Department of Telecooperation

Issues and Challenges in Model Versioning

Tutorial at iiWAS2009/MoMM2009
(Kuala Lumpur, Malaysia)
14.12.2009

Dipl.-Ing. Kerstin Altmanninger
(Presenter)

Dipl.-Ing. Kerstin Altmanninger

Johannes Kepler University Linz
Department of Telecooperation (TK)
Altenberger Straße 69, 4040 Linz, Austria

Phone: +43 (0)70/2468-9262
Fax: +43 (0)70/2468-9829
Email: kerstin.altmanninger@jku.at
WWW: http://www.tk.jku.at

Projects:

- ModelCVS: A Semantic Infrastructure for Model-based Tool Integration
- AMOR – Adaptable Model Versioning

PhD thesis:

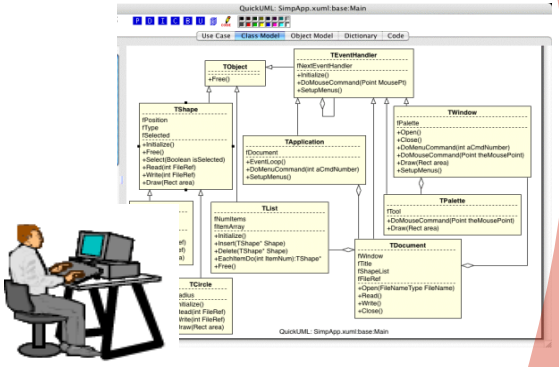
Models in Conflict – A Semantically Enhanced Version Control System for Model Artifacts

Outline

- **Introduction to Model-driven Engineering**
- **Features of Version Control Systems**
  - Top-level Features
  - 3-way Merge Features
- **Challenges in Model Versioning**
  - Existing Systems
  - Requirements
  - Issues
- **Discussion**
- **AMOR Project**
- **Literature**

Introduction to Model-driven Engineering

Problem #1: Problem Area vs. Solution Area



modeler

GAP

GAP

GAP

Problem

- *How do analyze models associate with code?*
- *How does changes in the code affect the model?*
- *Are models consistent with the code?*
- *…*

Analysis → Design → Implementation → Test → …

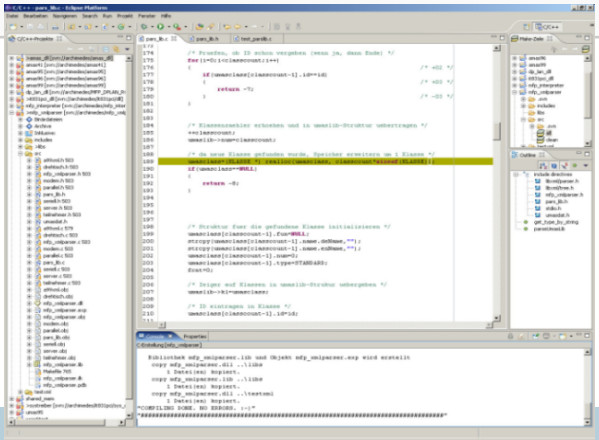
Traditional Development

GAP

GAP

GAP

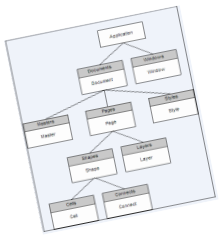
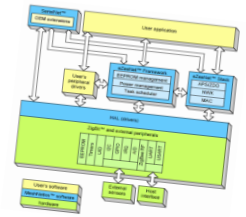
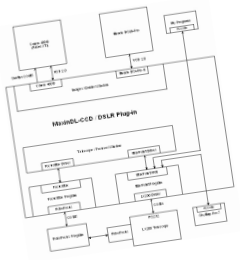
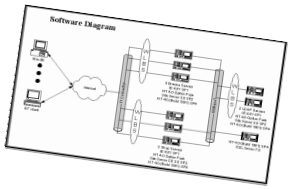
programmer

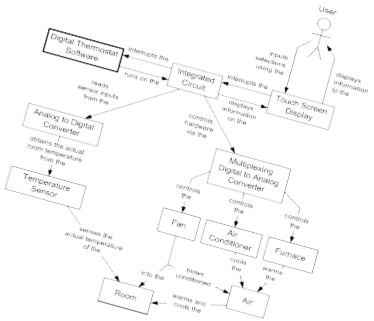


Problem #2: Modeling as a Necessary Evil

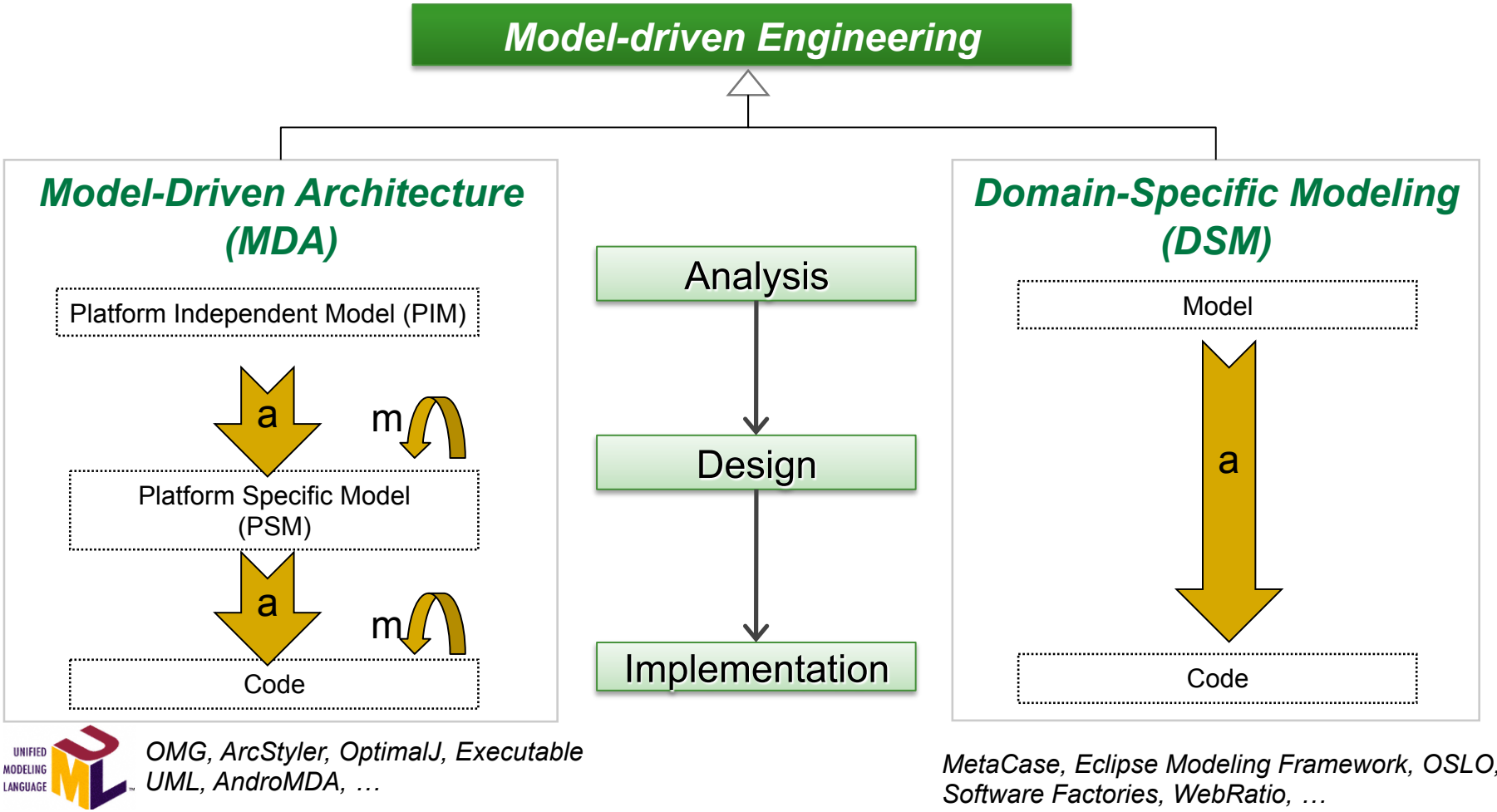
- »When it comes down to it, the real point of software development is cutting code«
- »Diagrams are, after all, just pretty pictures«
- »No user is going to thank you for pretty pictures; what a user wants is software that executes«

[M. Fowler, ”UML Distilled”, 1st edition, Addison Wesley, 1997]



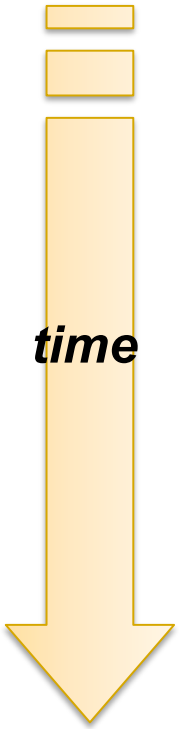
Solution: Model-driven Engineering (MDE)



OMG, ArcStyler, OptimalJ, Executable UML, AndroMDA, …

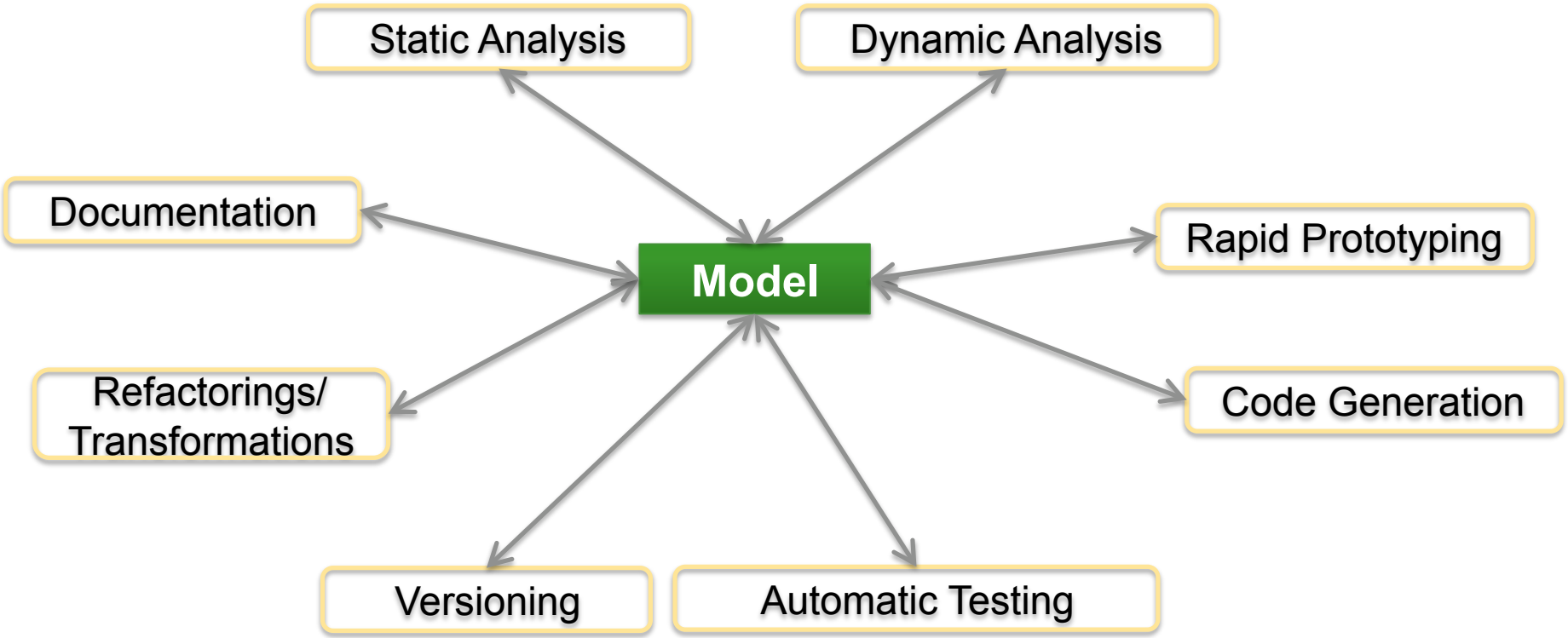
MetaCase, Eclipse Modeling Framework, OSLO, Software Factories, WebRatio, …

Effect #1: Evolution of Modeling



- ***Models as sketch***
  - Communication of ideas/alternatives
  - Goal: Modeling
- ***Models as templates***
  - Documentation of design decisions
  - Goal: Implementation guide
- ***Models as programs***
  - Generation of applications automatically
  - Goal: Models are source code and vice versa

Effect #2: Model as the Central Artifact



Effect #3: Models are more than just Pictures

Foundation: Formal language definition

- ***Modeling languages: MOF, Ecore, …***

Automatic model processing (e.g., PIM -> PSM, refactoring)

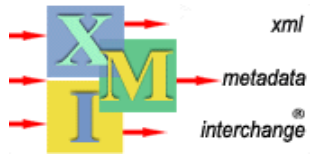
- ***Model transformation languages: QVT, ATL, …***

From the model to software

- ***Code generation languages: JET, Xpand, …***

Persistent model storage and model exchange

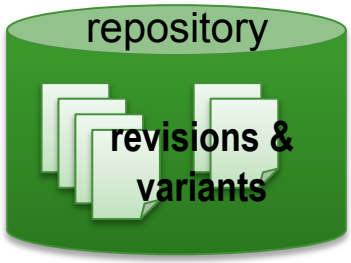
- ***Model serialization/deserialization: XMI***



