TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

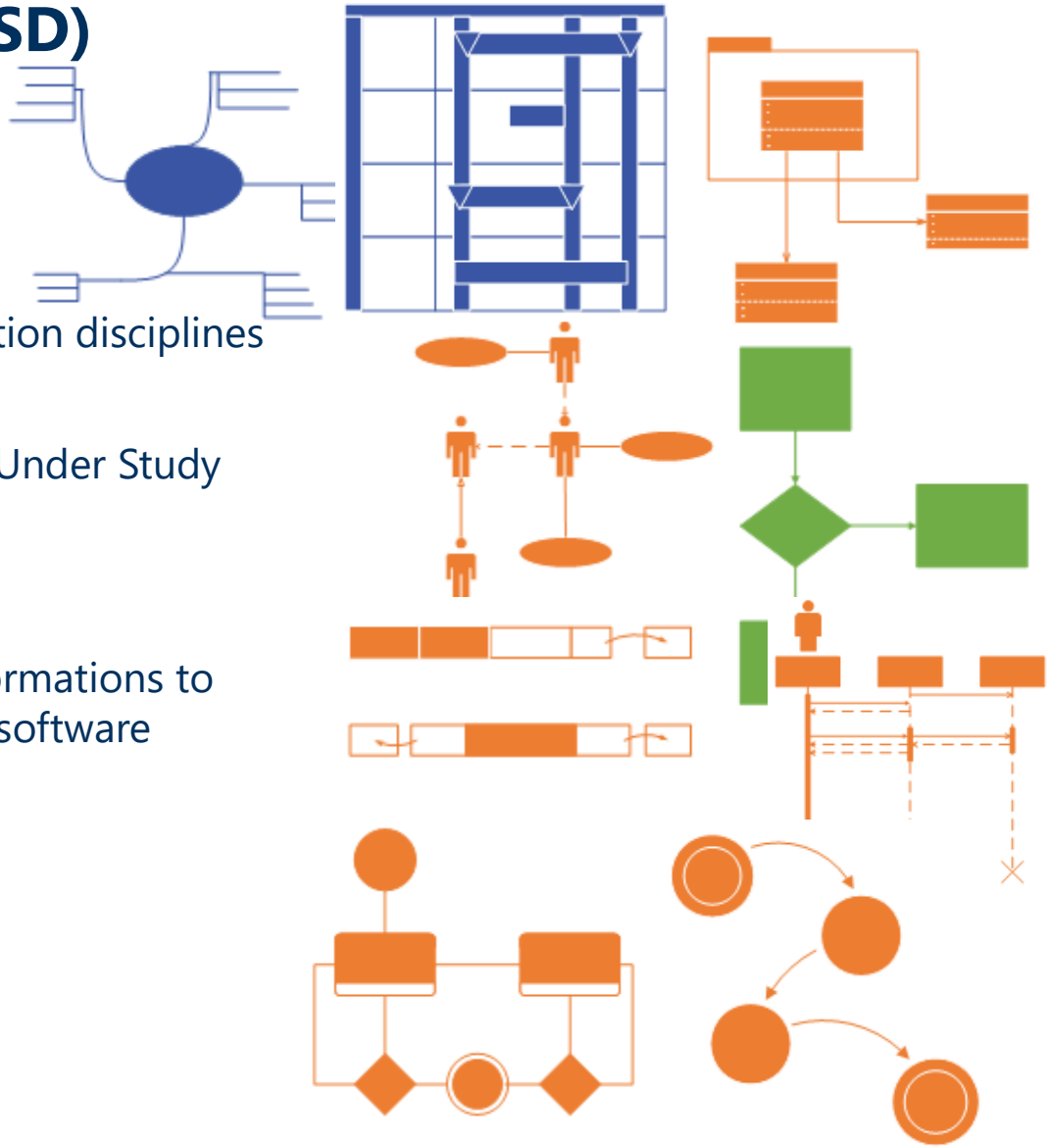
Christopher Werner
Institute of Software and Multimedia Technology, Software Technology Group

# Role-Modeling in Round-Trip Engineering for Megamodels

Supervisor: Prof. Aßmann
Co-supervisor: Prof. Strahringer

Lichtenwalde, 30th November 2018

# Model Driven Software Development (MDSD)



MDSD approaches *[Silva2015]*

- Focus on the requirements, analysis and design, and implementation disciplines (central development artifacts)
- Concrete ones define modeling languages to specify the System Under Study (SUS) at different levels of abstraction
- Problem: Inconsistency
- Provide model-to-model (M2M) and model-to-text (M2T) transformations to improve the productivity and quality of the process and the final software system

# Topic Definition

“Role-Modeling in Round-Trip Engineering for Megamodels”

# Topic Definition

"Role-Modeling in Round-Trip Engineering for Megamodels"

Round-Trip Engineering:

- "Round-Trip Engineering is the branch of software engineering that entails the design and application of techniques and tools to automatize the process of establishing global consistency across related software artifacts." *[Seifert2011]*
- Synchronizing software artefacts (e.g. models, source code, and documentation)
- Includes:
  - Forward engineering (creating software from specifications)
  - Reverse engineering (creating specifications from existing software)

# Topic Definition

"Role-Modeling in Round-Trip Engineering for Megamodels"

Model:

- "model is a system that helps to define and to give answers of the system under study without the need to consider it directly" *[Silva2015]*

Megamodel:

- "A megamodel is a model composed of related models" *[Perovich2009]*
- Model that contains models and relations between the models (ensemble of models)
- Examples: domain class diagrams, use case diagrams, sequence diagrams, source code, requirement documents, use case descriptions etc.
- Keep consistency between the models in the megamodel

# Running Example

3 simplified metamodels:

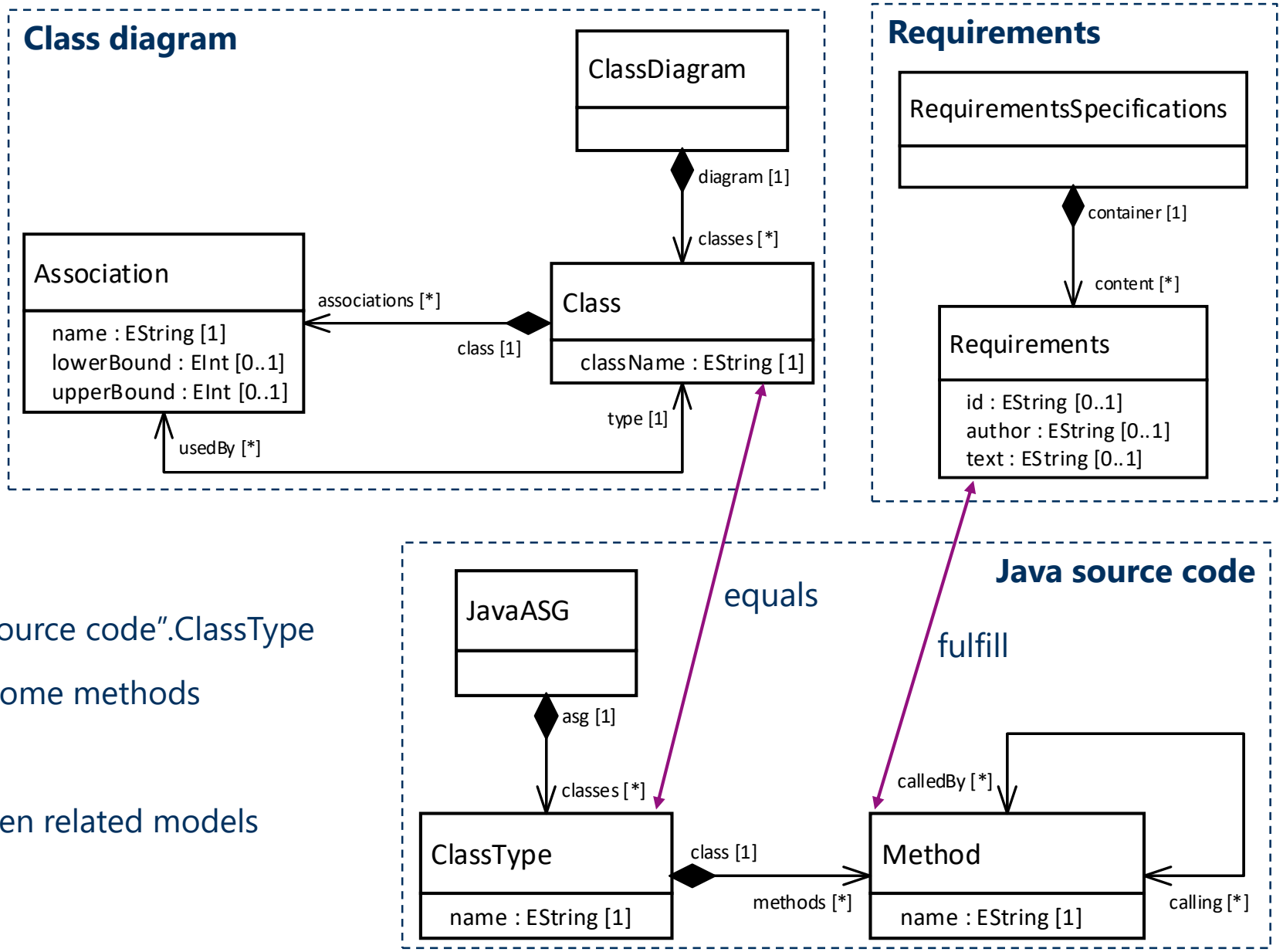
- Class diagrams metamodel
- Requirements metamodel
- Java source code metamodel

Overlaps between metamodels:

- “Class diagram”.class = “Java source code”.ClassType
- Requirements are fulfilled by some methods

Solutions:

1. Model synchronization between related models
2. Merge to one metamodel

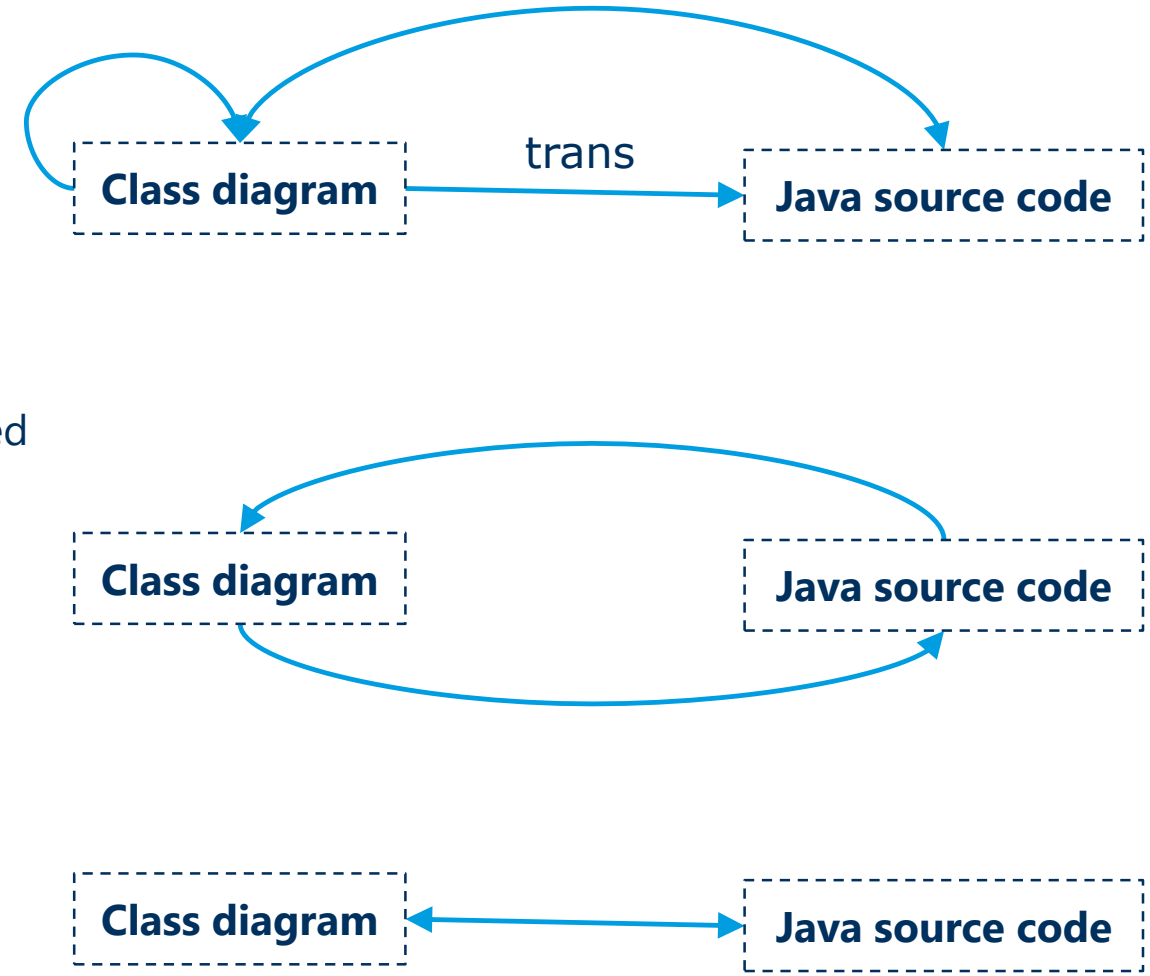


# 1. Model Synchronization

- Related models:
  - Contain redundant information
  - Independently editing leads to inconsistencies
  - Instances of different metamodels
- Solution:
  - Defining model synchronizations between interrelated models
- Unidirectional synchronization
- Bidirectional synchronization

**Transformation languages:**

- Query View Transformation (QVT), Atlas Transformation Language (ATL), Triple Graph Grammars (TGG), Lenses



