Business Informatics Group

Vienna University of Technology

State of the Art and Future Directions in Model Management Research

Japan-Austria Joint Workshop on ICT, October 18-19, 2010

Gerti Kappel

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

Content

- Introduction
  - *Model-Driven Engineering and Model Management*
- Model Management Use Case I: *Model Versioning*
- Model Management Use Case II: *Model Co-Evolution*
- Résumé

Content

Model-Driven Engineering (MDE)

Models, Models, Models, ...

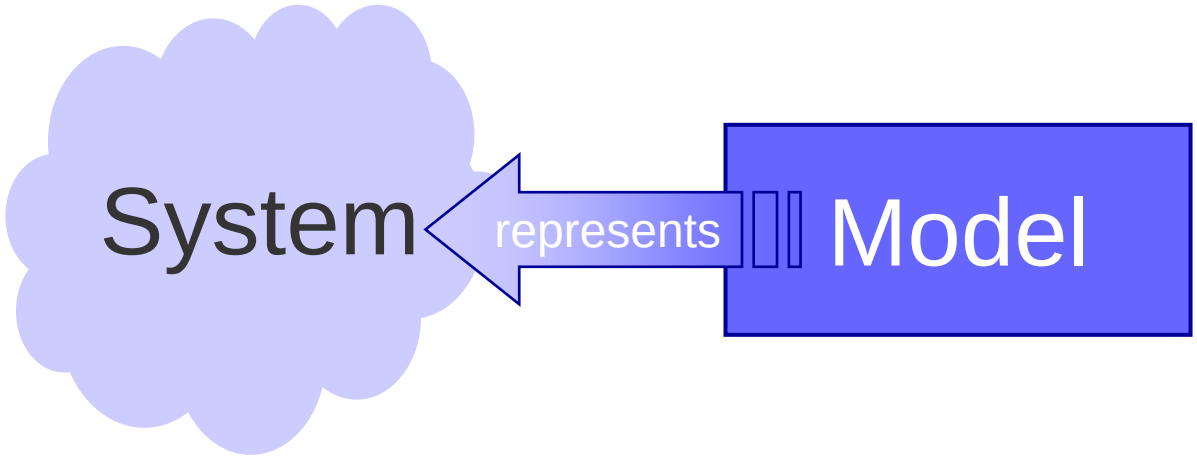
"Everything is a model"

- Analysis Model
- Transformation Model
- Metamodel
- Design Model
- Weaving Model
- Diff Model
- Metametamodel
- Test Model
- Change Model
- ...

Jean Bézivin. ***On the Unification Power of Models***. Software and System Modeling 4(2), pages 171-188, 2005.

Model-Driven Engineering (MDE)

Models, Models, Models, ...

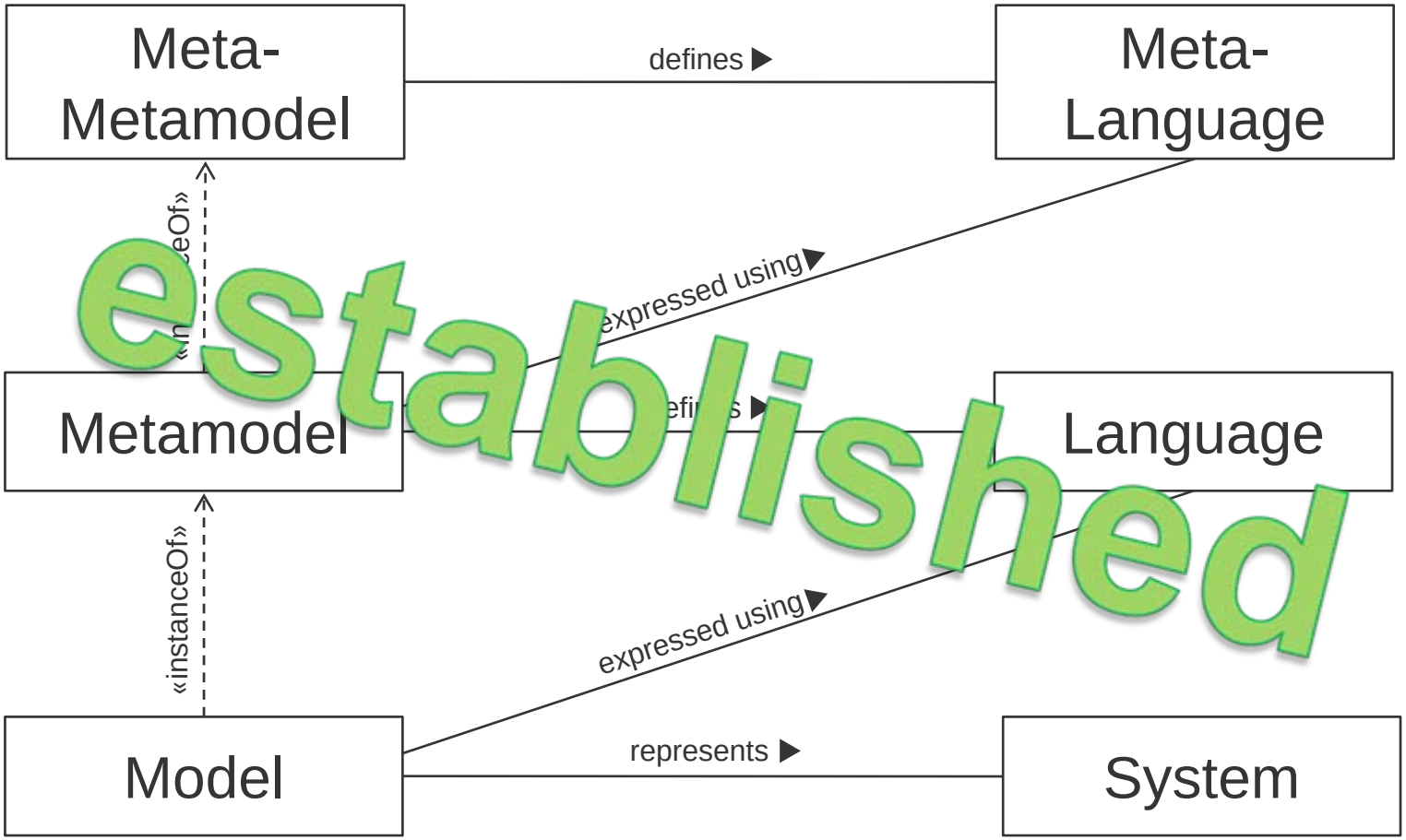


Mapping Feature	A model is based on an original (=system)
Reduction Feature	A model only reflects a (relevant) selection of an original‘s properties
Pragmatic Feature	A model needs to be usable in place of an original with respect to some purpose

Herbert Stachowiak. **Allgemeine Modelltheorie**. Springer, Wien, 1973.

Model-Driven Engineering (MDE)

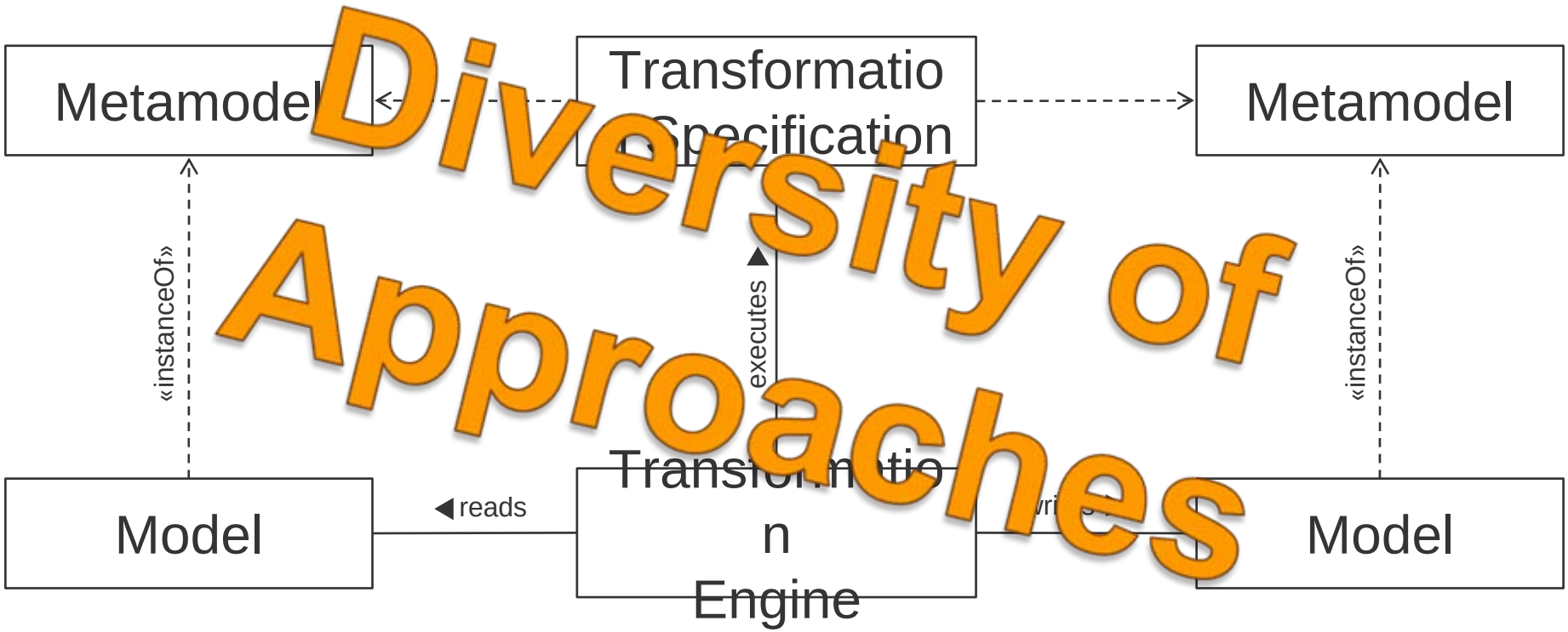
Building Block #1: Metamodeling



Thomas Kühne. ***Matters of (Meta-)Modeling***. Software and System Modeling 5(4), pages 369-385, 2006.

Model-Driven Engineering (MDE)

Building Block #2: Model Transformations



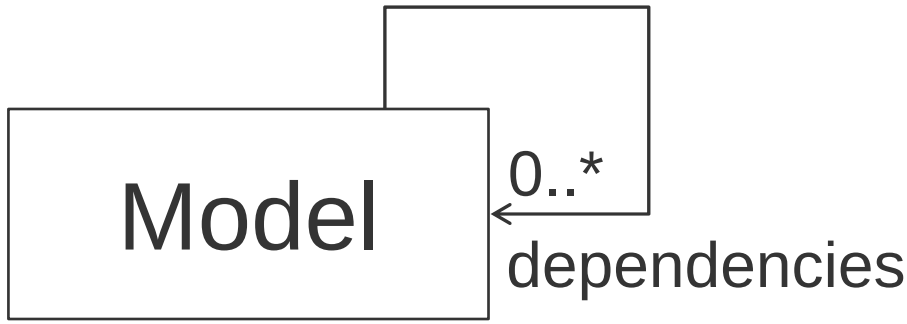
Krzysztof Czarnecki, Simon Helsen. ***Feature-based survey of model transformation approaches***. IBM Systems Journal 45(3), pages 621-646, 2006.

"Everything is a model"

"No model is an island"

- instanceOf
- generatedFrom
- refinementOf
- abstractionOf
- equivalentWith
- overlappingWith
- crossCuttingWith
- sameAs
- …

Model of "No model is an island"



Model Management

Old wine in new bottles?

- Origin: **Data Engineering**
  - Research issue for decades since integration of heterogeneous databases (1970)

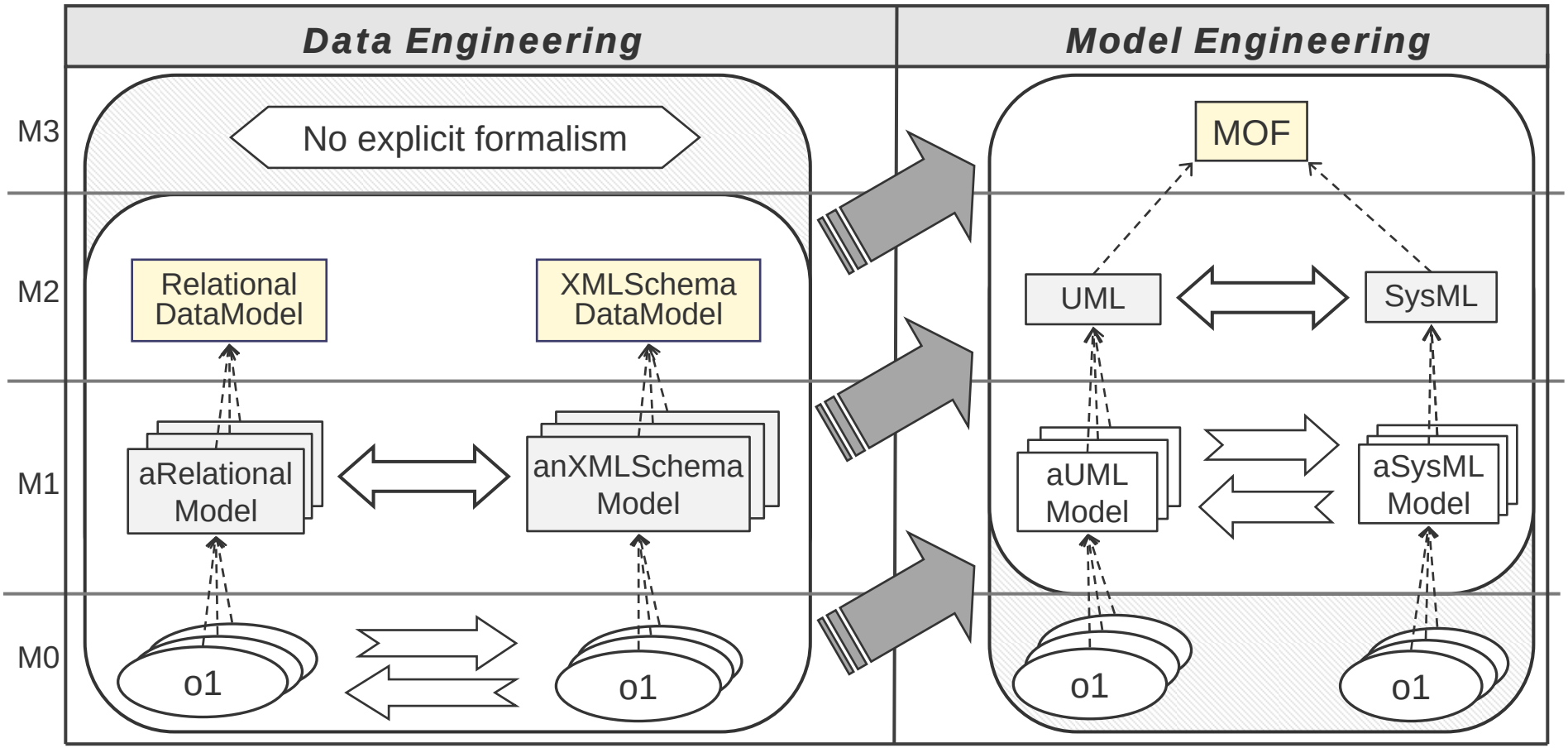
- Current Status: **Model Management 2.0**
  - Well-documented kinds of heterogeneities
  - **Global Model Management Operators**
    - *Diff, Merge, Match, Compose, ModelGen, TransGen, Inverse, …*
  - Much progress, but still **many challenges**

P. Bernstein, S. Melnik. ***Model Management 2.0: Manipulating Richer Mappings***.
ACM SIGMOD 2007 Keynote, China, June 2007.