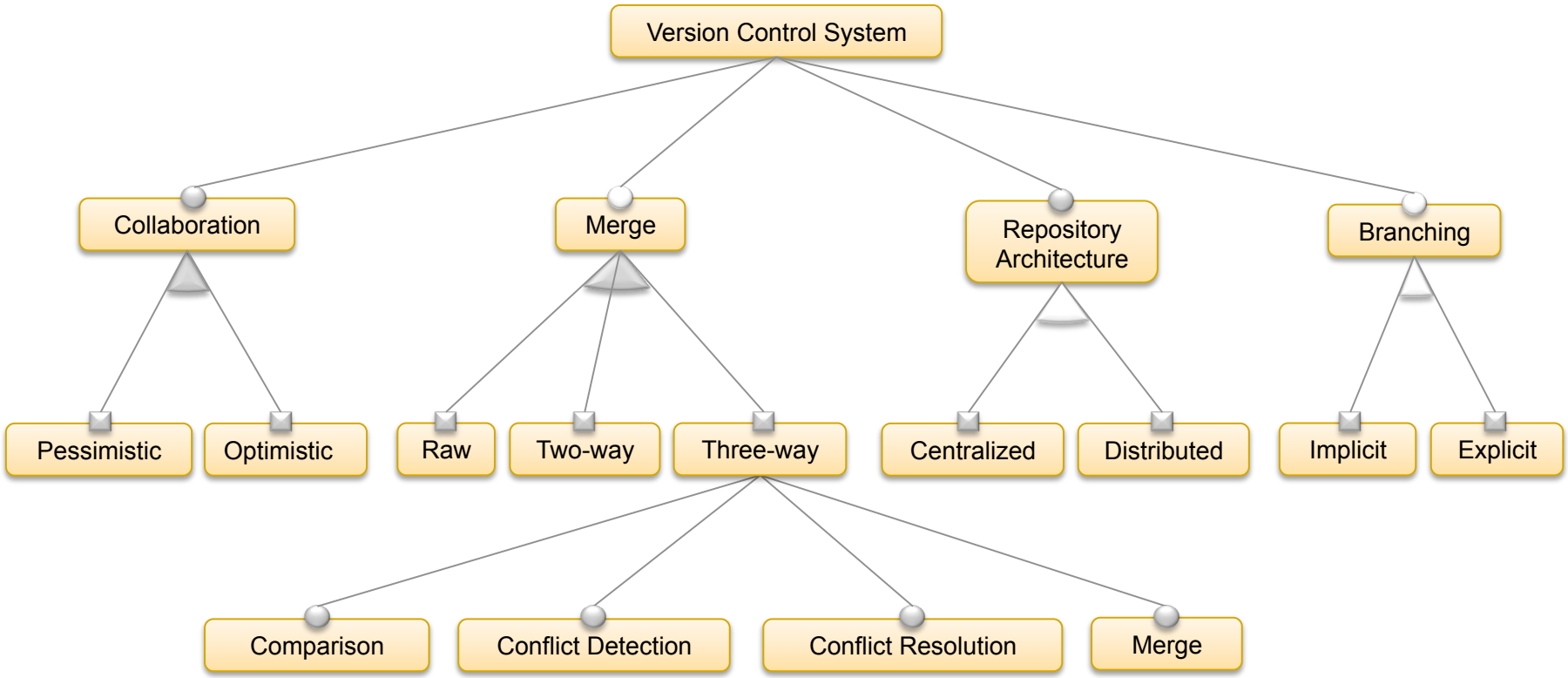
Features of Version Control Systems

Overview of Version Control Systems

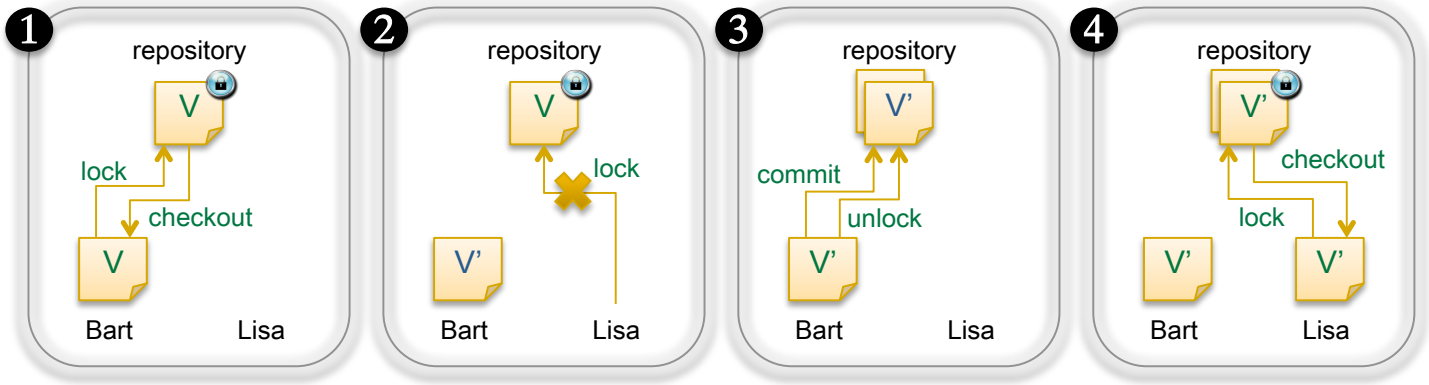
- Part of Software Configuration Management
- **Basic components:**
  - Repository
  - (Software) Artifacts => versions (revisions vs. variants)
- **Purpose:**
  - Historical archiving and management of changes in (software) artifacts
    - retrieving old revisions
    - keeping track of changes
  - Supports collaboration



Top-level Features

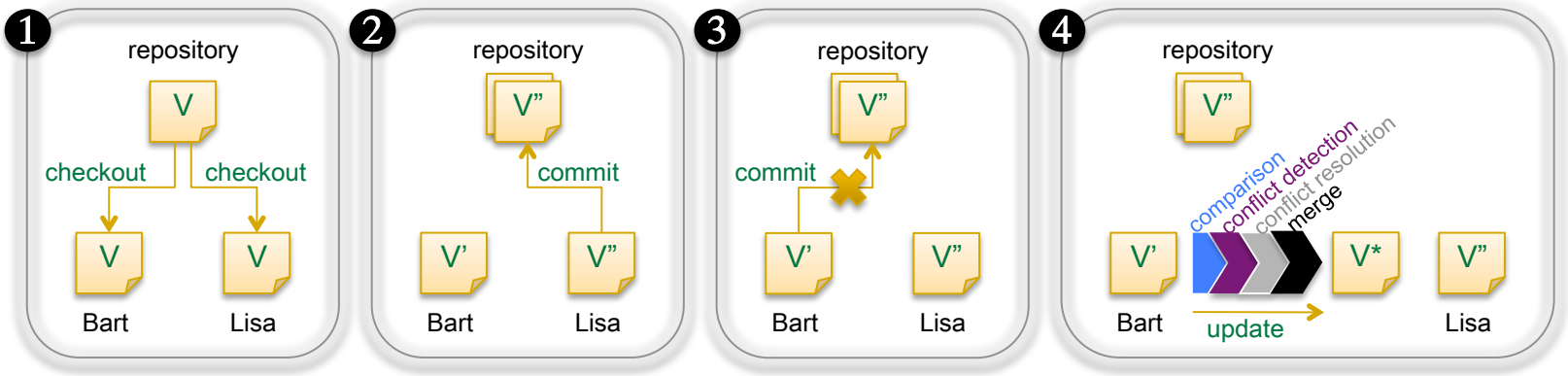


Pessimistic Versioning

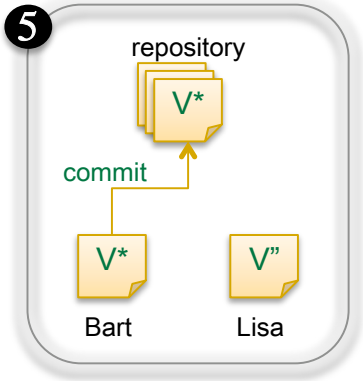


- **Lock-modify-unlock paradigm**
- **Disadvantages:**
  - Prohibits parallel working
  - Administration overhead
  - False sense of security

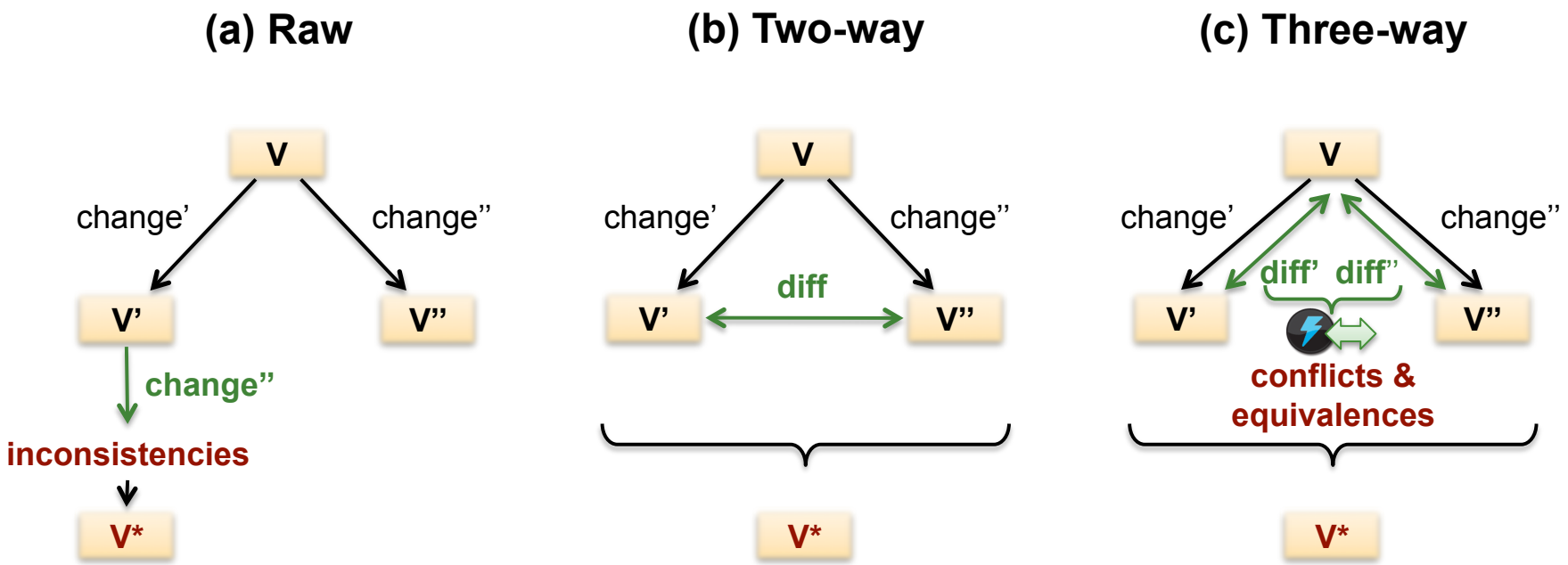
Optimistic Versioning



- **Copy-modify-merge paradigm**
- **Advantage:**
  - Saves time
- **Disadvantage:**
  - Result depends on quality of the merge



Raw vs. Two-way vs. Three-way



Centralized vs. Distributed

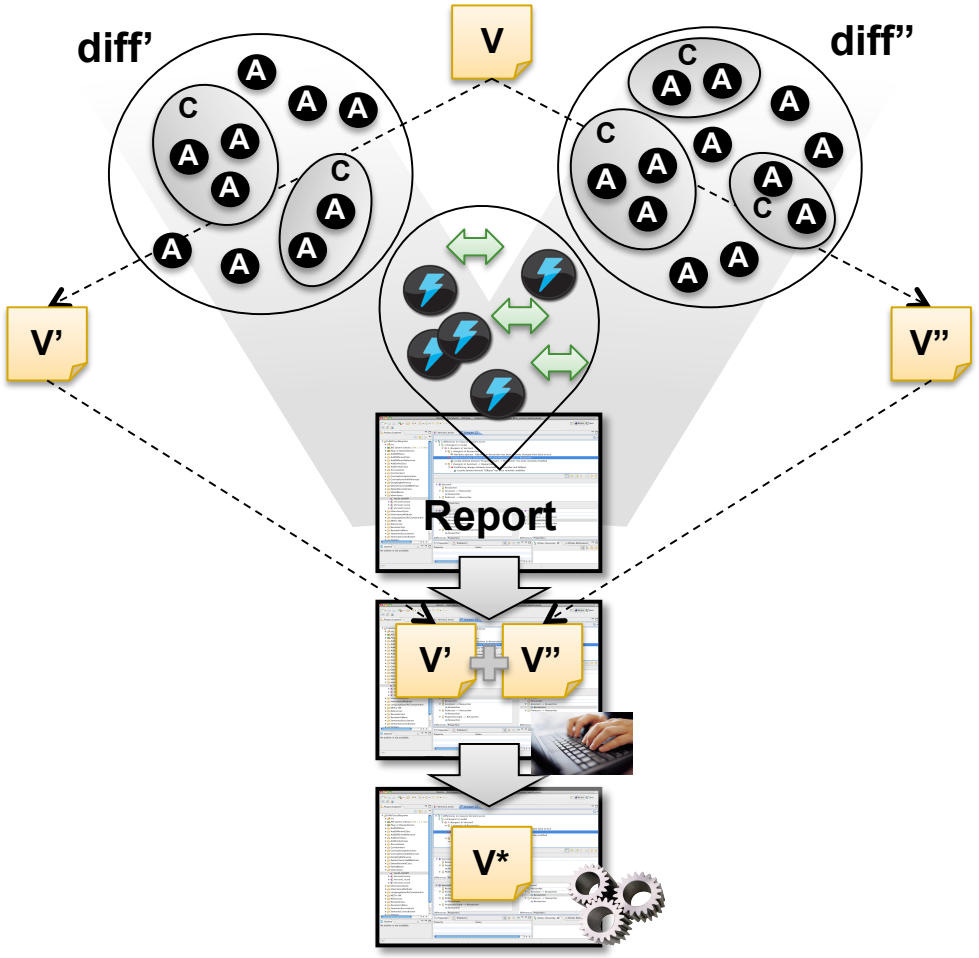
- **Centralized**
  - Users access a master repository via a client
  - Edited working copies must be committed before they can be propagated to other VCS users
  - Version history is only available in the master repository

- **Distributed**
  - Each user has an own repository
  - Repositories can be synchronized optionally with others
  - Version history is distributed on the different repositories

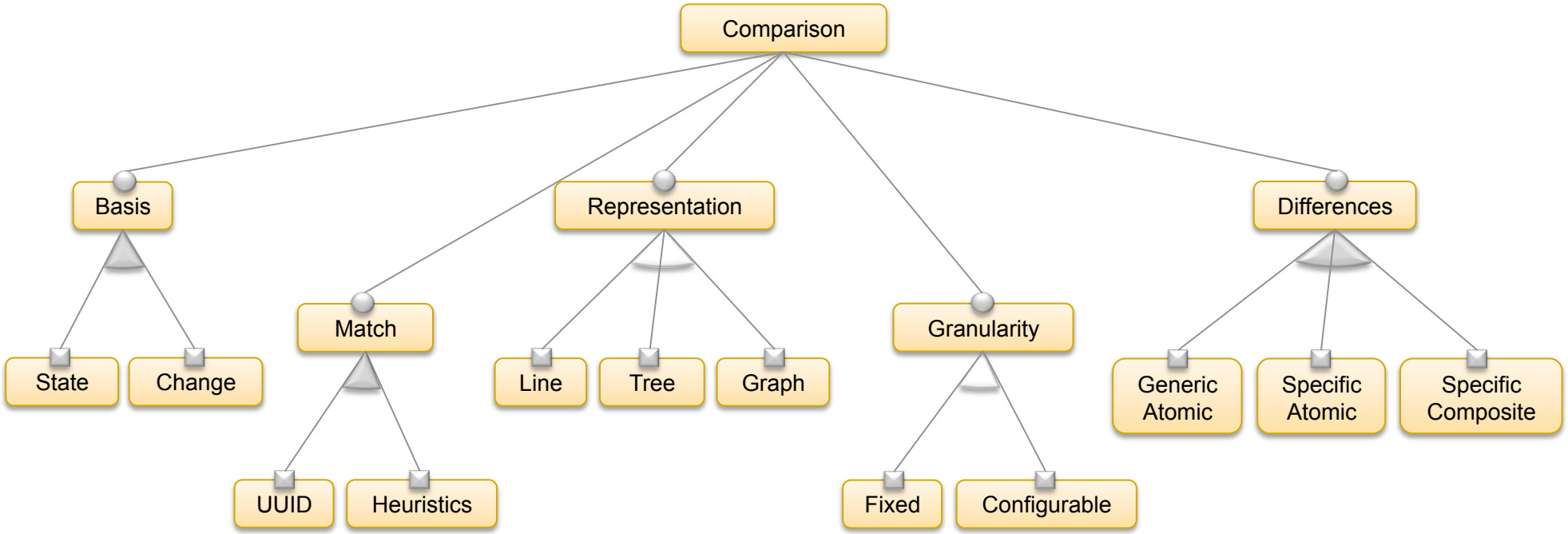
Branching Variants

- **Implicit/Explicit**
- **Used due to different intentions**
  - **Version Management**
    - software product releases
  - **Variant Management**
    - software product variants
    - common core
  - **Decentralized Version Control Systems**
  - **Resolution Support**
    - storage of the intermediate versions before a merge
    - to retrieve information about differences, conflicts and resolution patterns