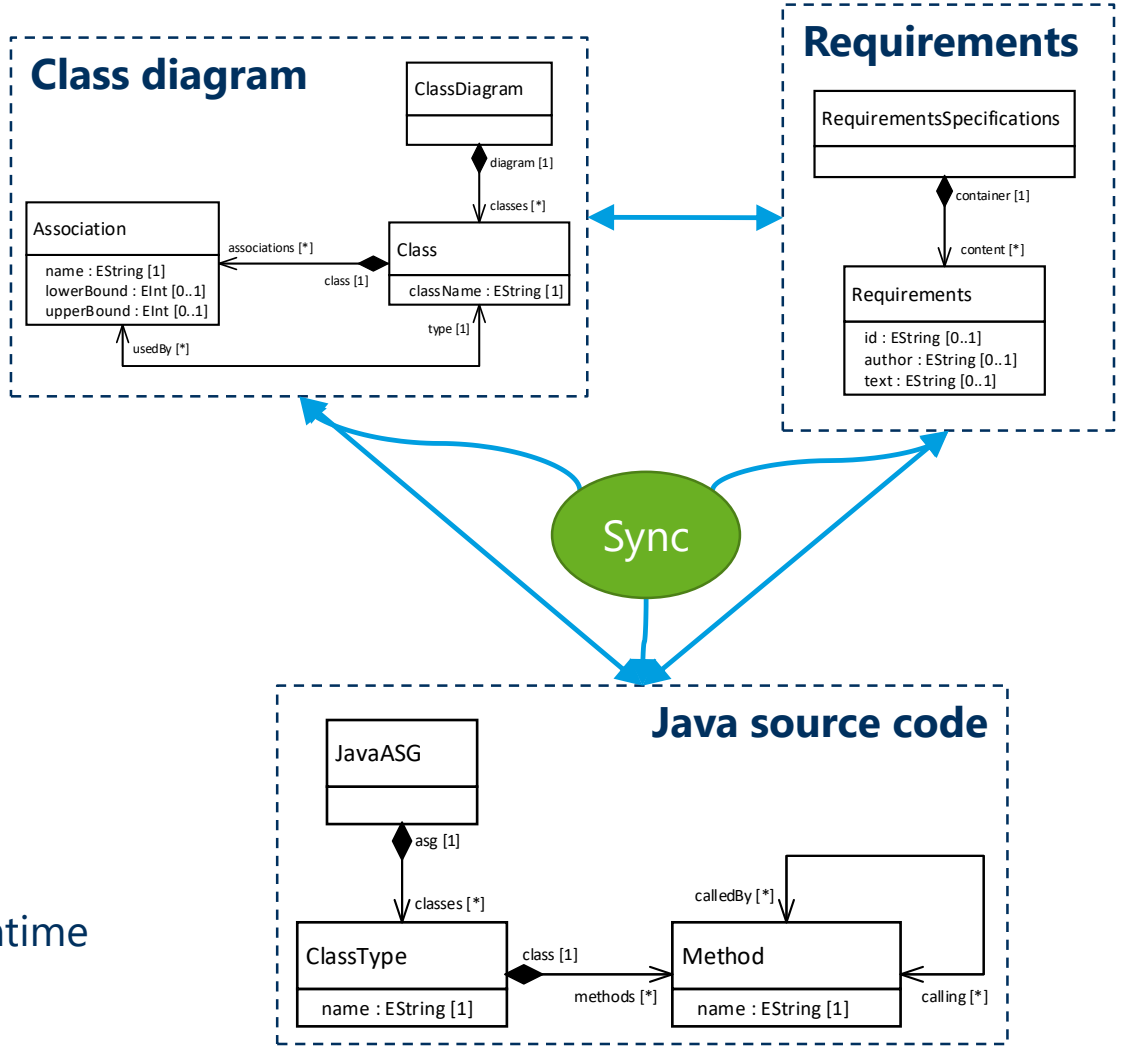
# 1. Model Synchronization between Related Models

Currently often used:

- Manually triggering of the synchronization process
- Batch updates of the whole model
- Specification of synchronization rules at design time
- Synchronization of two models
- $CC_{model\ consistency} = \frac{n \cdot (n-1)}{2}$

**Runtime** model synchronization:

- Automatically triggering of the synchronization process
- Incremental updates of small changes
- Specification and modification of synchronization rules at runtime
- Synchronization rules for multiple models



# Running Example

3 simplified metamodels:

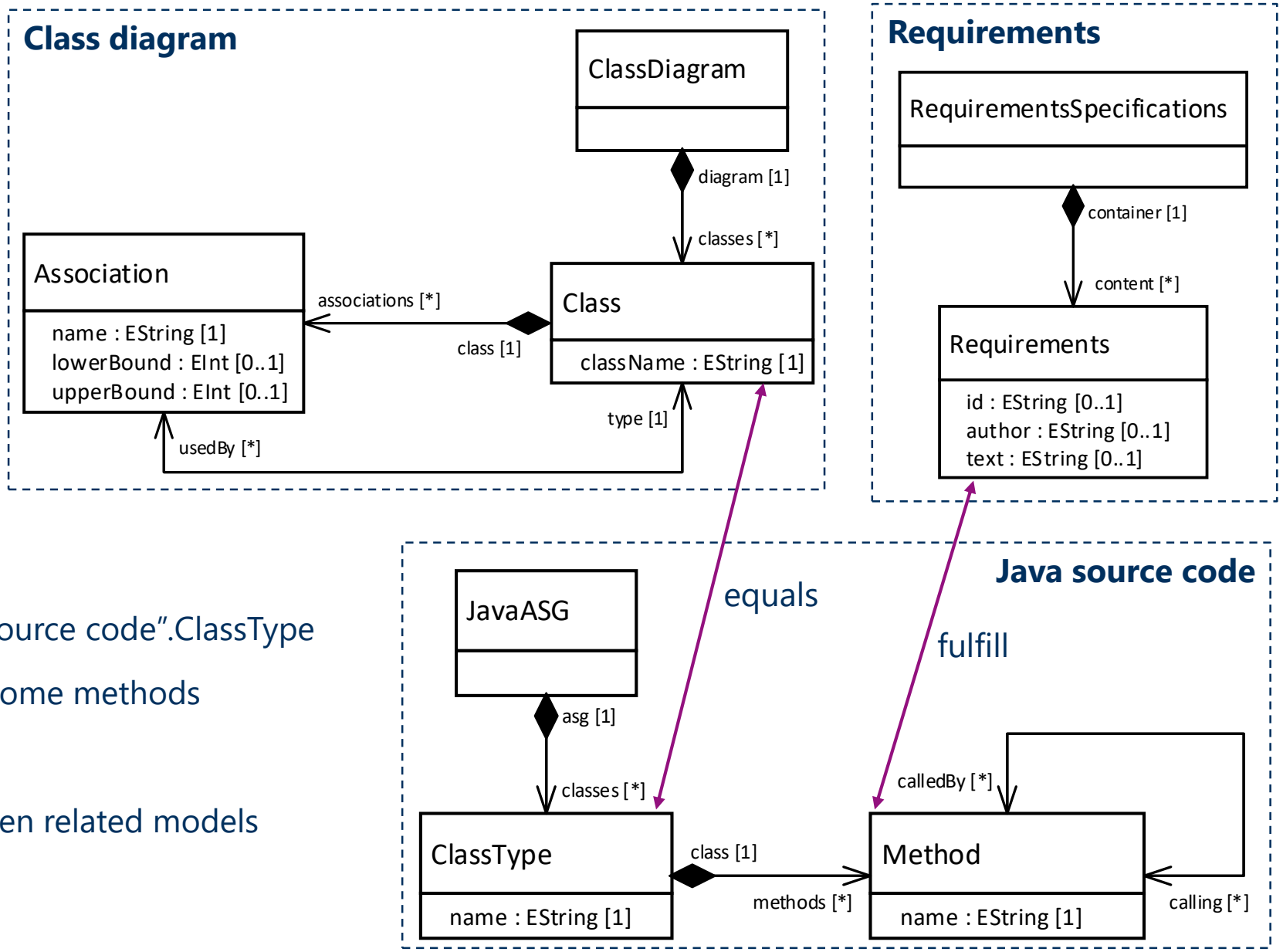
- Class diagrams metamodel
- Requirements metamodel
- Java source code metamodel

Overlaps between metamodels:

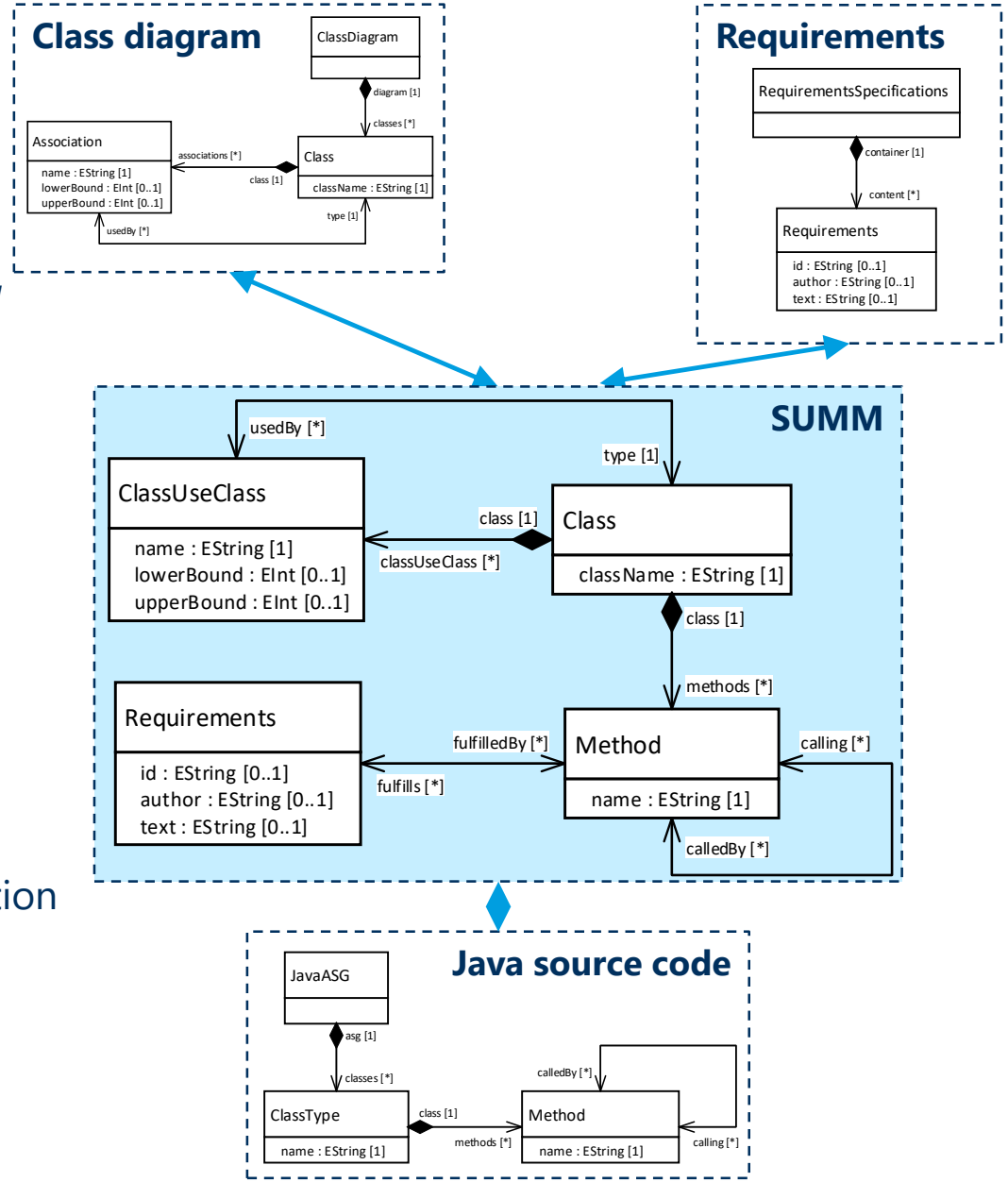
- “Class diagram”.class = “Java source code”.ClassType
- Requirements are fulfilled by some methods

Solutions:

1. Model synchronization between related models
2. Merge to one metamodel



# 2. Merge to one Metamodel



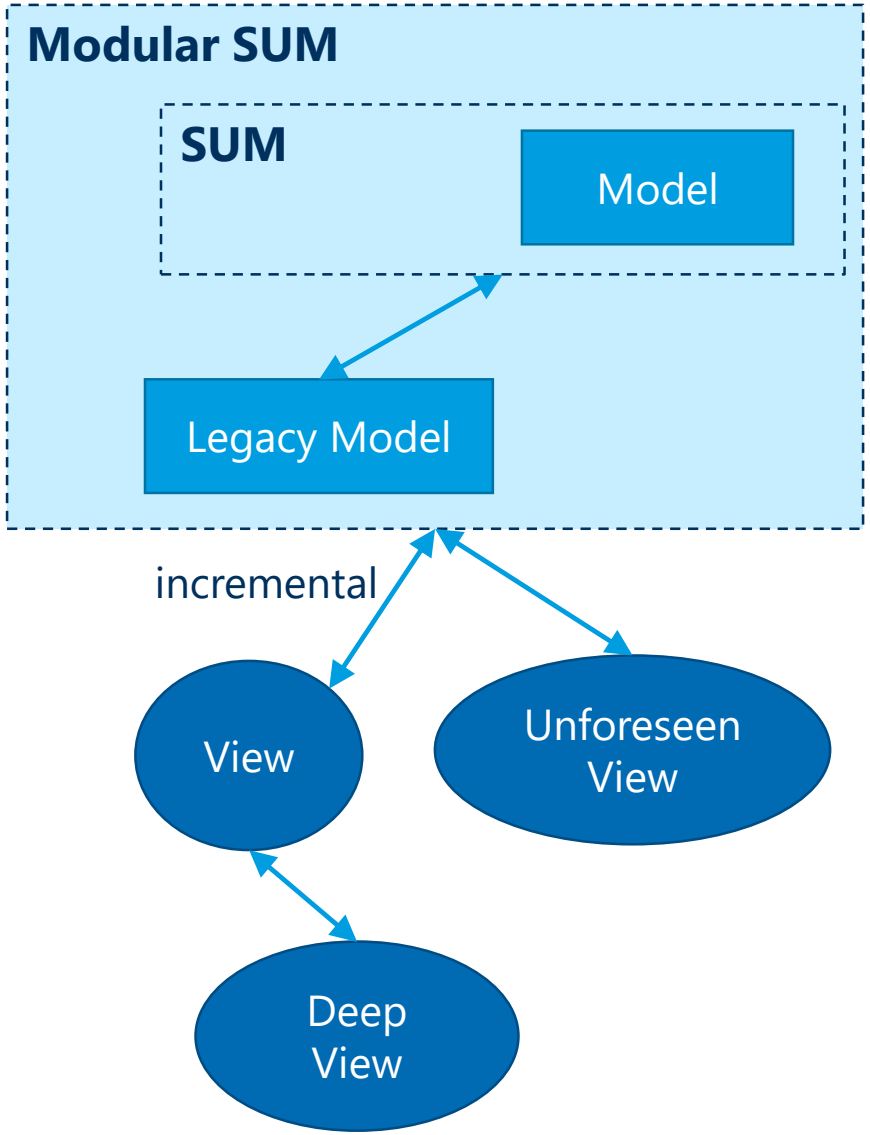

- Using a Single Underlying Model (SUM) approach *[Atkinson2010]* depending on one Single Underlying Metamodel (SUMM)
- Storing all known information in one SUM
- Consistent from scratch without redundancy
- Individual metamodels are viewtypes from the SUMM
- Individual models are views from the SUM
- m: number of views
- $cc_{SUM\ approach} = m$
- Manually created SUM or model operators for SUM construction

Problems:

- REDO construction process after metamodel evolution
- Complexity of the single underlying model

