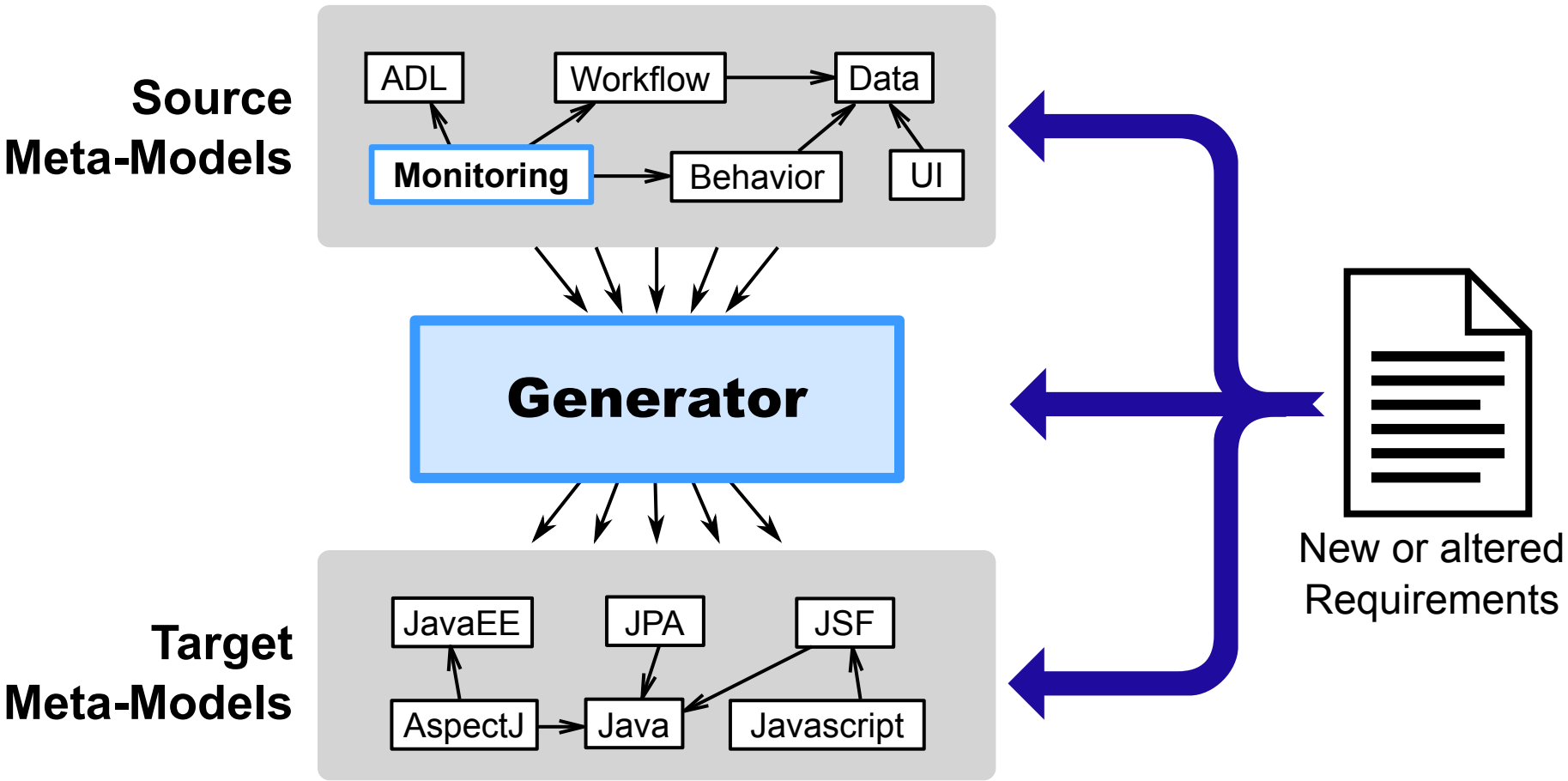
- Combination proofen transformations
- Uses small *correct* transformations
- Goal: correct compilers

Approach

Solution Overview

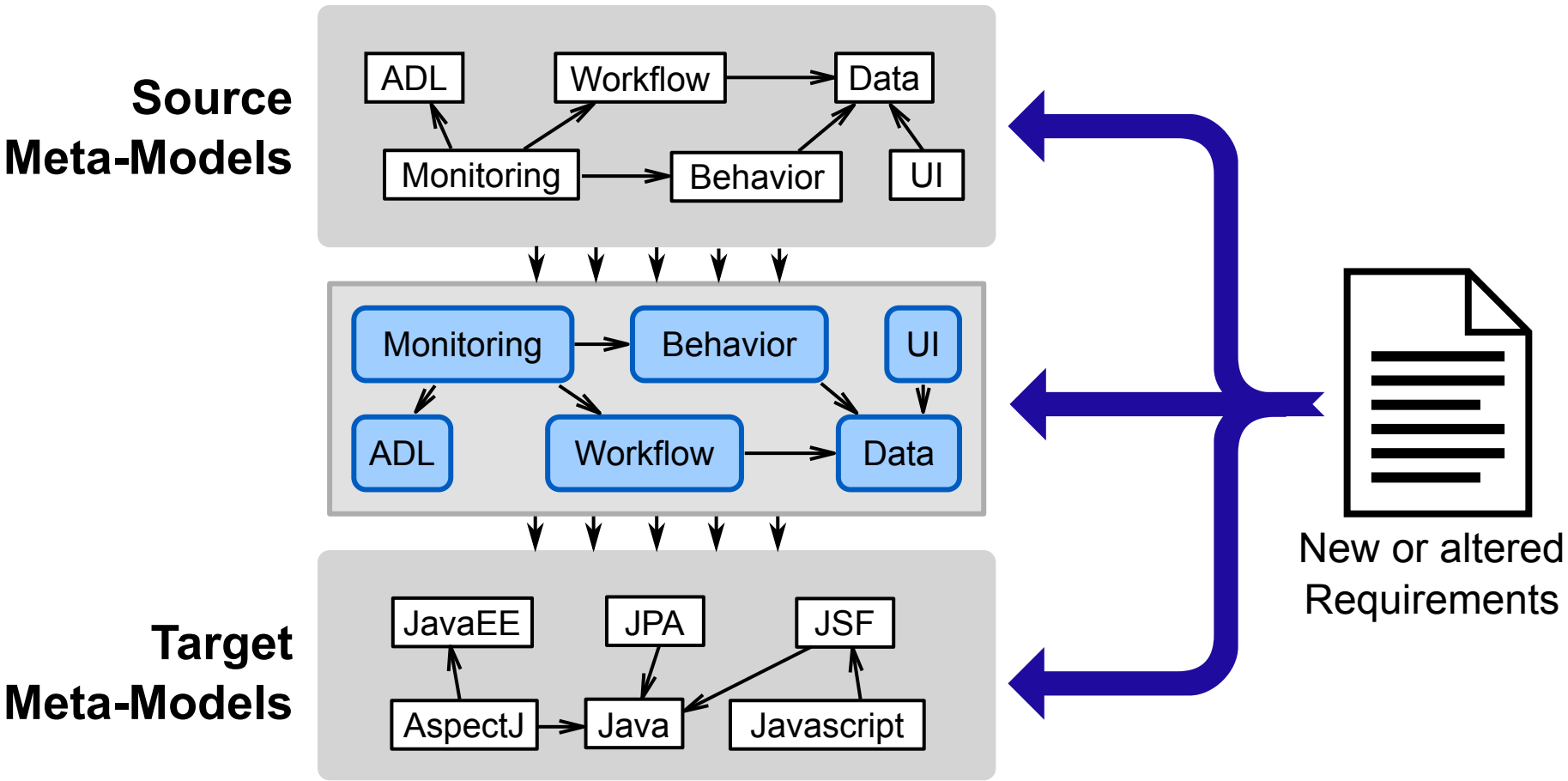


Key challange How to combine generator outputs?



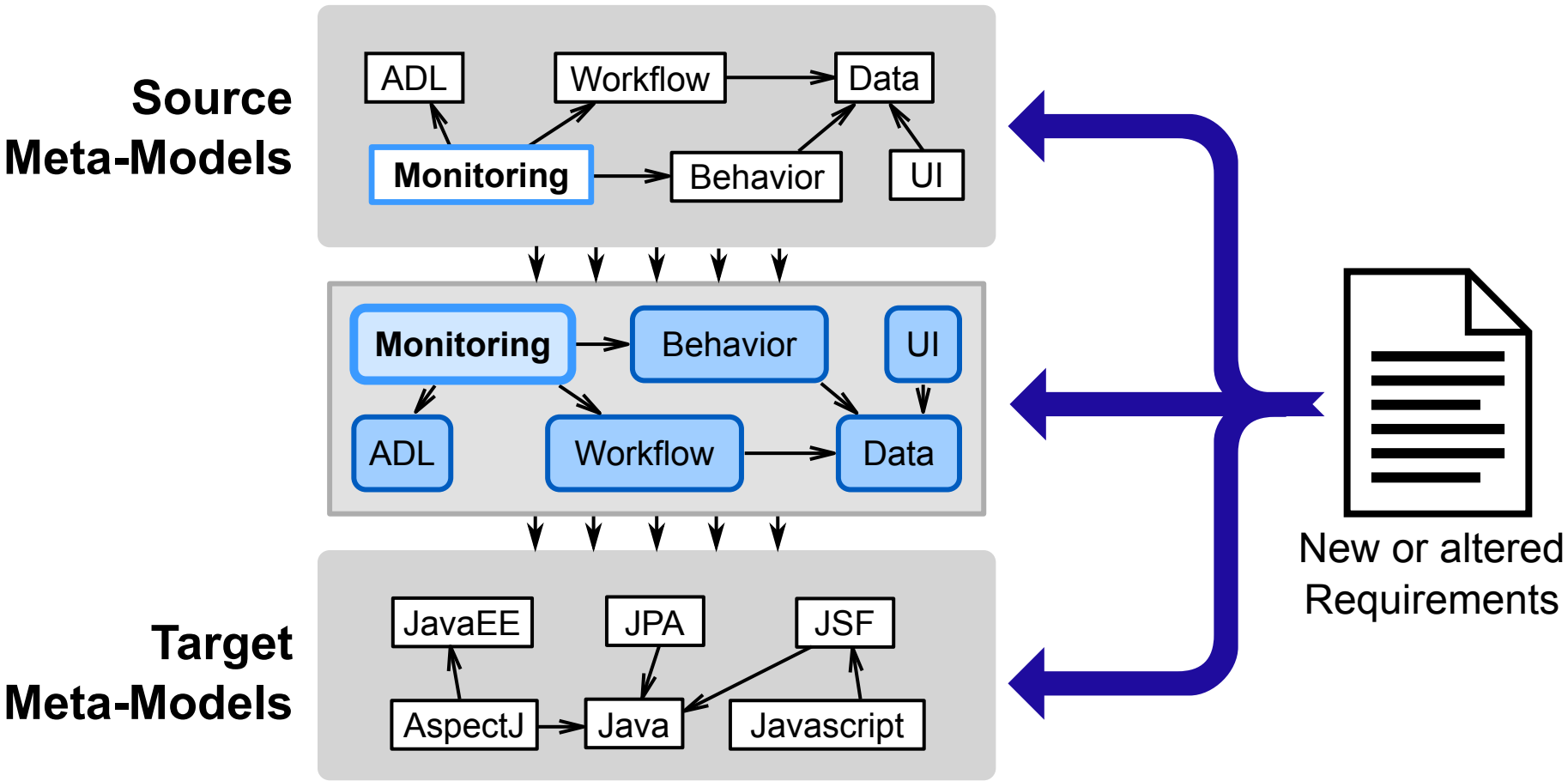
Key challange How to combine generator outputs?