3-way Merge Features

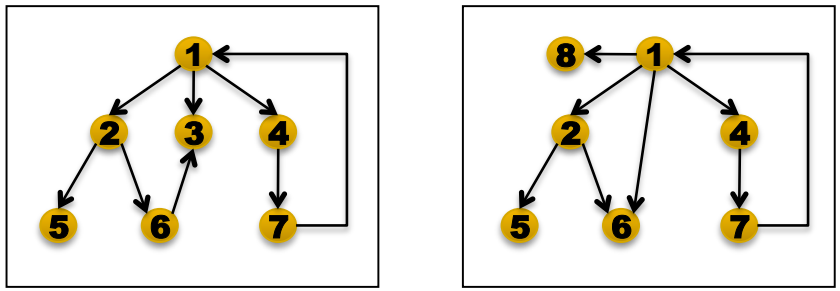


Comparison Phase



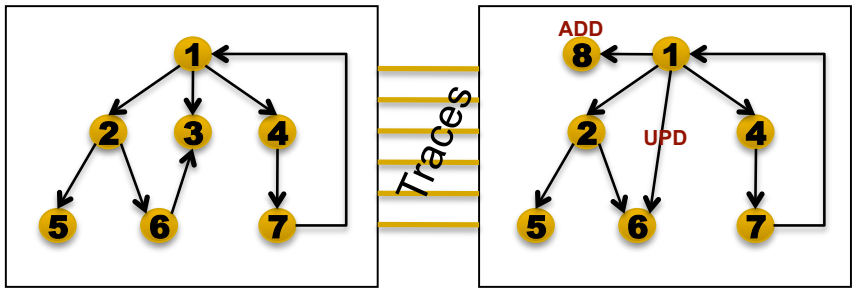
State vs. Change

State-based



- **Input:**
  - Two versions of an artifact
- **Advantage:**
  - Independent of a development environment / editor
- **Disadvantage:**
  - Matching techniques needed
- **Conflict detection** is computed on the resulting difference sets

Change-based



- **Input:**
  - Two versions of an artifact
  - Change protocol (log file)
- **Advantage:**
  - Composite operations available
- **Disadvantage:**
  - Bounded to an development environment / editor
- **Conflict detection** is computed on the logged changes

UUIDs vs. Heuristics

- Matching approaches particular important for state-based versioning
- Needed to identify the artifact's elements occurring in both versions

- **Universally Unique Identifiers (UUIDs)**
  - All elements in an artifact need to provide an UUID
  - Newly created elements must be assigned with an UUID
  - UUIDs must not be modified until the deletion of the element
  - Two elements occurring in different versions with the same UUID are considered as identical
  - **Advantages:**
    - easiest approach to realize a matching
    - elements can still be matched, although they have been changed fundamentally
    - unique matching
  - **Disadvantage:**
    - Assignment of UUIDs needed

UUIDs vs. Heuristics

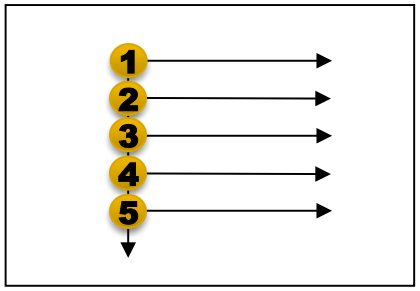
- **Heuristics**
  - May be based on structural similarities of two artifact versions
  - Often use metrics from information theory
  - **Advantage:**
    - flexible
  - **Disadvantage:**
    - matching might not be unique
    - performance may be an issue

- **UUIDs and Heuristics can be combined!**

Line vs. Tree vs. Graph

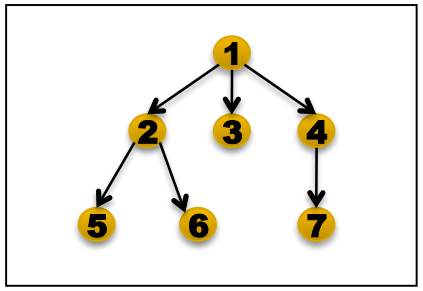
Line-based

- Any kind of artifacts



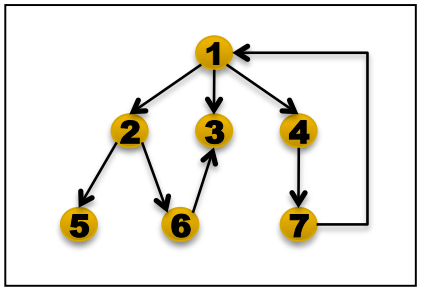
Tree-based

- HTML, XML file
- Latex document



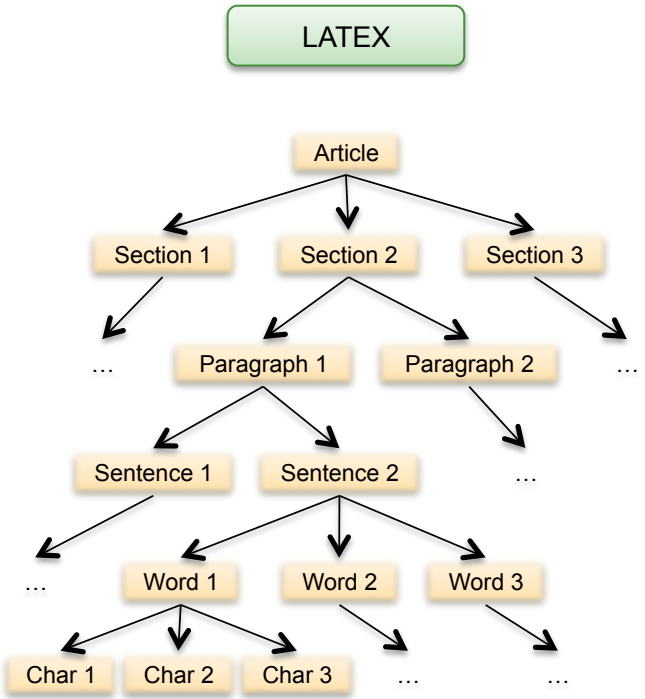
Graph-based

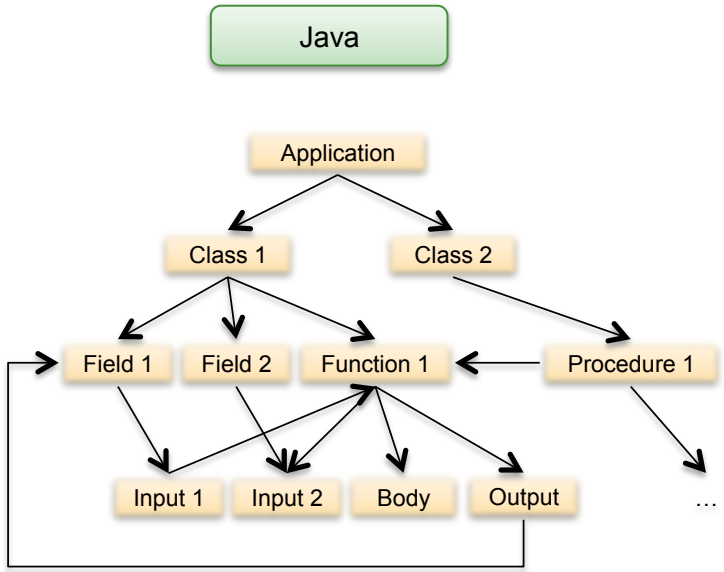
- Model artifacts
- Program source code



Fixed vs. Configurable Granularity

- Also known as version granularity, delta granularity or unit of conflicts
- Can be applied on any granularity level

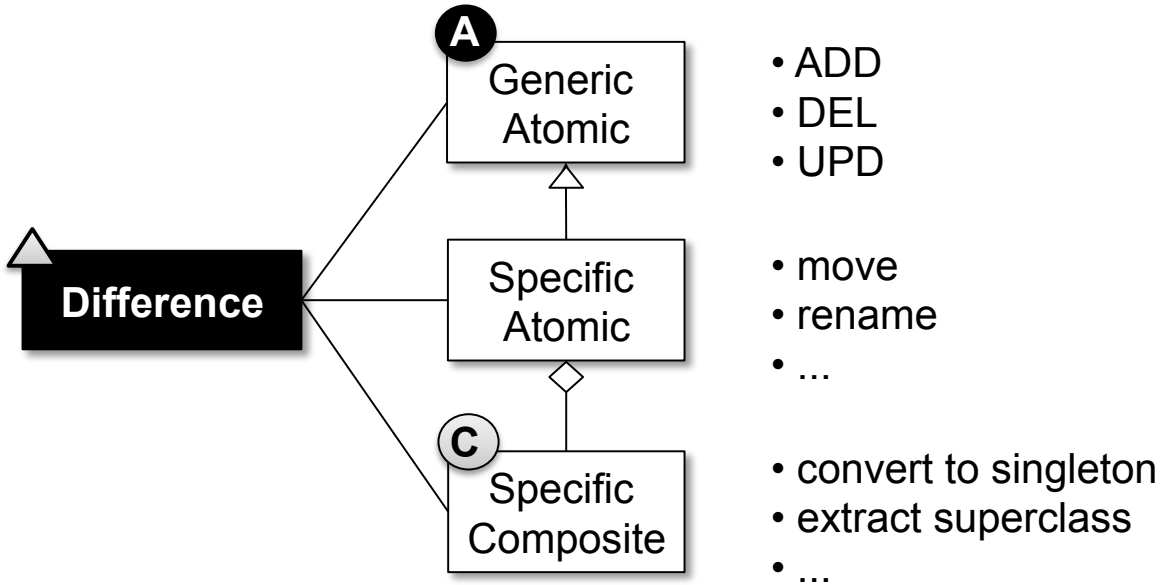




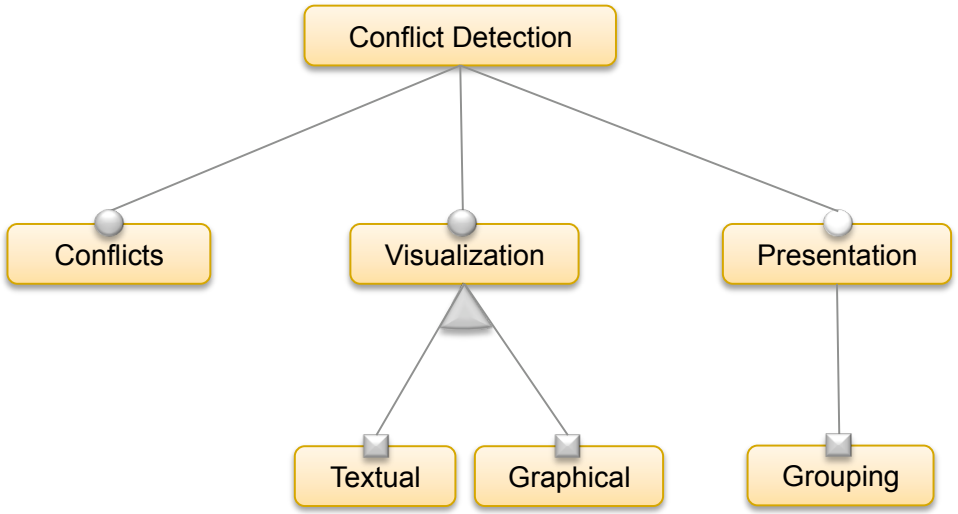
Outcome of the Comparison Phase

Resulting set of difference can be classified according to two orthogonal dimensions

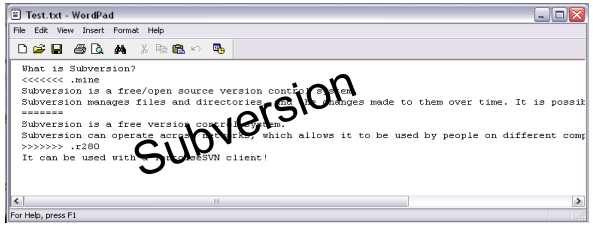
- Dependency on an underlying language
- Divisibility of a change operation



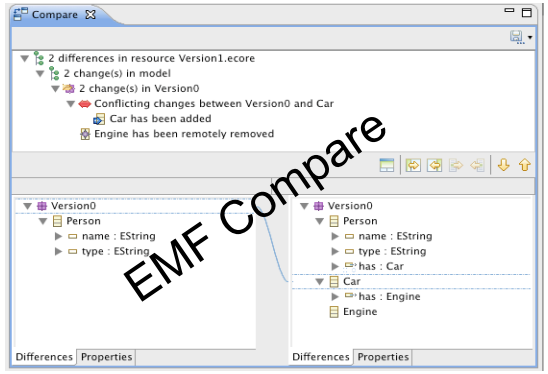
Conflict Detection Phase



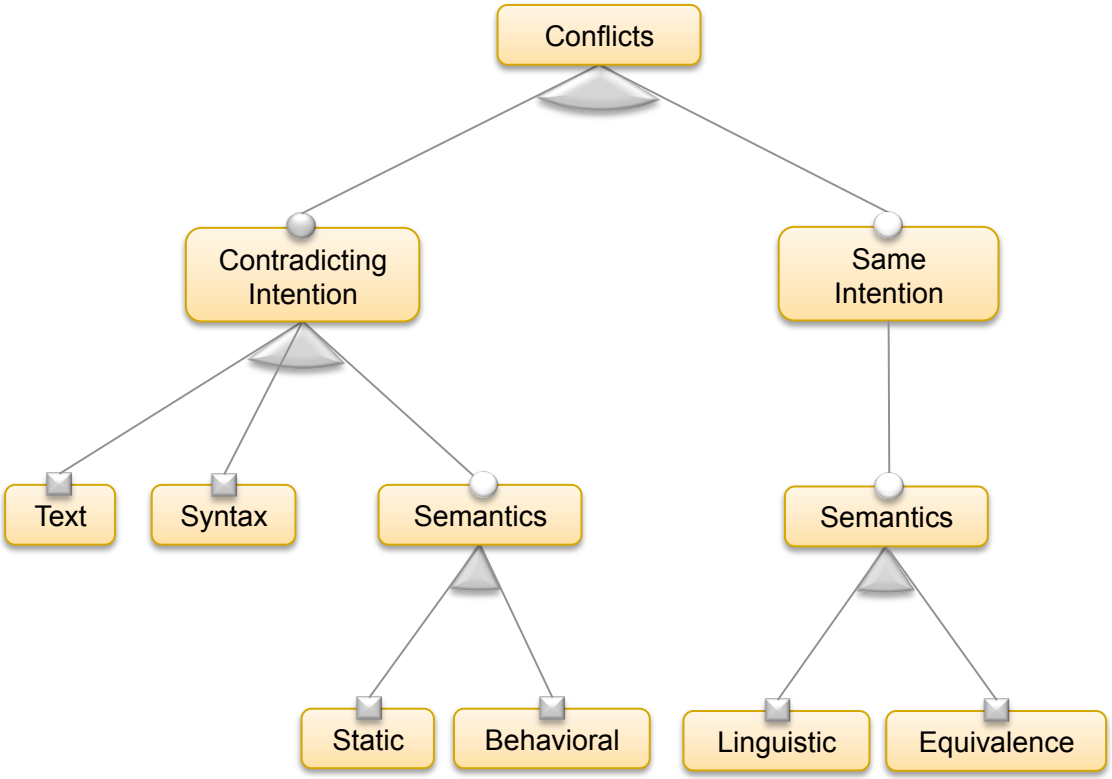
- **Textual**



- **Graphical**



Conflict Detection Phase



	ADD	DEL	UPD
ADD		n.a.	n.a.
DEL	n.a.	–	
UPD	n.a.		