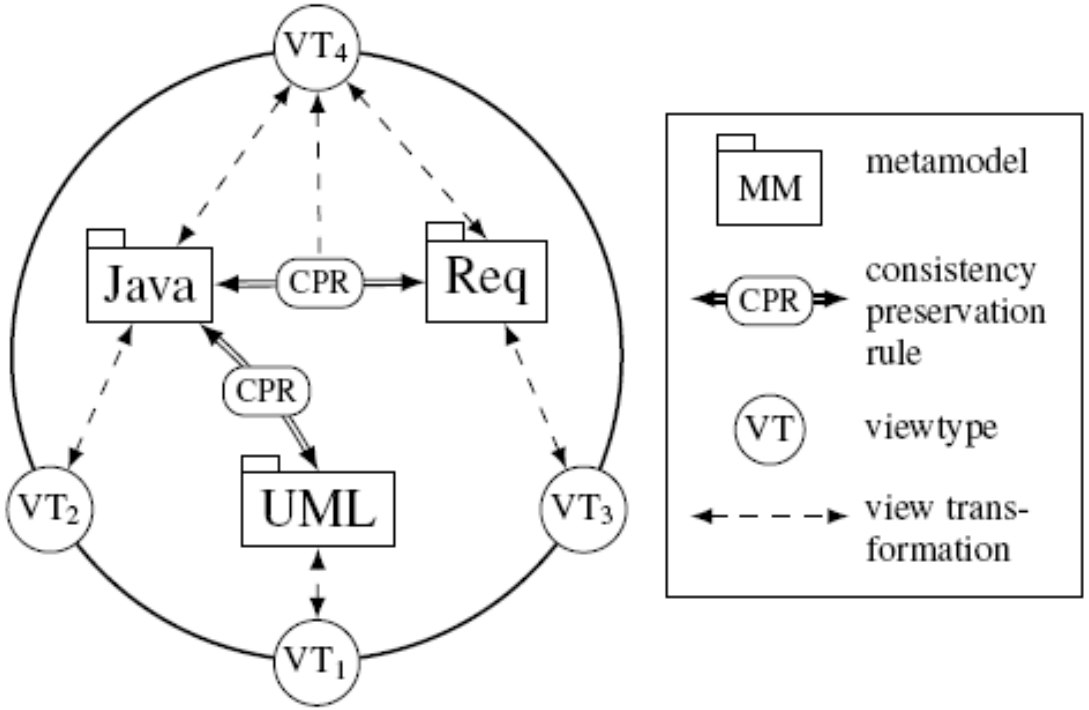
# Requirements

1. Model-view approach
   - Incremental & immediate synchronization between views and SUM
   - Using multiple (meta)models
   - Creation of unforeseen views
   - Construction of deep views
2. Model synchronization approach
   - Automatic triggering of the synchronization process
   - Incremental runtime updates
   - Specification and modification of synchronization rules at runtime
   - Synchronization rules for multiple models
3. Megamodel approach
   - Extensibility of the SUM
   - Integration of legacy models into the SUM
   - Model management



# Vitruvius (Model Synchronization)

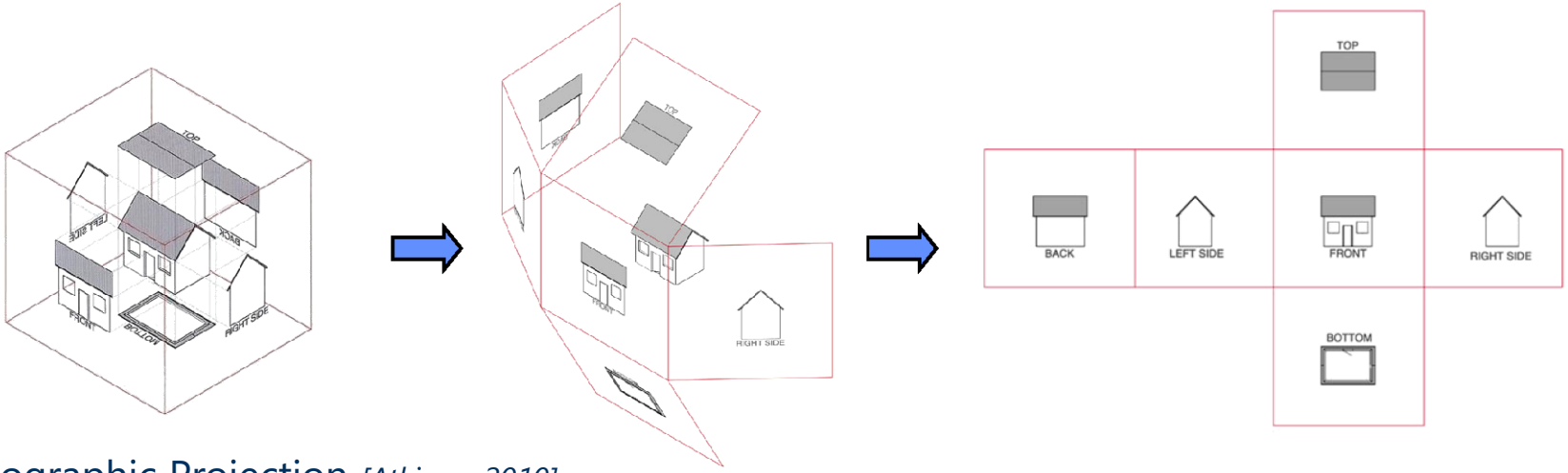
- Based on modular SUM concept that combinates several legacy metamodels
- Assume the existence of reusable metamodels for integration into the Virtual Single Underlying Metamodel (V-SUMM)
- Viewtypes are the only way to access and manipulate information
- Sub models and views are synchronized over bidirectional consistency transformations
- Bottom-up approach for the SUM construction process
- Advantages: Reusability and evolvability




Architecture of Vitruvius *[Kramer2013]*

# Orthographic Software Modeling (OSM) (Model Merge)

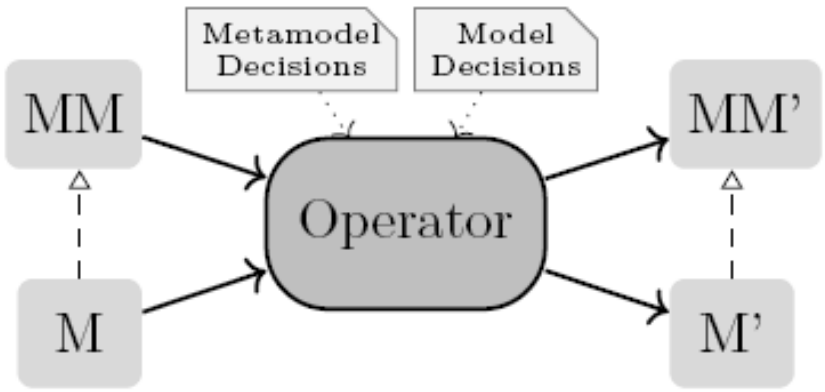
- Implementation of Single Underlying Model (SUM) approach
- Enforce “orthogonal” views with minimal overlap
- Data is free from dependencies and capture all relations between its inner elements in a redundancy-free way
- Top-down approach for the SUM construction process with problems in reusability of (meta)models



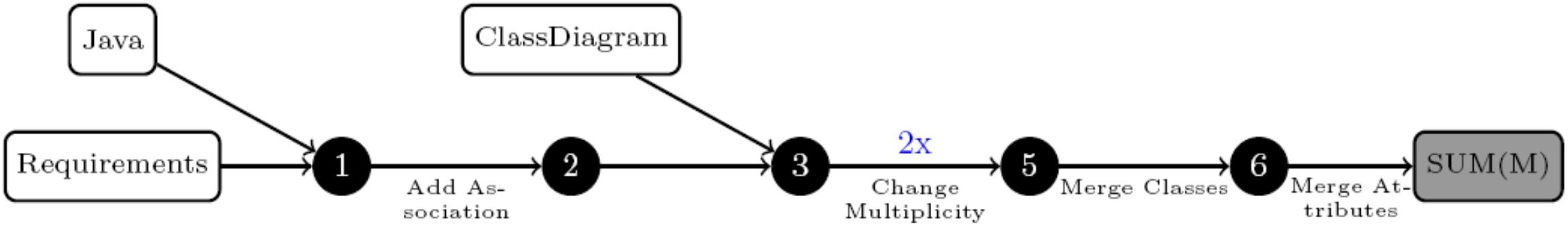

Orthographic Projection *[Atkinson2010]*

# MOdel CONSistency Ensured by Metamodel Integration (MoConseMI) (Model Merge)

- Combine major features of OSM and Vitruvius
- Creating one SUM by operator-based transformations
- Reusing existing (meta)models
- Chain of operators that changes the current meta(model) in a stepwise way
- Solve evolution problems with iterating back and front the chain



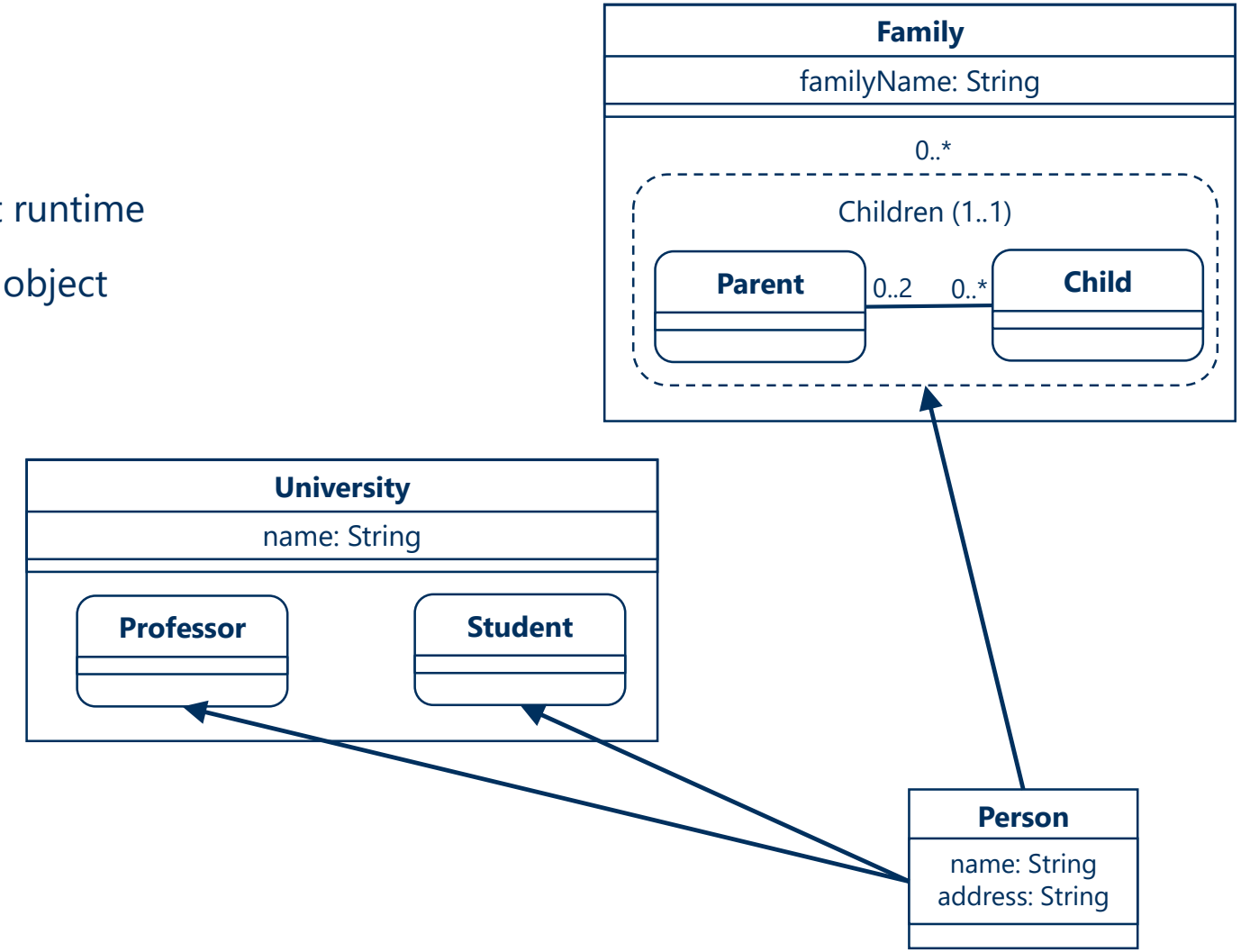

Signature of Operators *[Meier2018]*




Chain of configured Operators for Source code, Requirements, and Class Diagram *[Meier2018]*

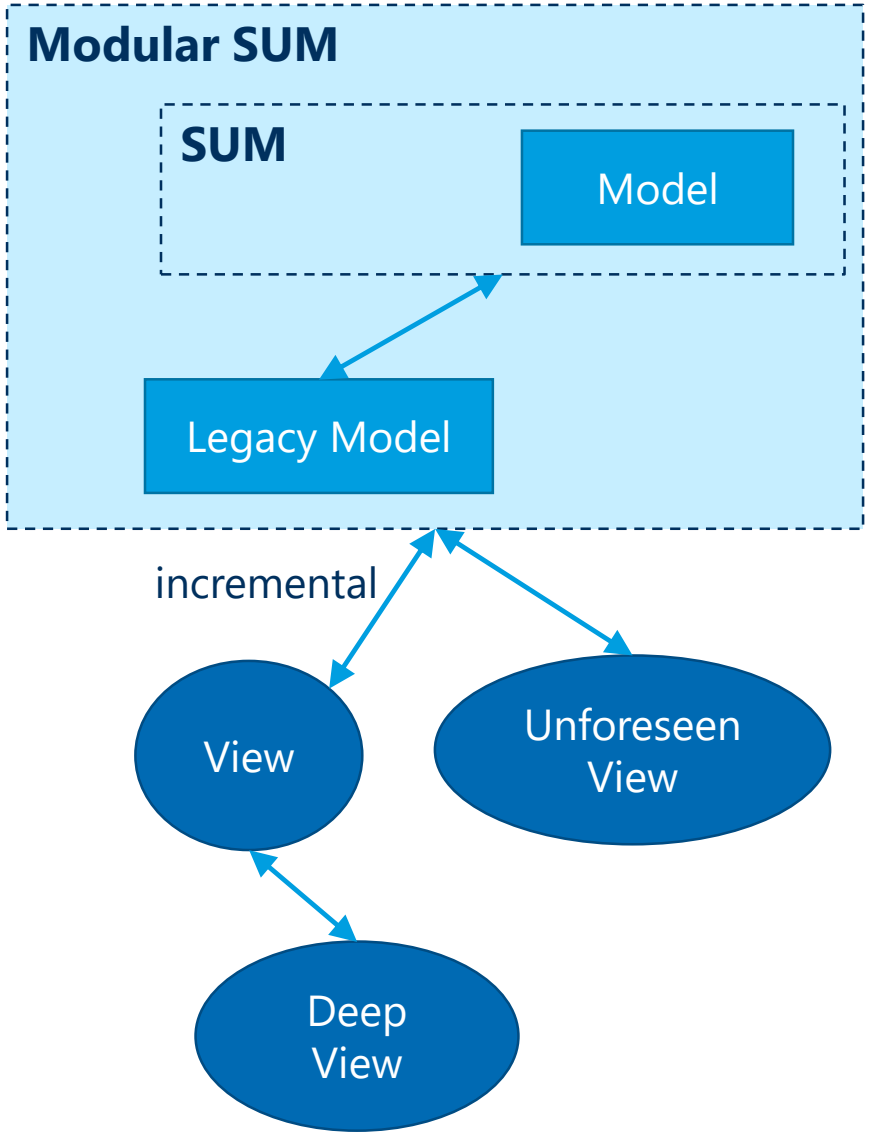
# Using Roles in the Area of Model Driven Software Development