Solution Overview



Key challange How to combine generator outputs?

Solution Overview



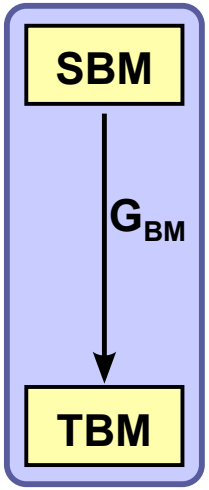
Key challange How to combine generator outputs?

Different Generator Architecture Styles

Different Generator Architecture Styles

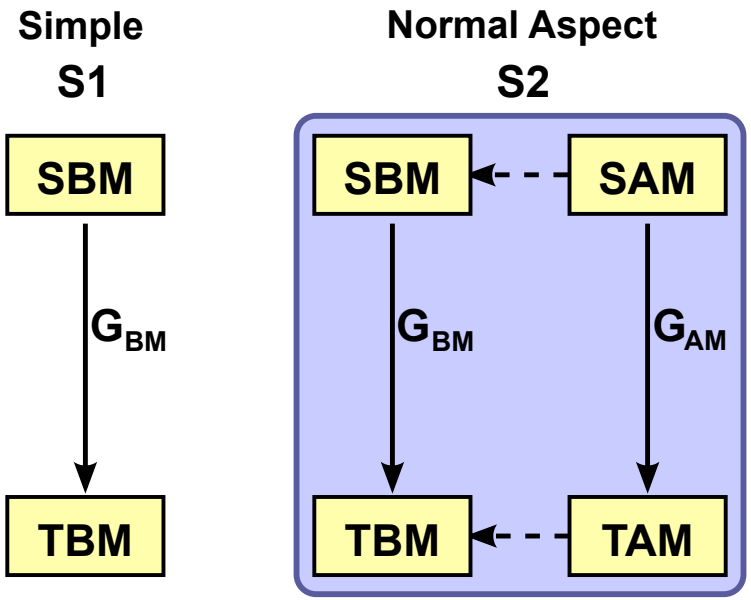
Simple

S1



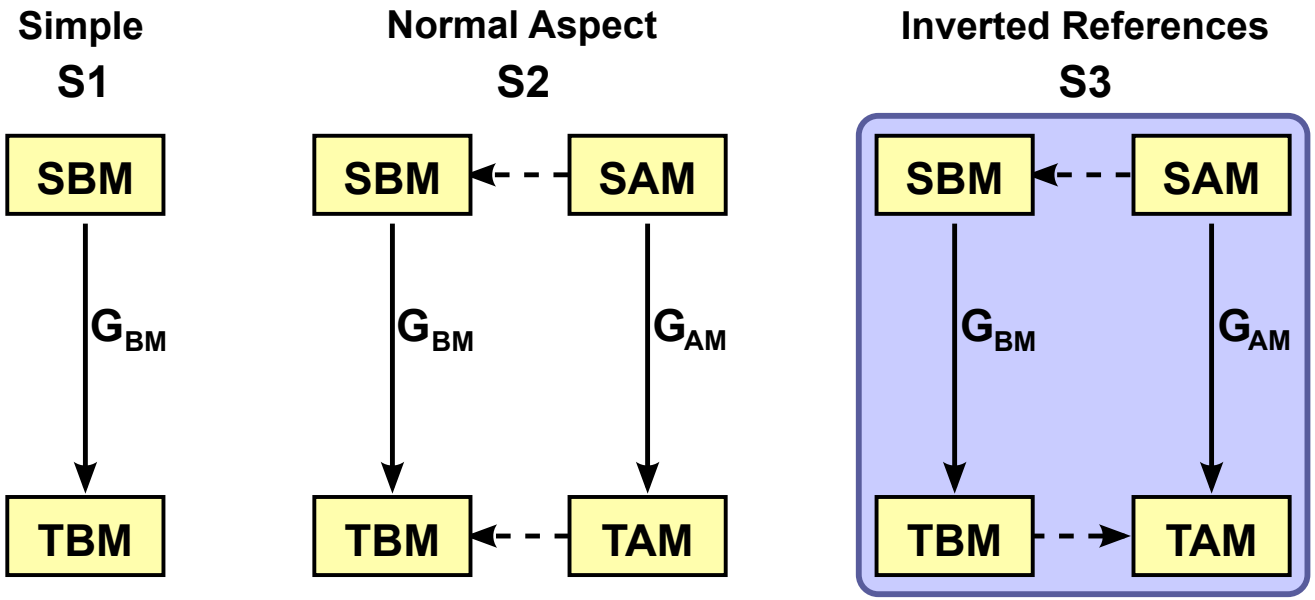
SBM Source Base Meta-Model
TBM Target Base Meta-Model
SAM Source Aspect Meta-Model
TAM Target Aspect Meta-Model
G Transformation
References

Different Generator Architecture Styles



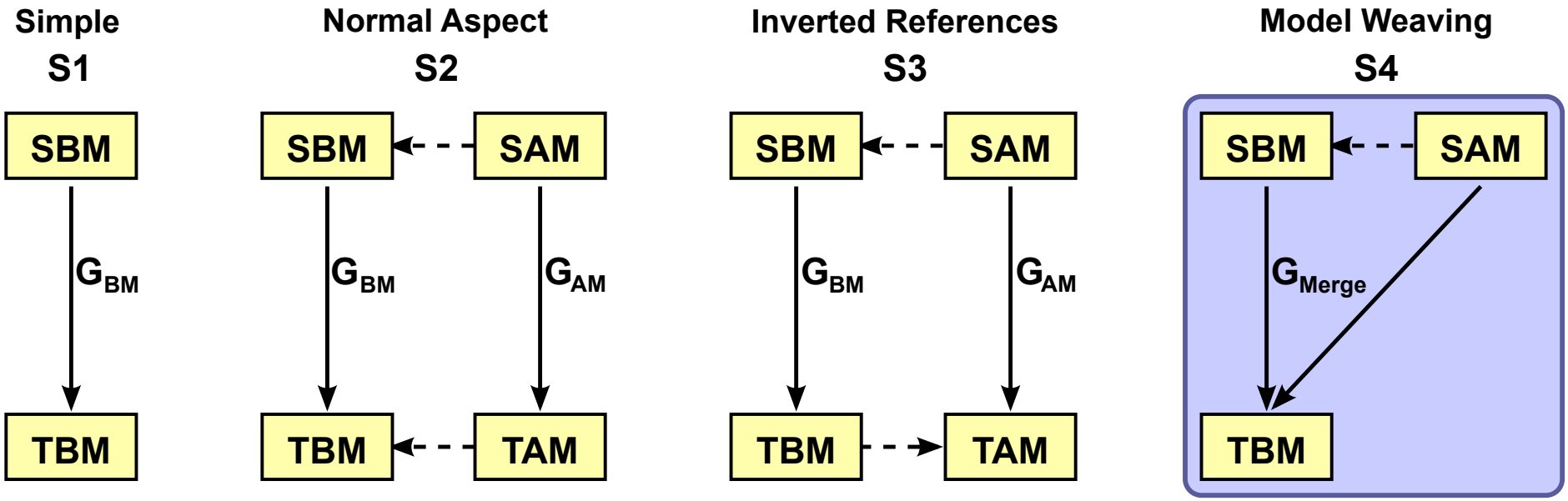
SBM Source Base Meta-Model
TBM Target Base Meta-Model
SAM Source Aspect Meta-Model
TAM Target Aspect Meta-Model
G Transformation
References

Different Generator Architecture Styles



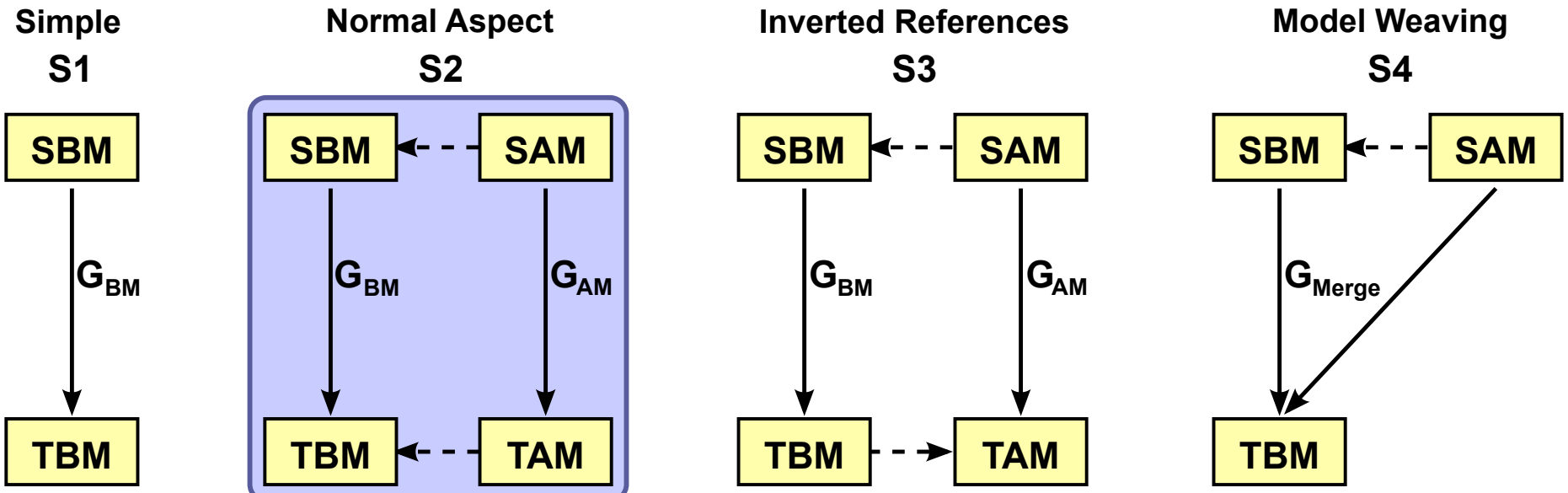
SBM Source Base Meta-Model
TBM Target Base Meta-Model
SAM Source Aspect Meta-Model
TAM Target Aspect Meta-Model
G Transformation
References