- Ongoing Work: **Model Management 3.0 = Model Management 2.0 + Model Engineering**
  - Raising the level of abstraction
    - Too many data models, formats, technologies, tools, …
  - Built on top of powerful model engineering technologies

Model Management

Old wine in new bottles?



Legend

Correspondences — Transformation — ------> InstanceOf

Model Management in a Nutshell

- **Model Management 2.0**
  - **Predefined** set of **generic operators**
    - Act upon models and produce models
  - **Limited** set of model **types**
    - Models and Maps
  - Model management **scripts**
    - Composition of given model management operators

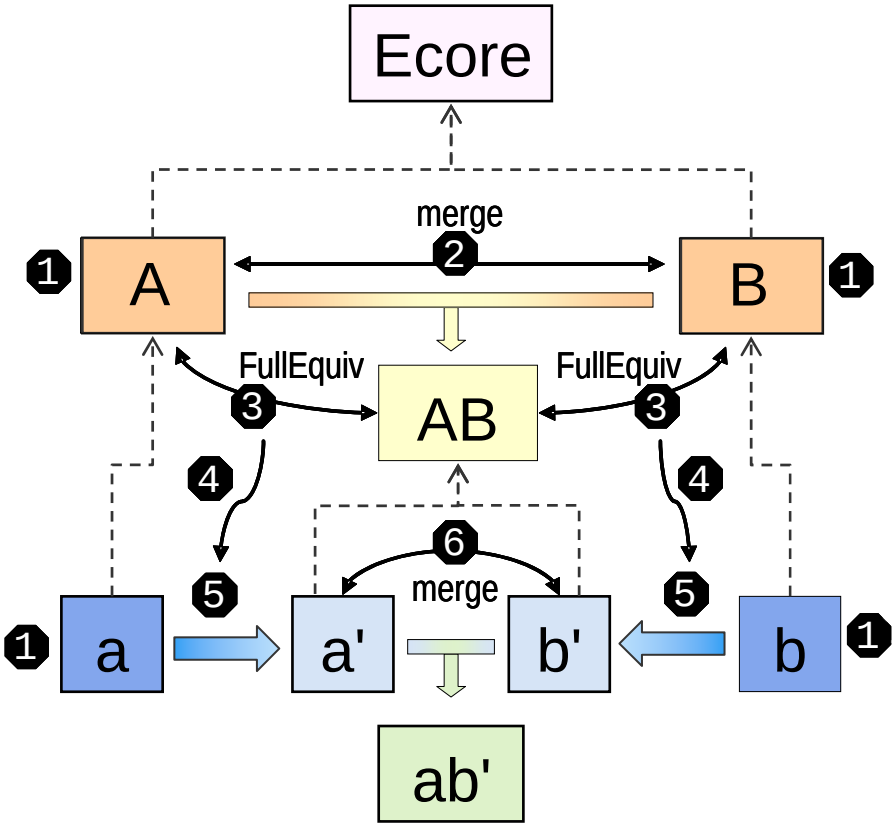
1. Extensions for Model Management 3.0

- **Scripts** are **object-oriented** programs
- **Models** are **typed** based on their **metamodels**
- **Operators** are tied to **metamodels**
- Operators **expect typed models** as input and **produce typed models** as output

Thomas Reiter, Kerstin Altmanninger, Werner Retschitzegger. ***Think Global, Act Local: Implementing Model Management with Domain-Specific Integration Languages***. Revised Selected Papers of Workshops and Symposia at MoDELS 2006, Springer LNCS 4363, pages 263-276, 2007.

Model Management

Motivating example: Exogenous merge



„Exogenous Merge“ Script

1. `Ecore A = …`
   `Ecore B = …`
   `Model a = …`
   `Model b = …`
2. `Ecore AB = A.merge(B);`
3. `FullEquiv map_a = AB.FE_match(A );`
   `FullEquiv map_b = AB.FE_match(B );`
4. `Transformation t_a = map_a.genTrafo();`
   `Transformation t_b = map_b.genTrafo();`
5. `AB a' = a.t_a();`
   `AB b' = b.t_b();`
6. `AB ab' = a'.merge(b');`

Content

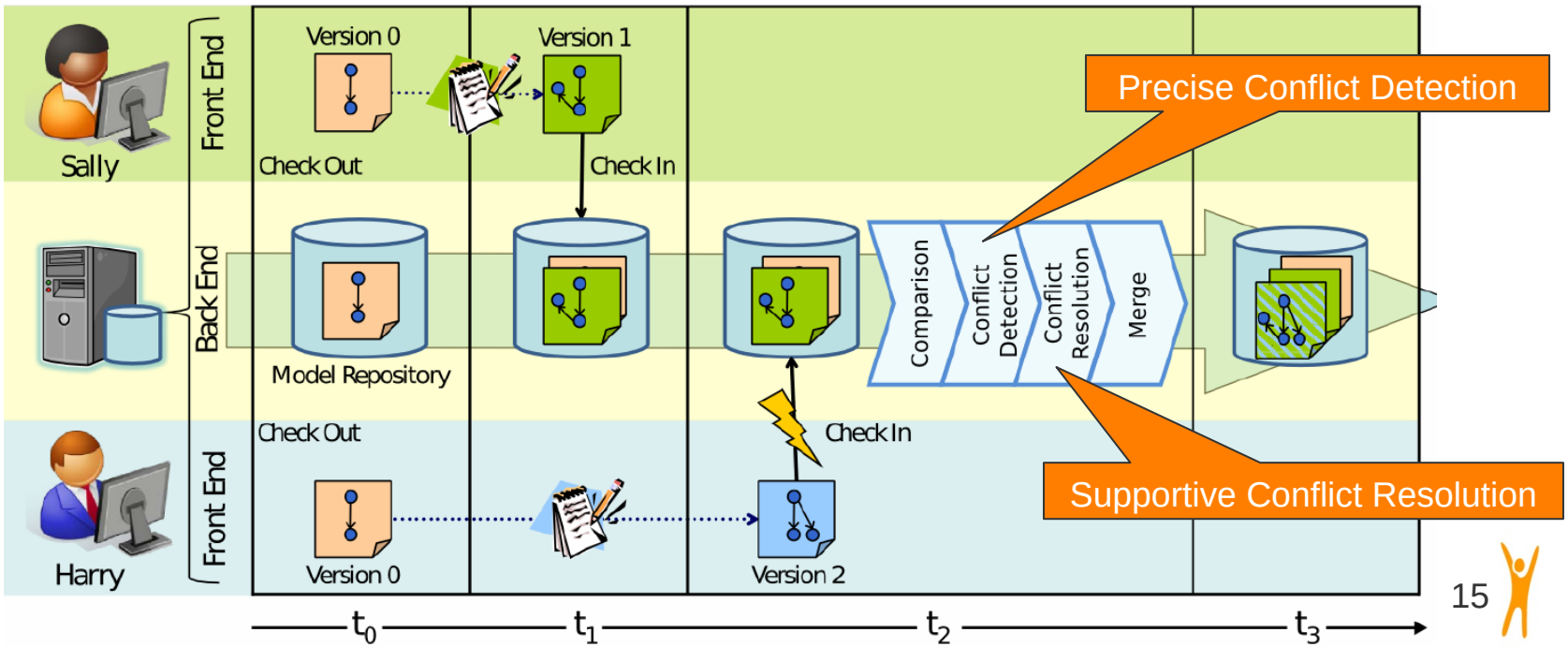
- Introduction
  - *Modeling, Model-Driven Engineering, and Model Management*
- **Model Management Use Case I: Model Versioning**
- Model Management Use Case II: Model Co-Evolution
- Résumé

Model Management Use Case I: Model Versioning

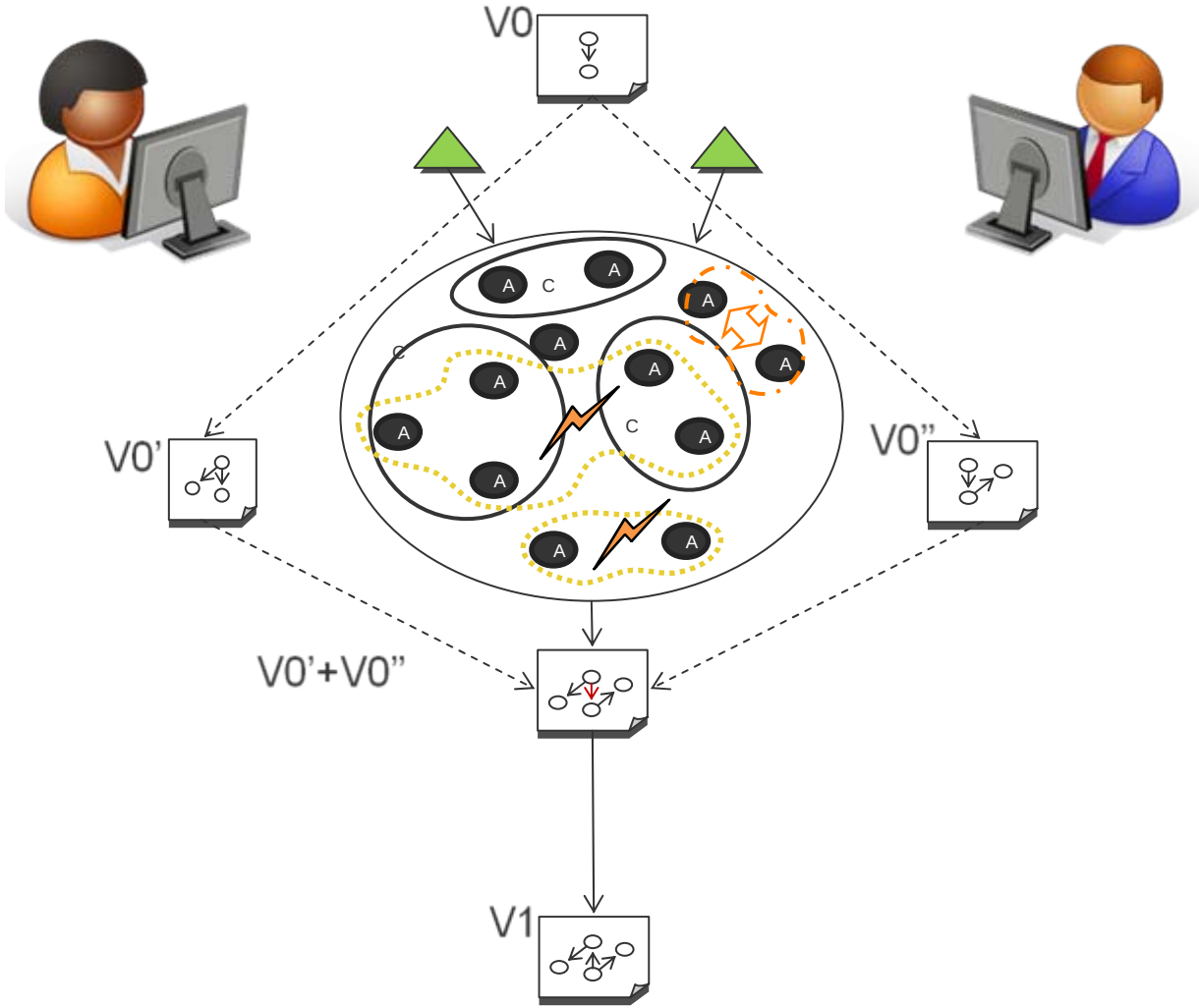
Motivation

- Some definitions of Software Engineering (SE)
  - SE is defined as the **multi-person construction** of multi-version software
    – *David Lorge Parnas*
  - SE deals with the building of software systems that are so large or so complex that they are **built by teams** of engineers
    – *Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli*

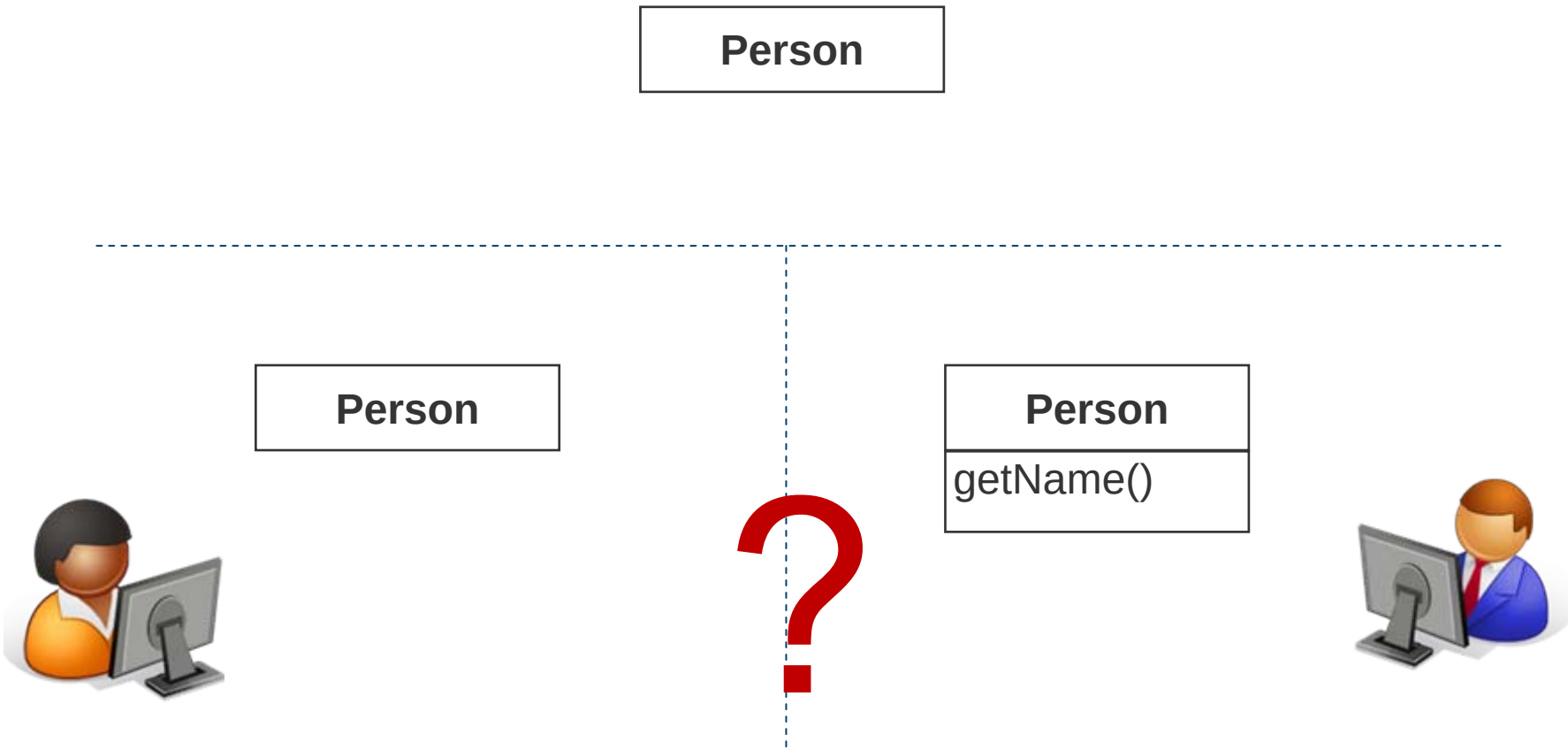
- Implication for Model Engineering: **Models must be built in teams!**