- Separation of concerns
- Dynamic acquiring and abandoning of roles at runtime
- Natural and bounded roles form a compound object
- Roles and naturals share identity
- Naturals can play as much roles as necessary
- Loading naturals, roles, and compartments at runtime
- Manage access to methods and attributes



# Requirements

1. Model-view approach
   - Incremental & immediate synchronization between views and SUM
   - Using multiple (meta)models
   - Creation of unforeseen views
   - Construction of deep views
2. Model synchronization approach
   - Automatic triggering of the synchronization process
   - Incremental runtime updates
   - Specification and modification of synchronization rules at runtime
   - Synchronization rules for multiple models
3. Megamodel approach
   - Extensibility of the SUM
   - Integration of legacy models into the SUM
   - Model management



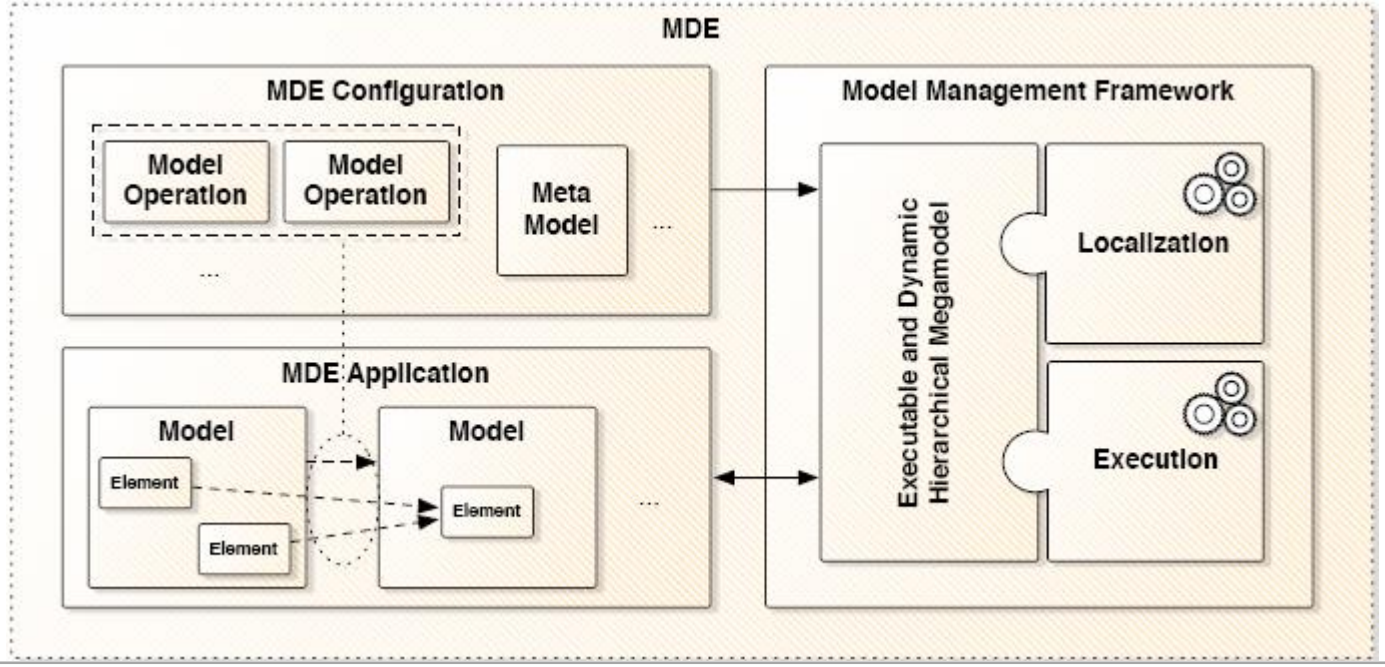
# Megamodel Approaches

Complexity of a running software cannot be captured by a runtime model with a fixed metamodel

Dissertations:

- Traceability and Model Management with Executable and Dynamic Hierarchical Megamodels *[Seibel2012]*
- Model-Driven Engineering of Self-Adaptive Software *[Vogel2018]* about the ExecUtable RuntimE MegAmodels (EUREMA)

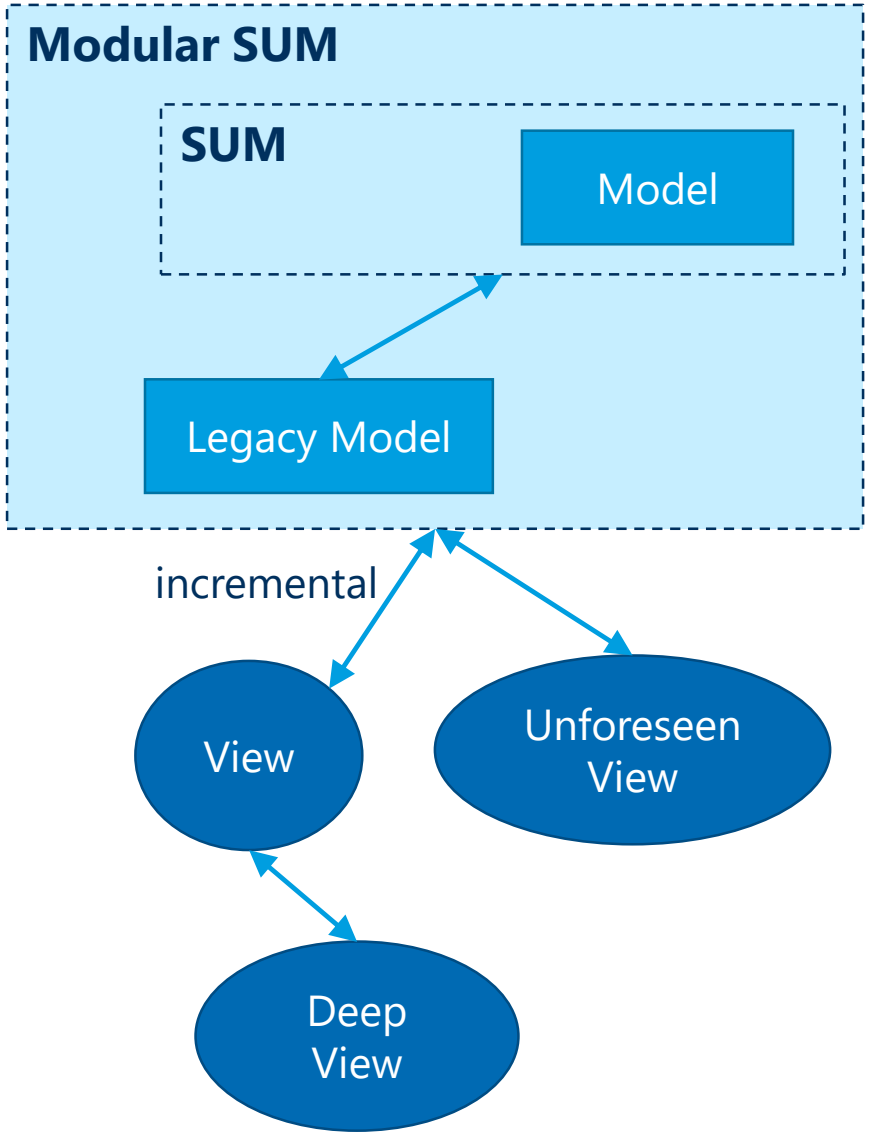
- Traceability, Localization, and Execution Functionality in the megamodel
- Managing the runtime models for autonomic systems in a megamodel




*[Seibel2012]*

# Requirements

1. Model-view approach
   - Incremental & immediate synchronization between views and SUM
   - Using multiple (meta)models
   - Creation of unforeseen views
   - Construction of deep views
2. Model synchronization approach
   - Automatic triggering of the synchronization process
   - Incremental runtime updates
   - Specification and modification of synchronization rules at runtime
   - Synchronization rules for multiple models
3. Megamodel approach
   - Extensibility of the SUM
   - Integration of legacy models into the SUM
   - Model management



# Model Synchronization *[Kahani2018]*

| | QVTo-Eclipse [Gerking2014] | Umple [Forward2009] | Epsilon [Kolovos2008] | Xtend [Xtend] | Reactive ATL *[Martinez2017]* | VIATRA3 *[Varró2016]* | TGGs *[Trollmann 2016]* |
|---|---|---|---|---|---|---|---|
| Incremental updates | ○ | ● | ● | ● | ● | ● | ● |
| Automatic updates | ◑ | ◑ | ◑ | ◑ | ◑ | ● | ● |
| Transformation direction | → | ↑↓←→ | → | → | → | → | ←→ |
| Runtime rule changes | ○ | ○ | ● | ○ | ● | ○ | ○ |
| Synchronization between multiple models | ○ | ● | ○ | ○ | ○ | ● | ● |

● Supported ◑ Semi supported ○ Not supported → Unidirectional ←→ Bidirectional ↑↓←→ Multidirectional

# Model Synchronization

**Reactive ATL** *[Martinez2017]*

- Successor of *Incremental ATL* and *Lazy ATL*
- Meta-modeling framework extended to support change propagation for individual properties of the source model (incremental ATL)
- Target model elements are only computed when they are explicitly requested (lazy ATL)

**VIATRA3** *[Varró2016]*

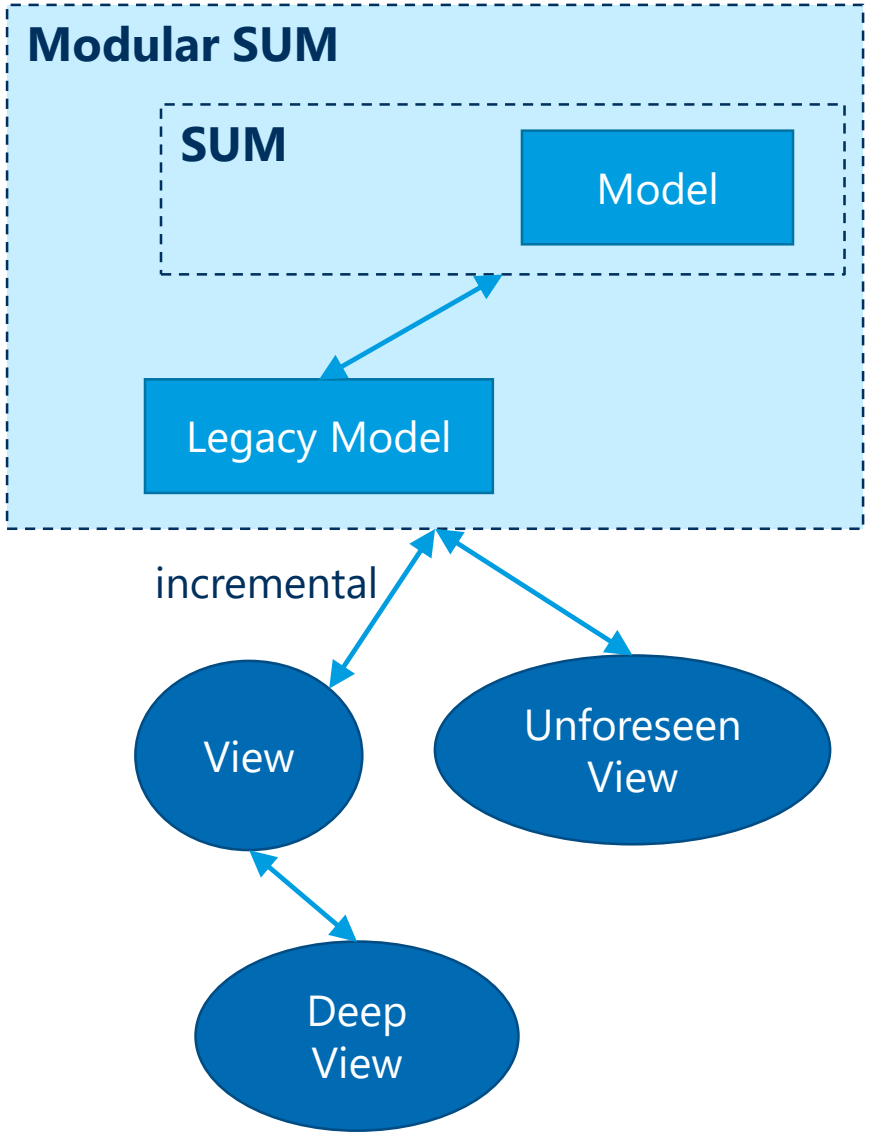
- Based on graph transformations
- Follows the reactive programming paradigm
- Aims at high scalability (suitable for very large models)

**TGGs** *[Trollmann2016, Giese2006, Vogel2010]*

- Used in alternative approaches to model synchronization
- Synchronize and generate models with triple graph grammars

# Requirements

1. Model-view approach
   - Incremental & immediate synchronization between views and SUM
   - Using multiple (meta)models
   - Creation of unforeseen views
   - Construction of deep views
2. Model synchronization approach
   - Automatic triggering of the synchronization process
   - Incremental runtime updates
   - Specification and modification of synchronization rules at runtime
   - Synchronization rules for multiple models
3. Megamodel approach
   - Extensibility of the SUM
   - Integration of legacy models into the SUM
   - Model management