Textual Conflict

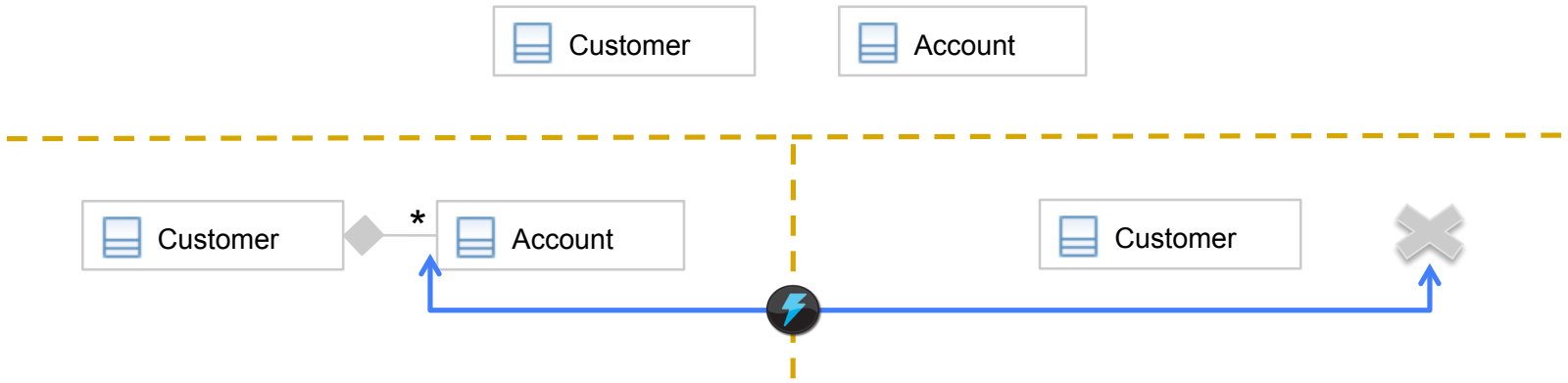

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:uml="http://
www.eclipse.org/uml2/3.0.0/UML" xmi:id="_tLh4kJ7TEd67t-3wTeUA-A" name="Version0">
3  <packagedElement xmi:type="uml:Class" xmi:id="_v0dpsJ7TEd67t-3wTeUA-A" name="Person">
4   <ownedAttribute xmi:id="_zvIvgJ7TEd67t-3wTeUA-A" name="birthday" aggregation="composite">
5    <type xmi:type="uml:PrimitiveType" href="pathmap://UML_METAMODELS/UML.metamodel.uml#String"/>
6   </ownedAttribute>
7  </packagedElement>
8 </uml:Package>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <uml:Package xmi:version="2.1" xmlns:xmi="http://
schema.omg.org/spec/XMI/2.1" xmlns:uml="http://
www.eclipse.org/uml2/3.0.0/UML" xmi:id="_tLh4kJ7TEd67t-3wTeUA-
A" name="Version0">
3  <packagedElement xmi:type="uml:Class"
    xmi:id="_v0dpsJ7TEd67t-3wTeUA-A" name="Person">
4   <ownedAttribute xmi:id="_zvIvgJ7TEd67t-3wTeUA-A"
    name="dateOfBirth" aggregation="composite">
5    <type xmi:type="uml:PrimitiveType" href="pathmap://
UML_METAMODELS/UML.metamodel.uml#String"/>
6   </ownedAttribute>
7   <ownedAttribute xmi:id="_azZRcKCFEd67t-3wTeUA-A"
name="name" aggregation="composite"/>
8  </packagedElement>
9 </uml:Package>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <uml:Package xmi:version="2.1" xmlns:xmi="http://
schema.omg.org/spec/XMI/2.1" xmlns:uml="http://
www.eclipse.org/uml2/3.0.0/UML" xmi:id="_tLh4kJ7TEd67t-3wTeUA-
A" name="Version0">
3  <packagedElement xmi:type="uml:Class"
xmi:id="_v0dpsJ7TEd67t-3wTeUA-A" name="Person">
4   <ownedAttribute xmi:id="_zvIvgJ7TEd67t-3wTeUA-A"
name="birthday" aggregation="composite">
5    <type xmi:type="uml:PrimitiveType" href="pathmap://
UML_METAMODELS/Ecore.metamodel.uml#EDate"/>
6   </ownedAttribute>
7   <ownedAttribute xmi:id="_c9UyYKCFEd67t-3wTeUA-A"
name="lastName" aggregation="composite"/>
8  </packagedElement>
9 </uml:Package>
```


Syntactic Conflict

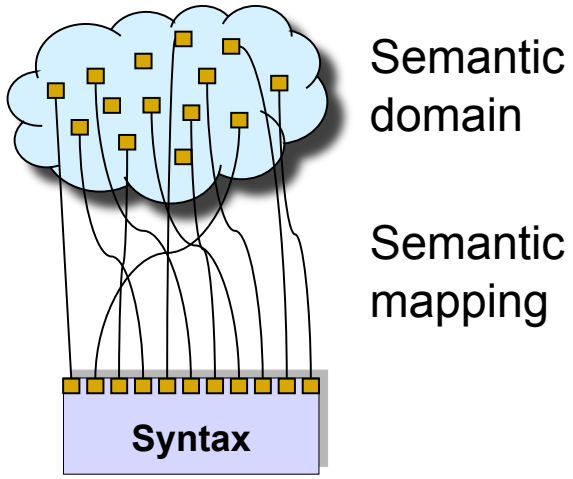
- »Detected by a structural comparison of artifacts, which are represented either in a tree or graph reflecting the syntax of the artifact's language.«
- »Syntactic conflicts are resulting from modifications, which produce a violation of the language's syntax when merged.«



Semantic Conflicts

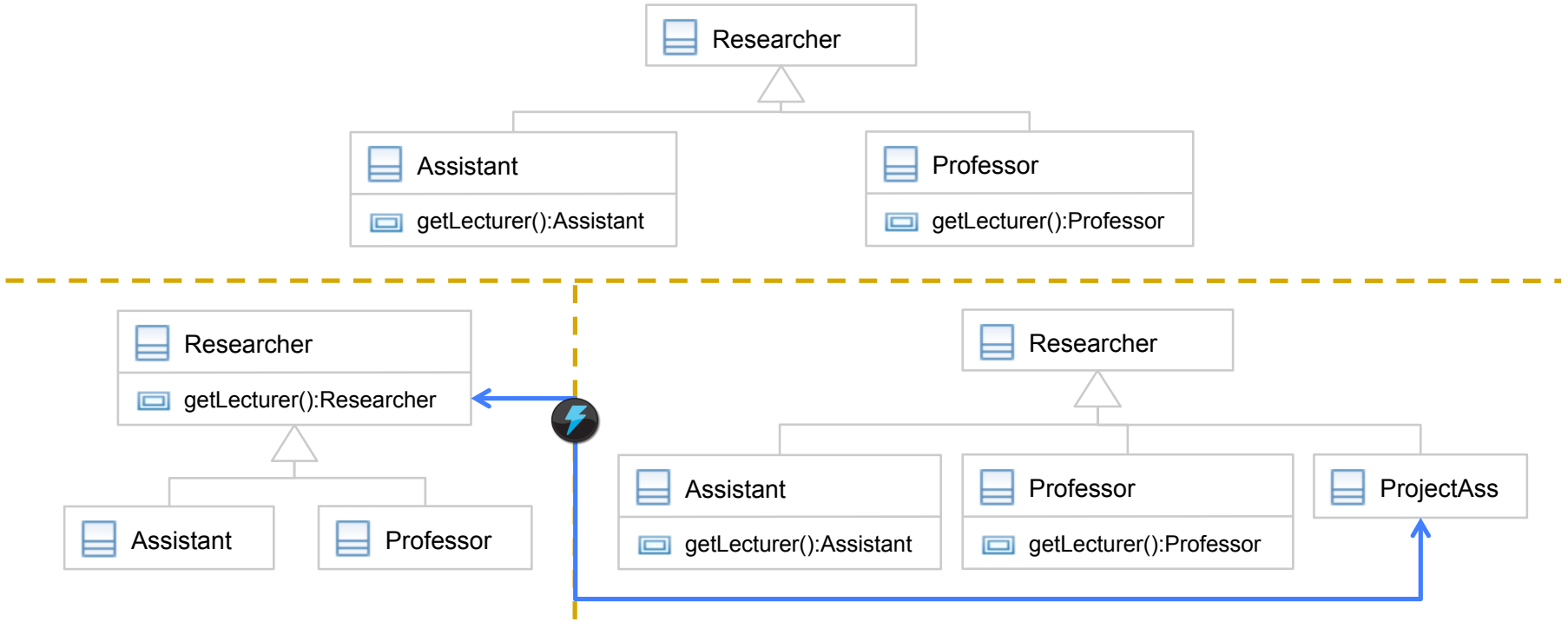
- Some conflicts cannot be detected by a structural comparison due to
  - Semantic meaning of linguistic expressions
  - Syntactic different parts have been edited which may
    - violate language constraints or
    - the execution behavior

    if merged
- Formal semantics are needed to detect such kinds of conflicts
- Formal semantics consists of three parts:
  - Syntax of a language
  - Semantic domain
  - Semantic mapping



Static Semantic Conflict

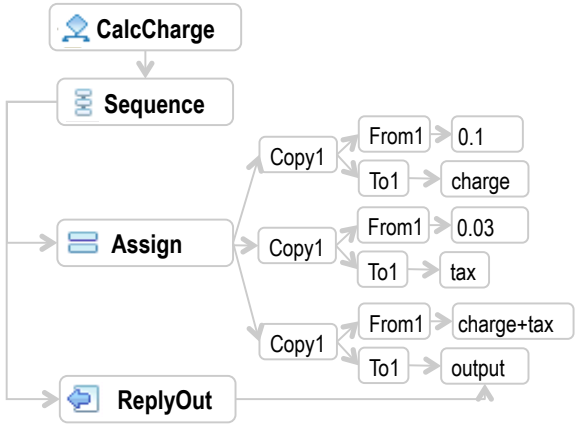
- Occurs due to violation of language constraints
- In modeling e.g., by violation of relationships, context conditions, consistency or violation of integrity



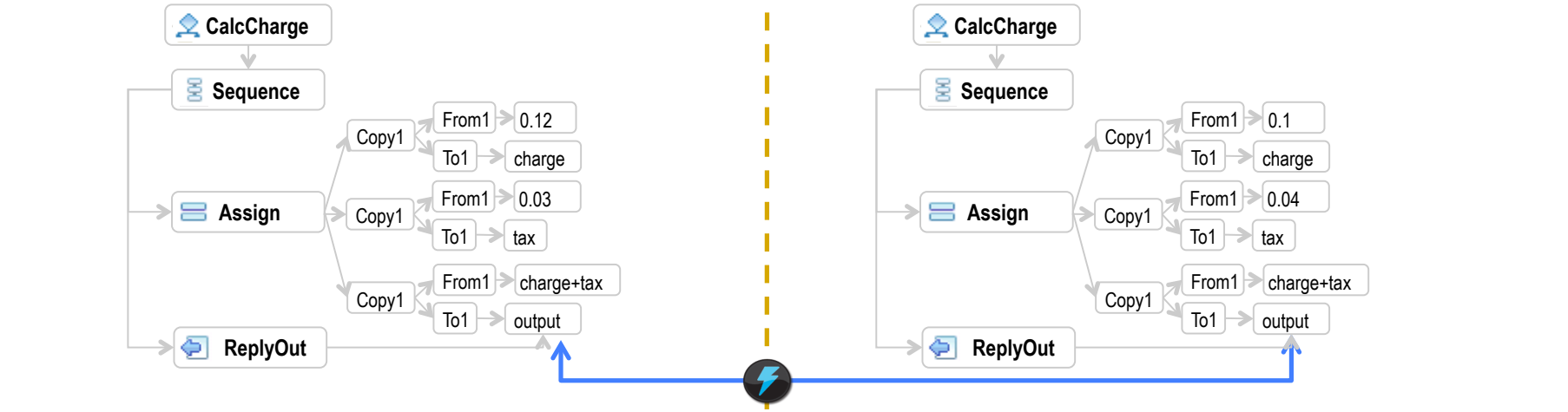
Behavioral Semantic Conflict

- Text
- Syntax
- **Semantics**
- Equivalence

- Occurs due to concurrent changes of the execution behavior

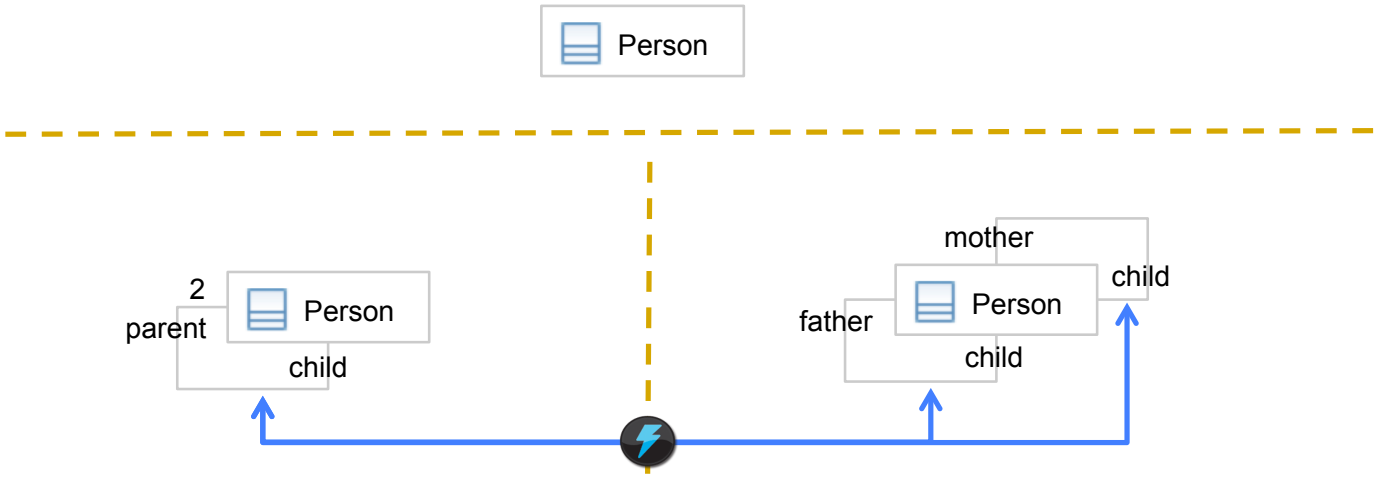


- E.g., concurrent changes on the data or control flow may yield to behavioral side effects



Linguistic Semantic Conflict

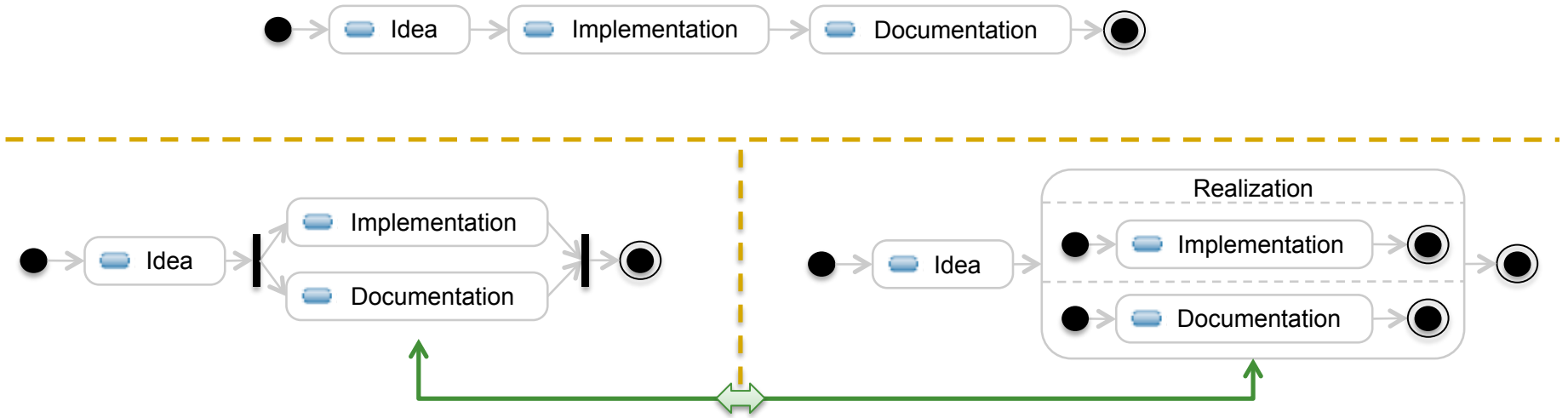
Occurs e.g., if two VCS users edit the same artifact with the same intentions but utilized different literal designations.



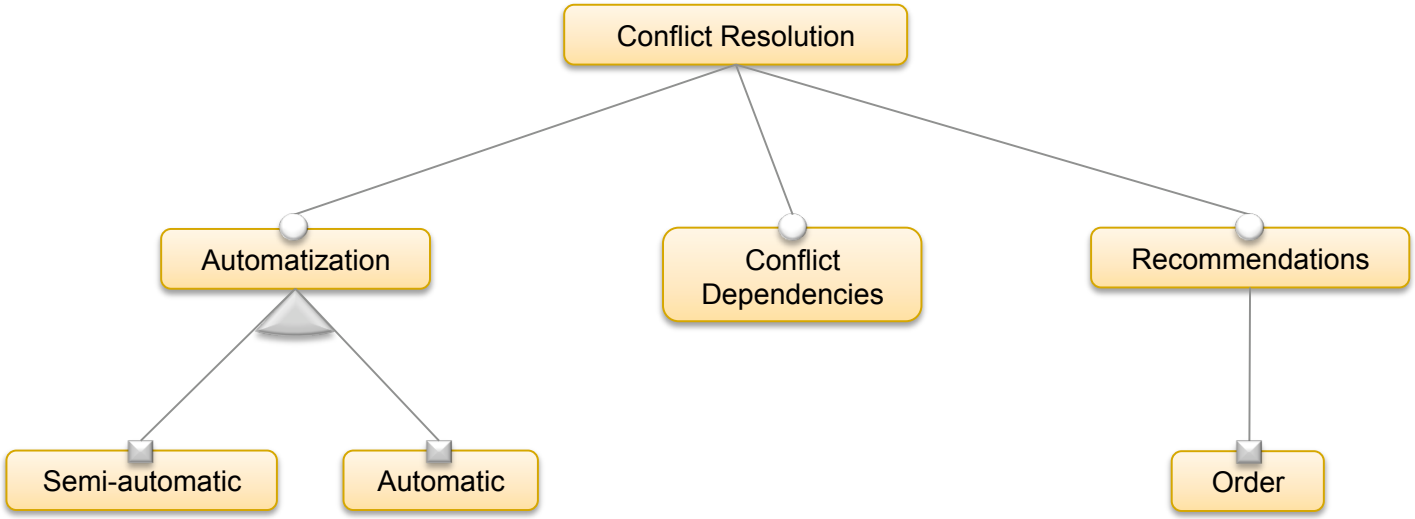
Conflict due to Equivalence

May occur due to utilization of

- semantically equivalent concepts
- refactorings



Conflict Resolution Phase



Support for semi-automatic resolution

- Dependency analysis to identify resolution orders
- Resolution patterns/policies

Challenges in Model Versioning

Existing Systems

Audodesk Vault

Alienbrain

Vesta

OpenCVS

GNU arch

CVS

Aldon

Subversion

FirePublish

AllChange

Monotone

LibreSource

SVK

FileHamster

Mercurial

CollabNet

git

Bazaar

Aegis

ArX

AllFusion Harvest

DVCS

Evolution

CVSNT

EVS

Codeville

ClearCase

Fossil

Darcs

BitKeeper

Survey Result about available Model VCSs

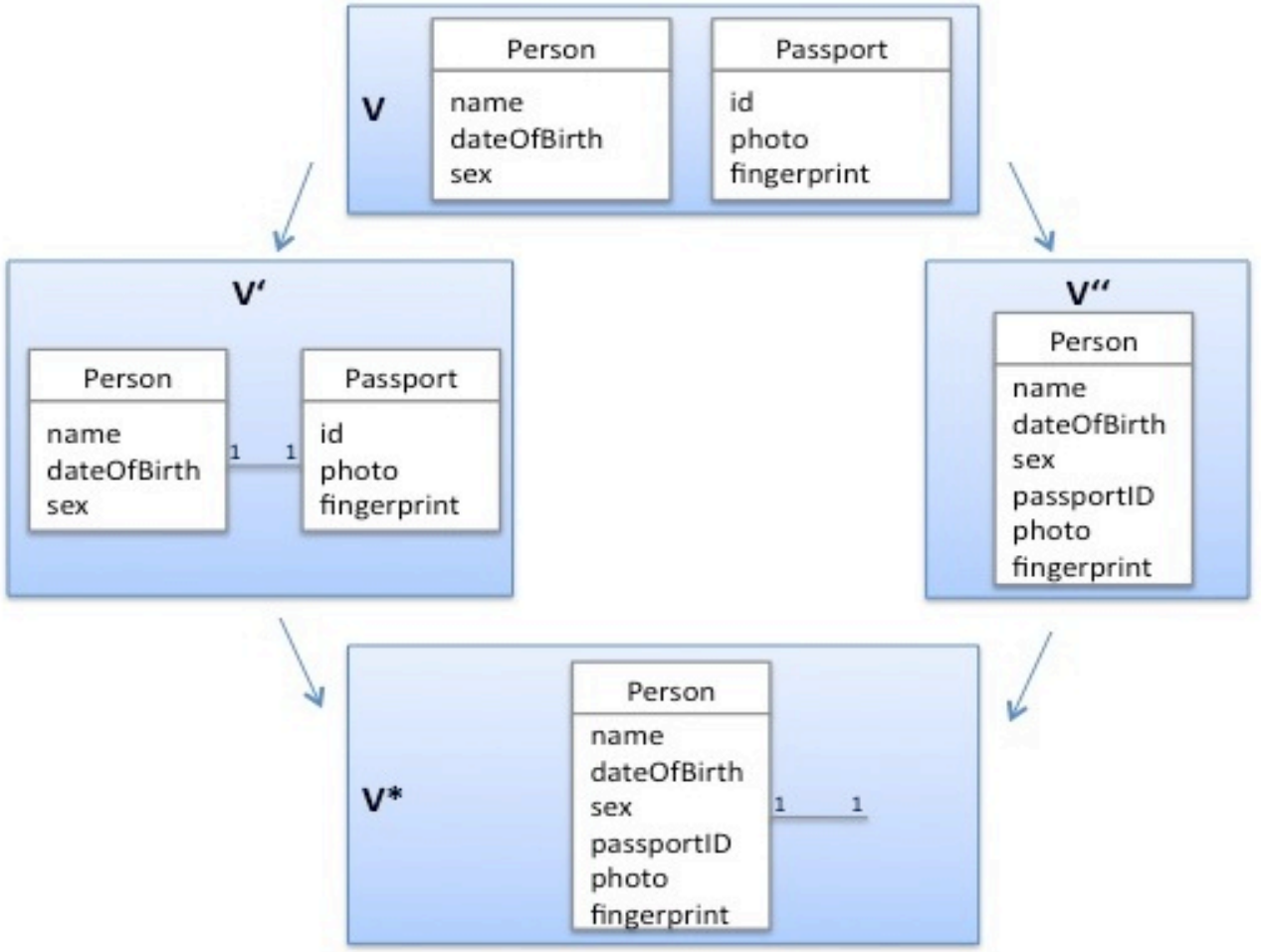
Approaches	Generic	Comparison								Conflict Detection									Conflict Resolution				Merge
		Basis		Match		Repre-sentation			Granularity	Conflicts						Visualization		Presentation	Automati-zation				
		State	Change	UUID	Heuristics	Line	Tree	Graph	Configurable	Textual	Syntactic	Composite Op.	Equivalence	Static	Behavioral	Textual	Graphical	Grouping	Semi-automatic	Automatic	Dependencies	Recommendations	Validation
Subversion	+	+	−	−	+	+	−	−	−	+	−	−	−	−	−	+	−	−	−	−	−	−	−
RSA	−	▨	▨	+	−	−	−	+	−	−	+	−	−	−	−	−	+	+	−	−	−	−	~
EMF Comp.	+	+	−	+	+	−	−	+	−	−	+	−	−	−	−	−	+	+	−	−	−	−	−
Alanen & Porres	+	+	−	+	−	−	−	+	−	−	+	−	−	−	−	−	−	−	▨	−	−	−	~
Unicase	−	−	+	+	−	−	−	+	−	−	+	+	−	~	−	−	+	+	+	−	−	−	−
Oda & Saeki	~	−	+	+	−	−	▨	▨	−	−	+	−	−	−	−	▨	▨	▨	−	−	−	−	−
CoObRA	−	−	+	+	−	−	−	+	−	−	+	−	−	−	−	+	−	−	−	−	−	−	−
Cicchetti et al.	−	−	+	+	−	−	−	+	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−
Odyssey-VCS	−	+	−	+	−	−	−	+	+	−	+	−	−	−	−	▨	▨	▨	−	−	−	−	−
SMoVer	+	+	−	+	−	−	−	+	+	−	+	−	+	+	+	+	−	+	−	−	−	−	−

Legend: + Supported ~ partly supported − not supported ▨ not applicable

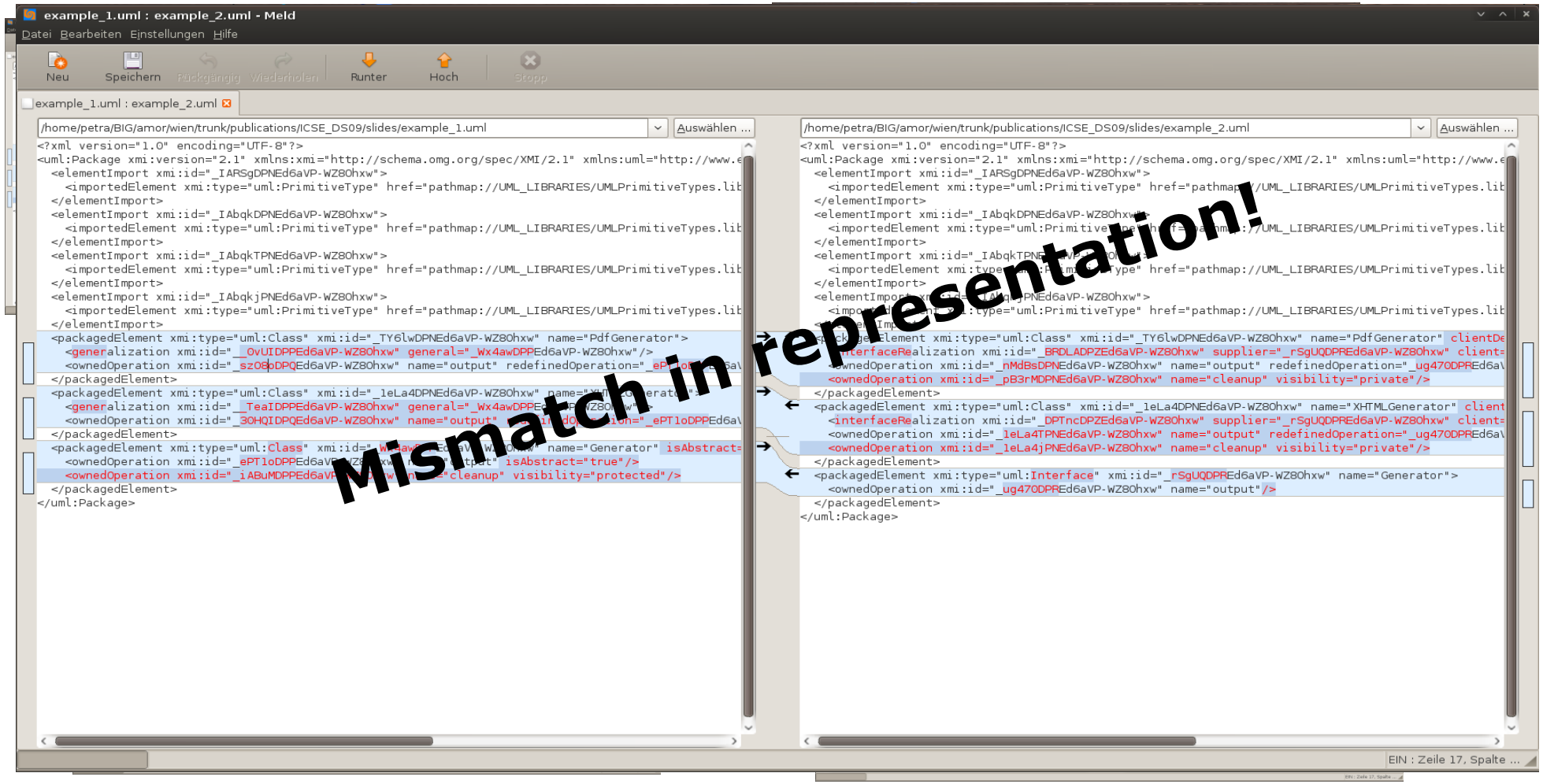