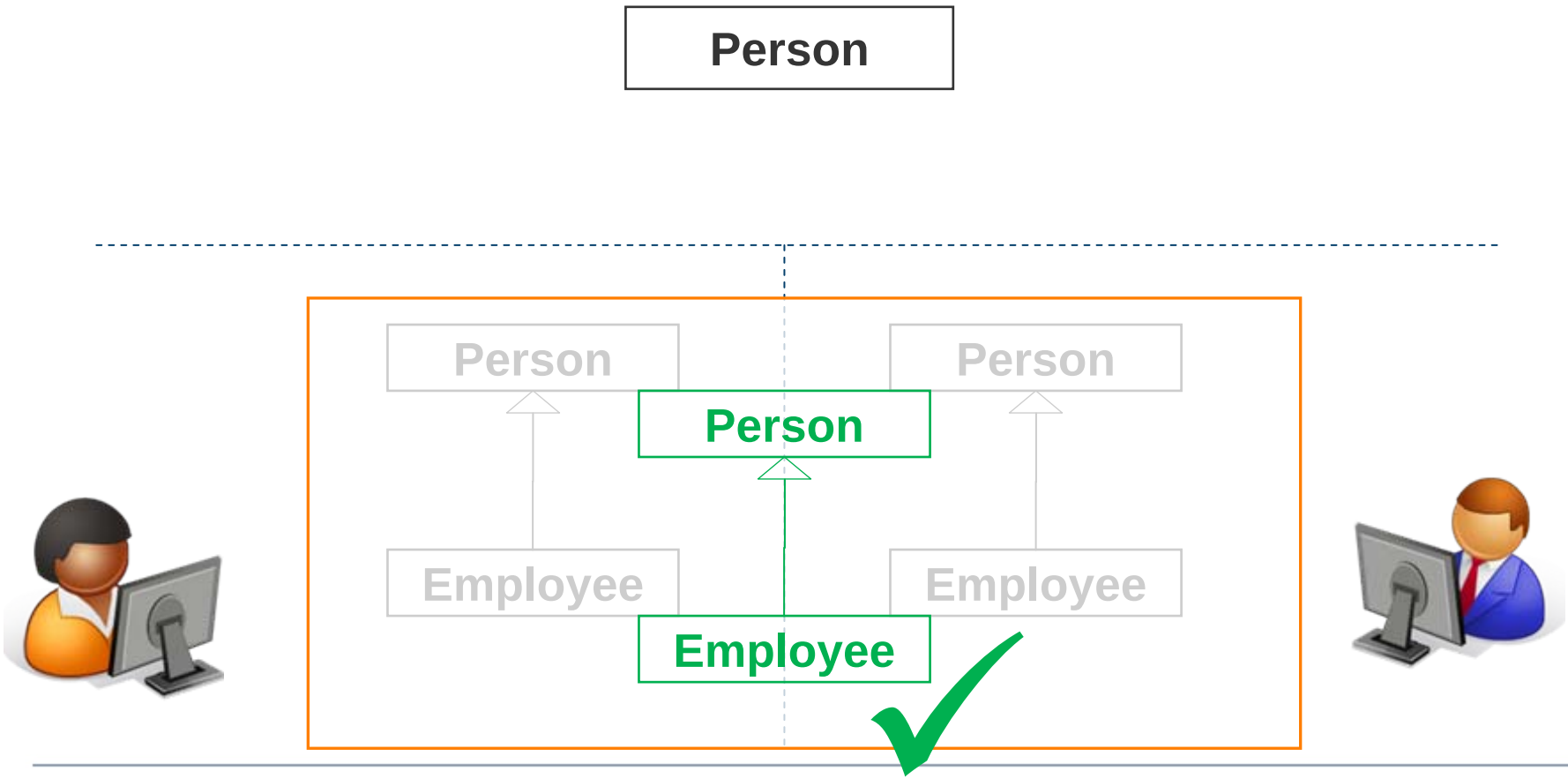
The Model Versioning Process Revised



Example 1: Contradicting Change

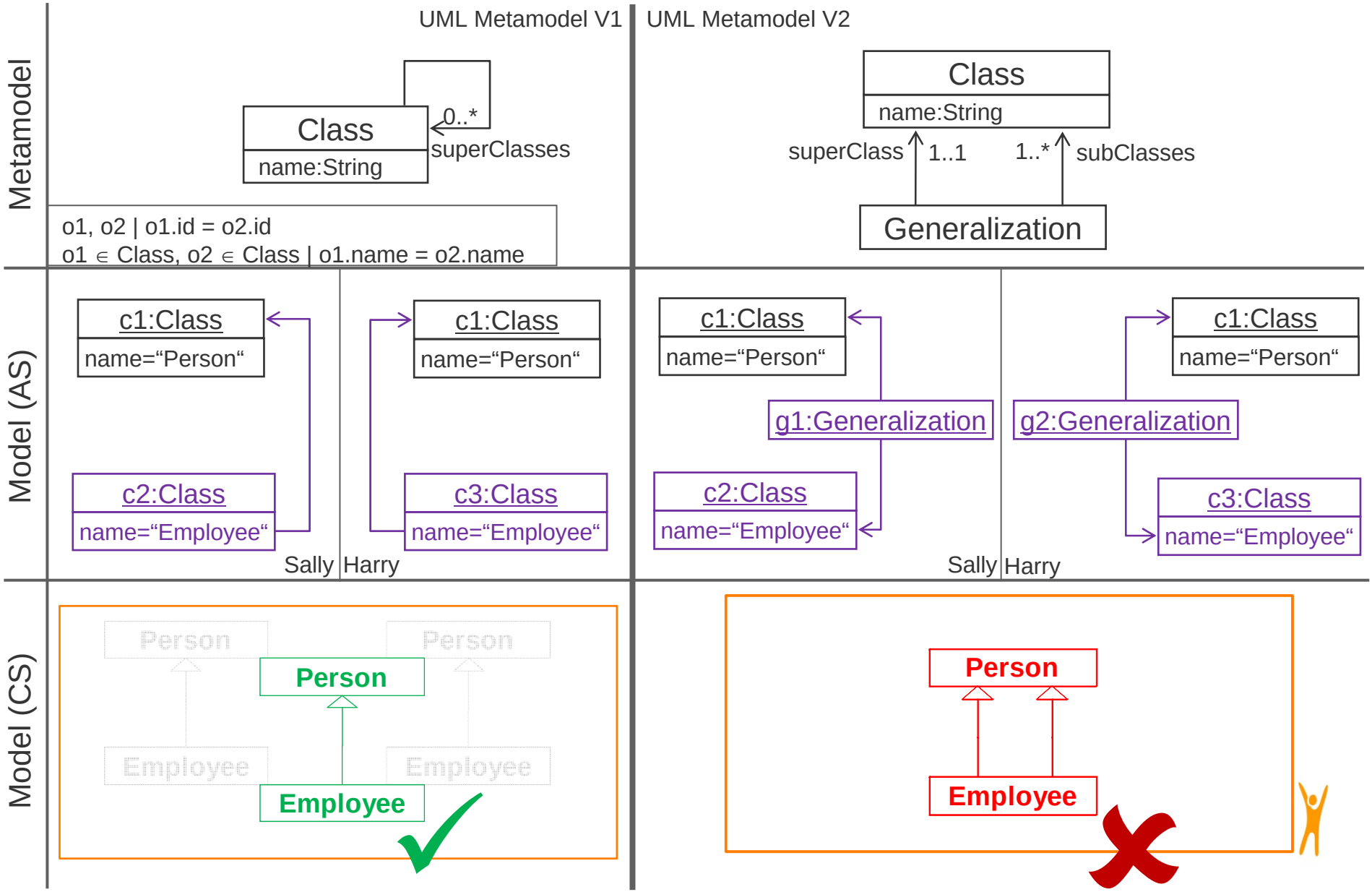


Example 2: Equivalent Change

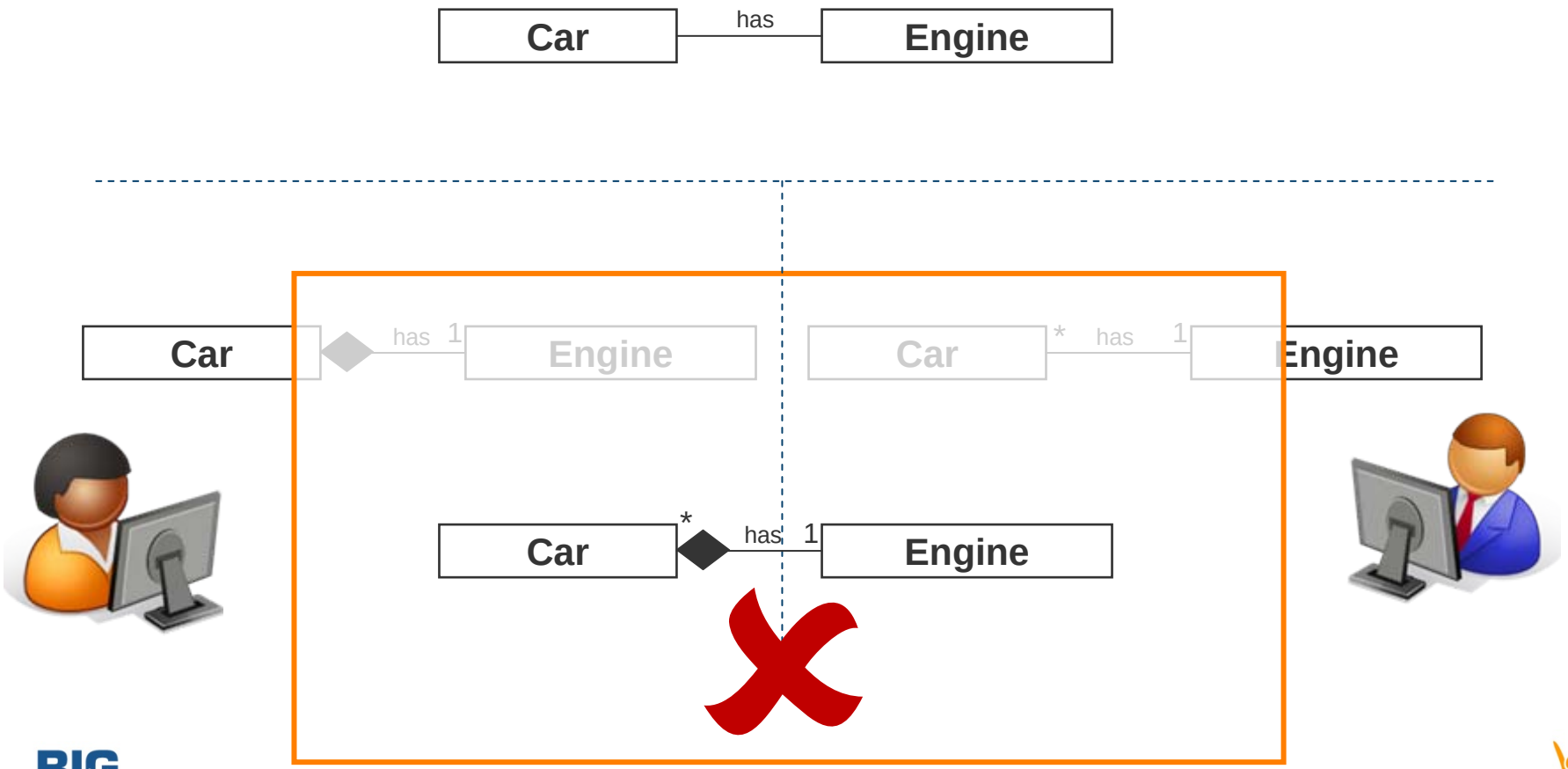


Example 2: Equivalent Change

Harder than we thought!?