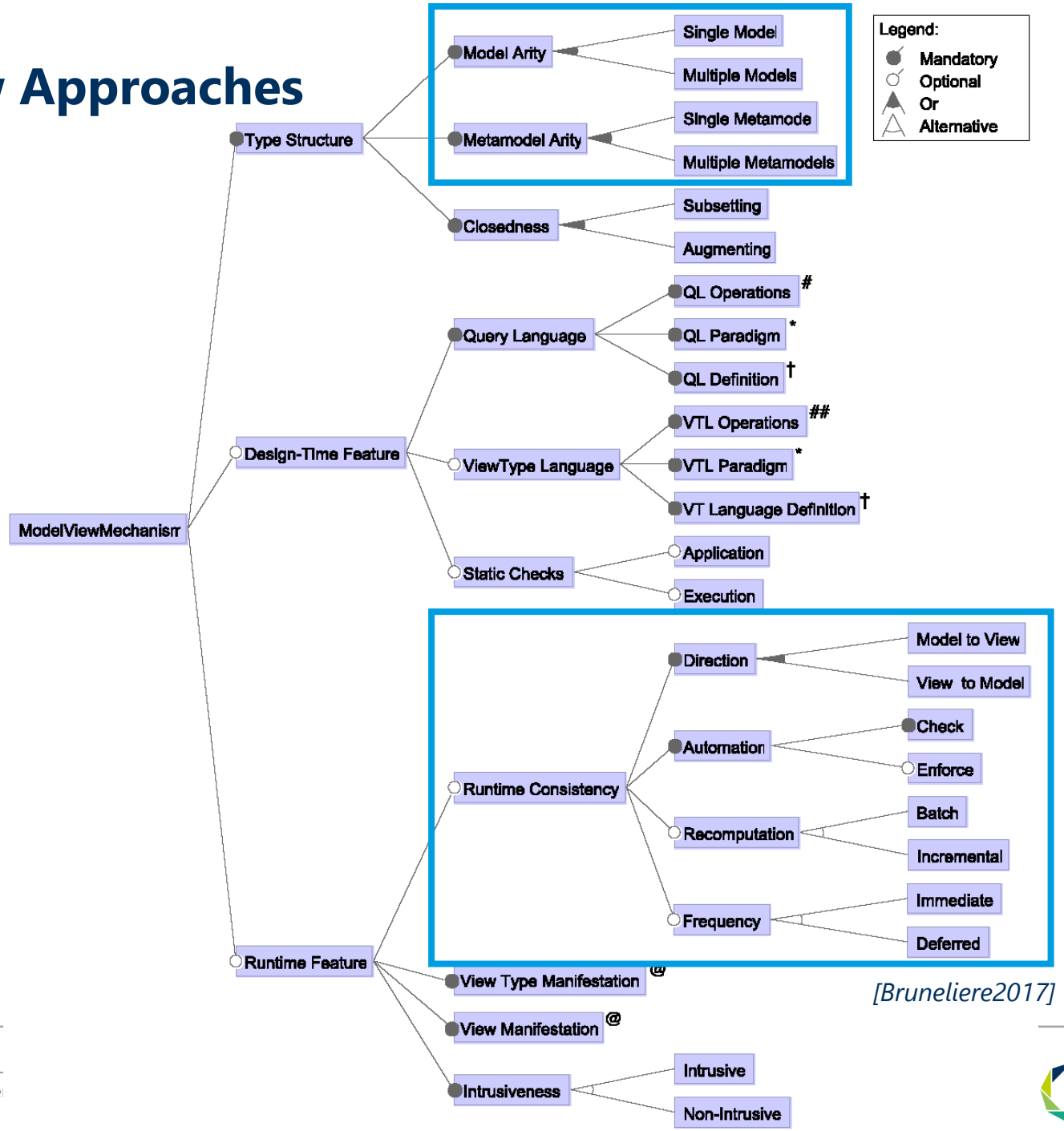


# Model View Approaches

- Survey: Hugo Bruneliere, Erik Burger, Jordi Cabot, and Manuel Wimmer. 2017. “A feature-based survey of model view approaches”. In: Software & Systems Modeling (15 Sept. 2017), 1–22. *[Bruneliere2017]*
- 16 approaches: Cicchetti, EMF Facet, EMF Profiles, EMF Views, Epsilon Decoration, Epsilon Merge, FacadeMetamodel, Kitalpha, ModelJoin, OpenFlexo, OSM, Sirius, TGGvv, TGGmv, VIATRA Viewers, VUML

# Feature Model for Model View Approaches

- Start with 10 distinct approaches (own knowledge)
- Own approaches: ModelJoin, EMF Profiles, and EMF Views
- Search on database dump of Digital Bibliography & Library Project (DBLP)
- Keywords: Model and view
- Snowballing
- Feature Model: Contains only functional features




*[Bruneliere2017]*

# Classified Model View Approaches

| Features | | EMF Facet [EclipseFacet] | EMF Views [Bruneliere2015] | Epsilon Merge [Kolovos2010] | Kitalpha [Langlois2014] | ModelJoin [Burger2014] | Sirius [EclipseSirius] | TGGvv [Jakob2006] | VUML [Anwar2010] | Cicchetti [Cicchetti2011] | Façade Metamodel [Noyrit2012] | EMF Profiles [Langer2012] | Epsilon Decoration [Kolovos2006] | OSM [Atkinson2010] | TGGmv [Anjorin2014] | OpenFlexo [Golra2016] | VIATRA Viewers [Debreceni2014] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Runtime | Immediate updates | ○ | ● | ○ | ● | ○ | ● | ● | ○ | ● | ● | ● | ● | ● | ● | ● | ● |
| | Incremental updates | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● | ● | ● | ● | ● | ● |
| Direction | Model -> View | ● | ● | ● | ● | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ● | ● | ● |
| | View -> Model | ○ | ● | ○ | ● | ● | ● | ● | ● | ○ | ● | ● | ● | ● | ● | ● | ● |
| Arity | Model | 1 | N | N | N | N | N | 1 | N | 1 | 1 | 1 | 1 | 1 | 1 | N | N |
| | Metamodel | 1 | N | N | N | N | N | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | N | N |
| Views | Virtual views | ● | ● | ○ | ○ | ○ | ○ | ● | ○ | ○ | ● | ○ | ○ | ● | ○ | ● | ● |
| | Deep views | ○ | ? | ? | ○ | ? | ○ | ● | ● | ○ | ○ | ○ | ○ | ○ | ● | ○ | ● |

● Supported ◑ Semi supported ○ Not supported ? unknown 1 Single (meta)model N Multiple (meta)models

# EMF Profiles *[Langer2012]*

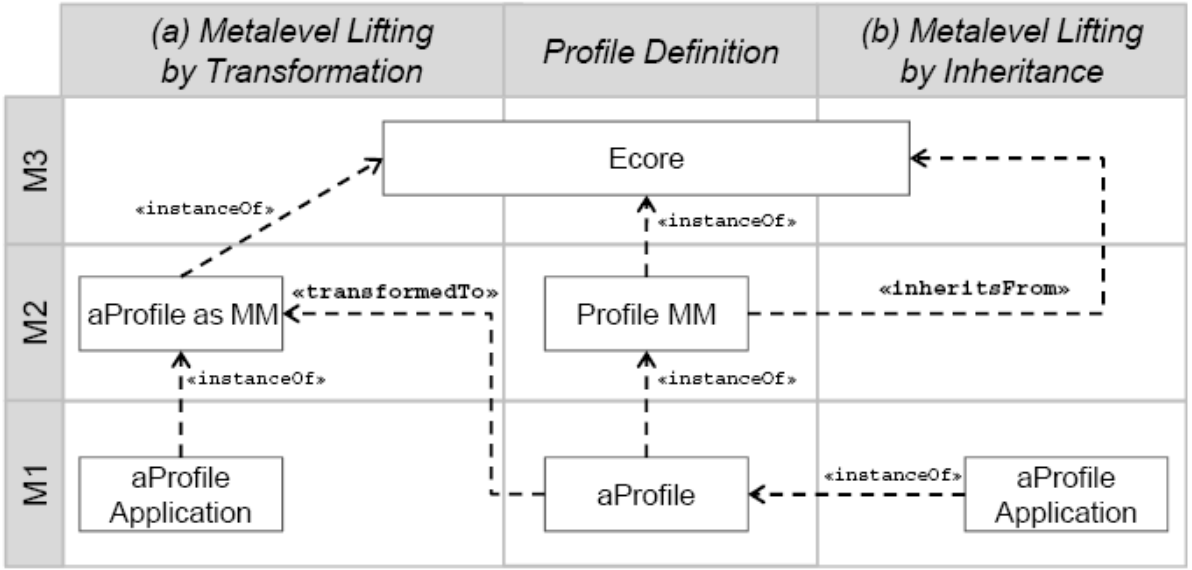
UML profiles (standardized by OMG) as basis for an extension of EMF models with profiles

Advantages:

- No pollution of metamodels (only addition of profiles)
- Reusable profile definitions
- Light-weight language extension
- Dynamic model extension
- Model based representation of new information

Users define or extend parts of relations and stereotypes

Stereotypes have a reference to the model element that they change




Implementation strategies *[Langer2012]*

# Openflexo *[Golra2016]*

Federation of data from heterogenous technical spaces (EMF, XML, OWL, Excel, etc.) into one conceptual space

Generic solution that assemble and relate data (without duplications)

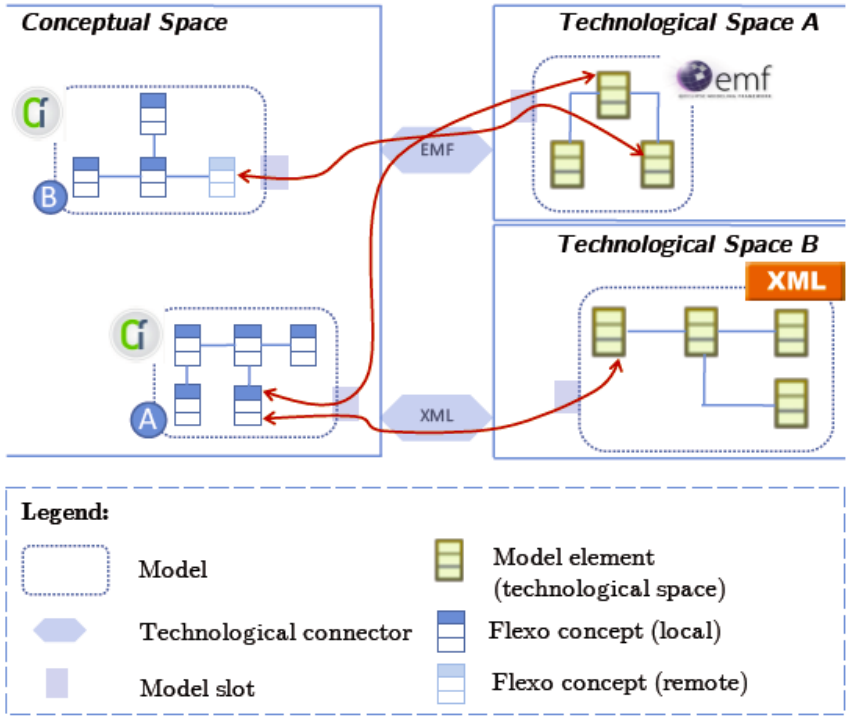
Including components:

- Viewpoint Modeler (specifying viewpoints, combine different data types)
- ViewEditor (provide regular view visualization and editing capabilities)
- Underlying model federation framework

Indirect bidirectional connecting between base models and views

Approach provides:

- Reusability, synchronization, flexibility, management, and conceptual alignment of existing (meta)models in the modeling space




Openflexo Concept *[Golra2016]*

# TGGmv (materialized view) *[Anjorin2014]*

Triple Graph Grammars (TGGs) specify consistency relations between two graph languages

Bidirectional transformations used for model transformation, comparison, and synchronization

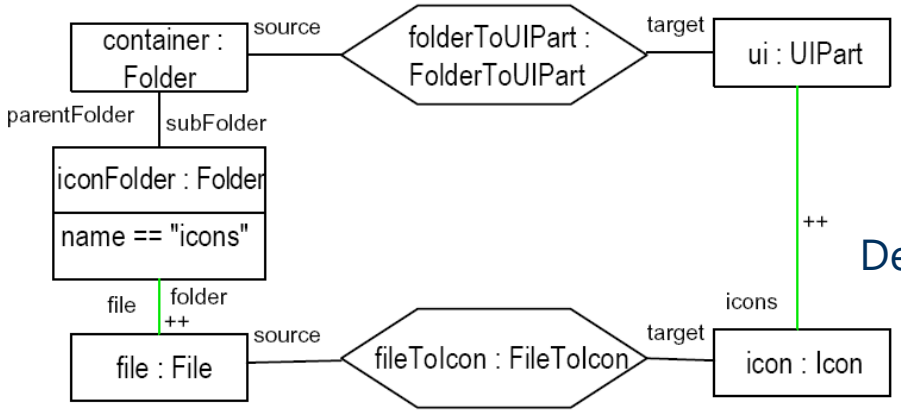
Formalization of ViewTGGs as a restriction of existing TGG without symmetry (not change source and target models)

Using eMoflon as model synchronization tool *[Leblebici2014]*

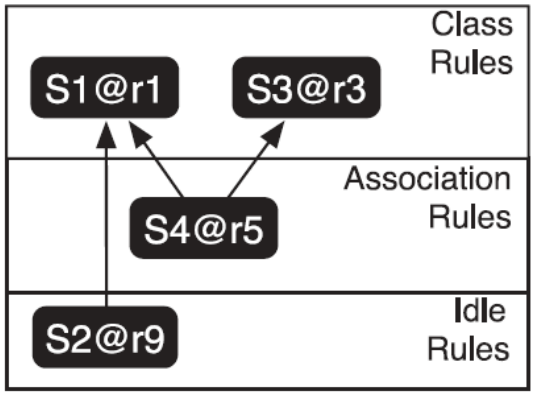
Allows separate view models without changing the base metamodels

Subdividing:

- Class rules (only create elements)
- Association rules (link existing elements)
- Idle rules (describe model development)




TGG Rule *[Anjorin2014]*




Dependency Graph *[Anjorin2014]*

# VIATRA Viewers *[Debreceni2014]*

Emerged from EMF IncQuery (framework to perform incremental model querying)

Not entirely evaluating query during model changes

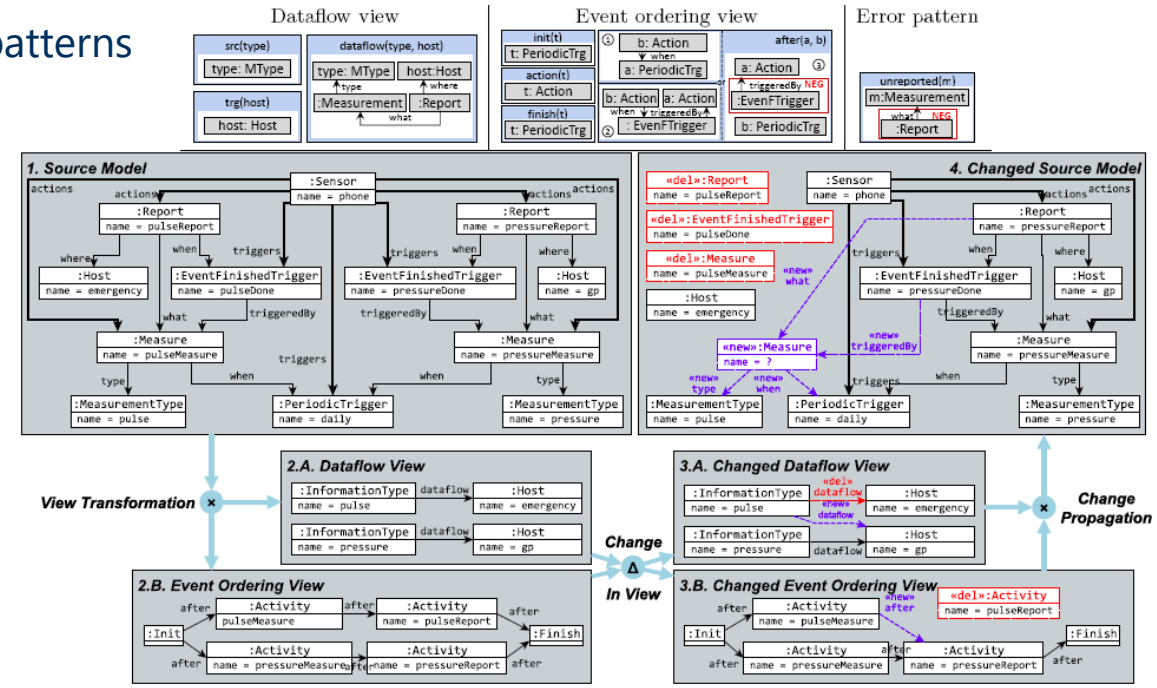
Derivation rules are define with annotations to query patterns