Overall Challenges

- **General**
  - Mismatch in representation
  - Bounded to modeling environment
  - Inflexible in modeling language
  - Too generic
  - Various serialization variants
  - Disregard of dependencies between artifacts
- **Comparison Phase**
  - Non-configurable
- **Conflict Detection**
  - Erroneous conflict detection
    - Missing interpretation of semantics of artifacts
    - Missing information about edit operations in state-based systems
  - Unsatisfactory conflict visualization
- Unsupportive **Conflict Resolution**
- Inconsistent **Merge**d Model Versions

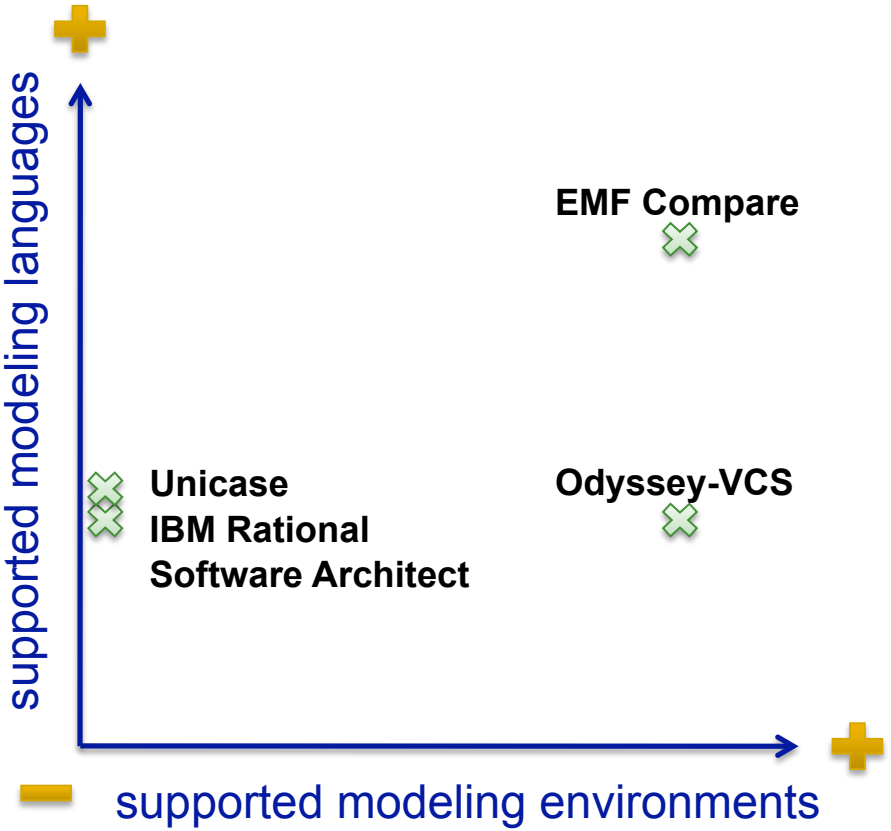
Why not using Subversion with Unix diff? (1/2)



Why not using Subversion with Unix diff? (2/2)



Specific vs. Generic VCSs for Model Artifacts



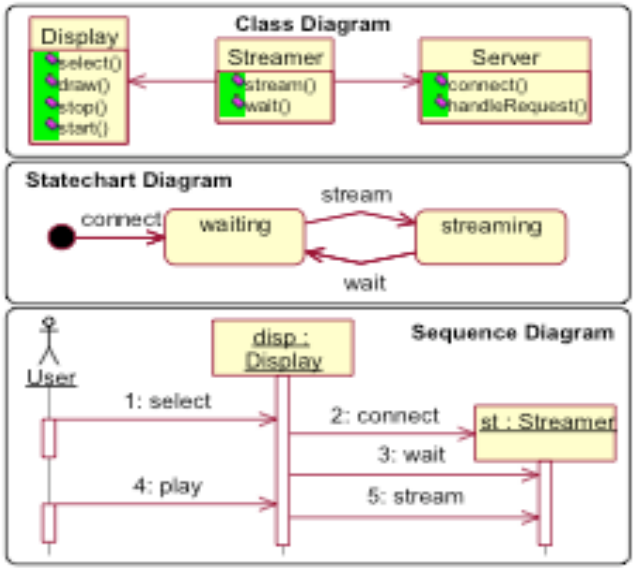
- How can a standalone VCS for model artifacts be realized without depending on the traces of changes on the artifact, performed by modelers?
- How can a VCS for model artifacts be realized without depending on a concrete modeling language?
- How can such a system be generic but provides accurate conflict detection and supportive conflict resolution support?

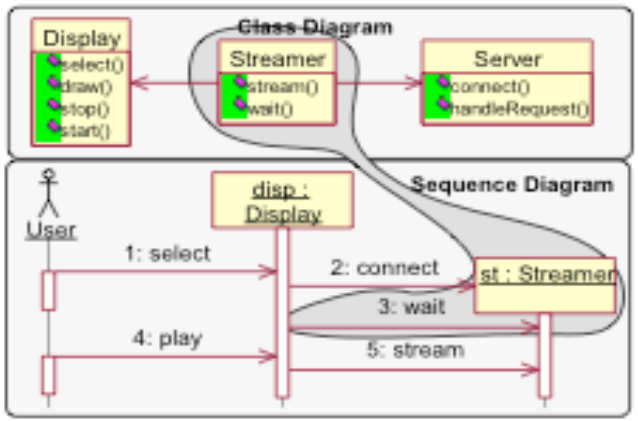
Serialization Variants

- For standalone VCSs the versions of model artifacts need to be exchanged between the modeling environment and the VCS
- Modeler may use different modeling environments to edit model artifacts from the repository
- **Problems:**
  - Modeling environments may export different XMI variants (those XMI representations of a model artifact may vary considerably)
  - Modeling environments may utilize divergent metamodels of a modeling language (e.g., various UML implementations exist)
- Summing up those cannot be utilized in the VCS right away
- **Solution:** Adapters to import different XMI representations are indispensable

Dependencies between Artifacts

- Artifacts in a repository may depend on each other (may have communalities)
- If e.g., a class is renamed in a UML class diagram this has an effect on the UML Sequence Diagram instantiating this class





Comparison Phase

- On which granularity level should the comparison be performed?
- For generic VCSs the granularity level should be configurable for specific languages and by VCS users. How can this be realized?
- How can composite operations (like refactorings) be detected in state-based systems?
- Which information is needed for the following phases conflict detection and resolution?

Conflict Detection Phase

- How can accurate conflict detection be achieved?
  - avoid falsely indicated conflicts
  - find undetected conflicts
- How can conflicts be presented?