Different Generator Architecture Styles

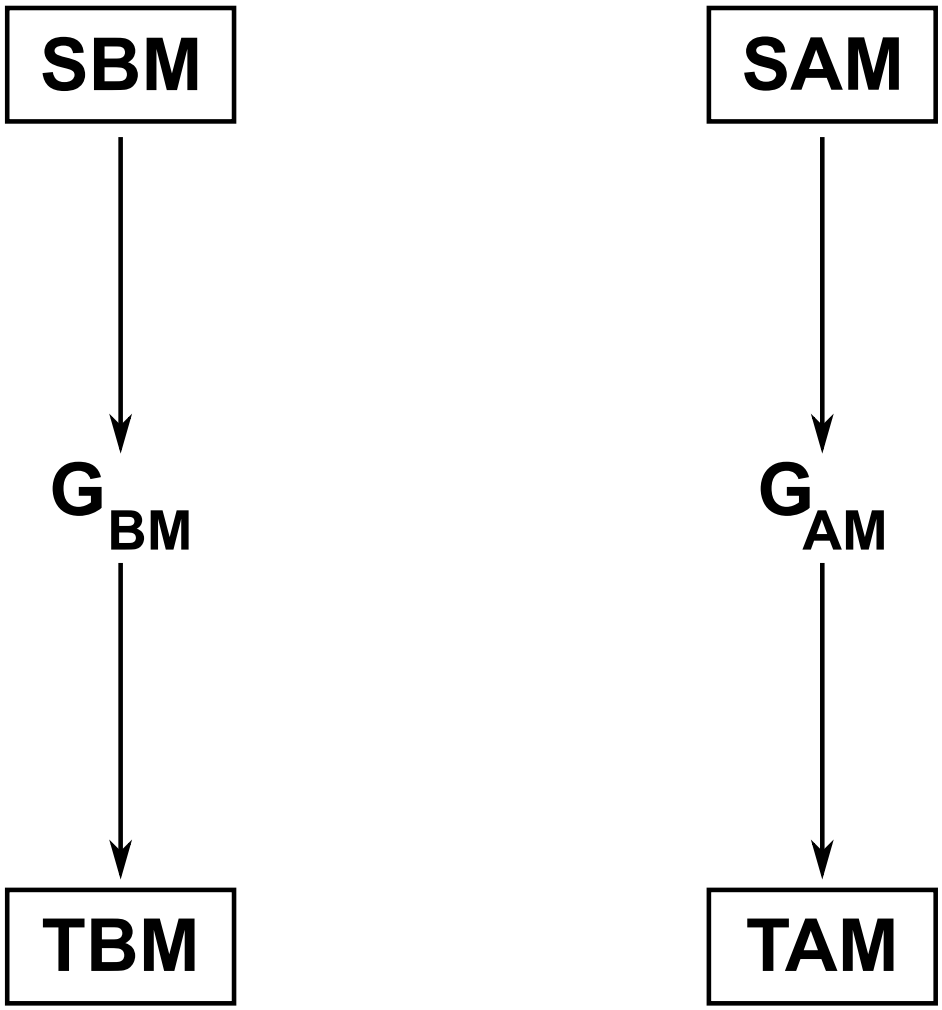


SBM Source Base Meta-Model
TBM Target Base Meta-Model
SAM Source Aspect Meta-Model
TAM Target Aspect Meta-Model
G Transformation
References

Different Generator Architecture Styles

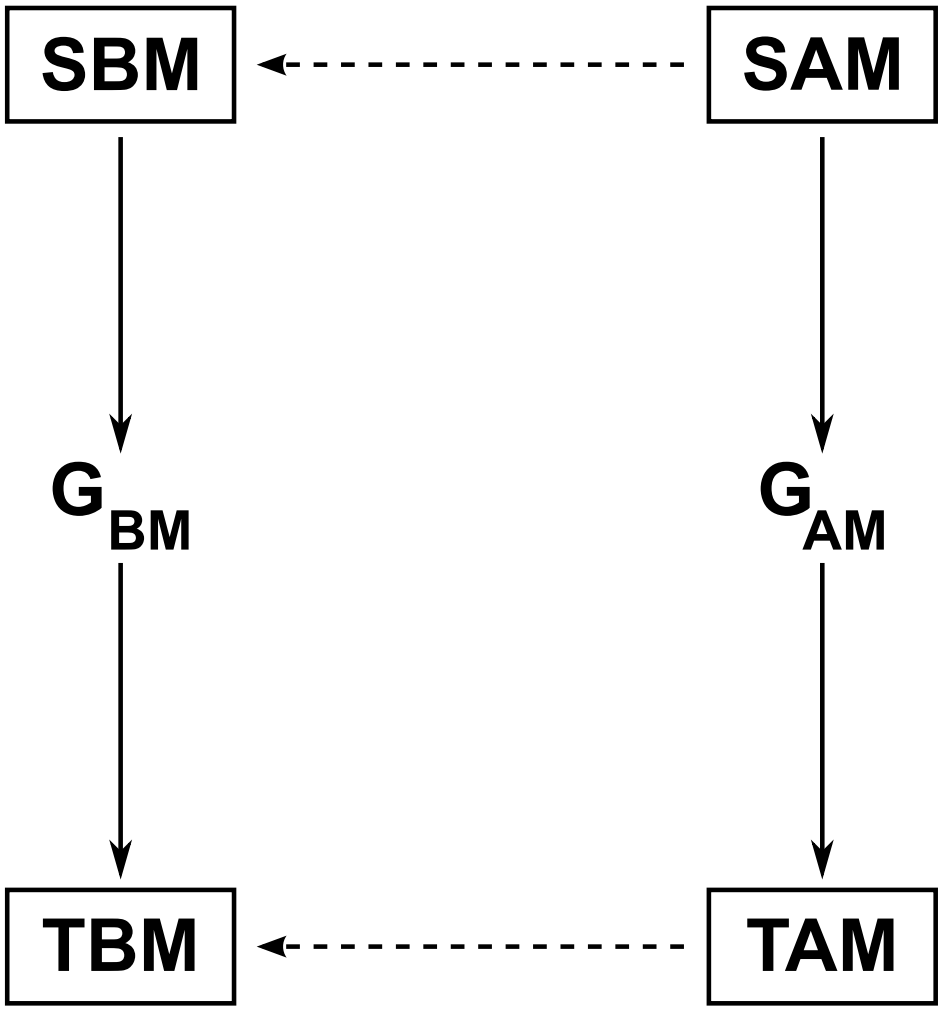


SBM Source Base Meta-Model
TBM Target Base Meta-Model
SAM Source Aspect Meta-Model
TAM Target Aspect Meta-Model
G Transformation
References

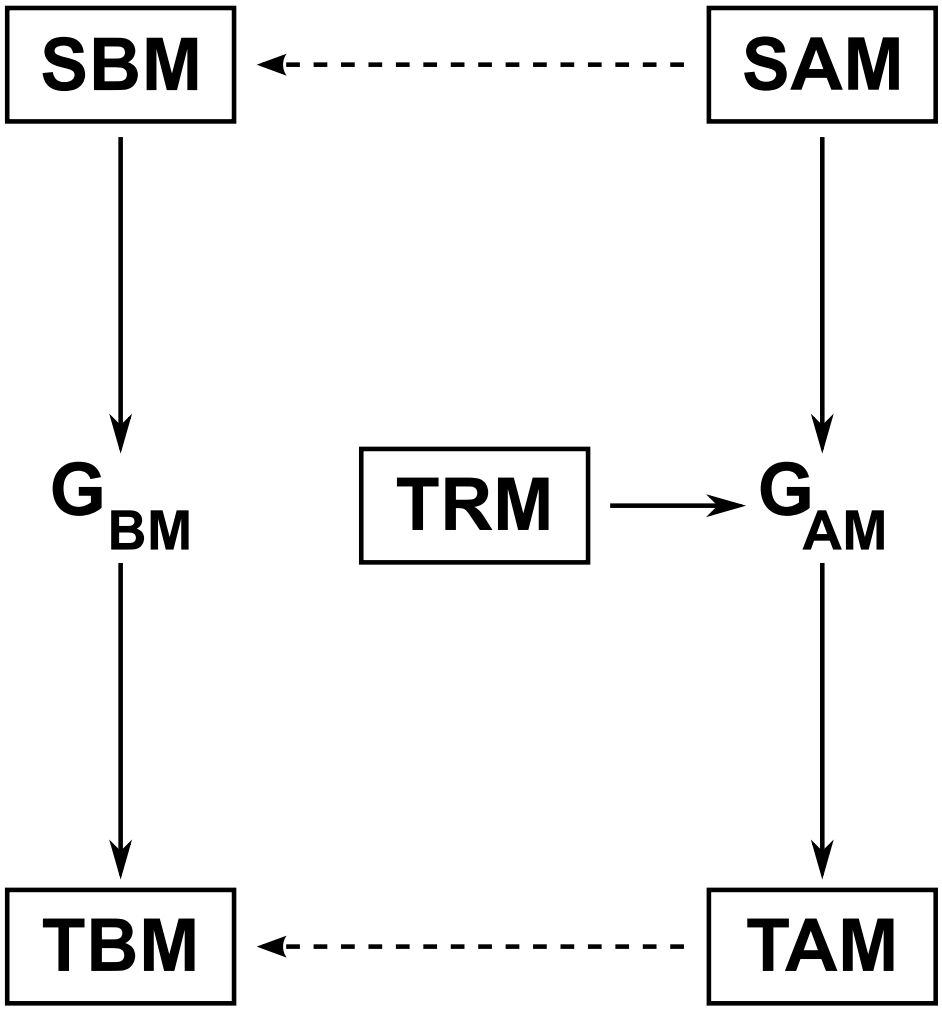


SBM Source Base Meta-Model
SAM Source Aspect Meta-Model
TBM Target Base Meta-Model
TAM Target Aspect Meta-Model
TRM Trace Meta-Model