Trace models between the views and the base models

Dealing with model views

Change reasoning of query pattern matches create synchronization support

Building chain views (deep views)




Example process *[Debreceni2014]*

# Own Concept

# Problem Definition

| Features | | VIATRA Viewers [Debreceni2014] | RSUM |
|---|---|---|---|
| Runtime | Immediate updates | ● | ● |
| | Incremental updates | ● | ● |
| Direction | Model -> View | ● | ● |
| | View -> Model | ● | ● |
| Arity | Model | N | (N) |
| | Metamodel | N | (N) |
| Views | Virtual views | ● | ● |
| | Deep views | ● | (●) |

Model View Approaches

| | VIATRA3 *[Varró2016]* | Role Sync |
|---|---|---|
| Incremental updates | ● | ● |
| Automatic updates | ● | ● |
| Transformation direction | → | ↑ ↓ ← → |
| Runtime rule changes | ○ | ● |
| Synchronization between multiple models | ● | ● |

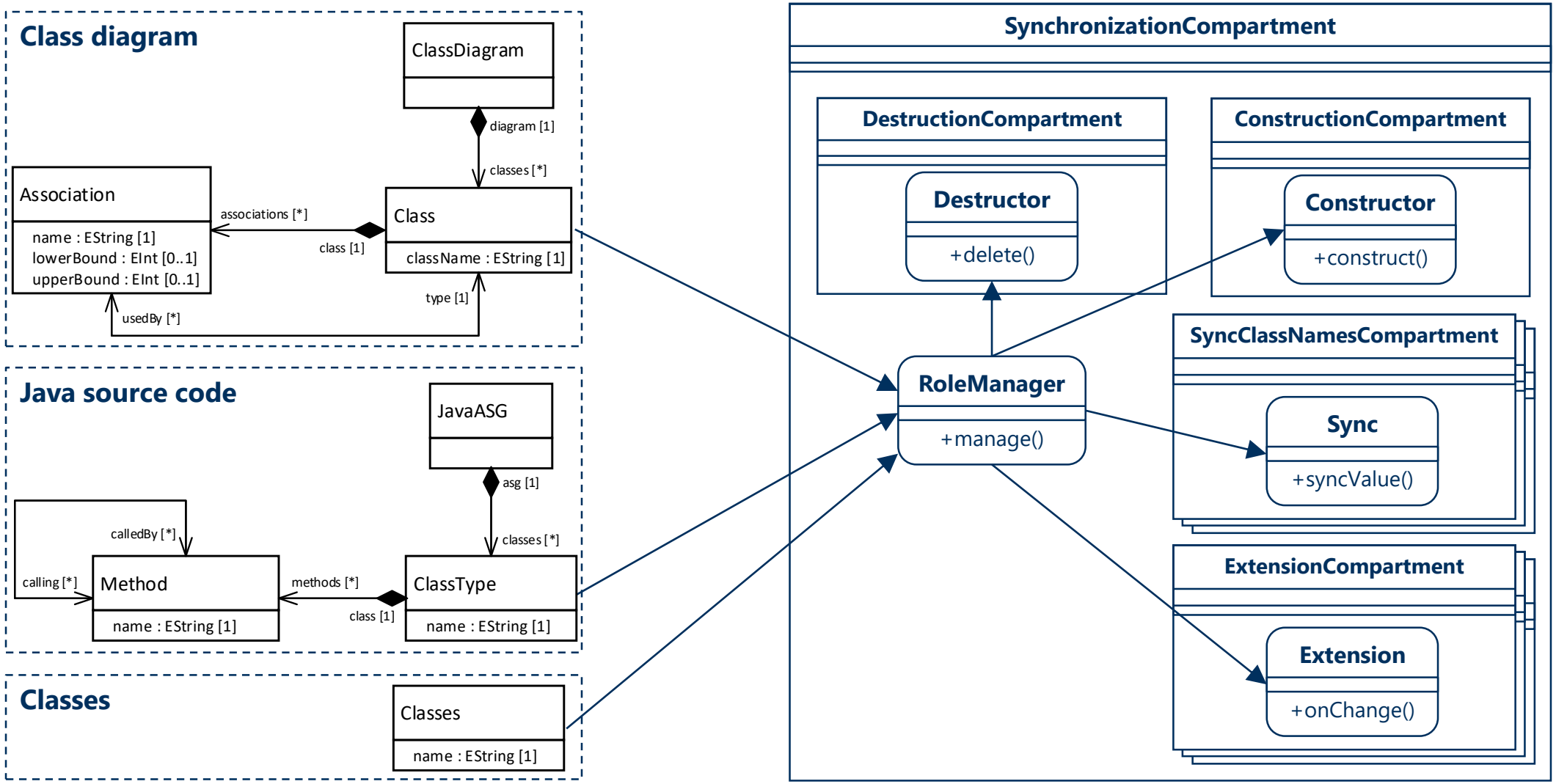
Model Synchronization

**Goals:**

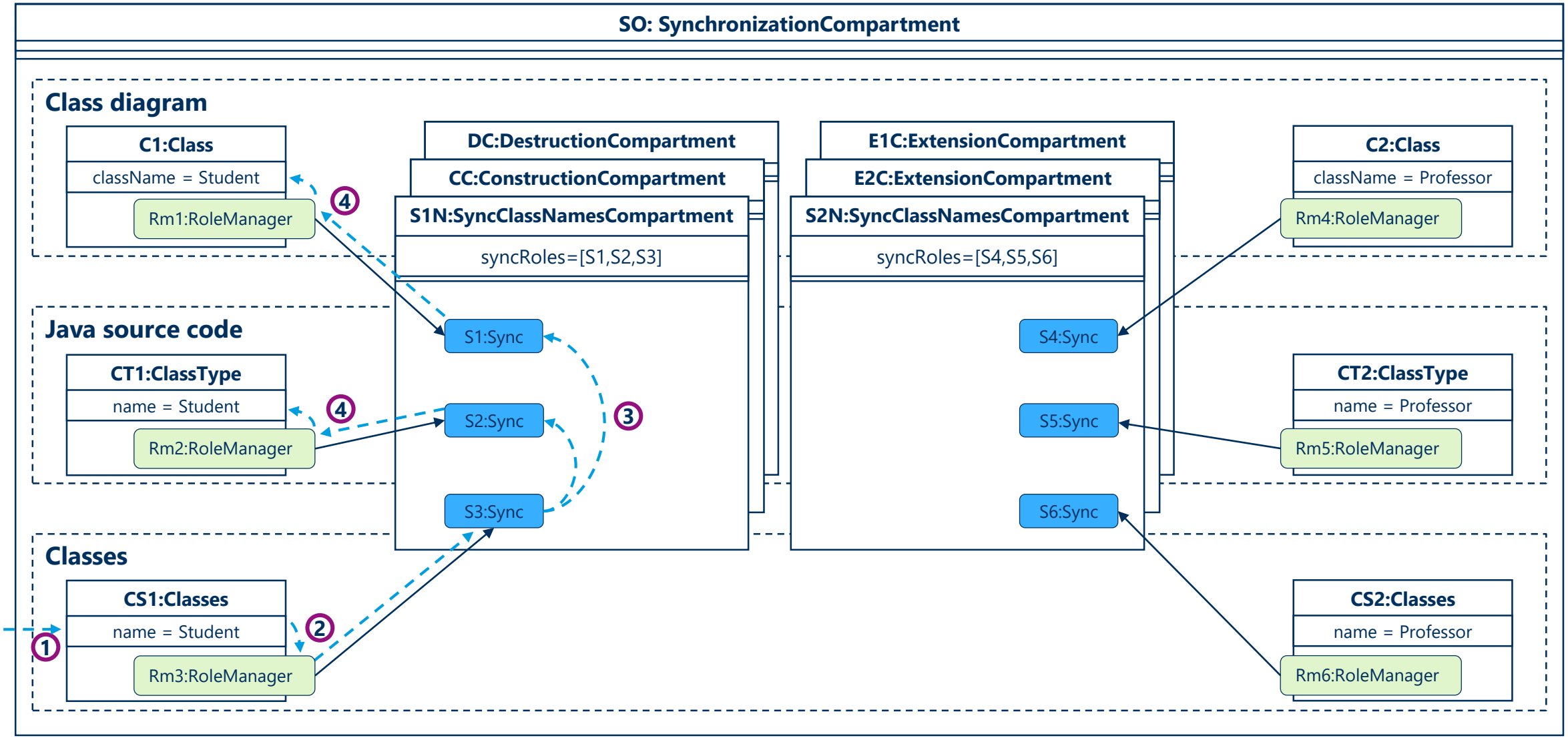
- Incremental runtime updates
- Runtime rule changes
- Synchronization in multiple directions
- Virtual and deep views
- Multiple (meta)model support