Example 3: Syntactic Inconsistency

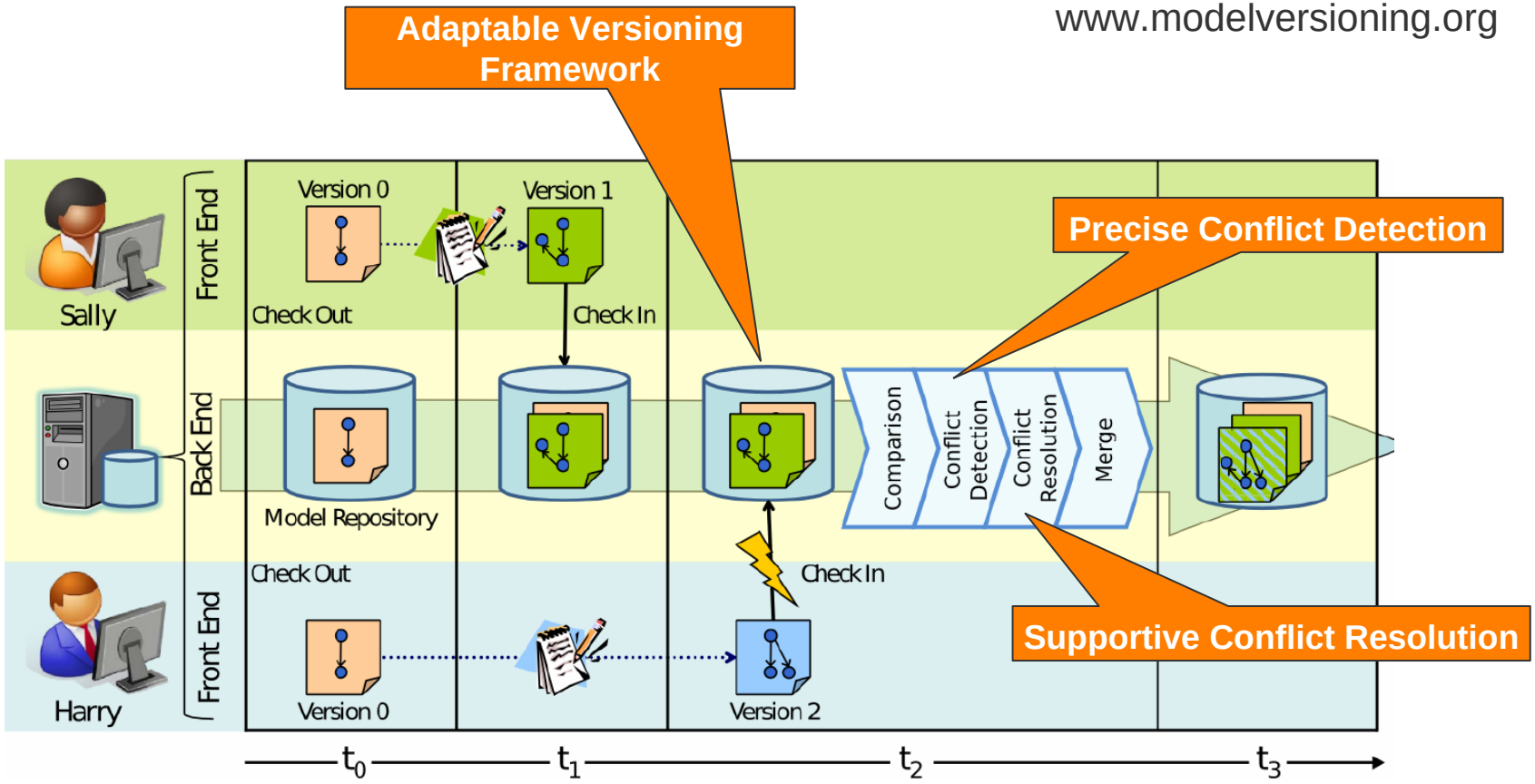


The AMOR Approach

Overview

AMOR
Adaptable Model Versioning

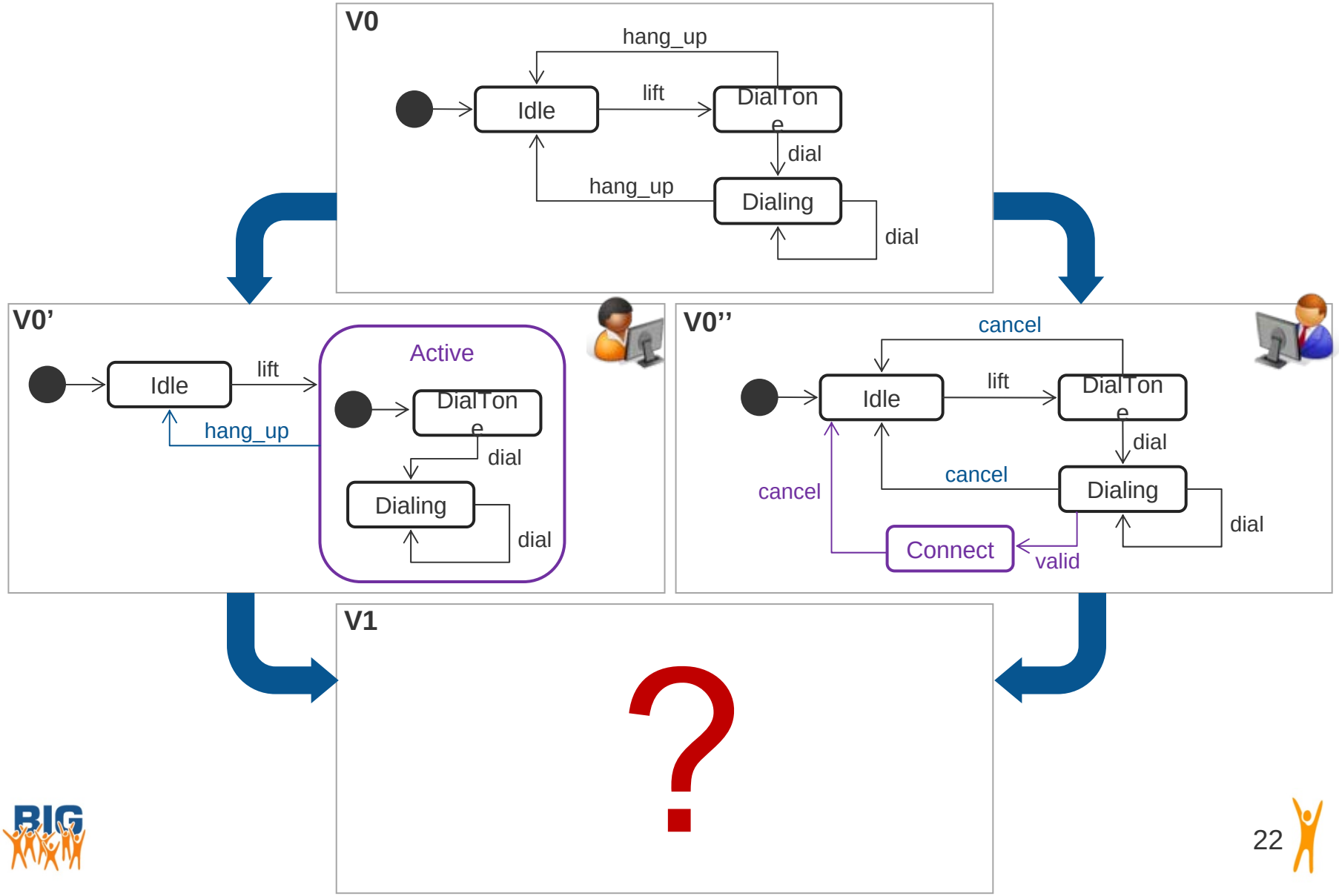
www.modelversioning.org



Gerti Kappel et al. ***Adaptable Model Versioning in Action***. In Proc. of Modellierung 2010, GI, LNI 161, pages 221-236, 2010.