Style 2: Normal Aspect

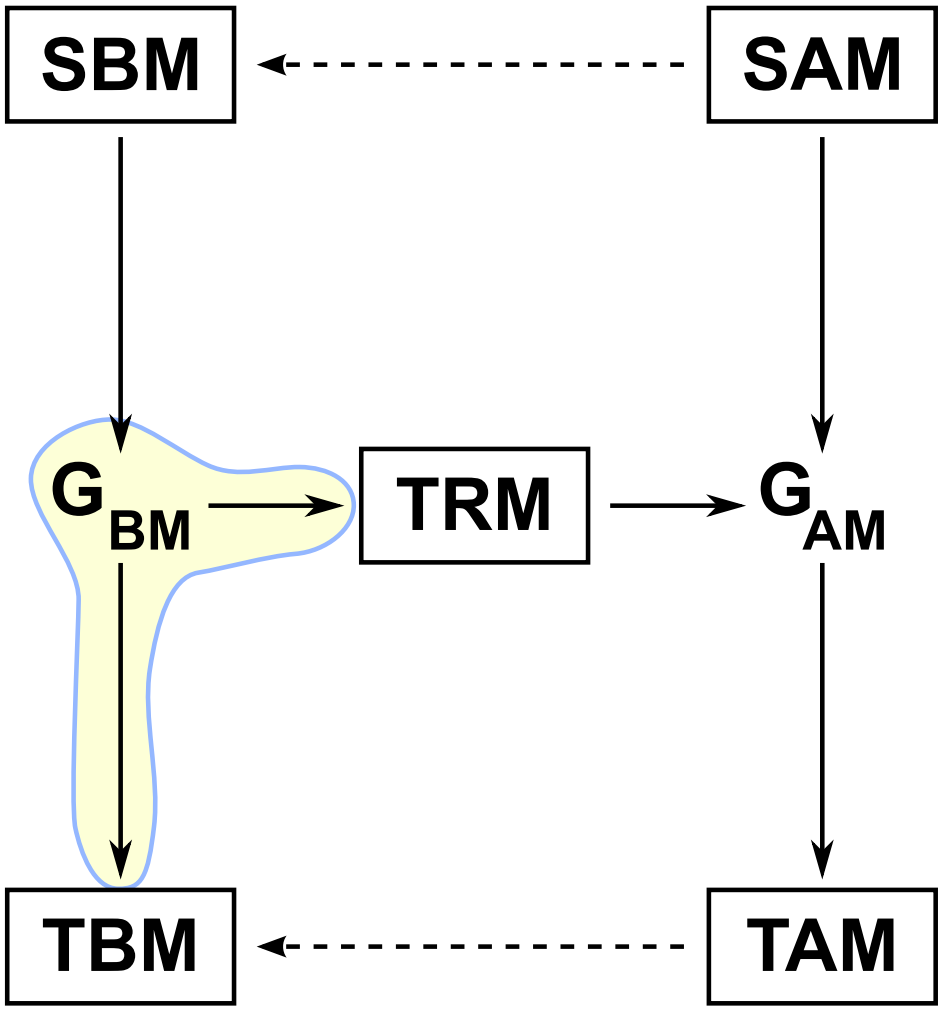


SBM Source Base Meta-Model
SAM Source Aspect Meta-Model
TBM Target Base Meta-Model
TAM Target Aspect Meta-Model
TRM Trace Meta-Model



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

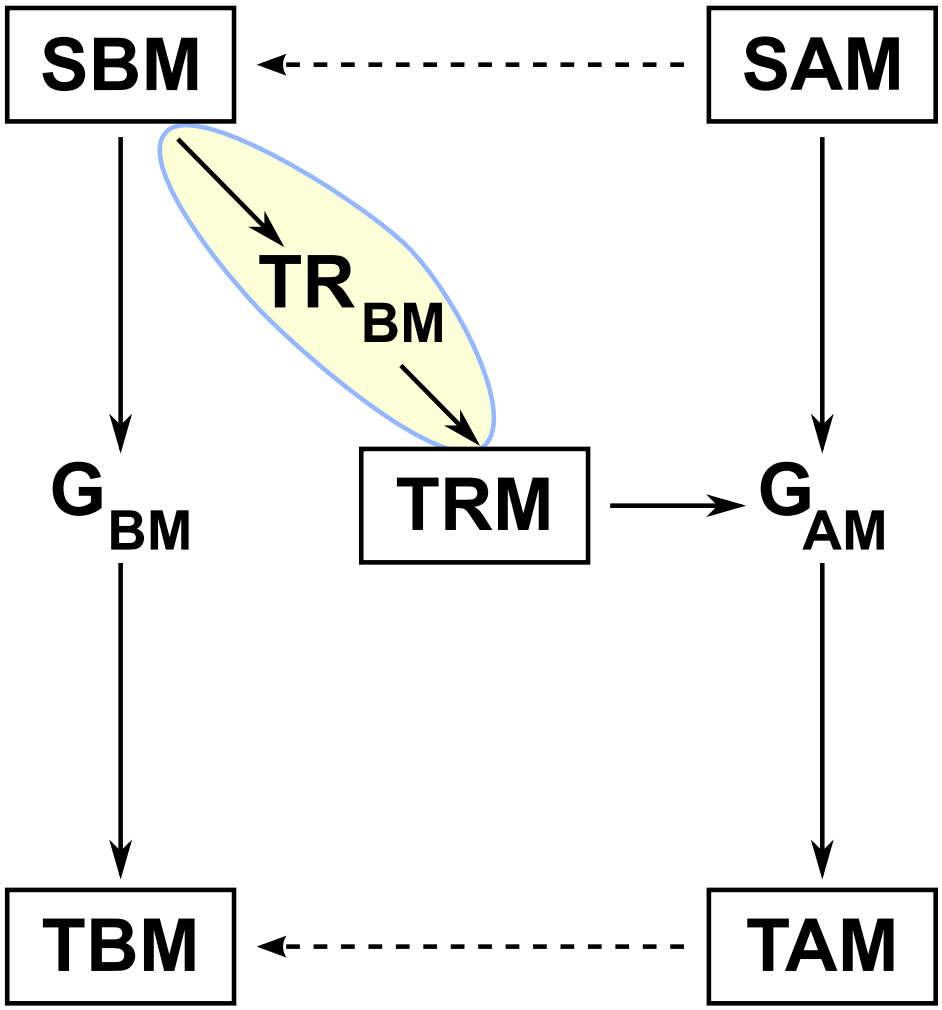
Style 2: Normal Aspect



Transformation with two outputs:

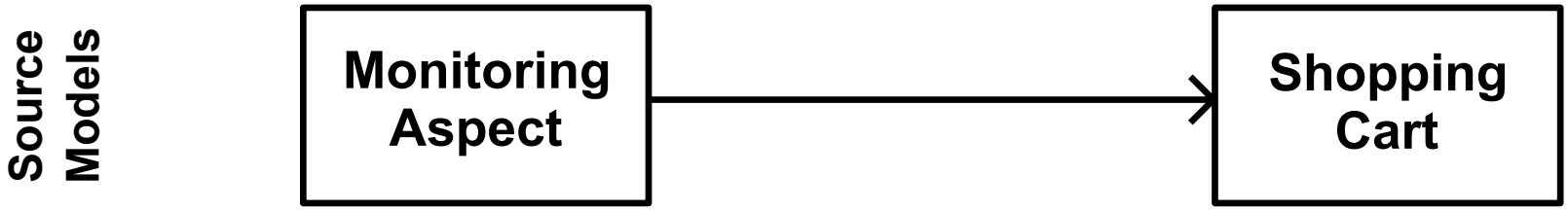
- **Base model TBM**
- **Trace model TRM**

SBM	**Source Base Meta-Model**
SAM	**Source Aspect Meta-Model**
TBM	**Target Base Meta-Model**
TAM	**Target Aspect Meta-Model**
TRM	**Trace Meta-Model**

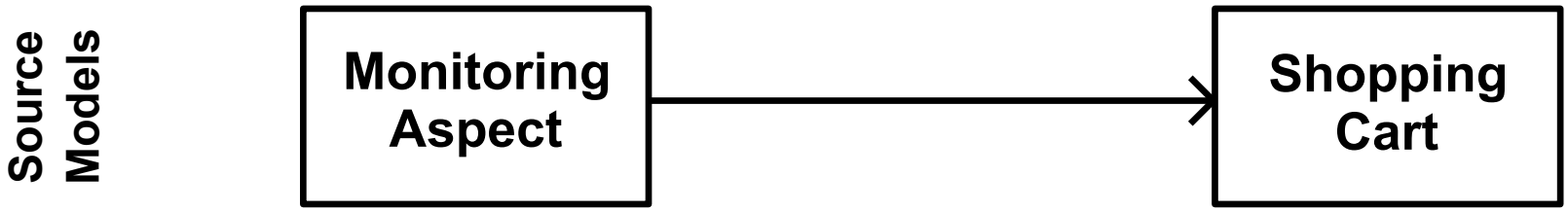


Auxiliary transformation. Used for:
- legacy base model transformations
- API constraints hinder two output models

SBM **Source Base Meta-Model**
SAM **Source Aspect Meta-Model**
TBM **Target Base Meta-Model**
TAM **Target Aspect Meta-Model**
TRM **Trace Meta-Model**