# RoleSync (Role-based Synchronization)

# RoleSync Concept

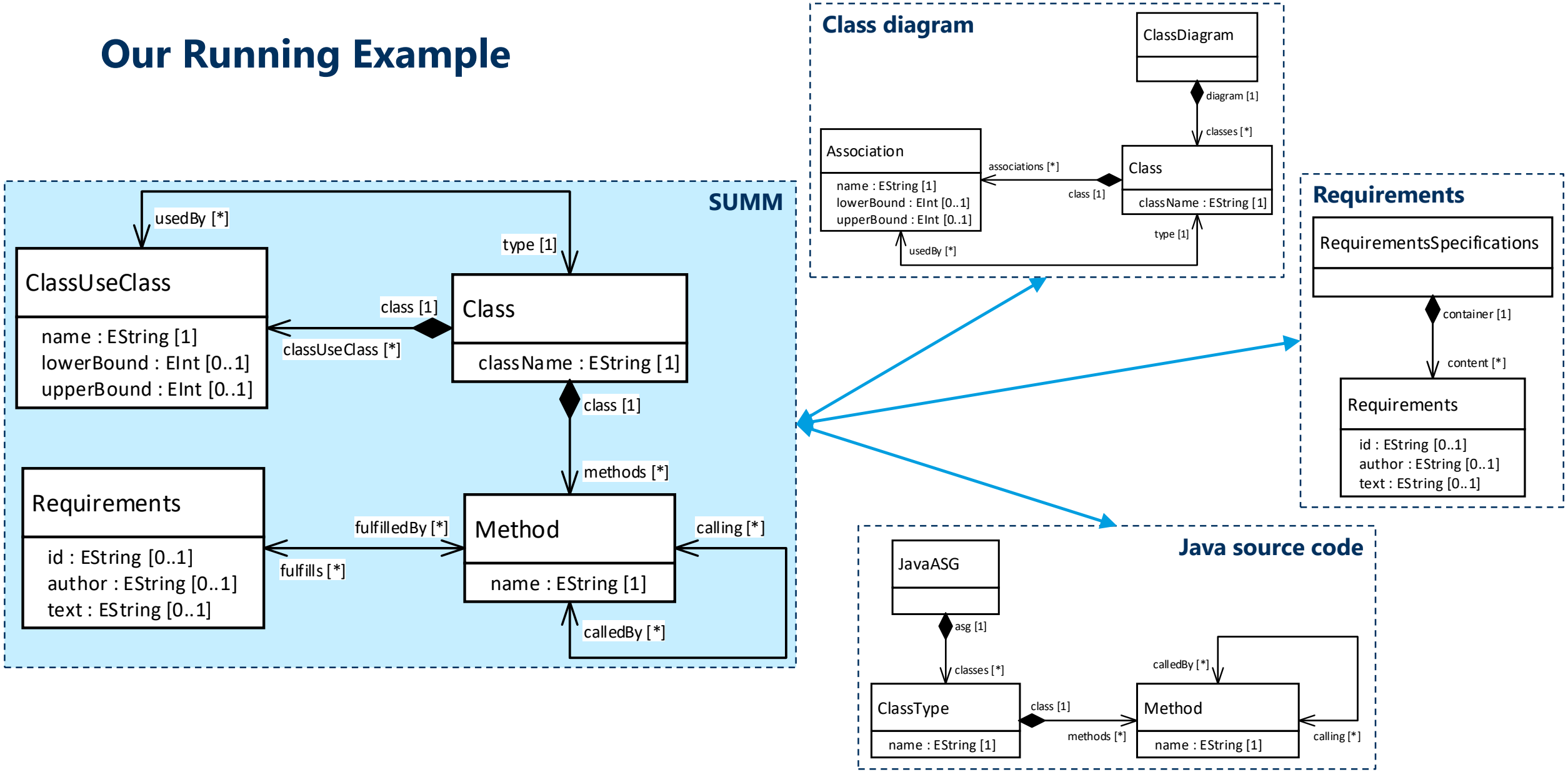


# Synchronization Process Example

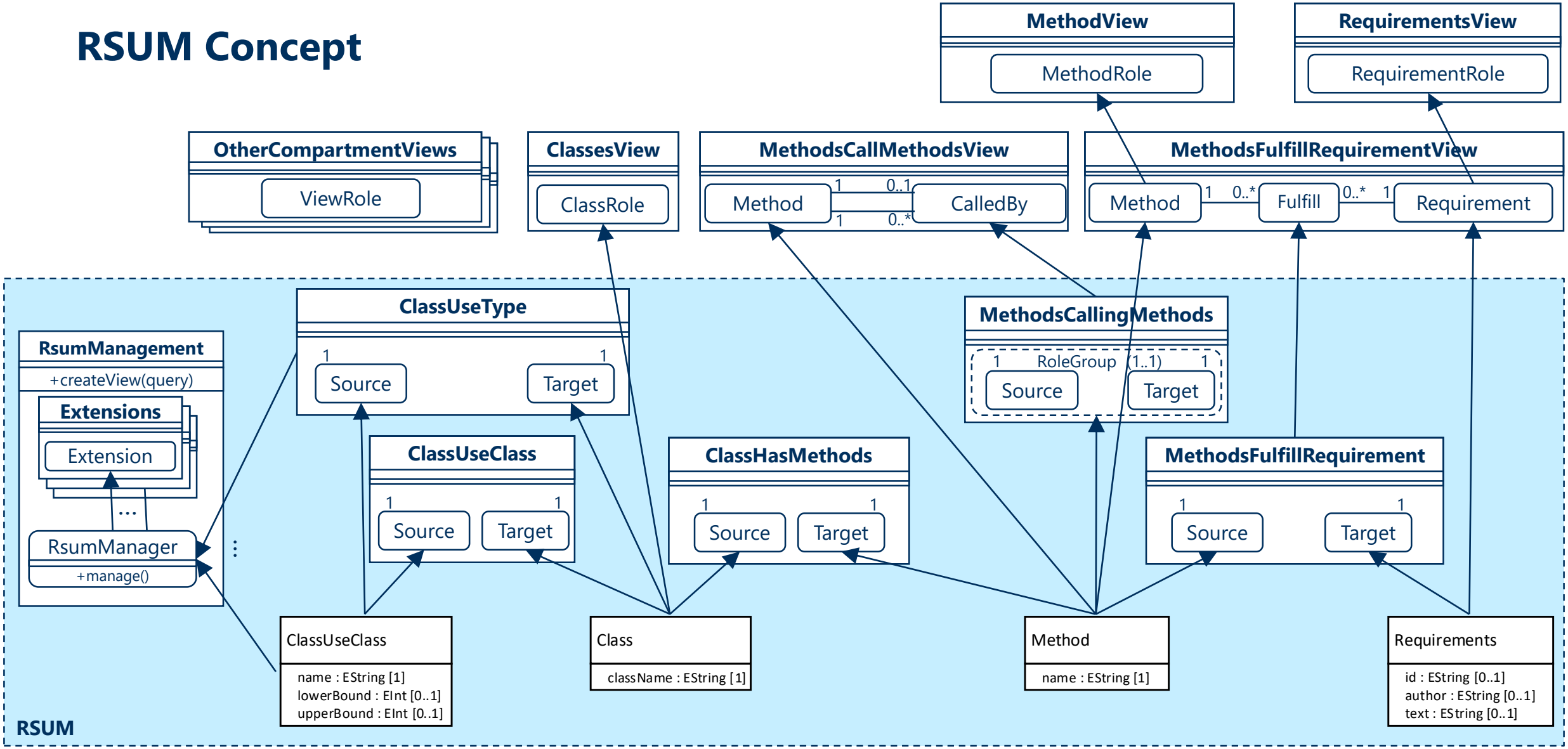


# RSUM (Role-based Single Underlying Model)

# Our Running Example



# RSUM Concept



# Concept of Relational Compartments

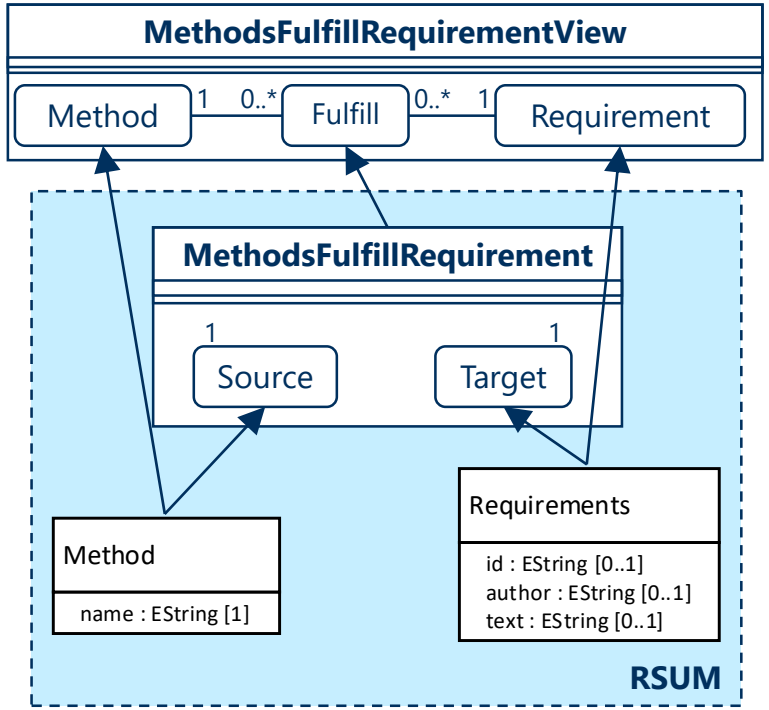
Relational compartments represent relations in the RSUM

<u>Advantages:</u>

1. Loading relations at runtime (using class loader functionality from programming languages like Scala and Java)
2. Extending naturals with new relations at runtime (playing of roles in relational compartments)
3. Add behavior and states to relations (methods and attributes in relational compartments)
4. n-ary relations (more role types in the relational compartment)

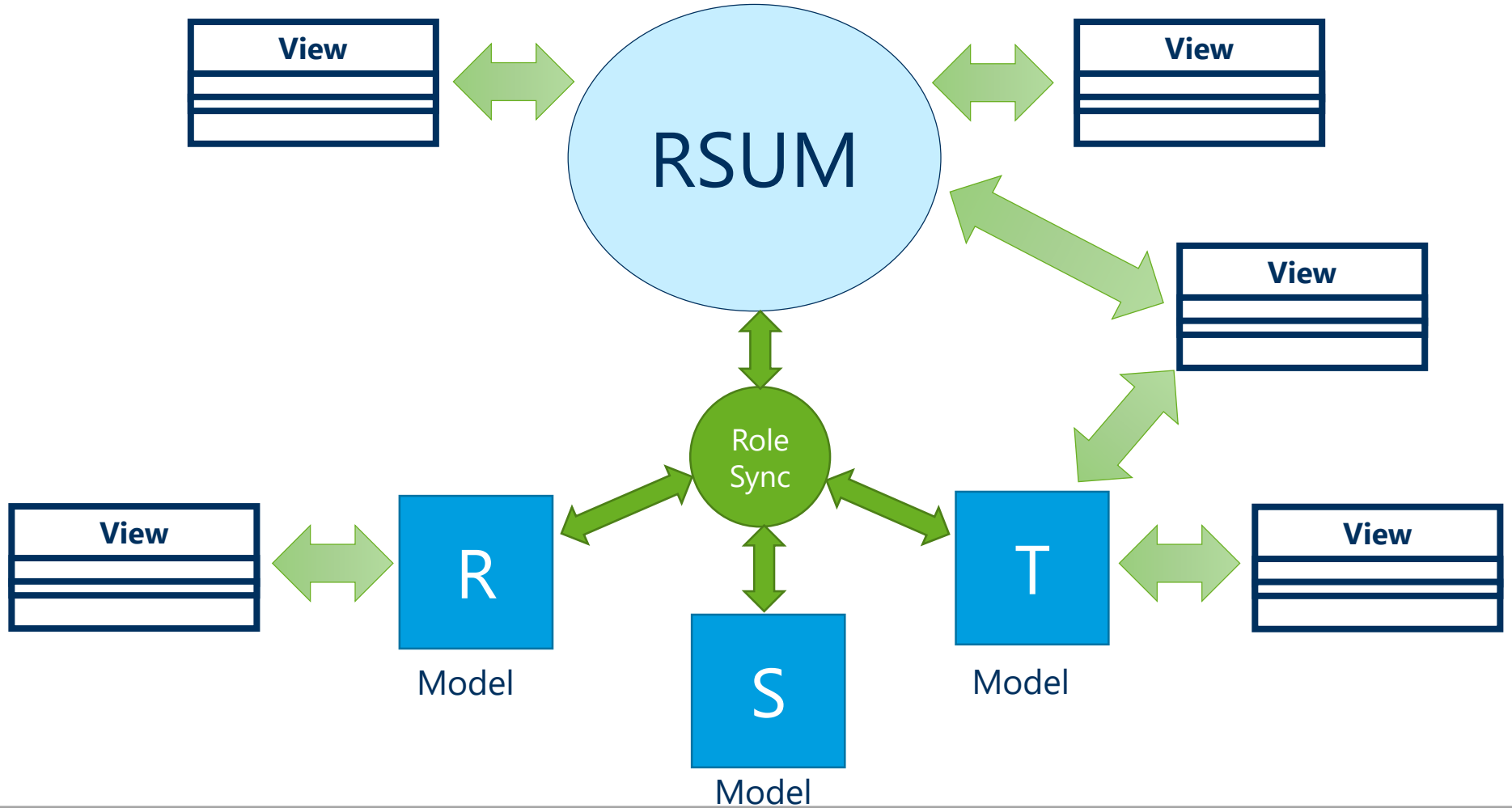
<u>Limitations:</u>

1. One relational compartment for each instance pair
2. Management of play relations of instances



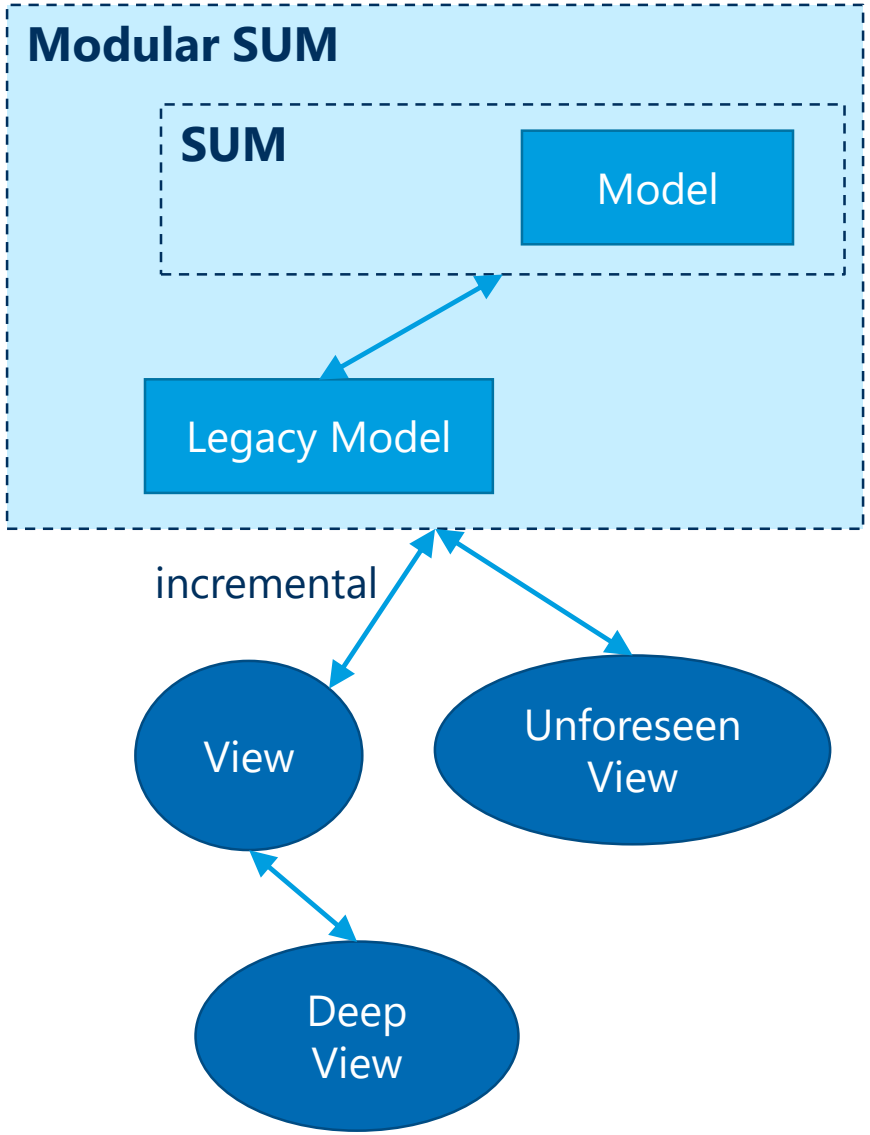
# RSUM & RoleSync Combination (future work)

# RSUM + RoleSync



# Contributions

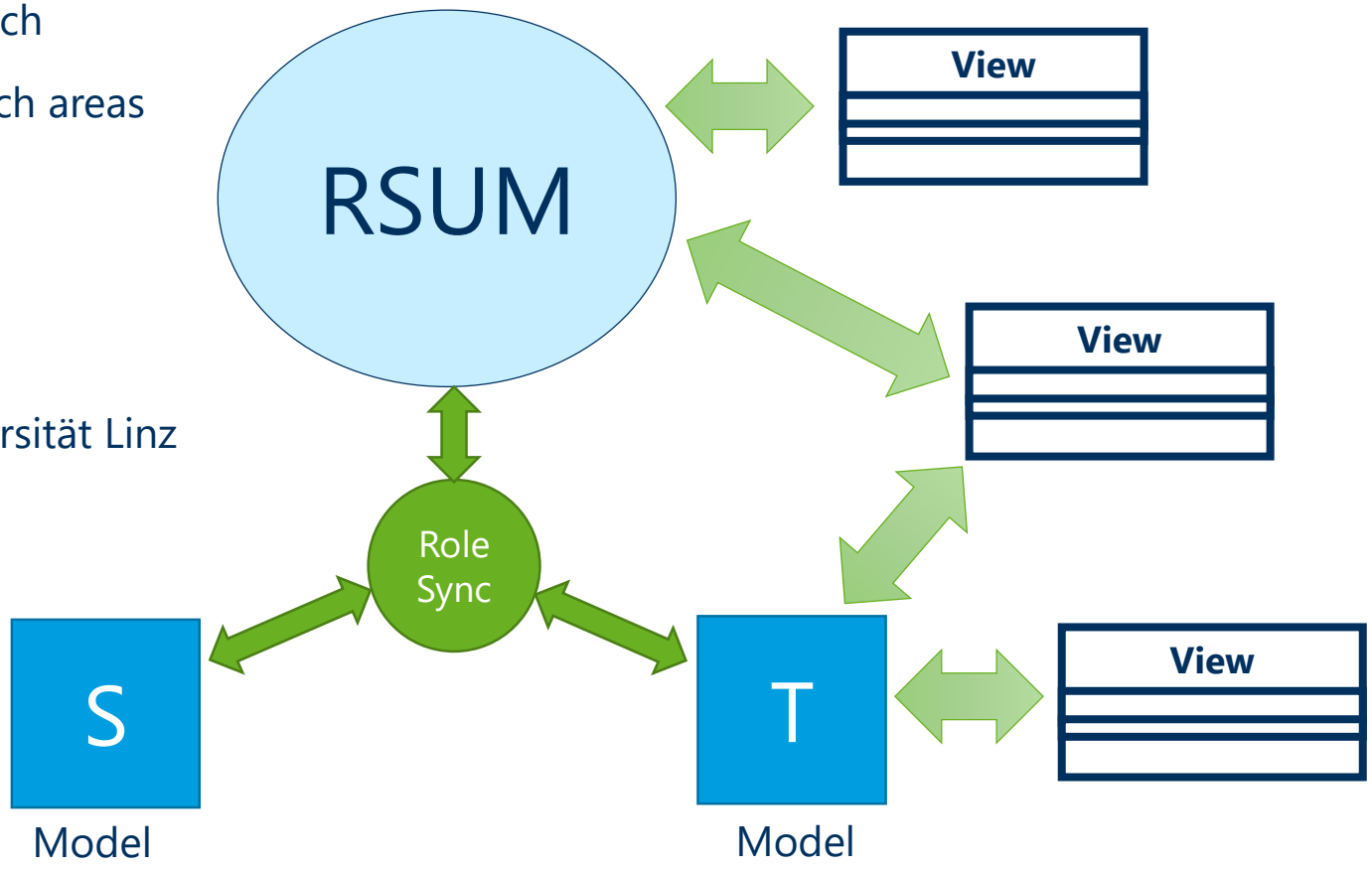
1. Model-view approach (×)
   - Incremental & immediate synchronization between views and SUM ✓
   - Using multiple (meta)models (×)
   - Creation of unforeseen views ✓
   - Construction of deep views (×)
2. Model synchronization approach ✓
   - Automatic triggering of the synchronization process ✓
   - Incremental runtime updates ✓
   - Specification and modification of synchronization rules at runtime ✓
   - Synchronization rules for multiple models ✓
3. Megamodel approach (×)
   - Extensibility of the SUM ✓
   - Integration of legacy models into the SUM (×)
   - Model management (×)



# Summary & Future Work

- Presented related work in the areas of megamodels, view modeling, and model synchronization
- Presentation of the RoleSync and RSUM approach
- Showing contributions in the considered research areas

- Q4 2018: Combinations of Approaches
- Q1 2019: Finish Implementation
- Q2 2019: Stay Abroad at Johannes Kepler Universität Linz
  Chair of Manuel Wimmer
- Q3+4 2019: Writing dissertation



# Publications

Main

- Christopher Werner, Hendrik Schön, Thomas Kühn, Sebastian Götz, and Uwe Aßmann: "Role-based Runtime Model Synchronization". In: Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications, IEEE, 2018, pp. 306-313.
- Christopher Werner and Uwe Aßmann: "Model Synchronization with the Role-oriented Single Underlying Model". MRT 2018 (Models@Runtime Workshop co-located to MODELS 2018).