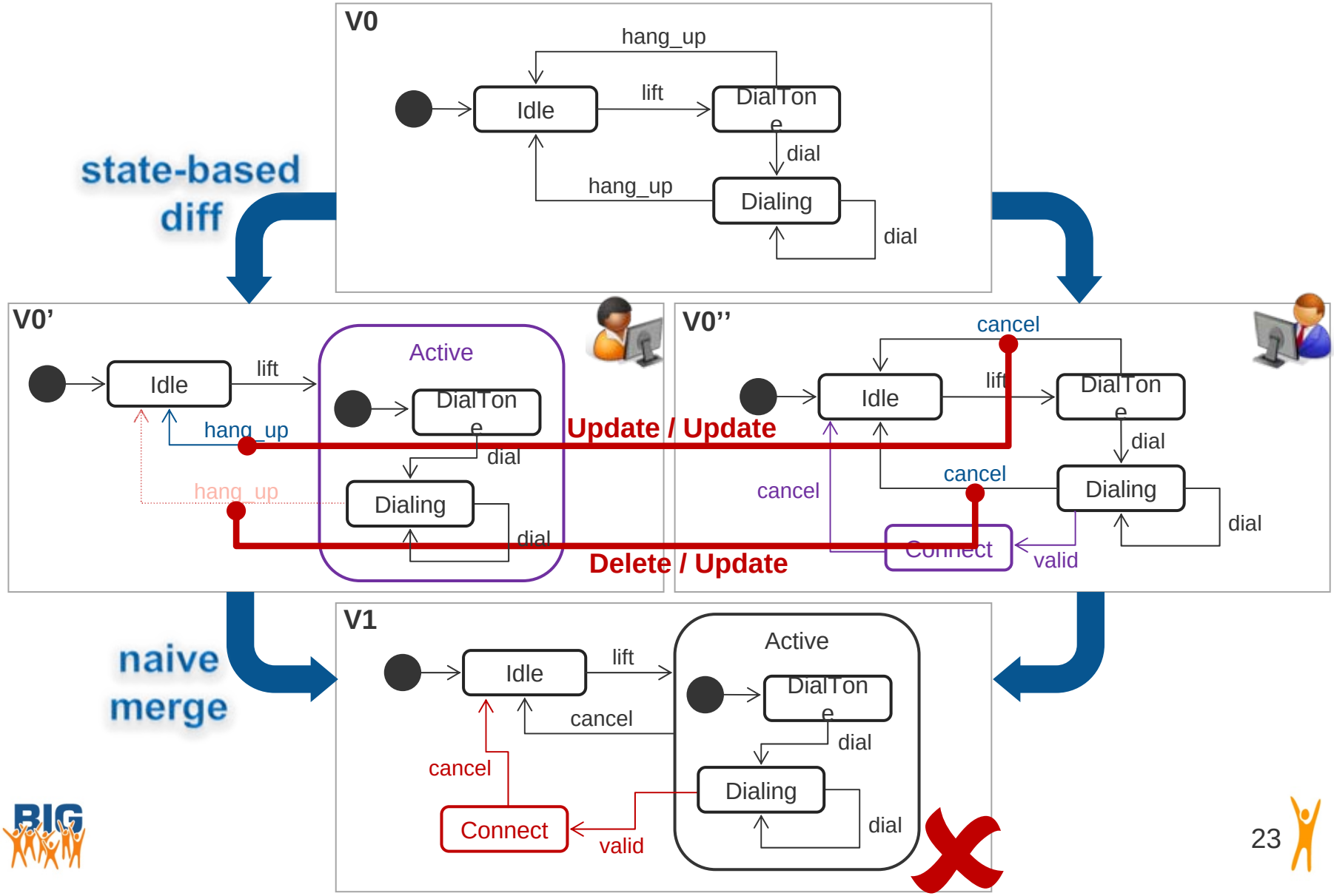
Versioning Example

State-based and generic merging



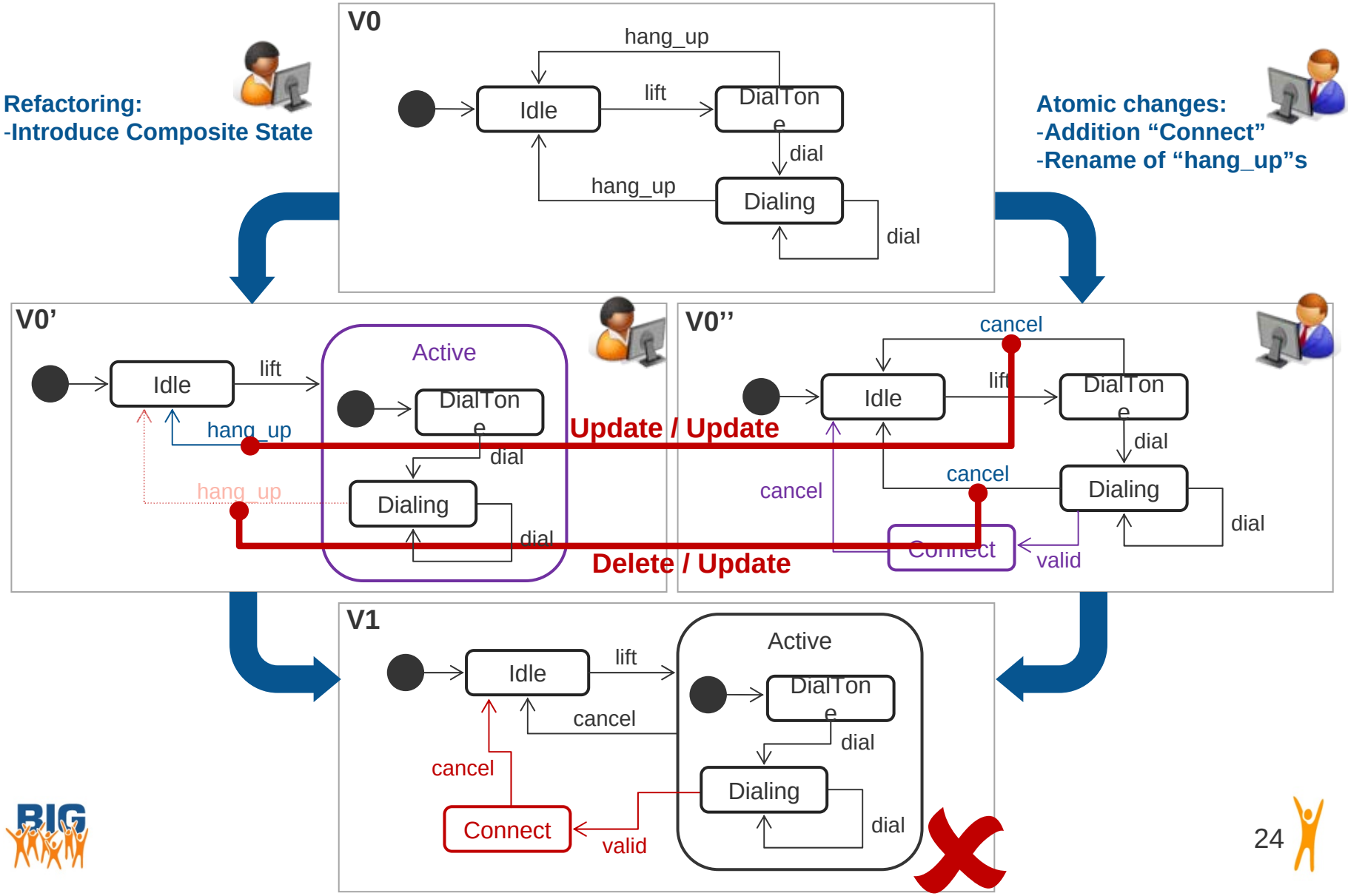
Versioning Example

State-based and generic merging



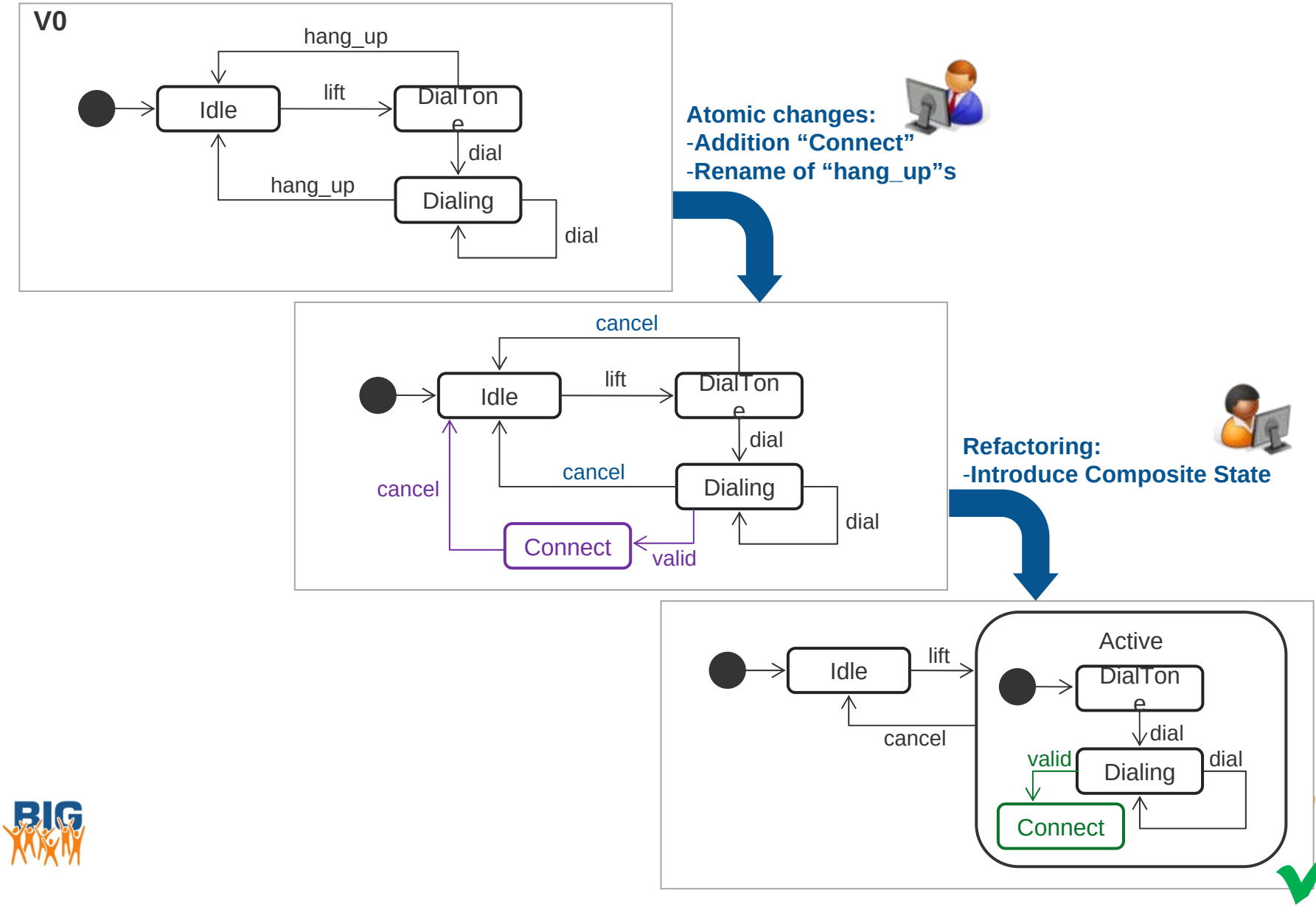
Versioning Example

State-based and generic merging



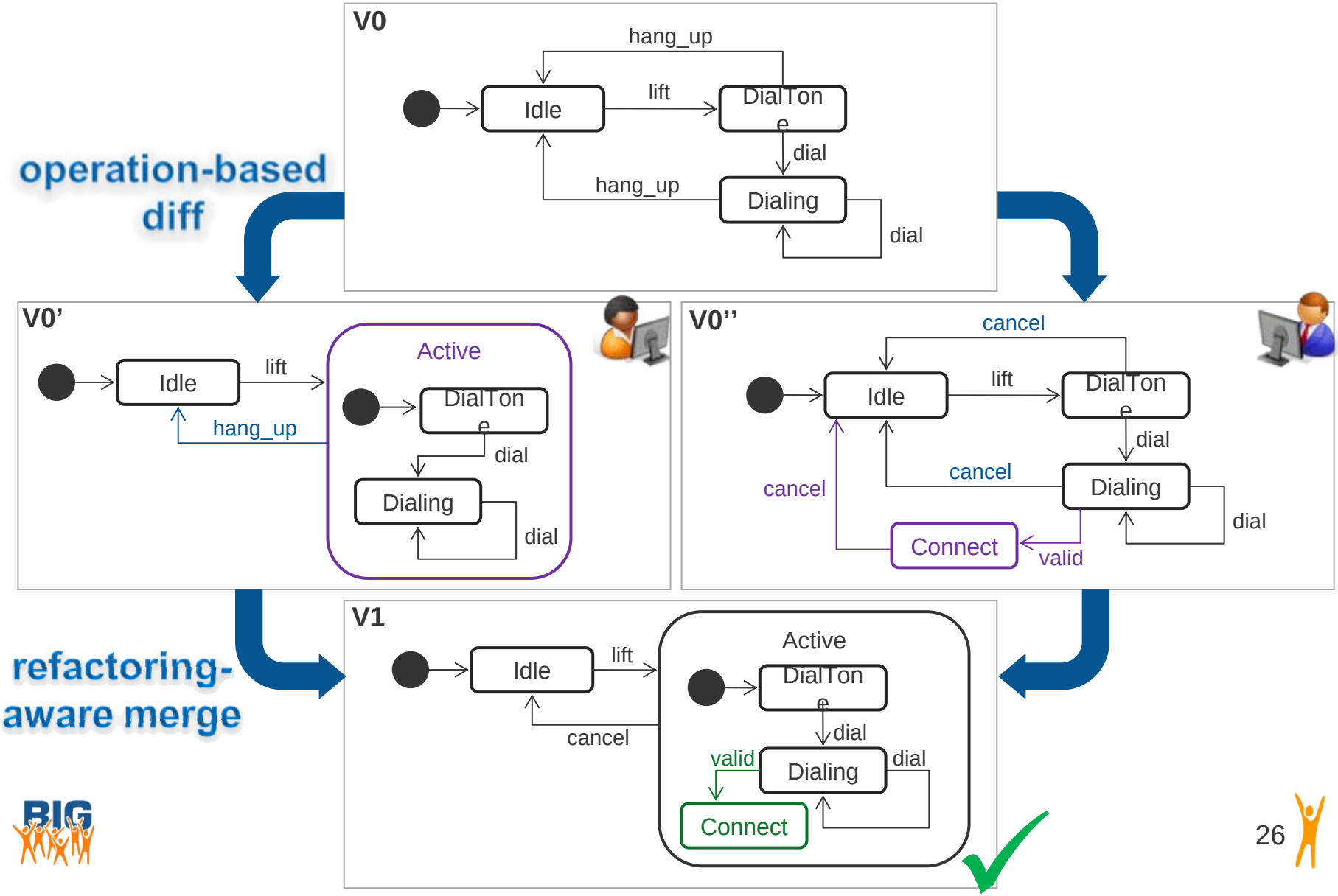
Versioning Example

Operation-based and refactoring-aware merging

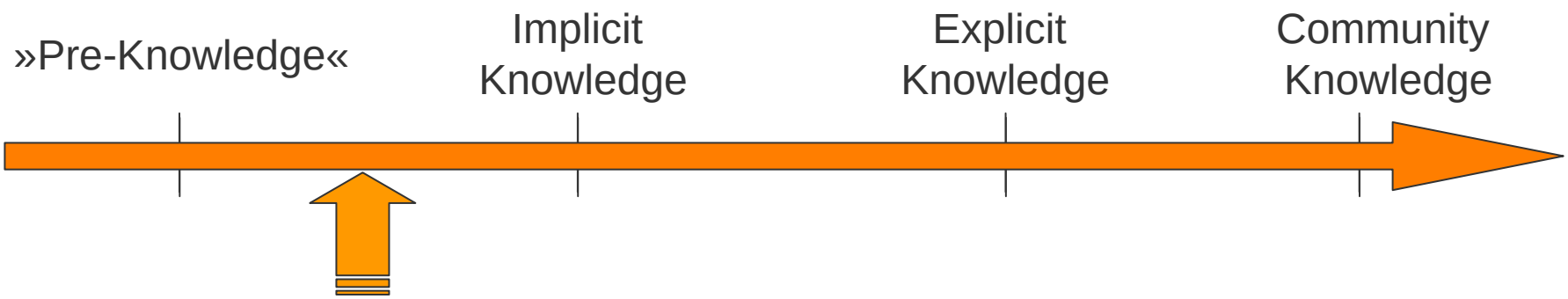


Versioning Example

Operation-based and refactoring-aware merging



Summary: Status Quo in Model Versioning Research



- Few **initial** approaches
- **No** common **terminology**
- No common problem definition
- No exactly formulated research goals
- **No test cases** for comparison of model versioning systems

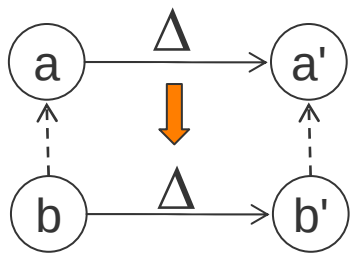
Content

Model Management Use Case II: Model Co-Evolution

Motivation

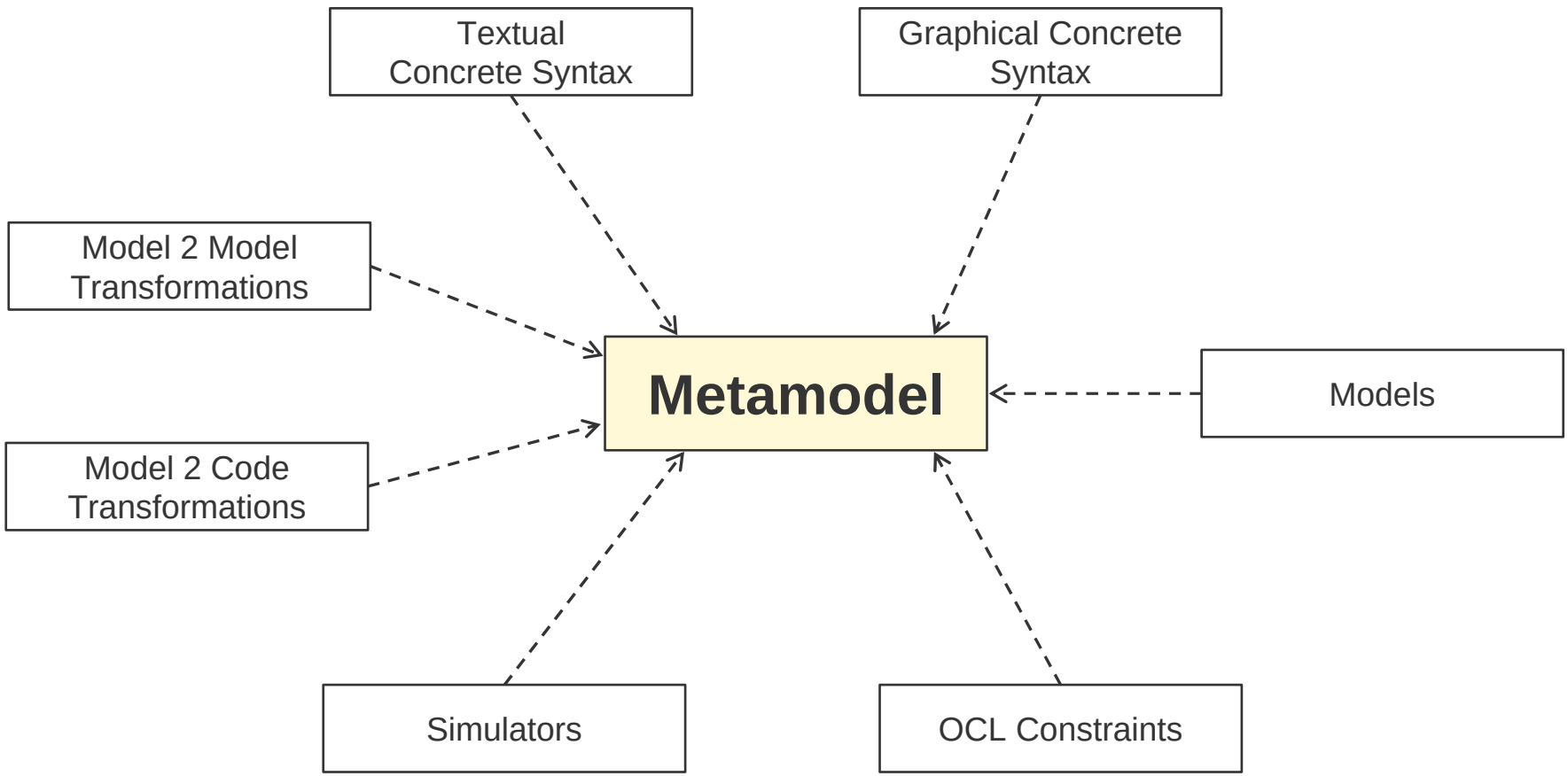
- Term “Co-Evolution” borrowed from Biology
  - Biological co-evolution is the **change** of a biological entity **triggered** by the **change** of a **related** entity
    - **One-to-one Relationships**: *Predator/Prey, Host/Symbiont, Host/Parasite, …*
    - **Diffuse Relationships**: An entity evolves in response to a number of other entities, each of which is also evolving in response to a set of entities

- Co-Evolution in MDE
  - Co-evolution is the **change** of a model **triggered** by the **change** of a **related** model
  - Current View
    - Relationship: r(a,b)
    - a $\rightarrow$ a’
    - b $\rightarrow$ b’ | r(a’,b’)
    - **Challenge: Relationship Reconciliation**
  - Current research focus is on one-to-one relationships