Source Models
Monitoring Aspect
Shopping Cart

Computing Target Model Join Points



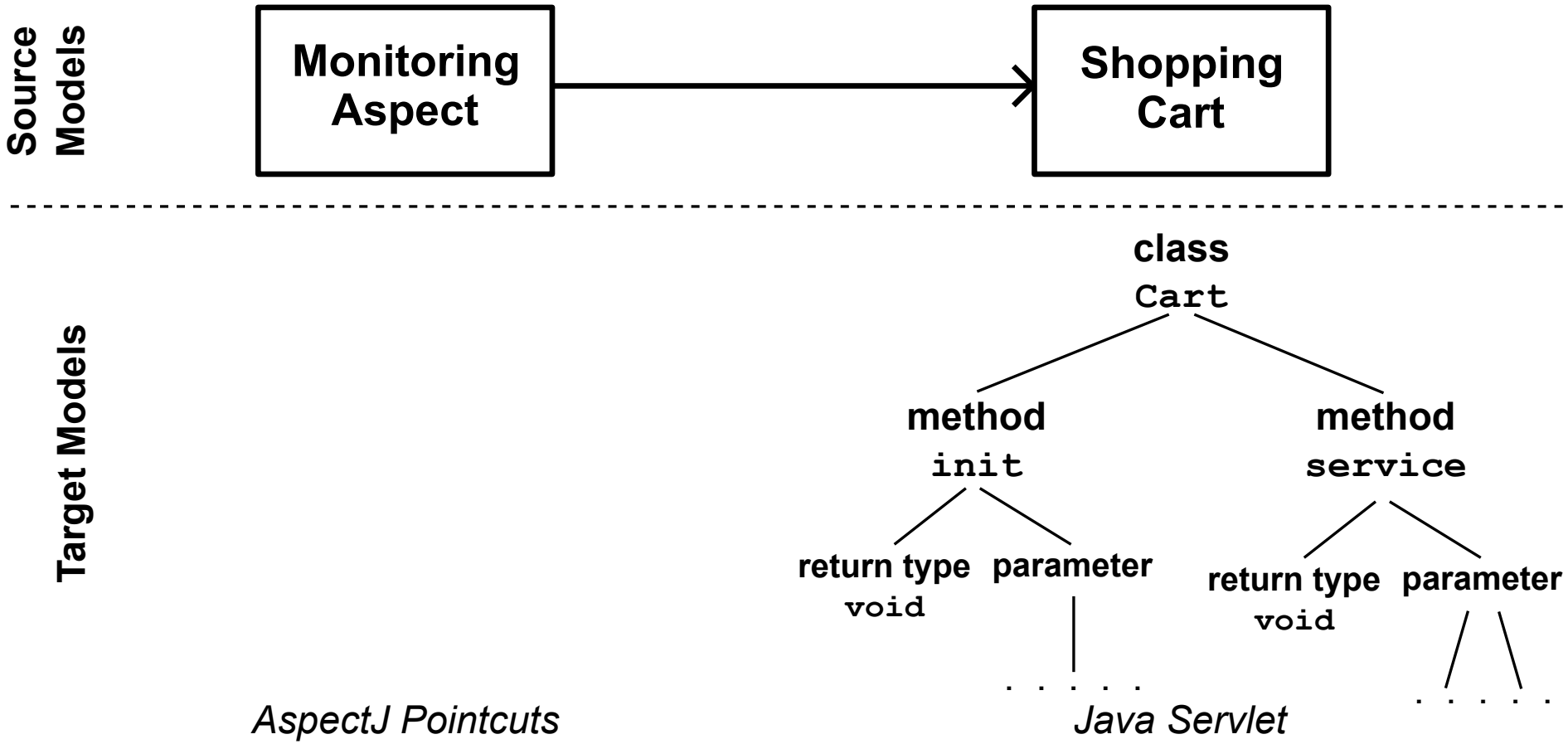
Target Models

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

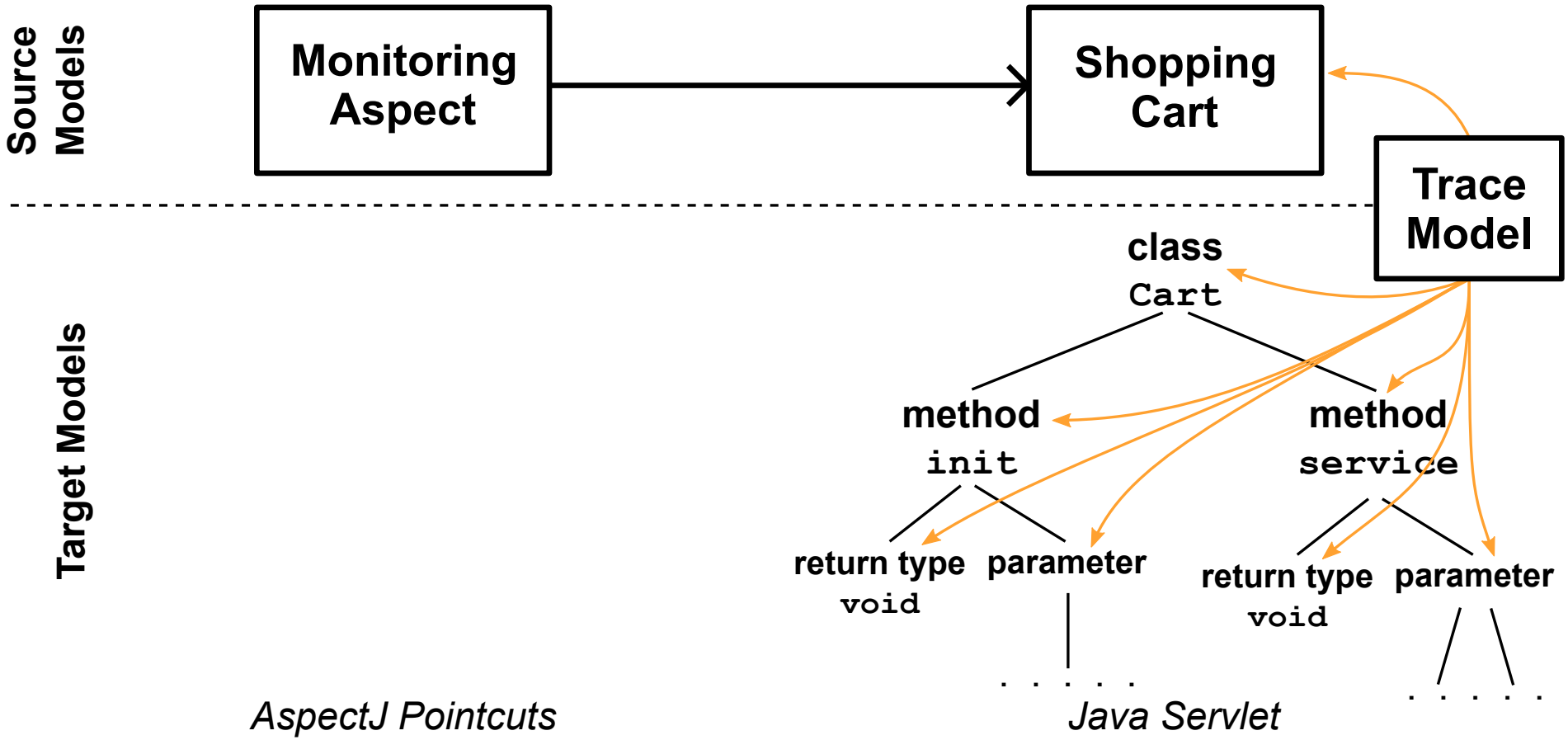
AspectJ Pointcuts

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

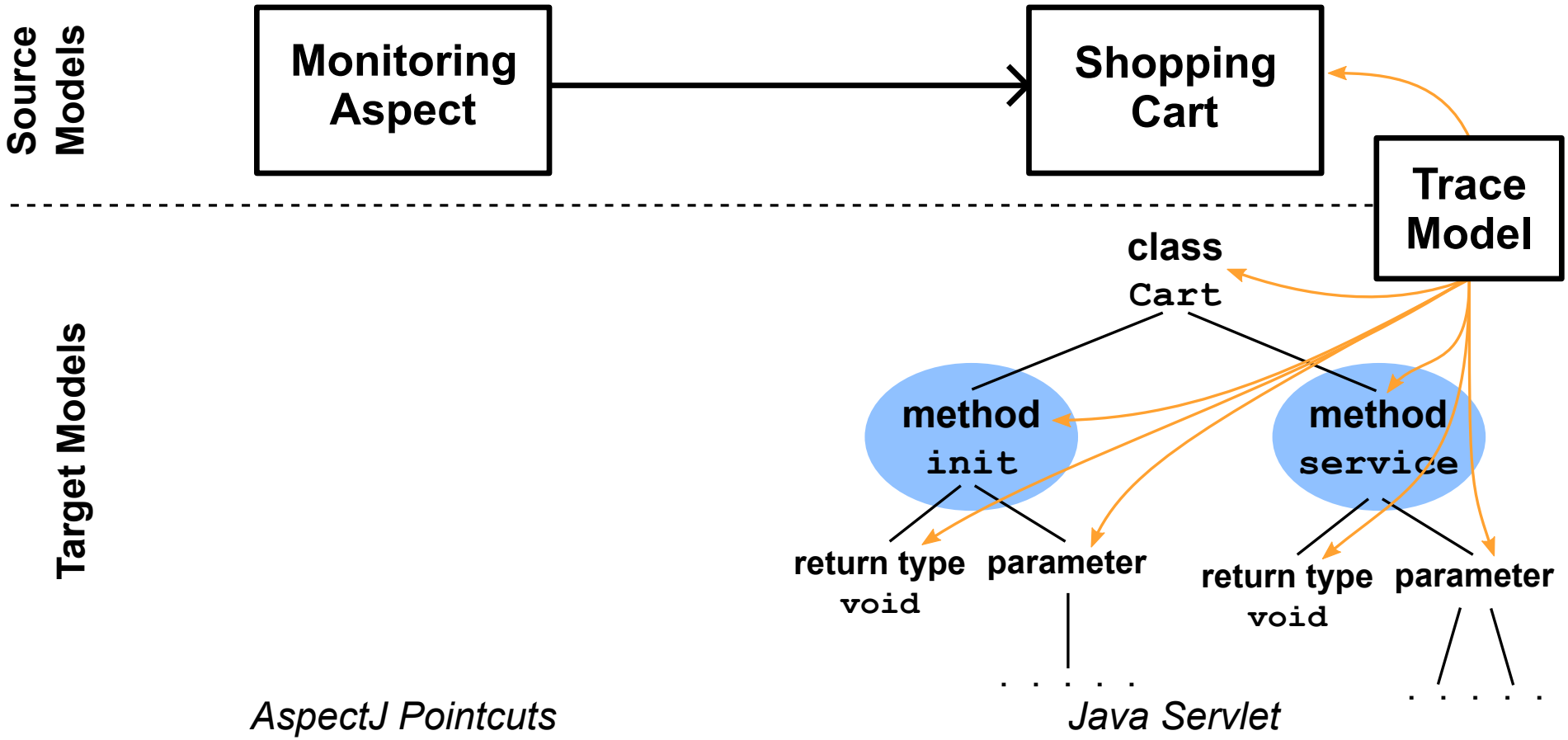
Java Servlet


Source Models
Monitoring Aspect
Shopping Cart
Target Models
class
Cart
method
init
method
service
return type
void
parameter
return type
void
parameter
. . . . .
. . . . .
AspectJ Pointcuts
Java Servlet