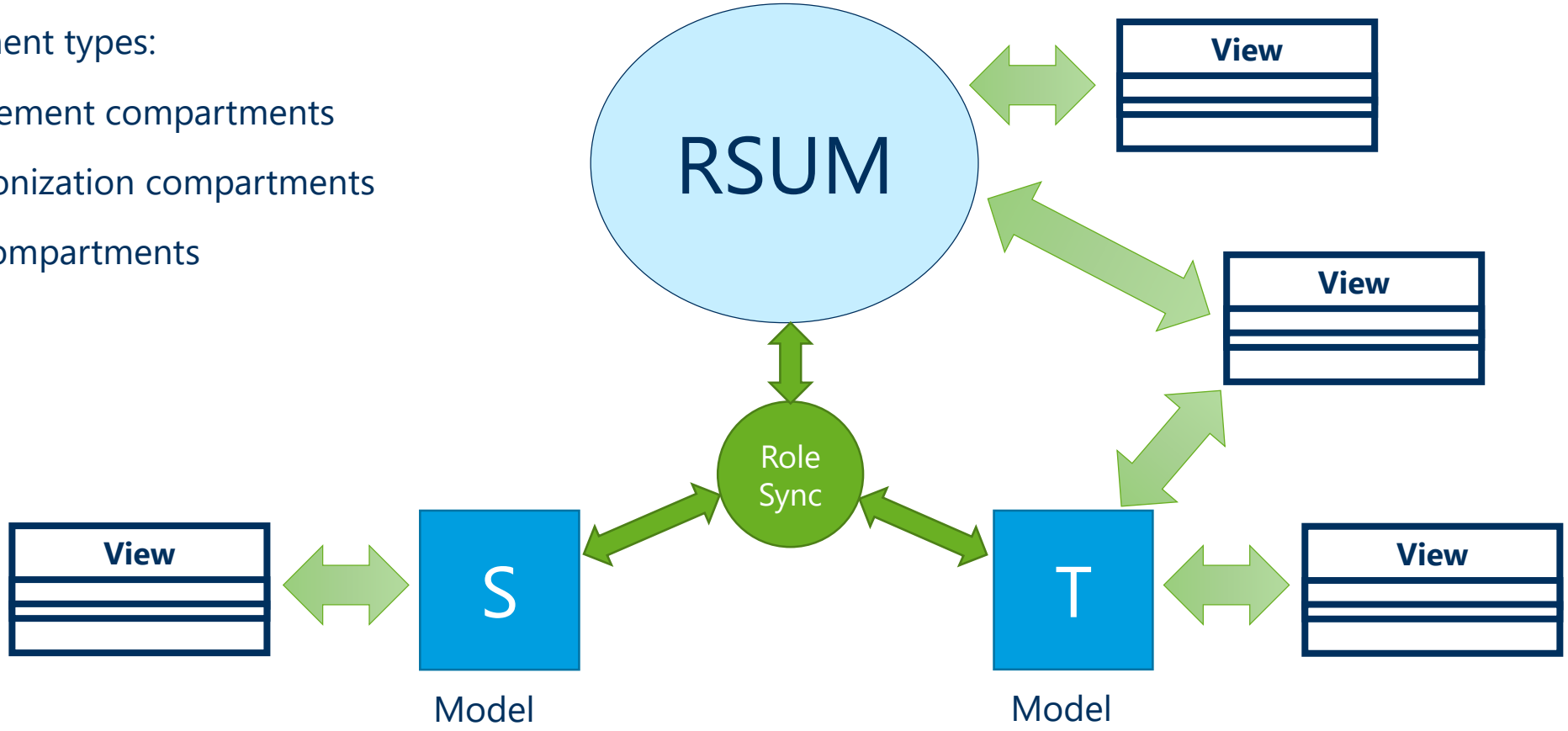
Others

- Christopher Werner, Sebastian Götz, and Uwe Aßmann: "A Simulation Framework to Analyze Knowledge Exchange Strategies in Distributed Self-adaptive Systems". MORSE 2017, Springer, Cham, pp. 280-294.
- Christopher Werner, Sebastian Werner, René Schöne, Sebastian Götz, and Uwe Aßmann: "Self-adaptive Synchronous Localization and Mapping using Runtime Feature Models". EDDY 2018 (Special Session DATA Conference).
- Julian Eberius, Christopher Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, and Wolfgang Lehner: "DeExcelerator: a framework for extracting relational data from partially structured documents". In Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM '13). ACM, New York, NY, USA, 2477-2480.
- Oliver Koch, Benjamin Beck, Georg Heß, Christian Richter, Volker Waurich, Jürgen Weber, Christopher Werner, and Uwe Aßmann: "Cloud-Based System Architecture for Driver Assistance in Mobile Machinery". In: Proceedings of 15th Scandinavian International Conference on Fluid Power, June 7-9, 2017, Linköping, Sweden, Linköping University Electronic Press, pp.81-90 2017.
- Uwe Aßmann, Johannes Mey, Dominik Grzelak, Dmytro Pukhkaiev, René Schöne, Christopher Werner and Georg Püschel: "Cross-Layer Adaptation in Multi-Layer Autonomic Systems". SOFSEM 2019.

# Questions?

Compartment types:

- Management compartments
- Synchronization compartments
- View compartments



# References

**[Anjorin2014]** A. Anjorin, S. Rose, F. Deckwerth, and A. Schürr. 2014. “Efficient Model Synchronization with View Triple Graph Grammars”. In Modelling Foundations and Applications. Springer International Publishing, Cham, pp. 1–17.

**[Anwar2010]** A. Anwar, S. Ebersold, B. Coulette, M. Nassar, and A. Kriouile. 2010. "A rule-driven approach for composing viewpoint-oriented models". J. Object Technol. 9(2), pp. 89–114.

**[Atkinson2010]** C. Atkinson, D. Stoll, and P. Bostan. 2010. "Orthographic software modeling: a practical approach to view-based development". In: Proceedings of the 4th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), pp. 206–219. Springer, Berlin.

**[Bruneliere2015]** H. Bruneliere, J. Garcia, P. Desfray, D.E. Khelladi, R. Hebig, R. Bendraou, and J. Cabot. 2015. "On lightweight metamodel extension to support modeling tools agility". In: Proceedings of the 11th European Conference on Modelling Foundations and Applications (ECMFA), pp. 62–74. Springer, Berlin.

**[Bruneliere2017]** H. Bruneliere, E. Burger, J. Cabot, and M. Wimmer. 2017. "A feature-based survey of model view approaches". In: Software & Systems Modeling.

**[Burger2014]** E. Burger, J. Henß, S. Kruse, M. Küster, A. Rentschler, and L. Happe. 2014. "ModelJoin. A textual domain-specific language for the combination of heterogeneous models". Tech. Rep. 1, Karlsruhe Institute of Technology, Faculty of Informatics.

# References

**[Cicchetti2011]** A. Cicchetti, F. Ciccozzi, and T. Leveque. 2011. "A hybrid approach for multi-view modeling". In ECEASST 50.

**[Debreceni2014]** C. Debreceni, A. Horváth, A. Hegedüs, Z. Ujhelyi, I. Ráth, and D. Varró. 2014. "Query-driven incremental synchronization of view models". In: Proceedings of the 2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling (VAO), pp. 31:31–31:38. ACM, London.

**[Diskin2011]** Z. Diskin, Y. Xiong, and K. Czarnecki.. 2011. “From State- to Delta-Based Bidirectional Model Transformations: The Symmetric Case”, Journal of Object Technology, pp. 10:6:1–25.

**[EclipseFacet]** Eclipse EMF Facet project. http://www.eclipse.org/facet/.

**[EclipseSirius]** Eclipse Sirius project. https://eclipse.org/sirius/.

**[Giese2006]** H. Giese and R. Wagner, "Incremental model synchronization with triple graph grammars," in International Conference on Model Driven Engineering Languages and Systems. Springer, 2006, pp. 543–557.

**[Golra2016]** F.R. Golra, A. Beugnard, F. Dagnat, S. Guérin, and C. Guychard. 2016. "Addressing modularity for heterogeneous multi-model systems using model federation. In: Companion Proceedings of the 15th International Conference on Modularity, pp. 206–211. ACM, London.

# References

**[Holmes2009]** T. Holmes, U. Zdun, and S. Dustdar. 2009. “MORSE: A Model-Aware service environment”. In 2009 IEEE Asia-Pacific Services Computing Conference (APSCC). 470–477.

**[Jakob2006]** J. Jakob, A. Königs, and A. Schürr. 2006. "Non-materialized model view specification with triple graph grammars". In: Proceedings of the 3rd International Conference on Graph Transformations (ICGT), pp. 321–335. Springer, Berlin.

**[Kolovos2006]** D.S. Kolovos, R.F. Paige, and F.A. Polack. 2006. "Merging models with the epsilon merging language (EML)". In: Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS), pp. 215–229. Springer, Berlin.

**[Kolovos2010]** D.S. Kolovos, L.M. Rose, N.D. Matragkas, R.F. Paige, F.A. Polack, and K.J. Fernandes. 2010. "Constructing and navigating non-invasive model decorations". In: Proceedings of the 3rd International Conference on Theory and Practice of Model Transformations (ICMT), pp. 138–152. Springer, Berlin.

**[Kramer2013]** M.E. Kramer, E. Burger, and M. Langhammer. 2013. "View-centric engineering with synchronized heterogeneous models". In Proceedings of the 1st workshop on view-based, aspect-oriented and orthographic software modelling pp. 5:1–5:6. VAO ’13. ACM.

# References

**[Langer2012]** P. Langer, K. Wieland, M. Wimmer, and J. Cabot. 2012. "EMF profiles: a lightweight extension approach for EMF models". J. Object Technol. 11(1), pp. 1–29.

**[Langlois2014]** B. Langlois, D. Exertier, and B. Zendagui. 2014. "Development of modelling frameworks and viewpoints with Kitalpha". In: Proceedings of the 14th Workshop on Domain-Specific Modeling (DSM), pp. 19–22. ACM, London.

**[Leblebici2014]** E. Leblebici, A. Anjorin, and A. Schürr. 2014. "Developing eMoflon with eMoflon". In: Proceedings of the 7th International Conference on the Theory and Practice of Model Transformations (ICMT), pp. 138–145. Springer, Berlin.

**[Martinez2017]** S. Martinez, M. Tisi, and R. Douence, "Reactive model transformation with atl," Science of Computer Programming, vol. 136, 2017, pp. 1–16.

**[Meier2018]** J. Meier, and A. Winter. 2018. "Model Consistency ensured by Metamodel Integration". 6th International Workshop on The Globalization of Modeling Languages, co-located with MODELS 2018.

**[Noyrit2012]** F. Noyrit, S. Gérard, and B. Selic. 2012. "FacadeMetamodel: Masking UML". In: Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems (MoDELS), pp. 20–35. Springer, Berlin.

# References

**[Perovich2009]** D. Perovich, M.C. Bastarrica, and C. Rojas. 2009. "Model-driven approach to software architecture design". In: Proceedings of the 2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge, pp. 1-8.

**[Seibel2012]** A. Seibel. 2013. "Traceability and model management with executable and dynamic hierarchical megamodels". Diss. Universität Potsdam.

**[Seifert2011]** M. Seifert. 2011. "Designing Round-Trip Systems by Model Partitioning and Change Propagation". PhD thesis, Technische Universität Dresden, Fakultät Informatik.

**[Silva2015]** A. R. da Silva. 2015. "Model-driven engineering: A survey supported by the unified conceptual model" Computer Languages, Systems & Structures, 43 , pp. 139-155.

**[Song2010]** H. Song, Y. Xiong, F. Chauvel, G. Huang, Z. Hu, and H. Mei. 2010. “Generating Synchronization Engines between Running Systems and Their Model-Based Views”. In Models in Software Engineering. Springer Berlin Heidelberg, 140–154.

**[Trollmann2016]** F. Trollmann and S. Albayrak, "Extending model synchronization results from triple graph grammars to multiple models," in International Conference on Theory and Practice of Model Transformations. Springer, 2016, pp. 91–106.

# References

**[Varró2016]** D. Varró, G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, and Z. Ujhelyi, "Road to a reactive and incremental model transformation platform: three generations of the viatra framework," Software & Systems Modeling, vol. 15, no. 3, 2016, pp. 609–629.

**[Vogel2010]** T. Vogel, S. Neumann, S. Hildebrandt, H. Giese, and B. Becker. 2010. “Incremental Model Synchronization for Efficient Run-Time Monitoring”. In Models in Software Engineering. Springer Berlin Heidelberg, 124–139.

**[Vogel2018]** T. Vogel. 2018. "Model-driven engineering of self-adaptive software". Diss. Universität Potsdam.

**[Xtend]** Xtend https://eclipse.org/xtend/index.html.

**[Gerking2014]** C. Gerking and C. Heinzemann. 2014. "Solving the movie database case with QVTo". In: TTC, pp. 98–102.

**[Forward2009]** A. Forward, T. Lethbridge, and D. Brestovansky. 2009. "Improving program comprehension by enhancing program constructs: An analysis of the Umple language". In: ICPC, pp. 311–312.

**[Kolovos2008]** D. Kolovos, R. Paige, F. Polack. 2008. "The Epsilon transformation language". In: Theory and Practice of Model Transformations, pp. 46–60.

# References

**[Kahani2018]** N. Kahani, M. Bagherzadeh, J.R. Cordy, J. Dingel, and D. Varró. 2018. "Survey and classification of model transformation tools". Software & Systems Modeling, pp. 1-37.