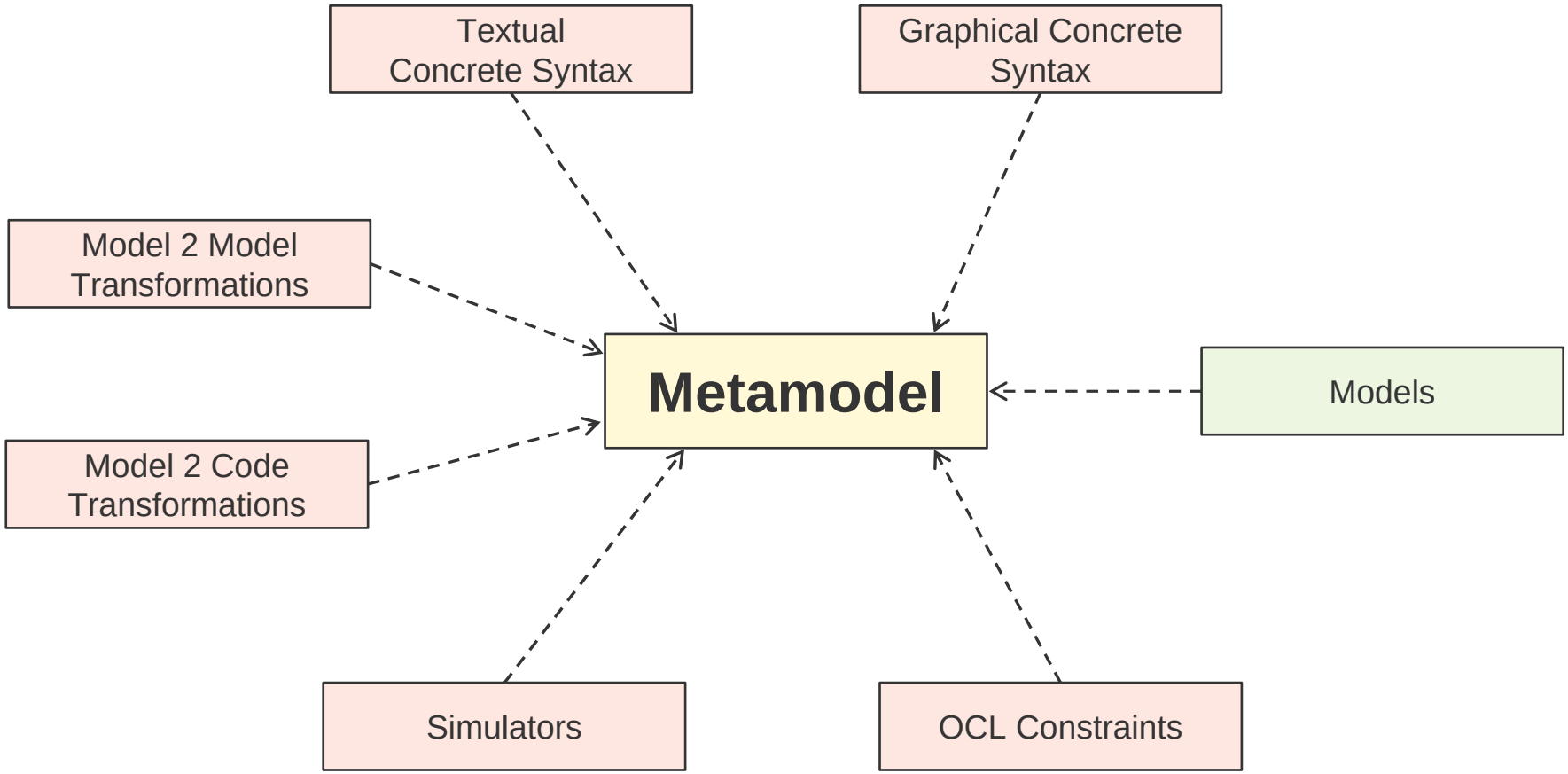
Metamodeling Level

Metamodels are the central artefacts



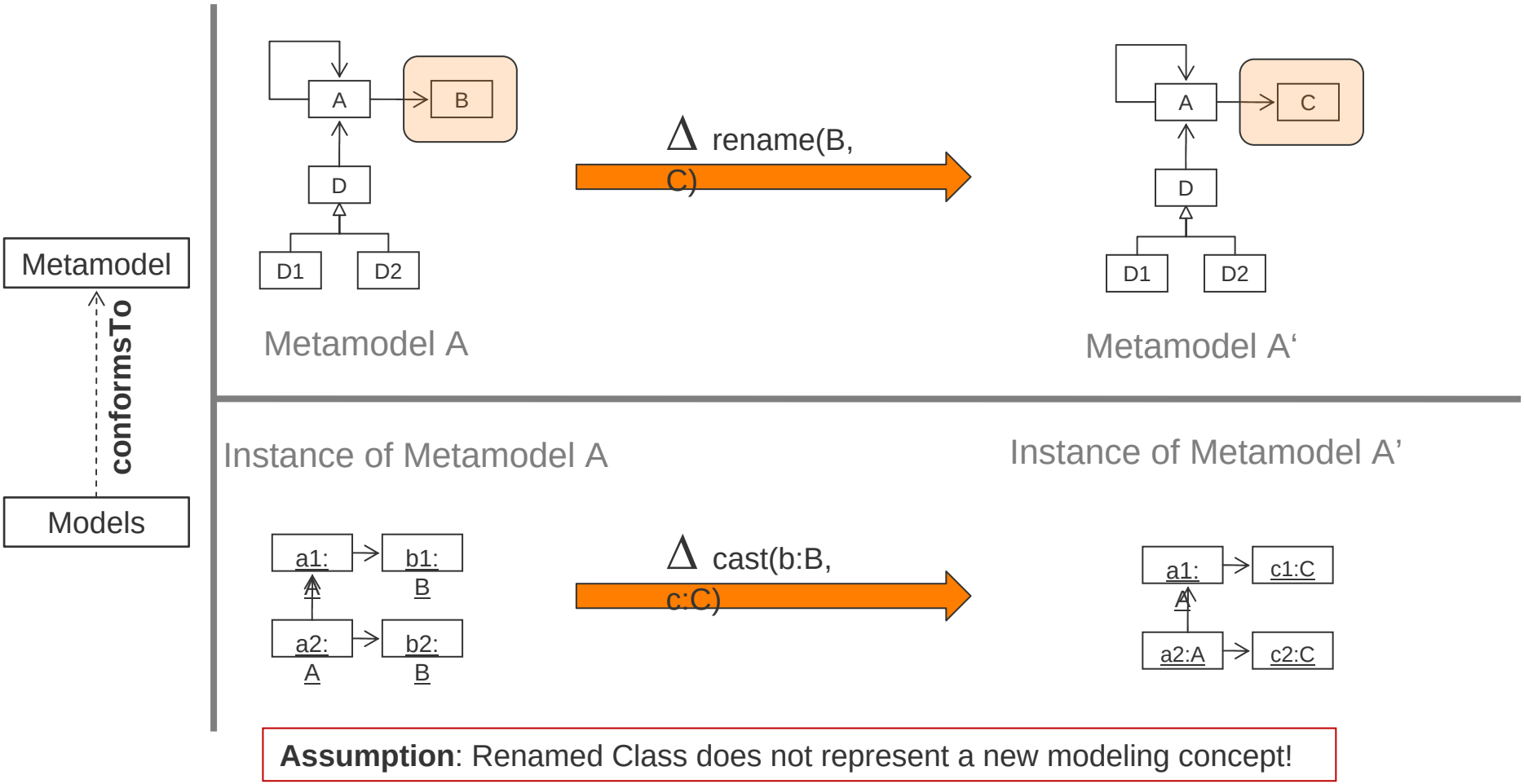
Metamodeling Level

Only models, i.e., instances of metamodels, are currently co-evolved!



Metamodel/Model (Co-)Evolution

Example

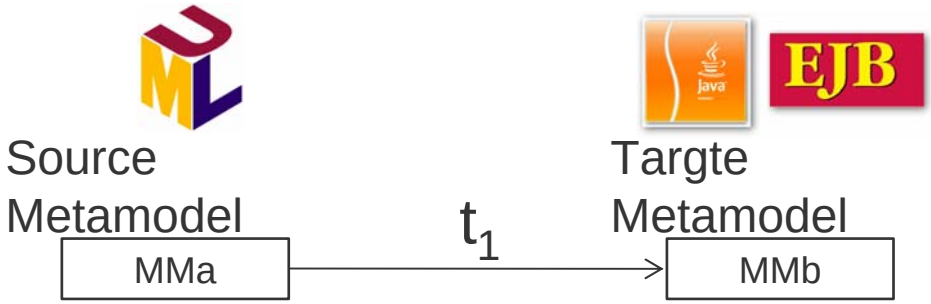


Assumption: Renamed Class does not represent a new modeling concept!

Gerti Kappel et al. ***On using Inplace Transformations for Model Co-Evolution***. In Proc. of the 2nd International Workshop on Model Transformations with ATL (MtATL) @ ICMT'10, 2010.

Metamodel/Transformation (Co-)Evolution

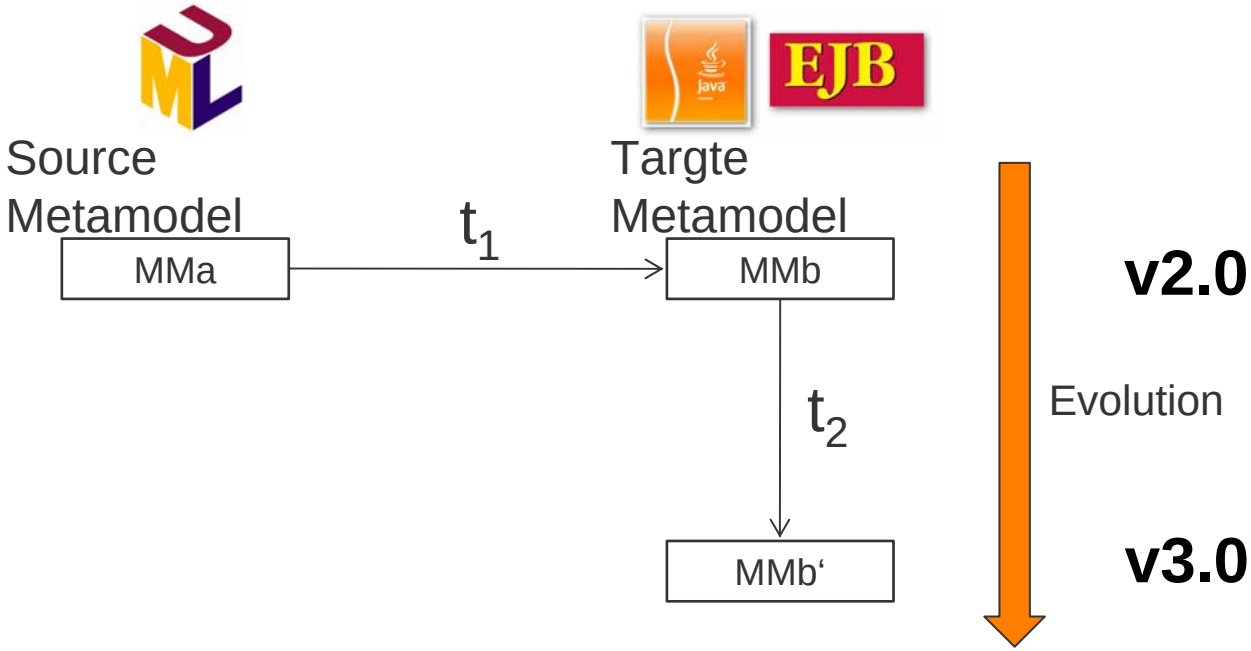
Initial example



t_1 … Forward Transformation

Metamodel/Transformation (Co-)Evolution

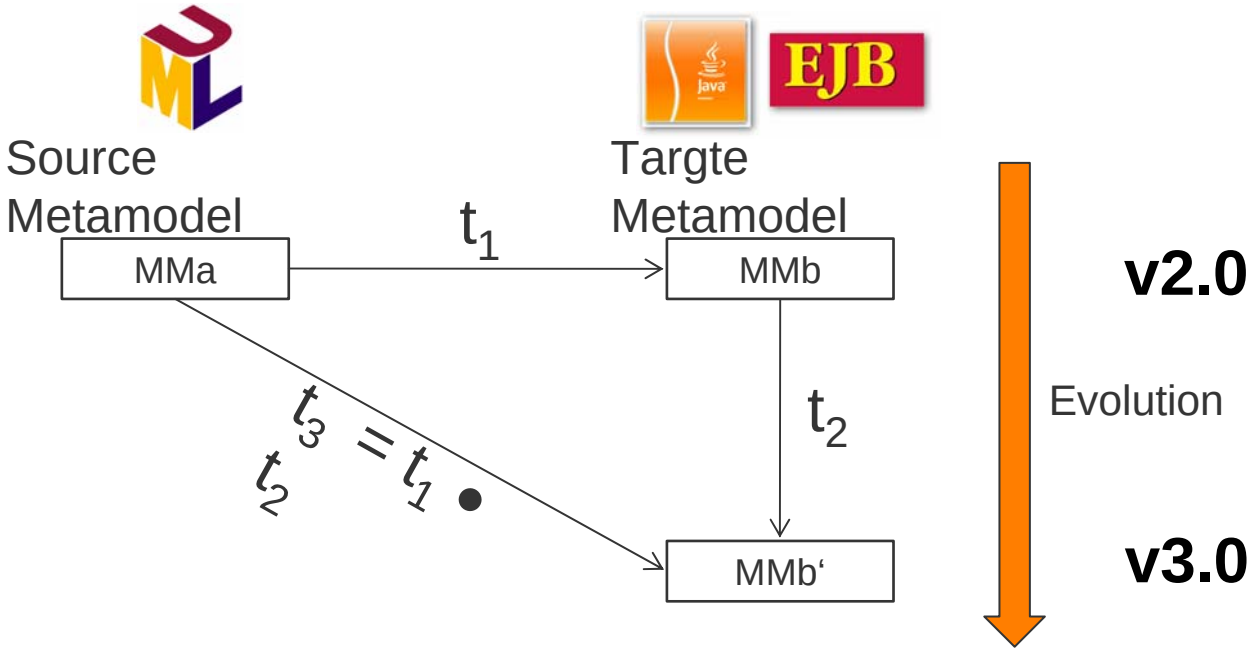
Target metamodel evolution



t_1 … Forward Transformation
t_2 … Migration Transformation

Metamodel/Transformation (Co-)Evolution

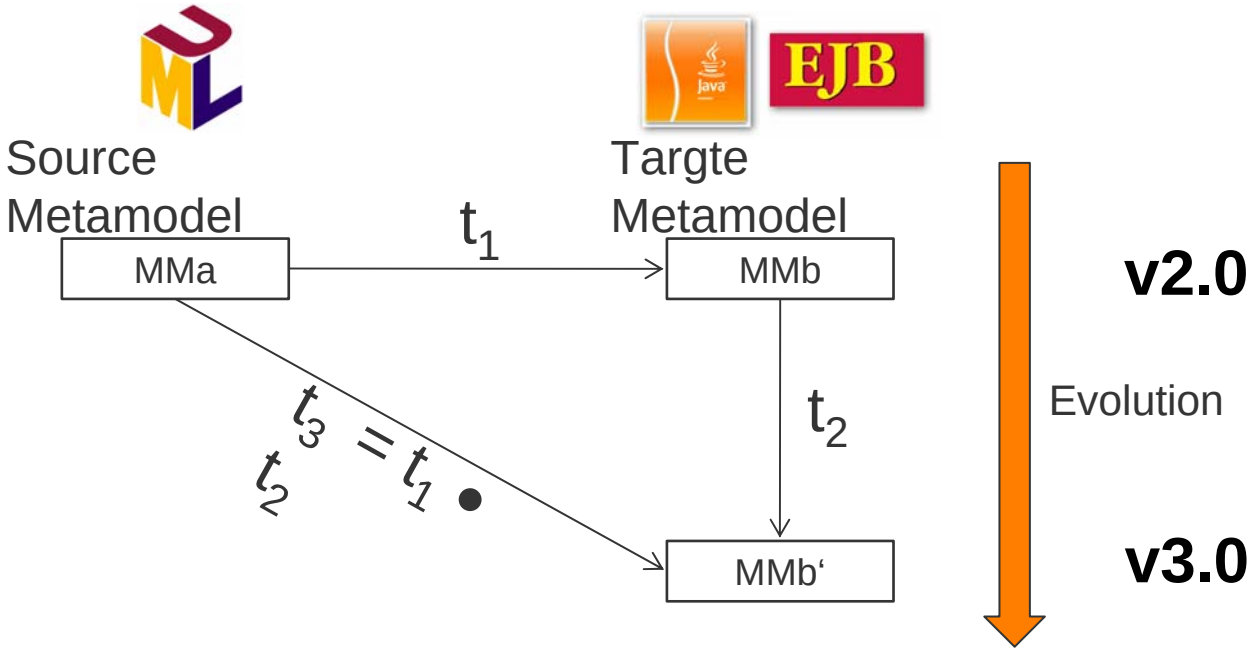
Transformation composition for evolving existing transformations