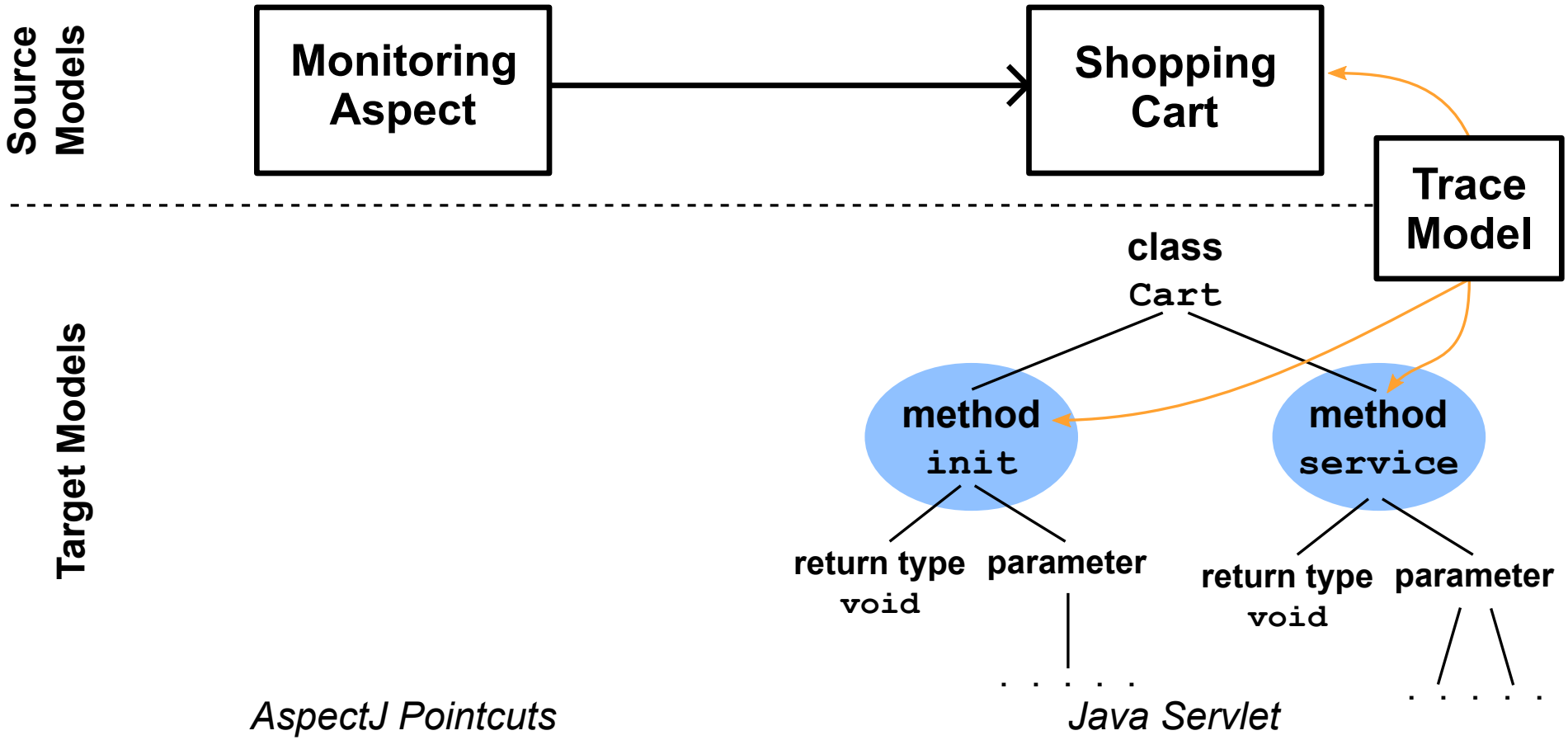
Computing Target Model Join Points



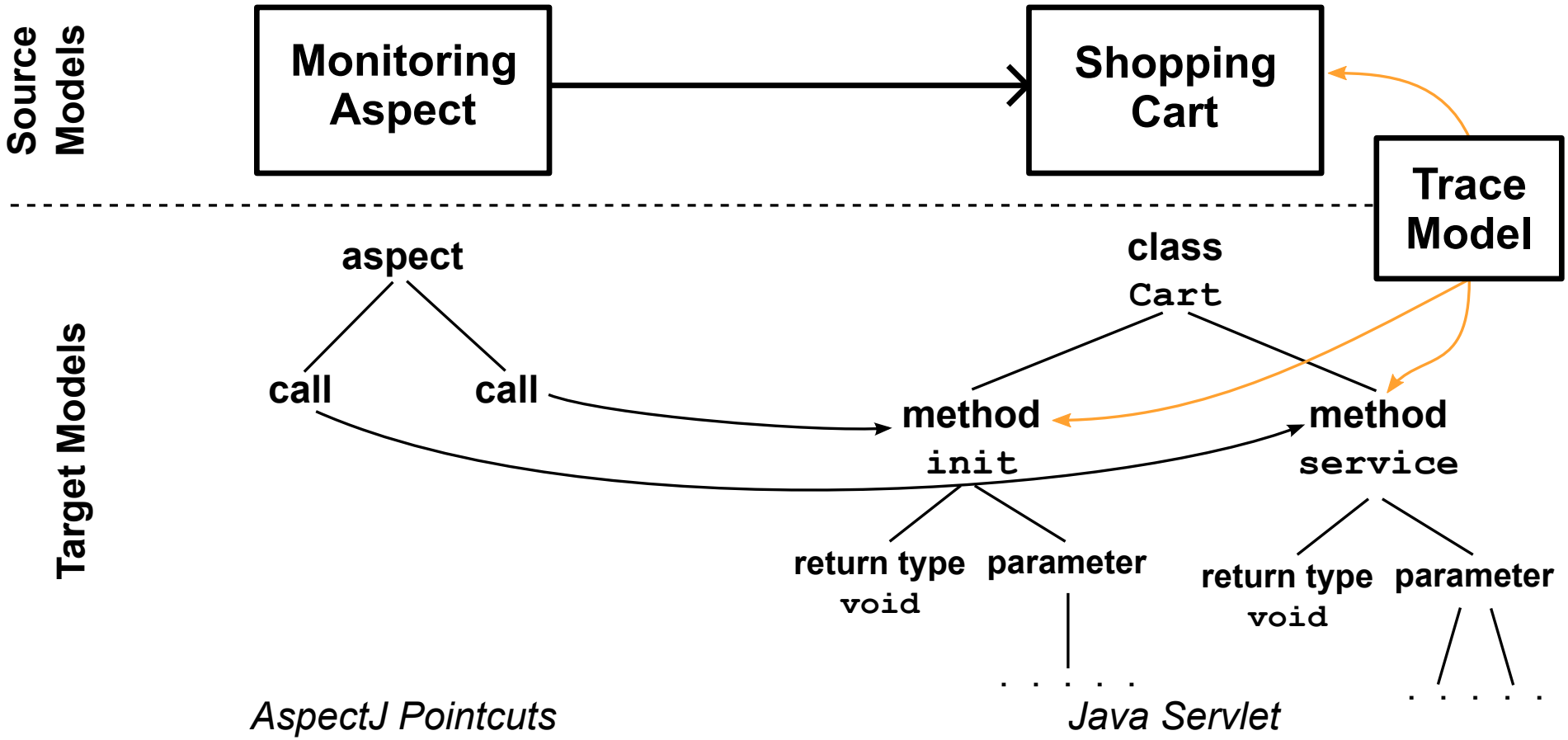
Computing Target Model Join Points



Computing Target Model Join Points



Computing Target Model Join Points



Trace recovery (cf. [Galvao and Goknil, 2007])

- Heuristic [Grammel and Kastenholz, 2010]
  - non-predictable results
- Deterministic
  - Based on node attribute matches
  - May miss relevant traces

Obtaining Model Traces

Trace recovery (cf. [Galvao and Goknil, 2007])

- Heuristic [Grammel and Kastenholz, 2010]
  - non-predictable results
- Deterministic
  - Based on node attribute matches
  - May miss relevant traces

Trace construction

- By the base model generator (fragment)
  - Cause more complex generators
  - Some transformations can be extended automatically [Jouault, 2005]
- By a separate trace model generator
  - Simpler generators
  - Support of legacy generators
  - Have to be adapted separately

Evaluation

Information Systems

- Example: Common Component Modeling Example (CoCoME)[1] [Rausch et al., 2011]
- Source meta-model: Palladio Component Model
- Target platform: Java EE
- Users: Projects of the DFG-SPP *Design for Future*[2]

Embedded Systems

- Example: Railway control center
- Source meta-model: MENGES DSL
- Target platform: PLCopen, C
- Users: Developers and engineers

[1] http://www.cocome.org/
[2] http://www.dfg-spp1593.de

Phases

- A: Technique and method evaluation
- B: Approach evaluation with information system scenario
- C: Approach evaluation with embedded systems scenario

Methods

- Controlled Experiments
- Simulation
- Expert Interviews
  - b+m Informatik, Stefan Zeug, Thomas Stahl
  - Scheidt & Bachmann, Hauke Fuhrmann

Experiment Setup

- Partial CoCoME model
- Source meta-model
  - Palladio Component Model [Becker et al., 2009]
  - Instrumentation aspect language [Jung et al., 2013a]
- Generators
  - ProtoCom for Palladio Component Model [Giacinto and Lehrig, 2013]
  - Generator for instrumentation aspect language

Objectives Proof of concept

- Evaluation of different pointcut concepts
- Evaluation of model/code weaving approach

Method

- Controlled Experiment
- Subjects: engineers and students

Experiment Setup

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

Objectives

- Integration test of method and tooling
- Practicability of approach and tooling (Java EE)

CoCoME Collaboration

- Projects of the Design for Future SPP
- iObserve
  - University of Duisburg-Essen
  - Karlsruhe Institute of Technology
  - Kiel University
- University of Stuttgart

Design For FUTURE

iObserve

EnSure

Experiment Setup

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

Objectives

- Feasibility for different target model/language domains (text, XML)
- Comparison with existing generator $\Rightarrow$ cost benefit

Method

- Simulation based on DSL versions of the original project

Collaboration

- b+m Informatik AG
- Scheidt & Bachmann GmbH

Goal A method for generator-composition for aspect-oriented DSLs

Publications

- Published
  - Type-Systems for DSLs [Jung et al., 2013b]
  - Data type language (DTL) [Jung, 2013]
  - Instrumentation aspect language [Jung et al., 2013a]
  - Design principles for meta-models [Jung et al., 2014]
- In progress
  - Survey on model join point notations
  - Evaluation of weaver tooling

Next step Realizing the first evaluation scenario

References I

S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *J. Syst. Softw.*, 82(1):3–22, Jan. 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2008.03.066. URL http://dx.doi.org/10.1016/j.jss.2008.03.066.

J. Bennett, K. Cooper, and L. Dai. Aspect-oriented model-driven skeleton code generation: A graph-based transformation approach. *Science of Computer Programming*, 75(8):689 – 725, 2010. ISSN 0167-6423. doi: http://dx.doi.org/10.1016/j.scico.2009.05.005. URL http://www.sciencedirect.com/science/article/pii/S0167642309000914. Designing high quality system/software architectures.

S. Clarke and E. Baniassad. *Aspect-Oriented Analysis and Design*. Addison-Wesley Professional, 2005. ISBN 0321246748.

M. Didonet, D. Fabro, J. Bézivin, and P. Valduriez. Weaving models with the eclipse amw plugin. In *In Eclipse Modeling Symposium, Eclipse Summit Europe*, 2006.

I. Galvao and A. Goknil. Survey of traceability approaches in model-driven engineering. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 313–313, 2007. doi: 10.1109/EDOC.2007.42.

D. Giacinto and S. Lehrig. Towards integrating java ee into protocom. In *KPDAYS*, pages 69–78, 2013.

References II

B. Grammel and S. Kastenholz. A generic traceability framework for facet-based traceability data extraction in model-driven software development. In *Proceedings of the 6th ECMFA Traceability Workshop*, pages 7–14, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-993-0. doi: 10.1145/1814392.1814394. URL http://doi.acm.org/10.1145/1814392.1814394.

S. Jörges. *Construction and Evolution of Code Generators - A Model-Driven and Service-Oriented Approach*, volume 7747 of *LNCS*. Springer, 2013. ISBN 978-3-642-36126-5.

F. Jouault. Loosely coupled traceability for atl. In *In Proceedings of the European Conference on Model Driven Architecture workshop on traceability*, pages 29–37, 2005.

R. Jung. Data type language, November 2013. URL http://www.dfg-spp1593.de/wiki/index.php/Model-driven_development_of_CoCoME.

R. Jung, R. Heinrich, and E. Schmieders. Model-driven instrumentation with kieker and palladio to forecast dynamic applications. In *KPDAYS*, pages 99–108, 2013a.

R. Jung, C. Schneider, and W. Hasselbring. Type systems for domain-specific languages. In S. Wagner and H. Lichter, editors, *Software Engineering (Workshops)*, volume 215 of *LNI*, pages 139–154. GI, 2013b. ISBN 978-3-88579-609-1.

R. Jung, R. Heinrich, E. Schmieders, M. Strittmatter, and W. Hasselbring. 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, pages 19:19–19:22, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2900-2. doi: 10.1145/2631675.2631681. URL http://doi.acm.org/10.1145/2631675.2631681.

L. Kapova, T. Goldschmidt, J. Happe, and R. H. Reussner. Domain-specific templates for refinement transformations. In *MDI '10: Proceedings of the First International Workshop on Model-Driven Interoperability*, pages 69–78, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0292-0. doi: http://doi.acm.org/10.1145/1866272.1866282.

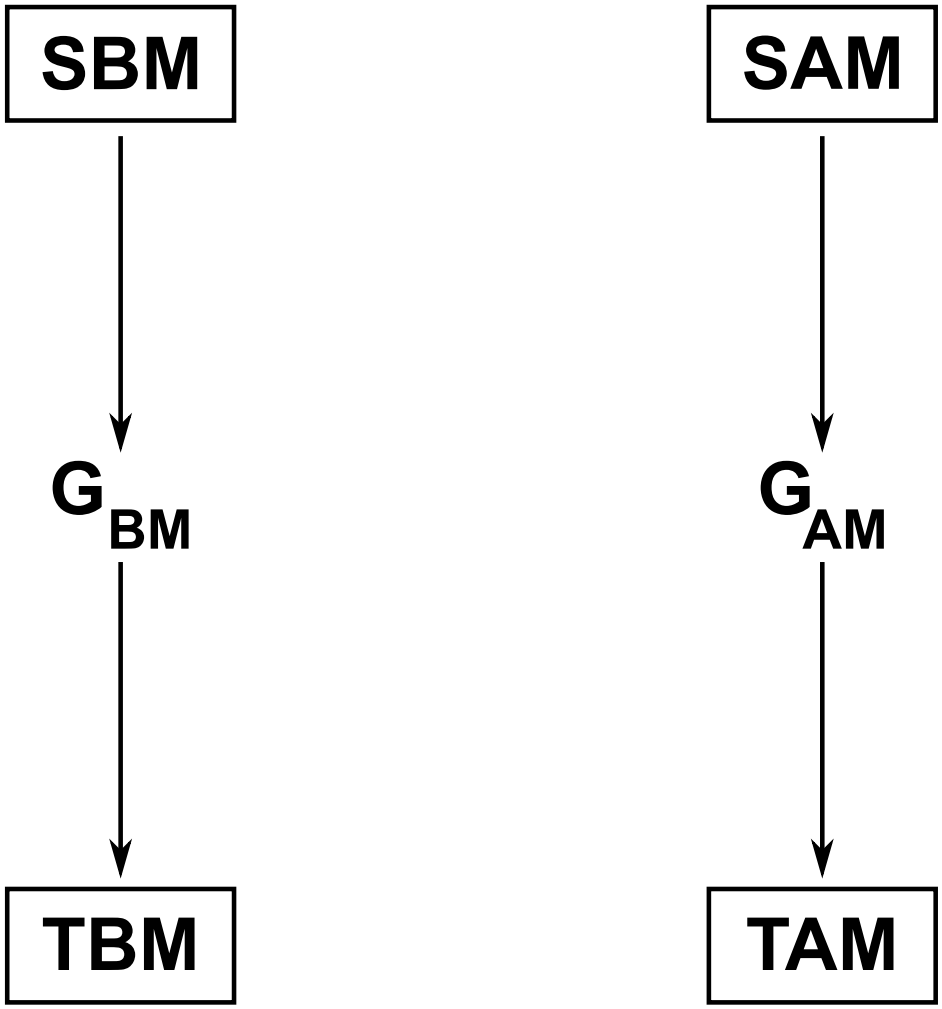
J. Klein and J. Kienzle. Reusable aspect models. In *11th Workshop on Aspect-Oriented Modeling, AOM at Models'07,*, 2007.

A. Mehmood and D. N. Jawawi. Aspect-oriented model-driven code generation: A systematic mapping study. *Information and Software Technology*, 55(2):395 – 411, 2013. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2012.09.003. URL http://www.sciencedirect.com/science/article/pii/S0950584912001863. Special Section: Component-Based Software Engineering (CBSE), 2011.

B. Morin, J. Klein, O. Barais, and J.-M. Jézéquel. A generic weaver for supporting product lines. In *Proceedings of the 13th International Workshop on Early Aspects*, EA '08, pages 11–18, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-032-6. doi: 10.1145/1370828.1370832. URL http://doi.acm.org/10.1145/1370828.1370832.

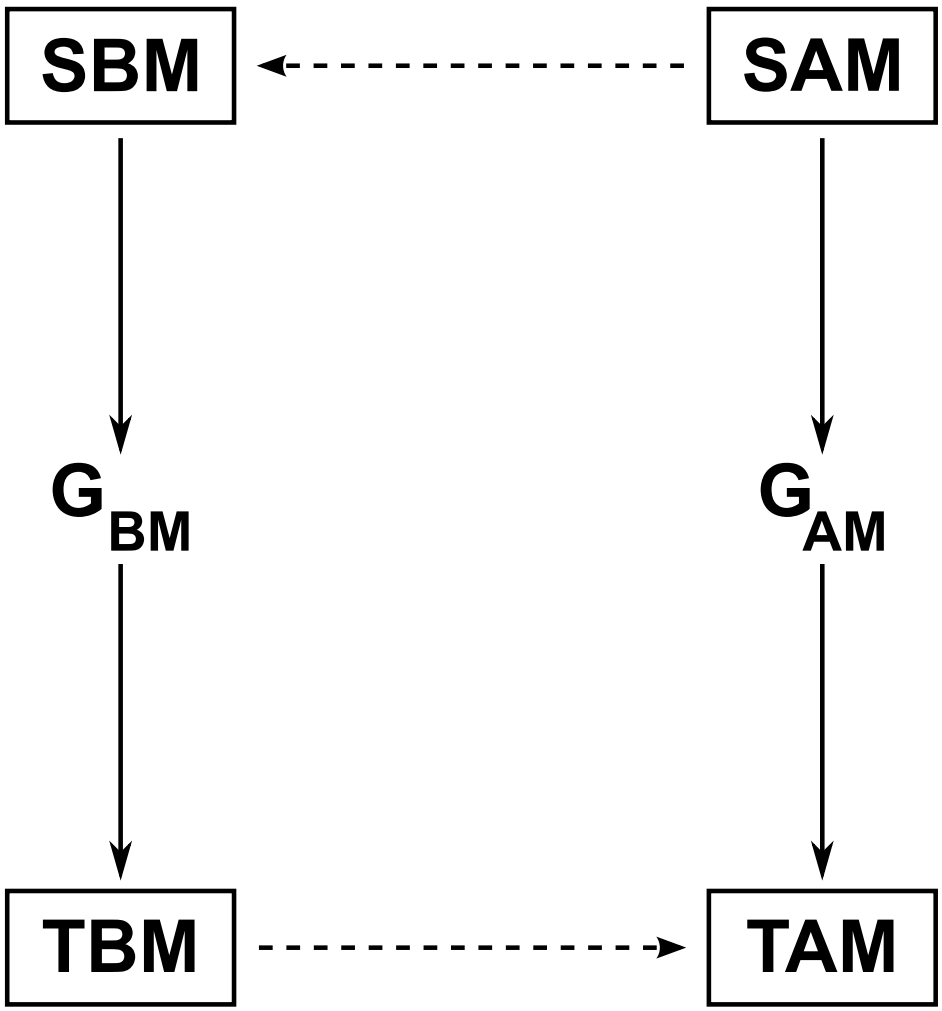
A. Rausch, R. Reussner, R. Mirandola, and F. Plasil, editors. *The Common Component Modelling Example (CoCoME)*, volume 5153 of *Lecture Notes in Computer Science*. Springer Verlag Berlin Heidelberg, 2011.