t_1 … Forward Transformation
t_2 … Migration Transformation

Metamodel/Transformation (Co-)Evolution

First initial results for composing graph transformations



Requirements on t_3

(1)Soundness and completeness

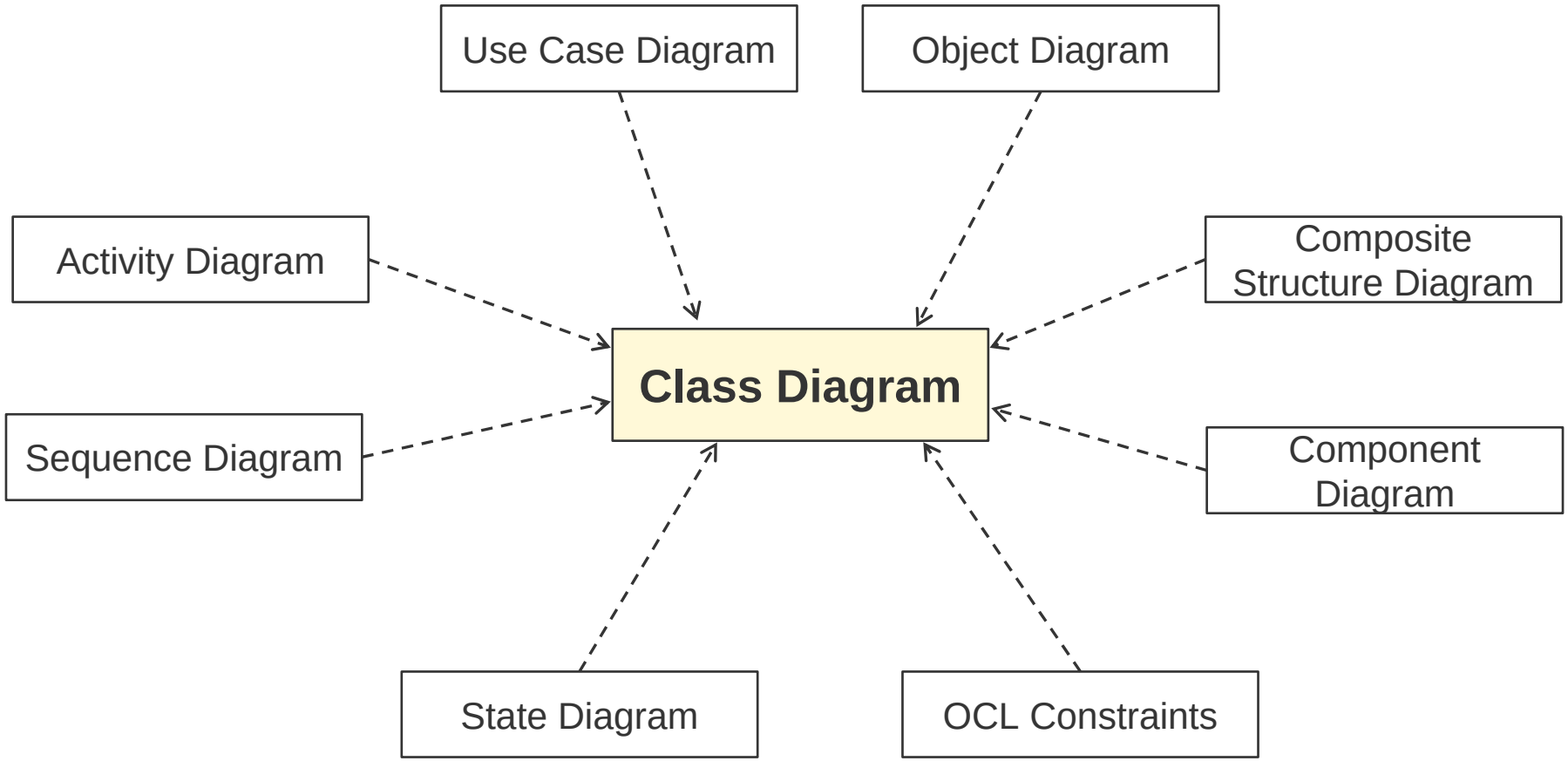
(2)Elimination

(3)Transitivity

Manuel Wimmer et al. ***Towards Transformation Rule Composition***. In Proc. of the 4th International Workshop on Multi-Paradigm Modeling (MPM) @ MoDELS‘10, 2010.