Style 3: Inverted References



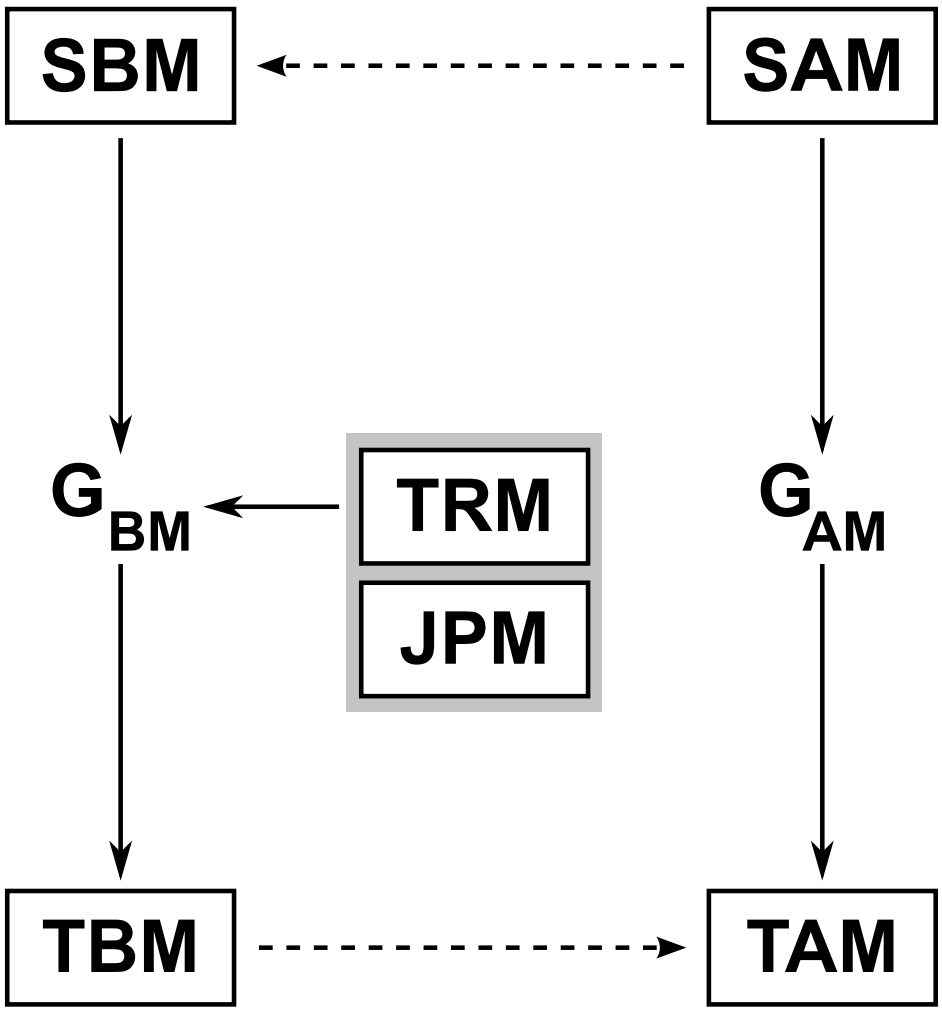
SBM	Source Base Meta-Model
SAM	Source Aspect Meta-Model
TBM	Target Base Meta-Model
TAM	Target Aspect Meta-Model
TRM	Trace Meta-Model
JPM	Join Point Meta-Model

Style 3: Inverted References



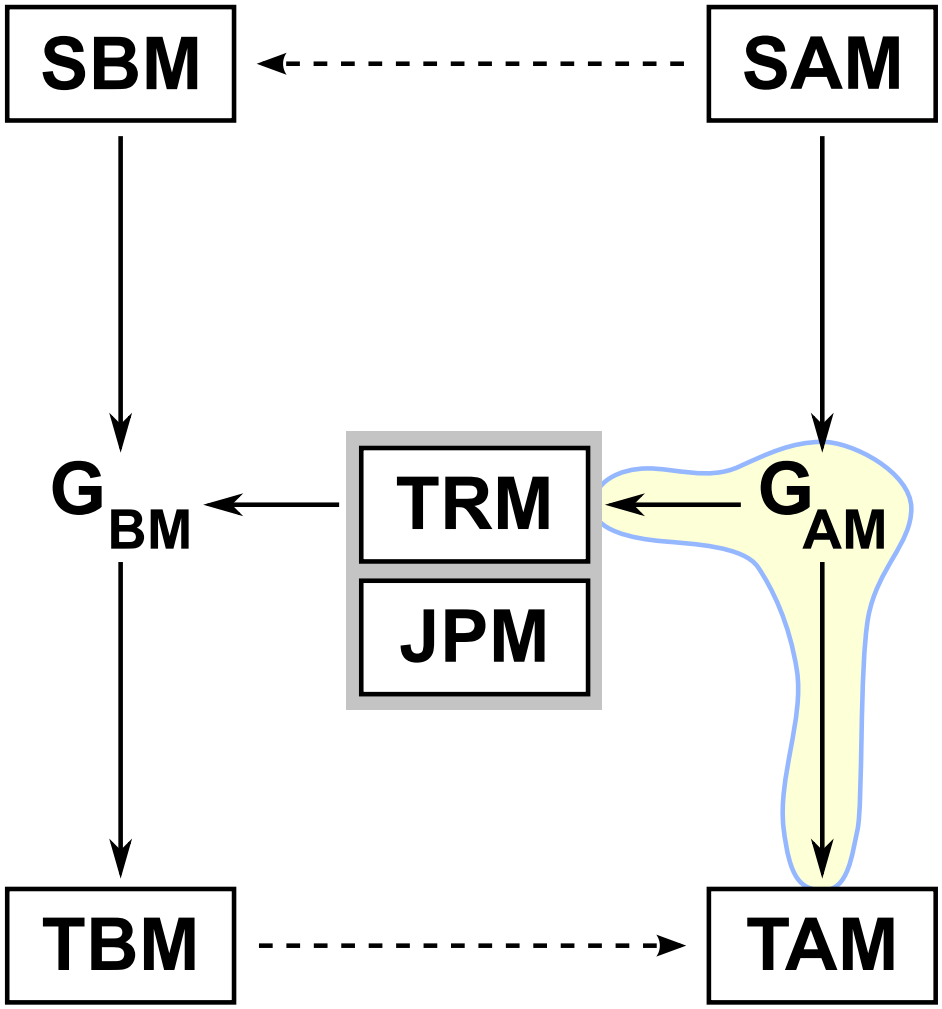
SBM	Source Base Meta-Model
SAM	Source Aspect Meta-Model
TBM	Target Base Meta-Model
TAM	Target Aspect Meta-Model
TRM	Trace Meta-Model
JPM	Join Point Meta-Model

Style 3: Inverted References



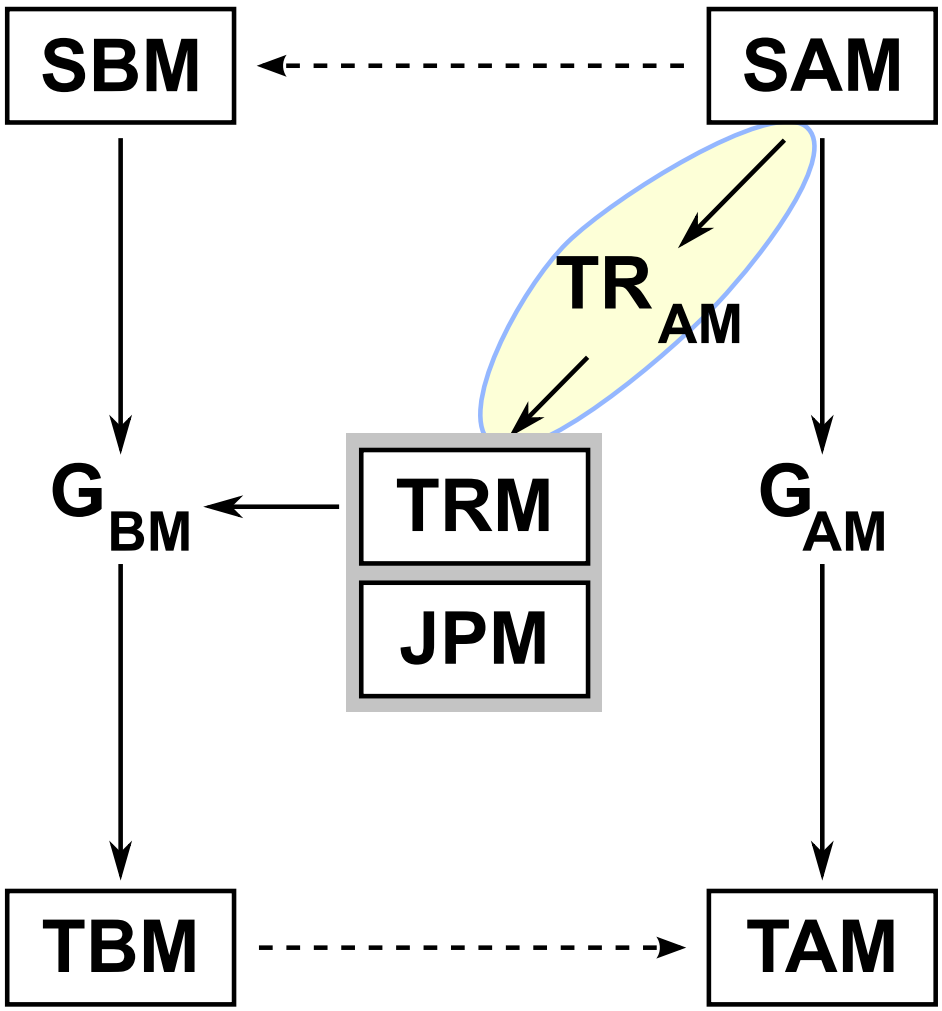
SBM	Source Base Meta-Model
SAM	Source Aspect Meta-Model
TBM	Target Base Meta-Model
TAM	Target Aspect Meta-Model
TRM	Trace Meta-Model
JPM	Join Point Meta-Model

Style 3: Inverted References



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

Style 3: Inverted References



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