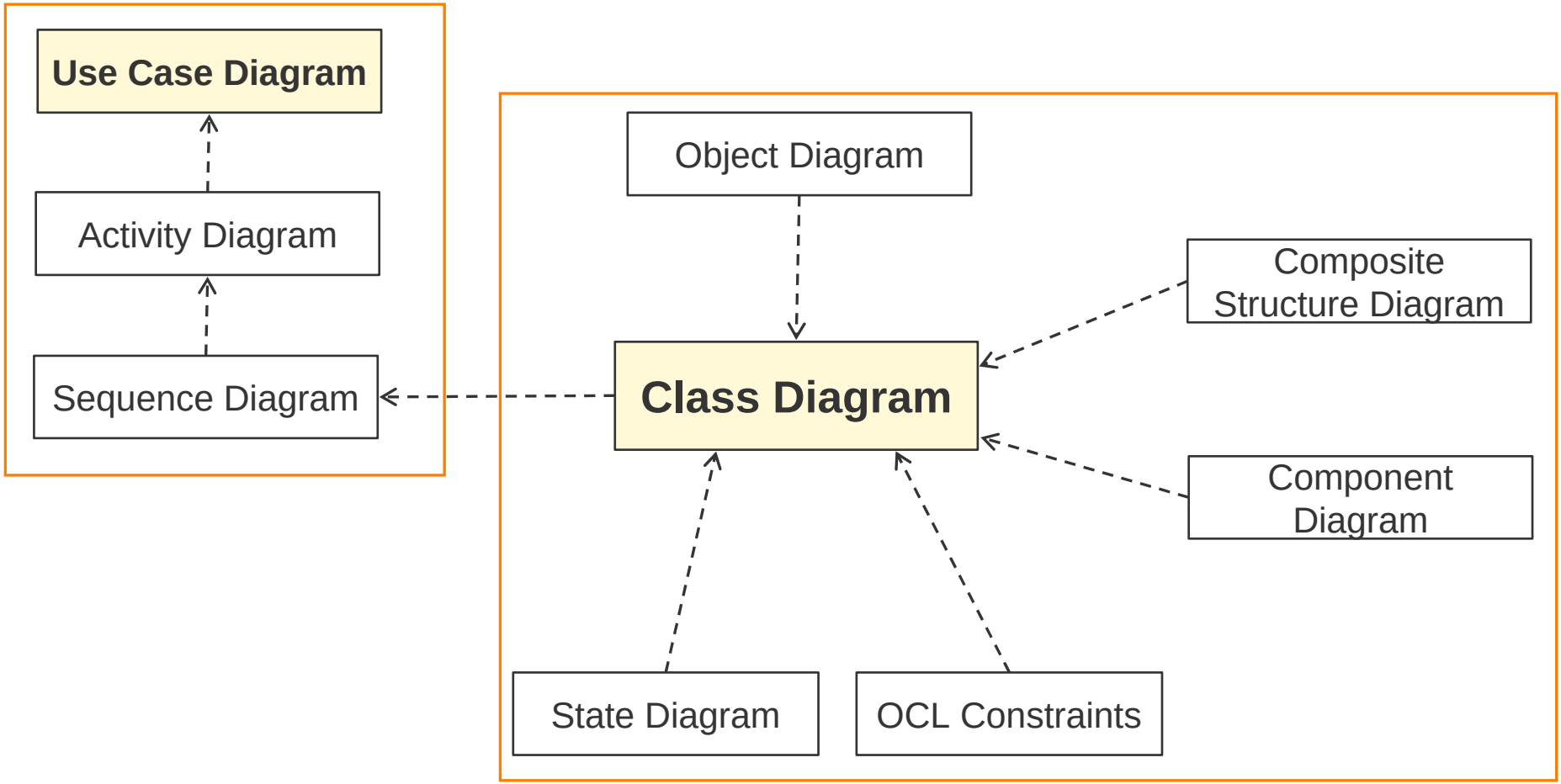
Modeling Level using UML

Question 1: *Are class diagrams in the centre of the development process?*



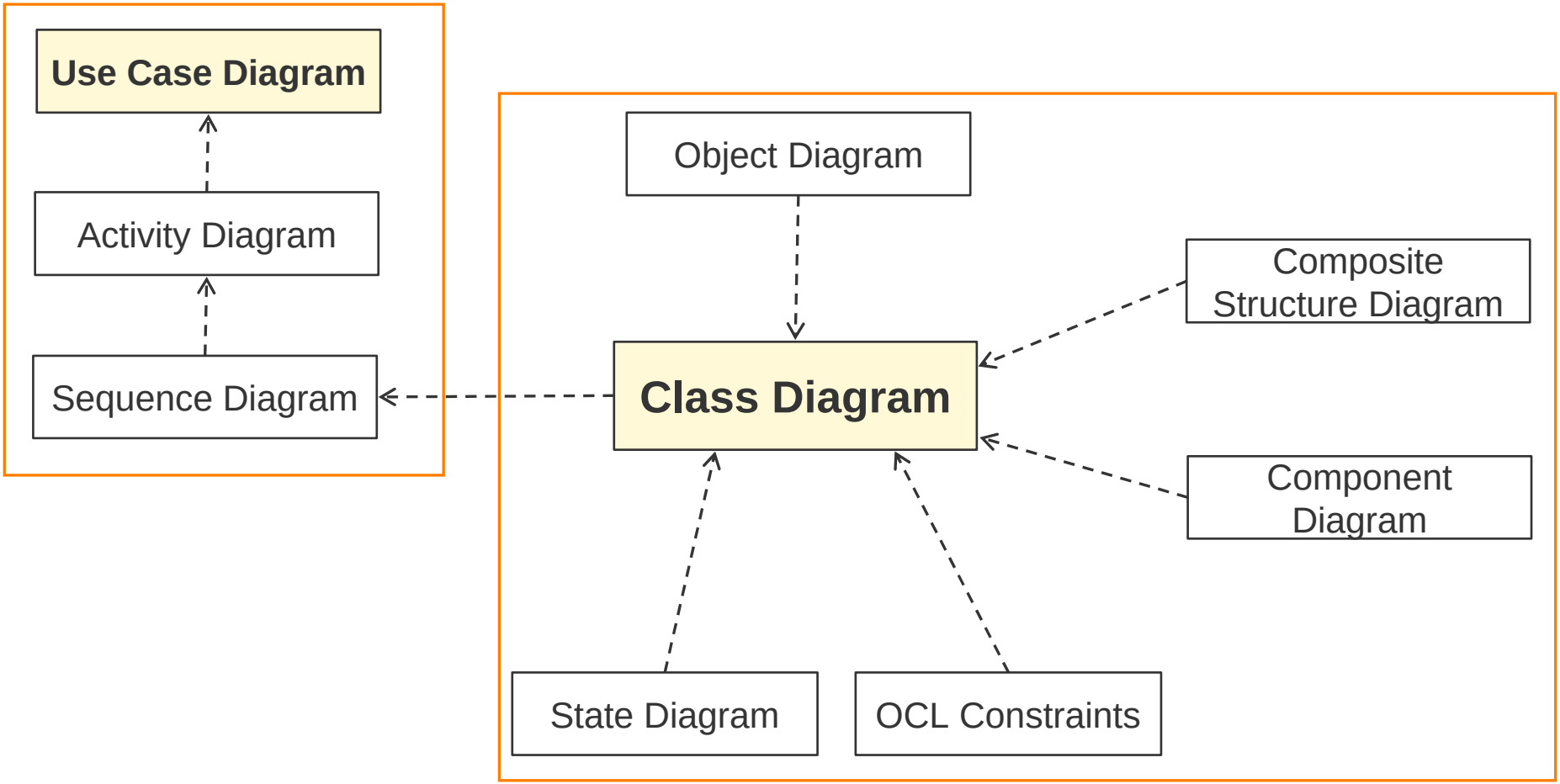
Modeling Level using UML

Question 1: *Are class diagrams in the centre of the development process?*



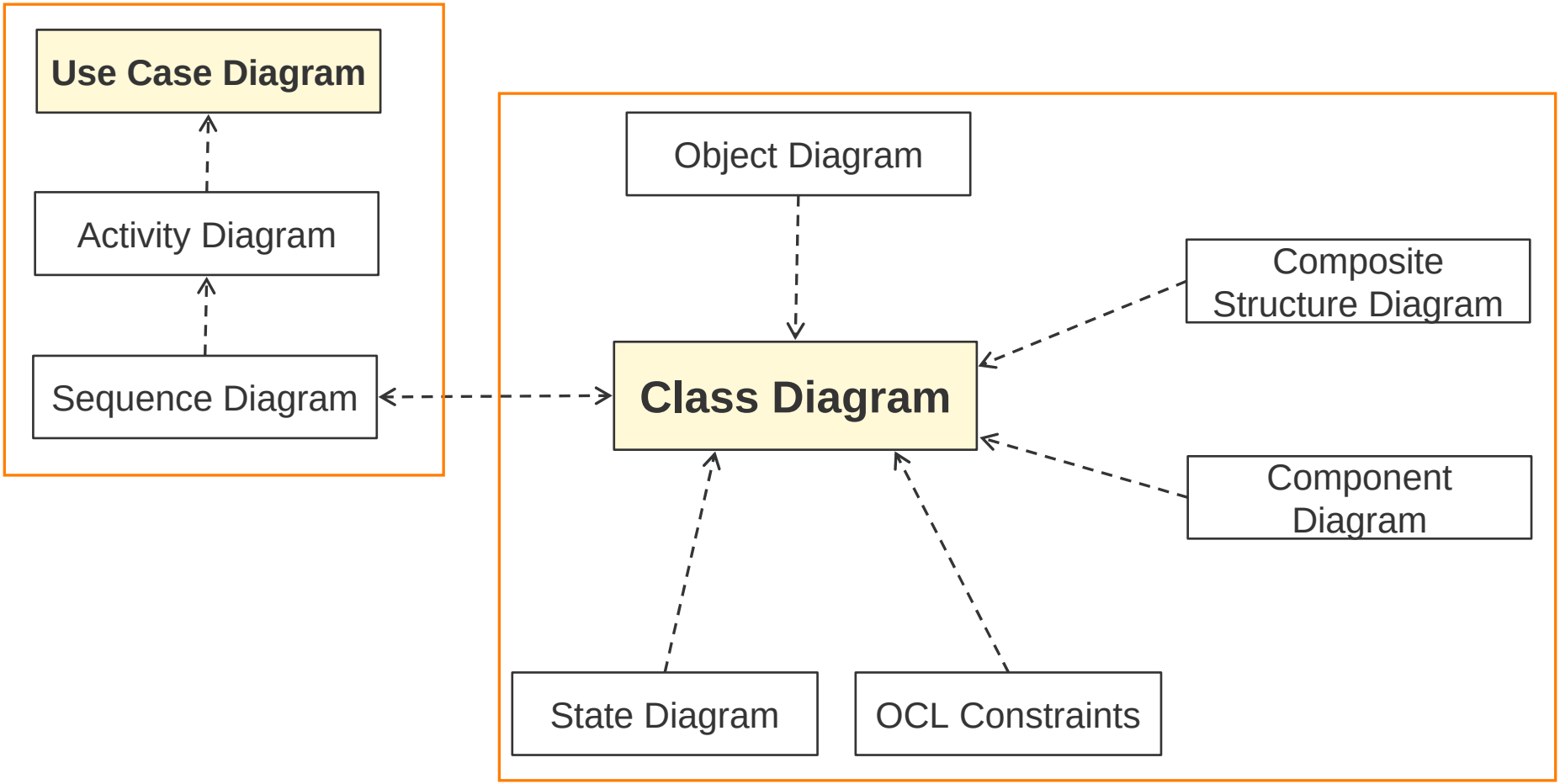
Modeling Level using UML

Question 2: *Do we have one-to-one relationships or diffuse relationships?*



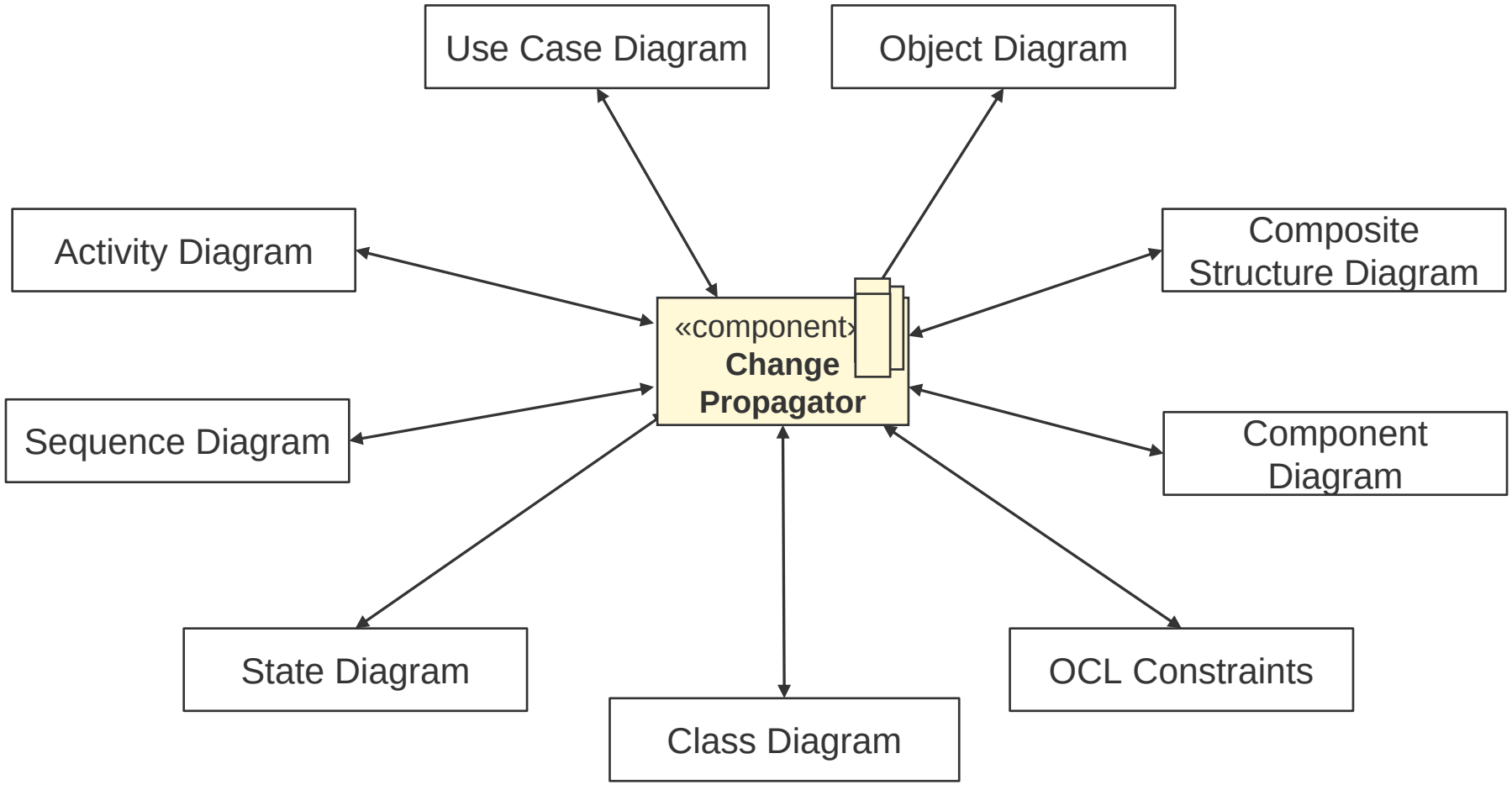
Modeling Level using UML

Question 2: *Do we have one-to-one relationships or diffuse relationships?*



Modeling Level using UML

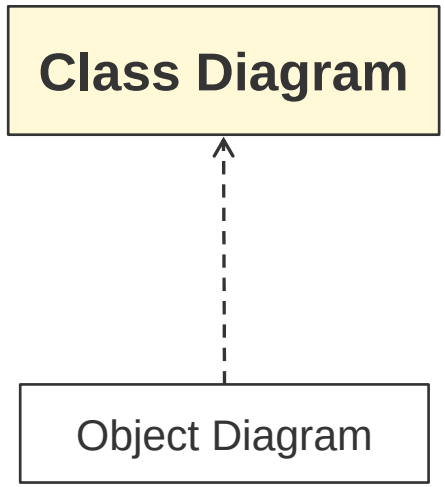
Question 2: *Do we have one-to-one relationships or diffuse relationships?*



Modeling Level using UML

Question 3: *Who is the host and who is the parasite?*

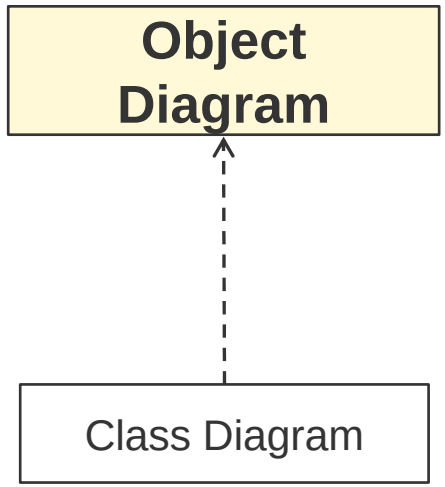
- Traditional Database Engineering View
  - Use the schema for populating the instances!
- **Consequence**
  - Each change in the Class Diagram has to be propagated to the Object Diagram



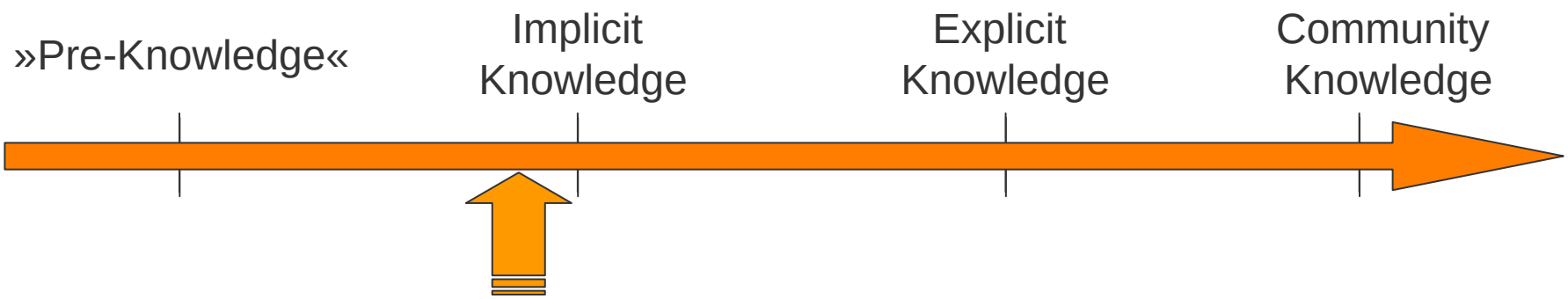
Modeling Level using UML

Question 3: *Who is the host and who is the parasite?*

- Prototype-based Engineering View
  - Abstract the schema from the instances!
- **Consequence**
  - Some changes in the Object Diagram have to be to propagated to the Class Diagram



Summary: Status Quo in Model Co-Evolution Research



- Several **initial** approaches
- Metamodel/model (co-)evolution has been solved for syntactical issues
- **Huge number** of **different** co-evolution scenarios
- Often **no** exactly **formulated relationships** between models
- **No** publicly accessible **model repositories** for studying model co-evolution

Content

Résumé

Model management infrastructure is the prerequisite for tackling evolution issues

- Find a **basic set** of model management operators
  - *Diff, Merge, Patch*, …
  - Explore **variations** and **properties** of these operators
- **Extend** existing **tools** with **model management capabilities**
  - Provide a **set of predefined** operators
  - Provide a **common** programming model for script development
- Based on these scripting languages, provide tool support for
  - *Versioning, Co-evolution, Merging*, …
  - Users should be enabled to **adapt** predefined scripts

Résumé

- Provides terminology for **evolution concerns**
- Allows to reason on a **high-level of abstraction**

- **Variations** of operators are a must
- Says nothing about **implementation** of operators

- Some operators seem to be **magical** (e.g., *match*, *merge*, …)
- **No** approved programming model for script development

Résumé

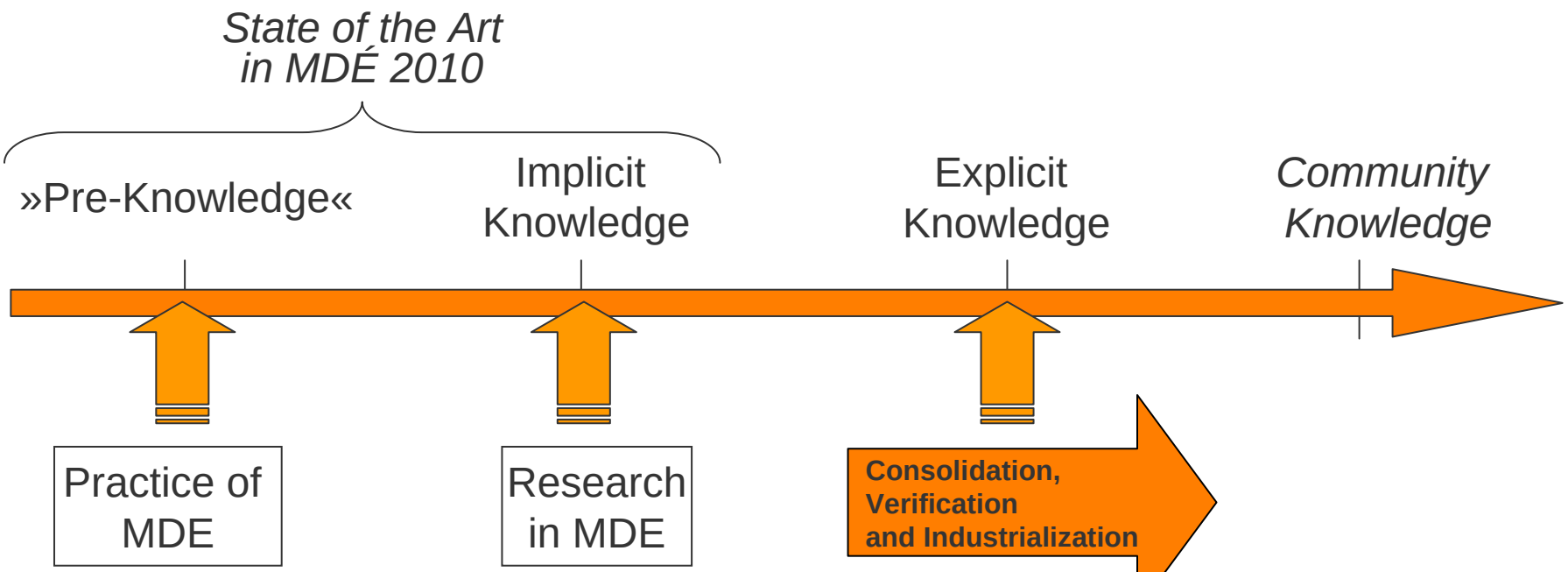
Future work needed!

Many research questions **remain open** in the **field of MDE**

- *What are the most important model management/evolution **scenarios**?*
- *What is the **sufficient set** of model management operators?*
- *What is an **appropriate programming model** for model management scripts?*
- *How to **implement** model management operators?*
- *How to **verify** model management operators?*
- *Is a **generic** model management approach feasible?*

Résumé

Model Management in Model-Driven Engineering – Still enough to do :-!



- Appropriateness of some standards questionable (QVT, UML2) – not yet adopted
- CASE-tool vendors jump on the MDE bandwagon

- Many different proposals, application areas and goals
- E.g., Model Management, Mega-Modeling, Runtime Models, AOM, MDWE, DSM, …

Thanks to …

- Petra Brosch
- Horst Kargl
- Philip Langer
- Thomas Reiter
- Werner Retschitzegger
- Wieland Schwinger
- Martina Seidl
- Konrad Wieland
- Manuel Wimmer
- and many more …