- What should be presented to the user?
  - Tentative merge
    (apply only non-conflicting changes)
  - Left change, right change, resolution patterns

Conflict Resolution Phase

- Which influence has the resolution order, in which conflicts are resolved, to the overall effort to combine two versions?
- How can a VCS user be assisted in the resolution of conflicts by the system?
- Can recurring conflict resolution activities by users be prevented?
- If VCSs can learn from conflict resolution activities, how can rules be applied on similar conflicting situations?

Merge Phase

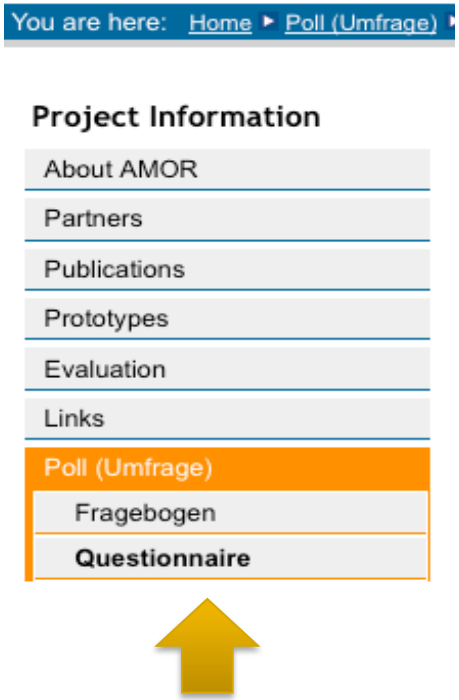
- How can it be ensured that only valid models are saved in the repository?
- How can it be prevented that information gets lost after the merge of two artifacts?

Discussion

Questions

- Are there more challenges and issues?
- What are your expectations on Version Control Systems for models?
- Do we really need Version Control Systems dedicated for model artifacts?
- Versioning habits?
- …

Questionnaire



http://www.modelversioning.org

Background

Which role do you mainly play in projects?

○ manager ○ developer ○ architect ○ tester ○ others

Which other roles do you occasionally have in projects?

☐ manager ☐ developer ☐ architect ☐ tester ☐ others

How is your team geographically distributed?

○ in the same building ○ in the same town ○ in the same time zone
○ all over the world

How many people typically participate in your projects?

○ up to 5 persons ○ up to 20 persons ○ up to 100 persons ○ more

Versioning Habits

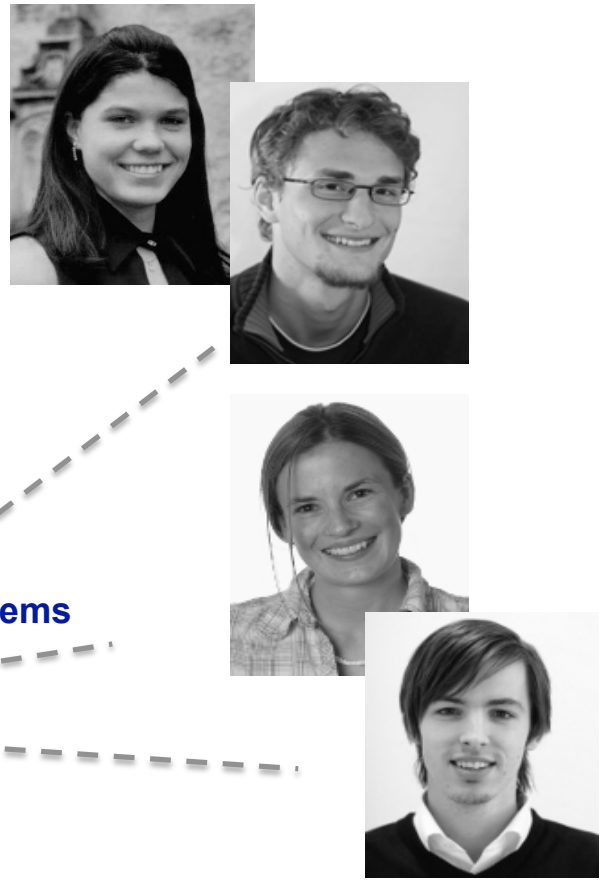
What version strategy do you apply?

○ pessimistic (lock/modify/unlock) ○ optimistic (modify/merge) ○ none at all

AMOR Project

Research Objectives

- **General**
  - **Mismatch in representation**
  - **Bounded to modeling environment**
  - **Inflexible in modeling language**
  - **Too generic**
  - Various serialization variants
  - Disregard of dependencies between artifacts
- **Comparison Phase**
  - **Non-configurable**
- **Conflict Detection (CD)**
  - **Erroneous conflict detection**
    - **Missing interpretation of semantics of artifacts**
    - **Missing information about edit operations in state-based systems**
  - Unsatisfactory conflict visualization
- **Unsupportive Conflict Resolution (CR)**
- Inconsistent **Merge**d Model Versions



Partners

- AMOR: Adaptable Model Versioning
  - FFG FIT-IT Semantic Systems Project
  - Duration: Feb. 2009 – Jan. 2011

Industrial Partner:

Academic Partners:

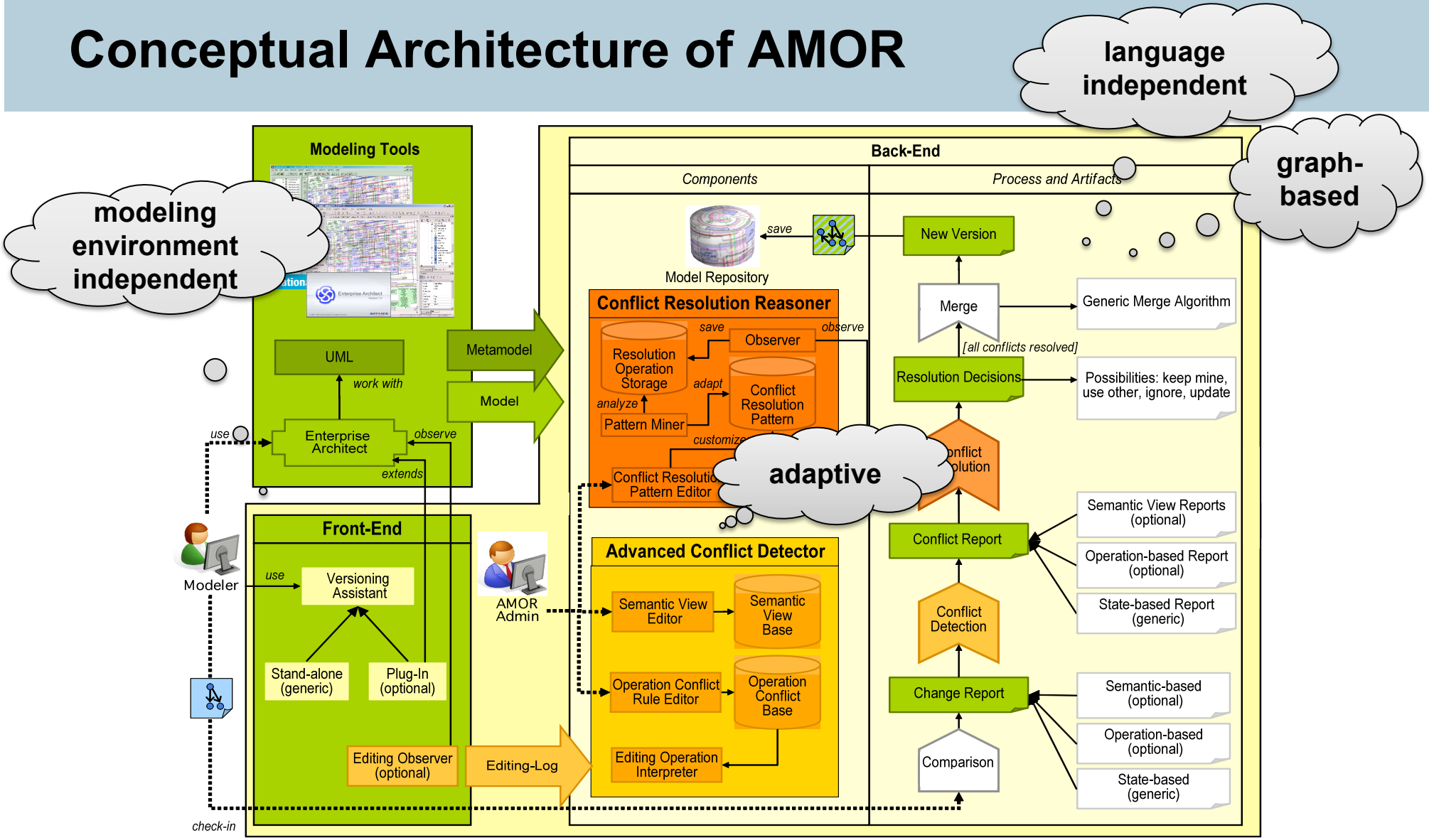
Business Informatics Group (BIG)
Institut für Software Technik und Interaktive Systeme

JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

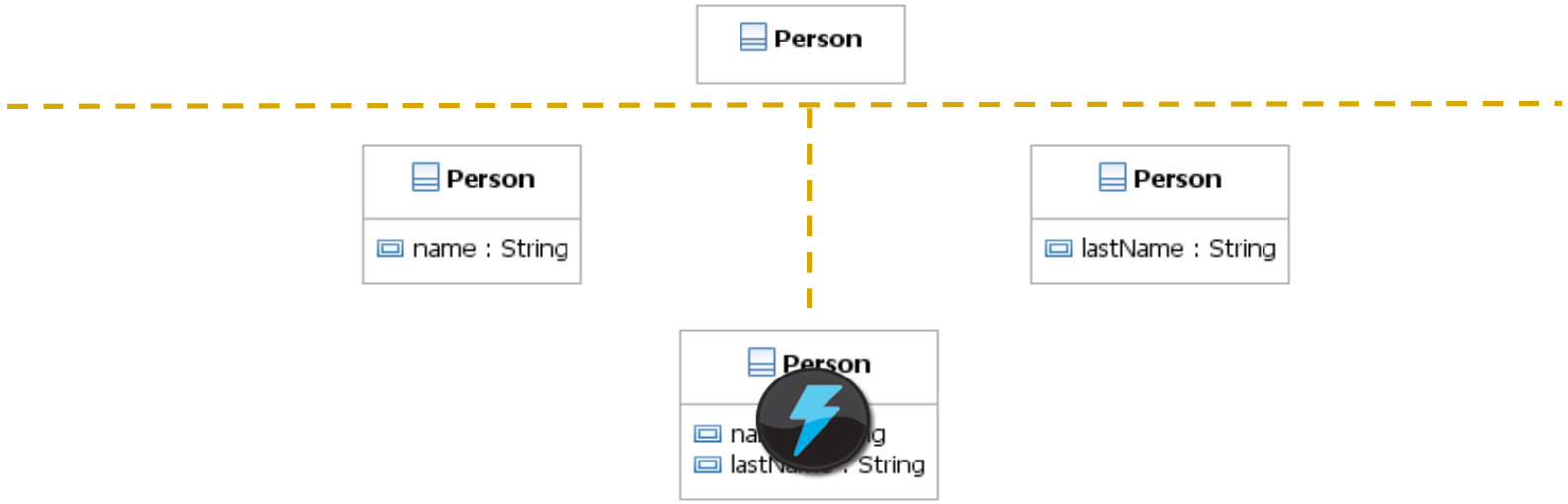
Arbeitsgruppe Information Systems (IFS)
Institut für Bioinformatik

Institut für Telekooperation (TK)

Conceptual Architecture of AMOR



Configurable Comparison – Motivation



Advantages:

- Configurable for specific user desires
- Configurable for specific modeling languages

Configurable Comparison

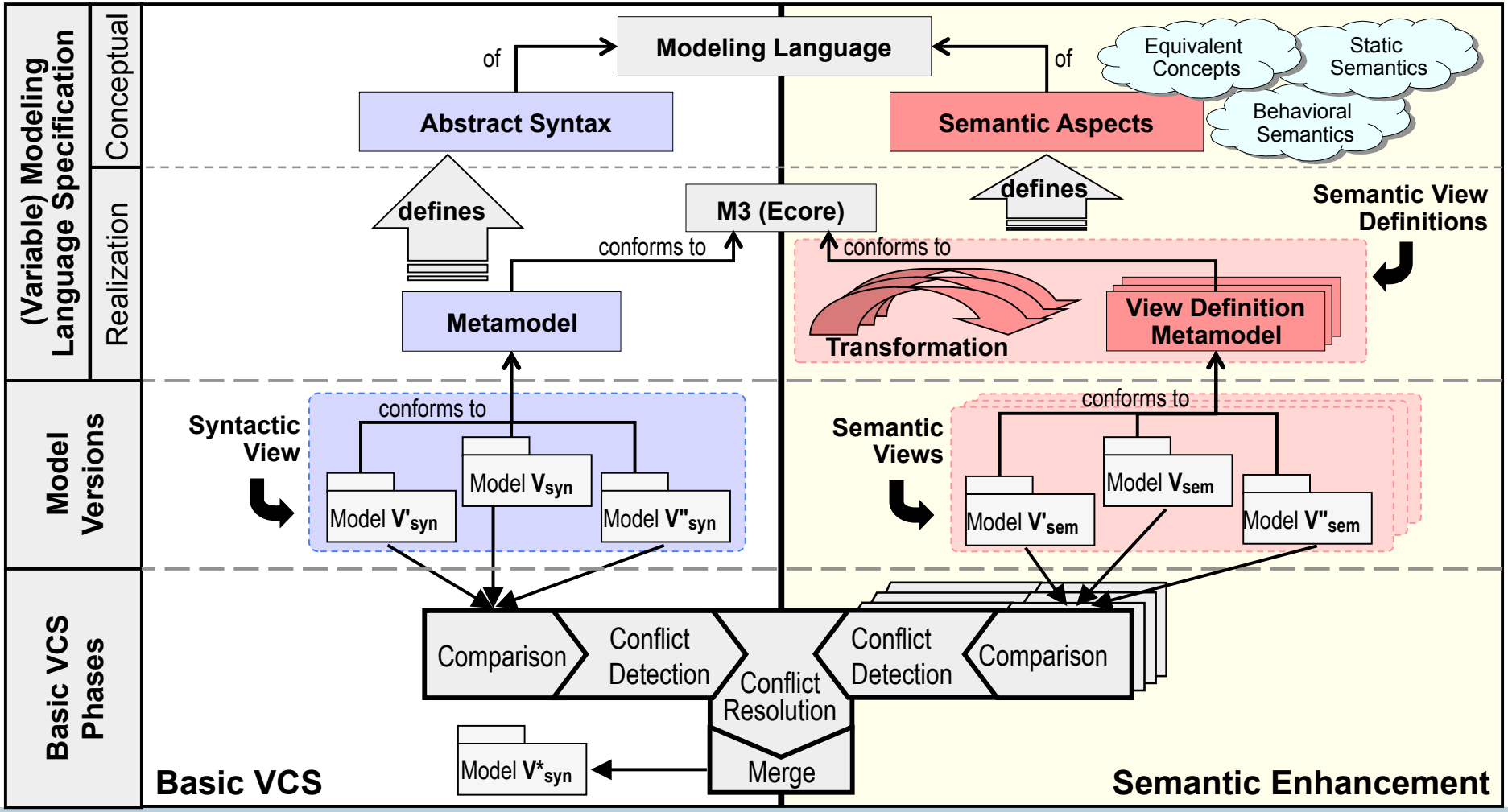
- By inspecting the structural features, namely
  - the attributes and
  - references of a model element

  one can determine whether the model element as a whole has been updated

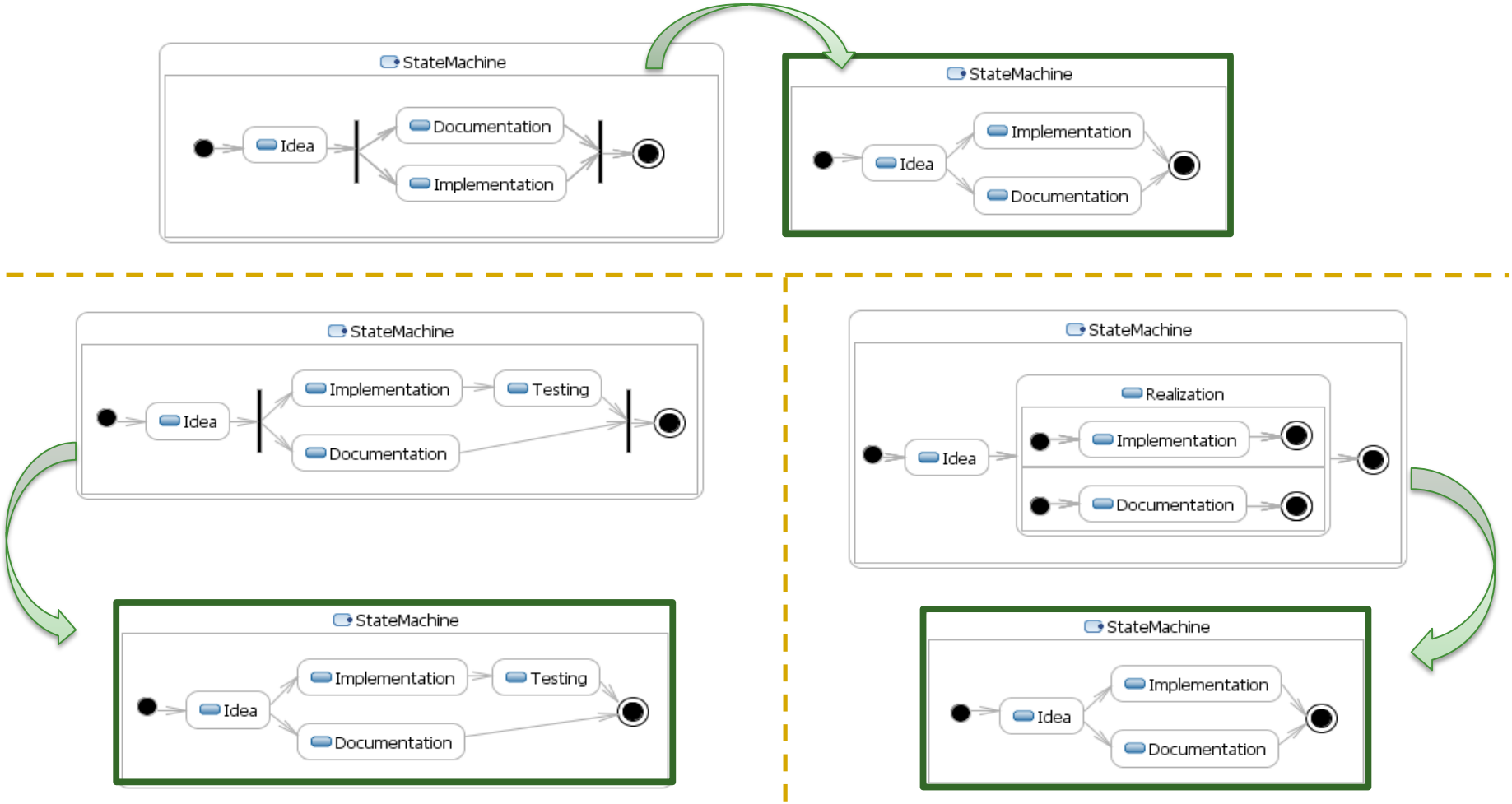
- Particularly four different strategies for the detection of structural changes in a graph that are of interest for conflict detection can be identified
  - Attribute update (**ATT**)
  - Reference update (**REFS**)
  - Role update (**ROL**)
  - Referenced element update (**REF**)

- By setting of those strategies on model elements and properties configurable comparison can be achieved

Architecture of Semantic Conflict Detection



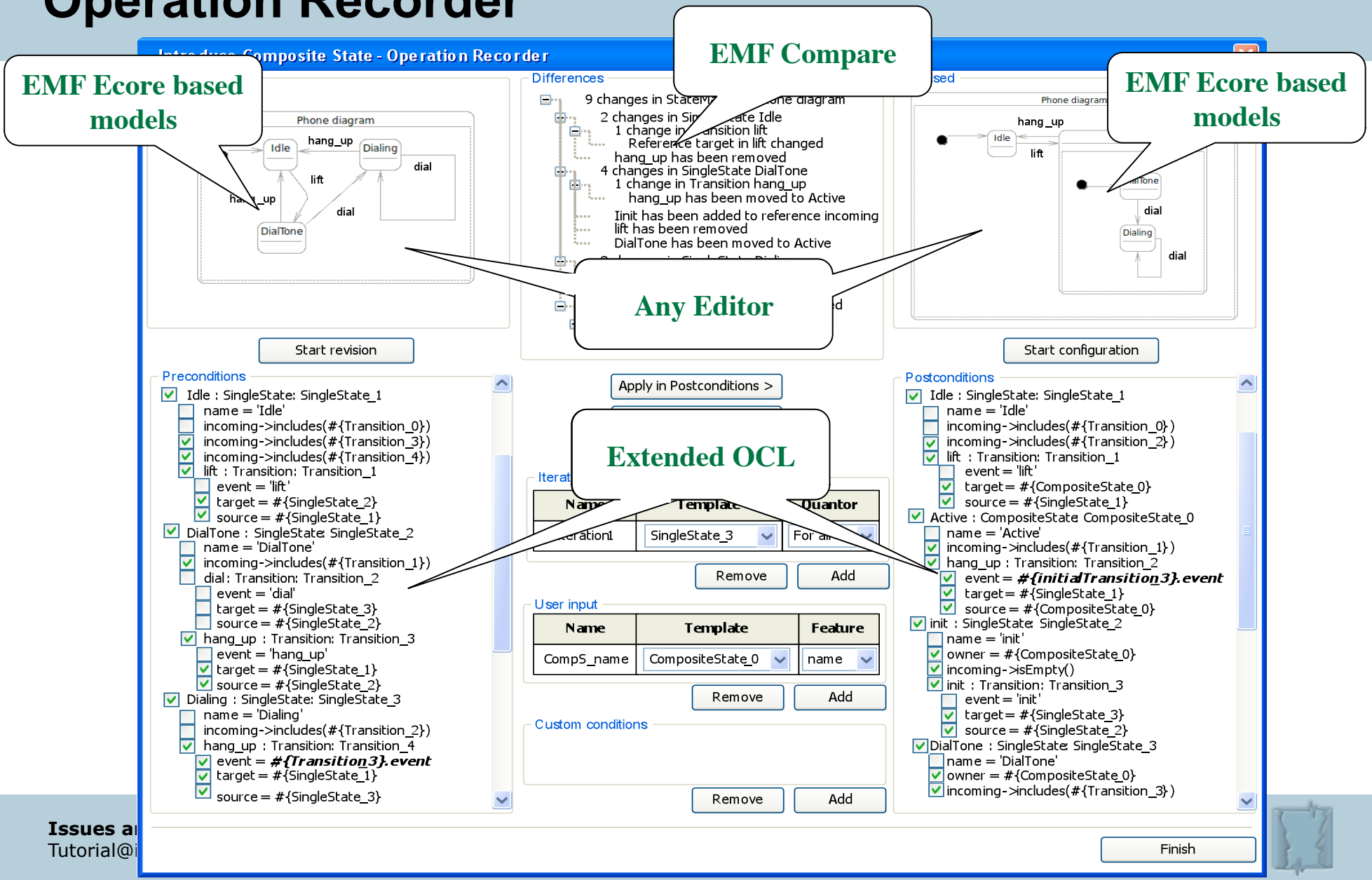
Example of Semantic Conflict Detection



Operation Recorder – Motivation

- Detection of **composite operations** and **refactorings** enables…
  - More accurate conflict detection
  - Smart conflict resolution
- Composite operations in modeling environments (accomplished by experts)
  → Fixed, pre-defined composite operations
- Hardly possible to predefine all relevant composite operations
  → Highly valuable to enable “usual modelers” to specify composite operations
- Modelers have domain knowledge
  → But, usually no knowledge on model transformation techniques

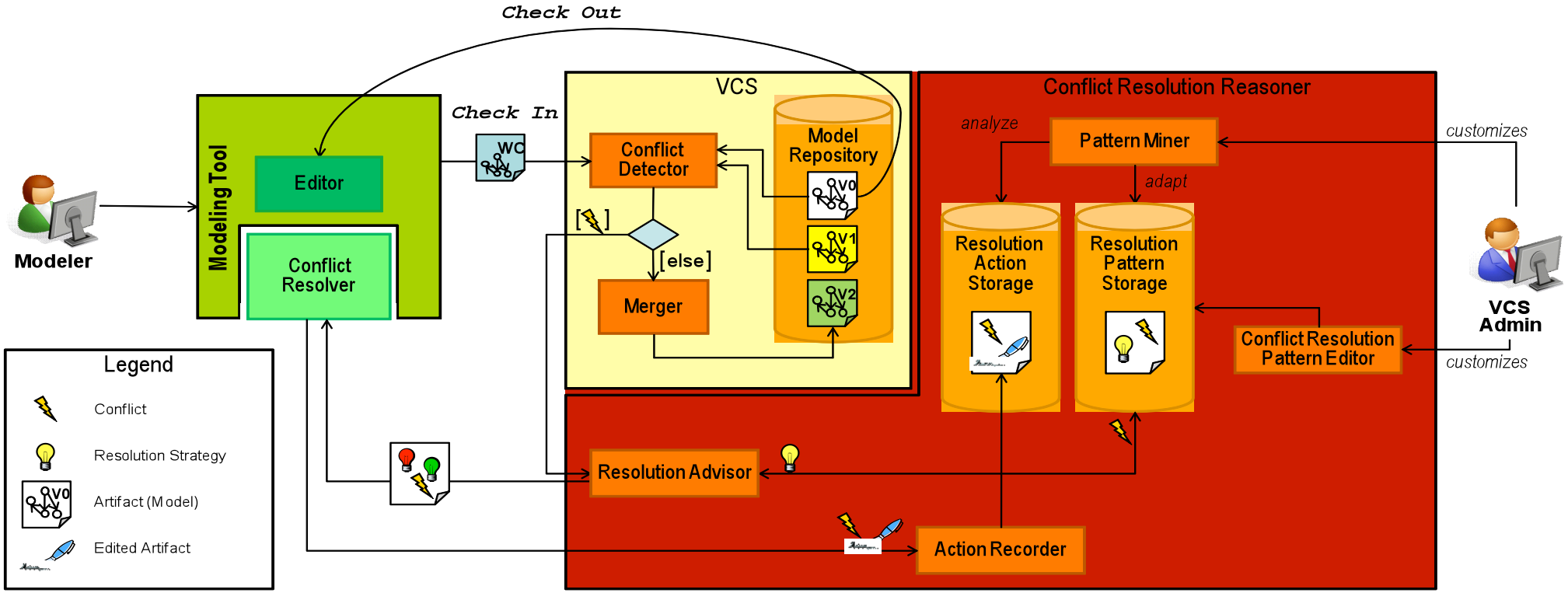
Operation Recorder



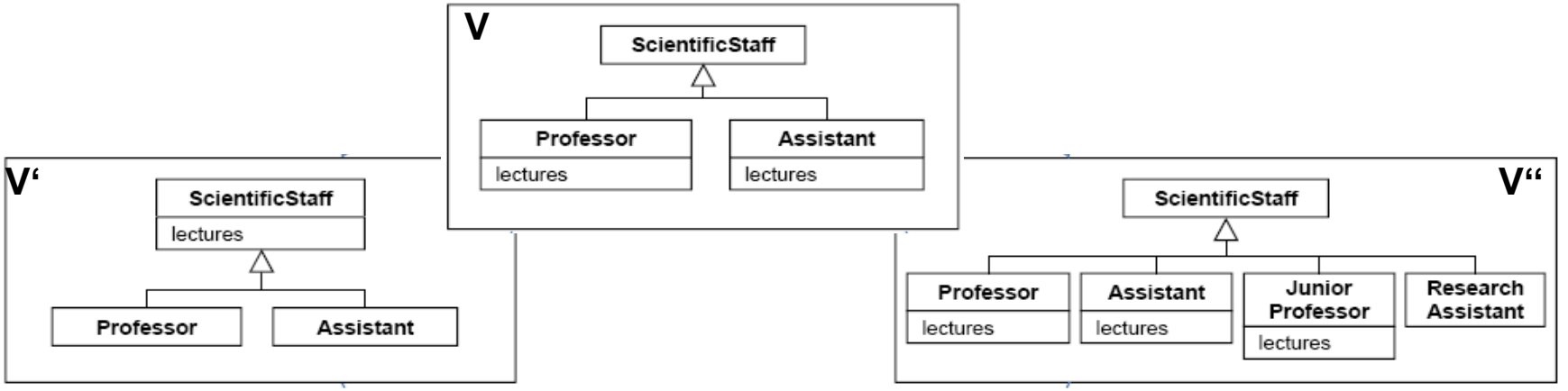
Supportive Conflict Resolution

- **Goal**
  - Accelerate conflict resolution
  - Minimize errors
- **Proposed Solution**
  - Infer general resolution strategies from accumulated historical data
  - Provide enhanced resolution support for repetitive conflict situations
  - Apply collaborative merge in order to capture all user intentions

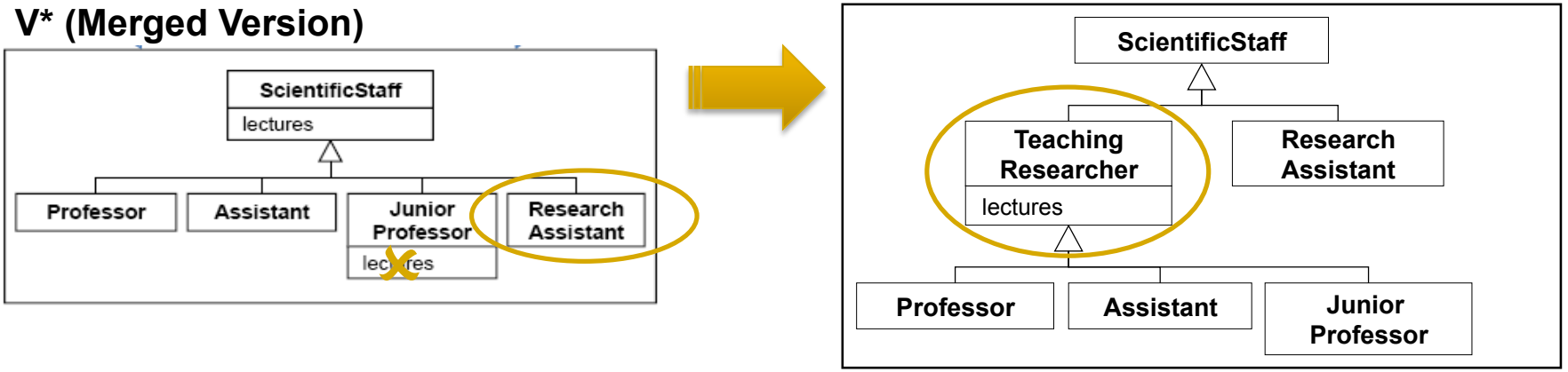
Architecture for Conflict Resolution



Conflict Resolution: Motivation Example



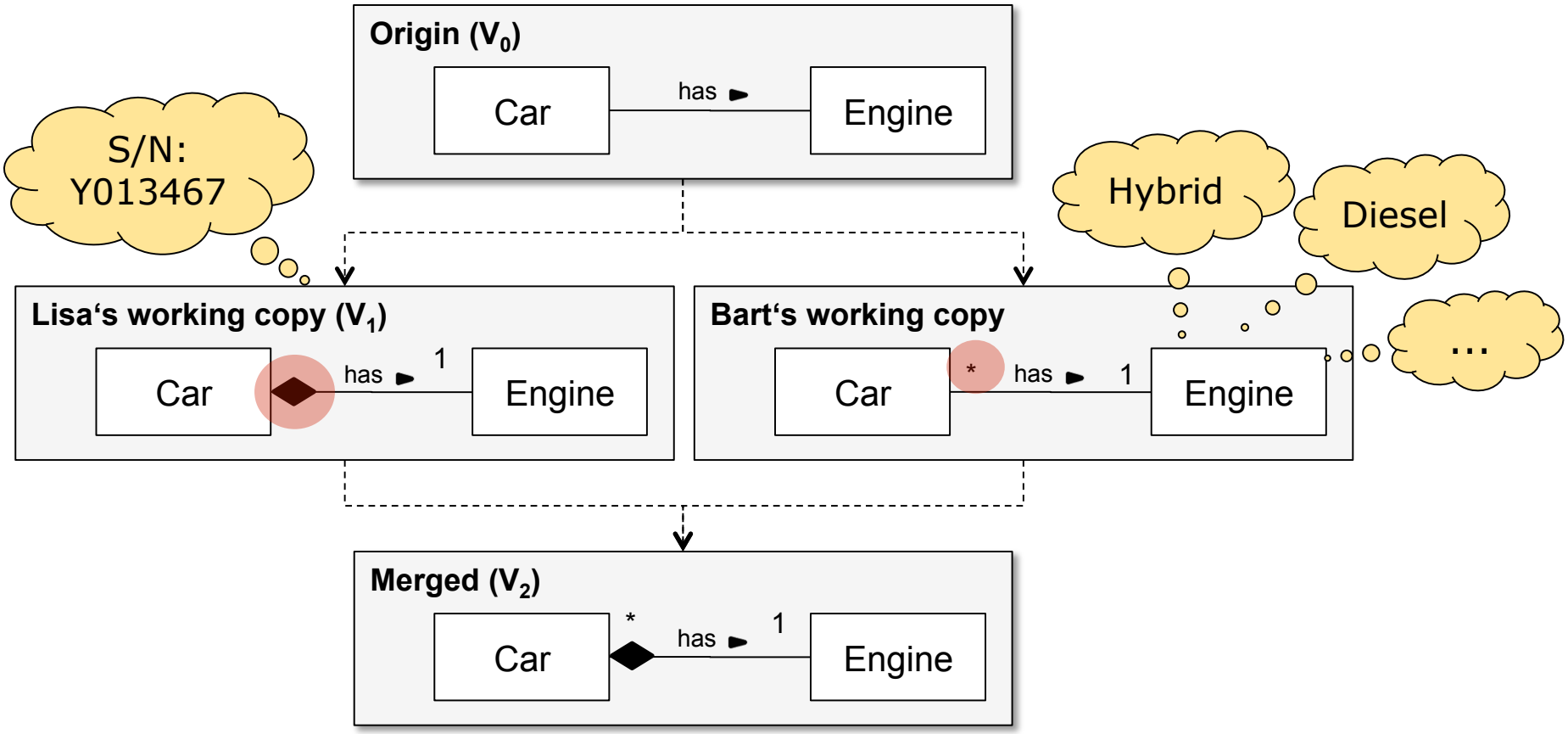
V* (Merged Version)



... but sometimes this will not be enough

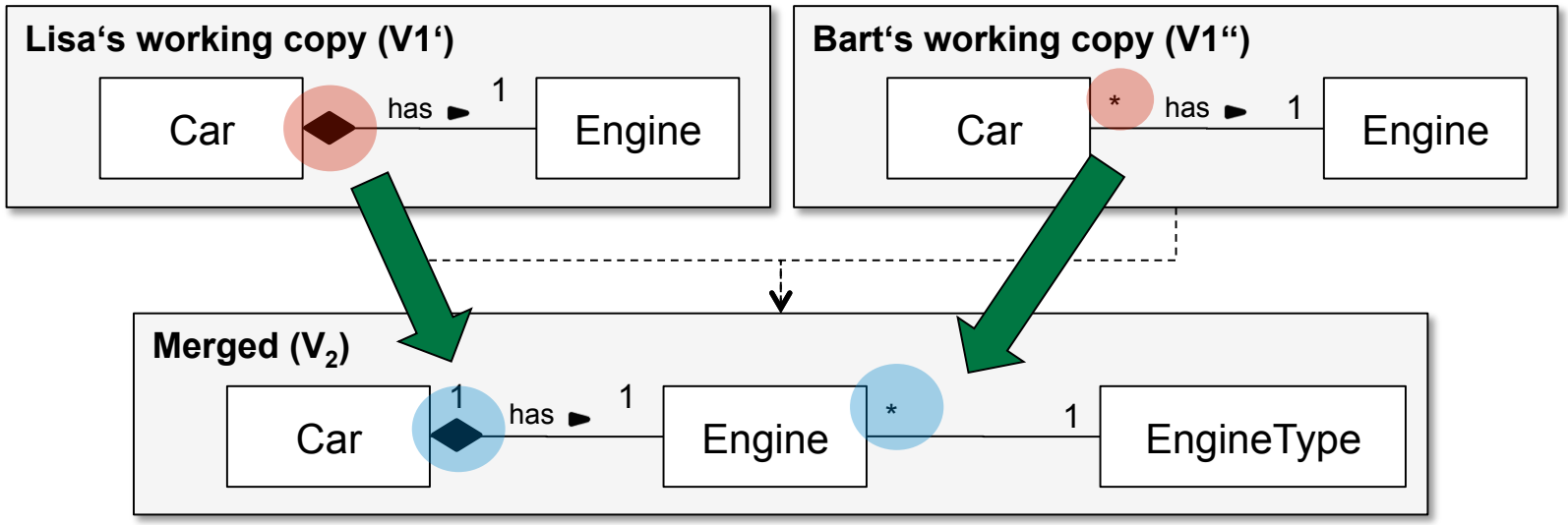
... then Lisa and Bart must resolve
the conflict together

Why Collaborative Conflict Resolution?



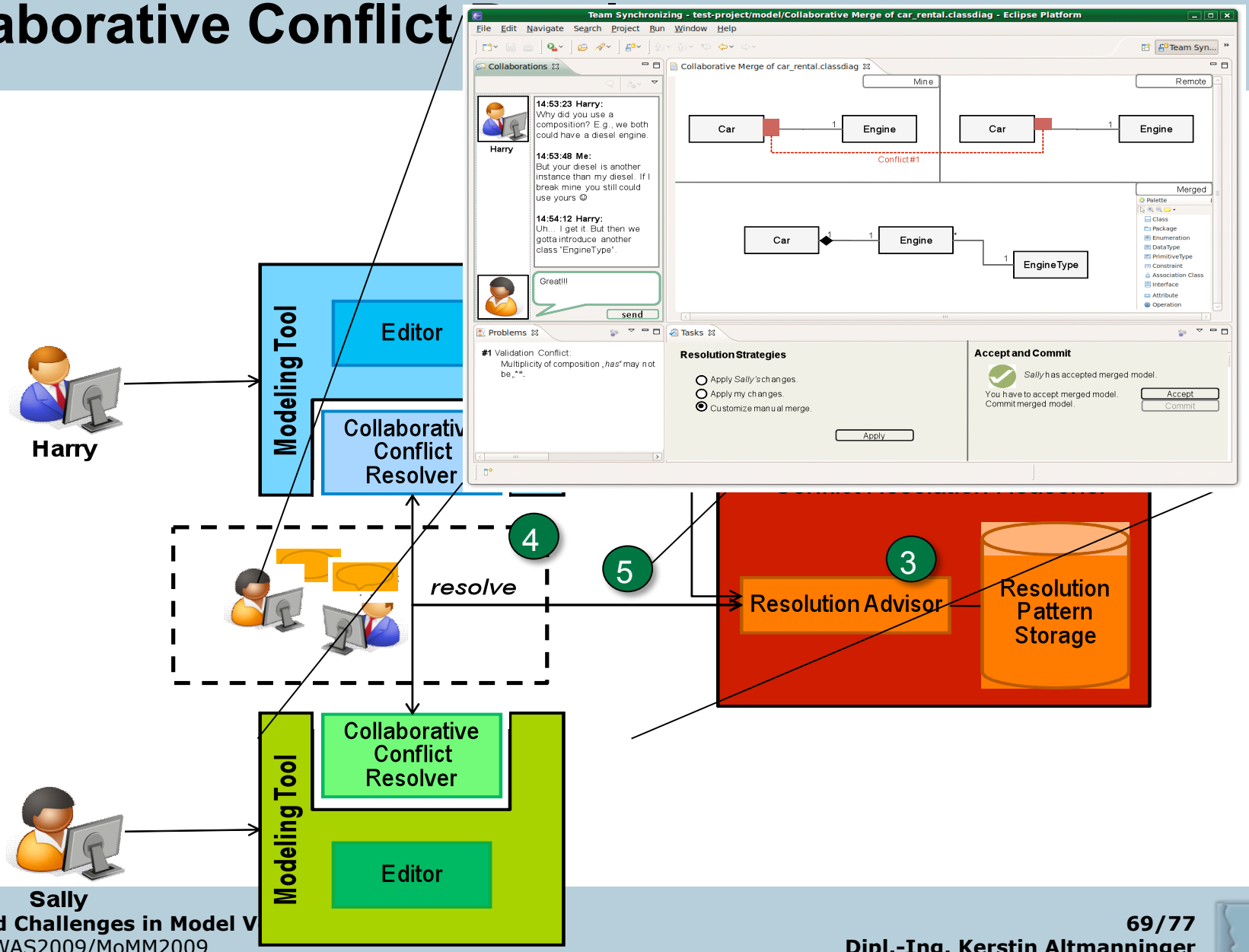
→ Automatic resolution is not possible here!

Why Collaborative Conflict Resolution?

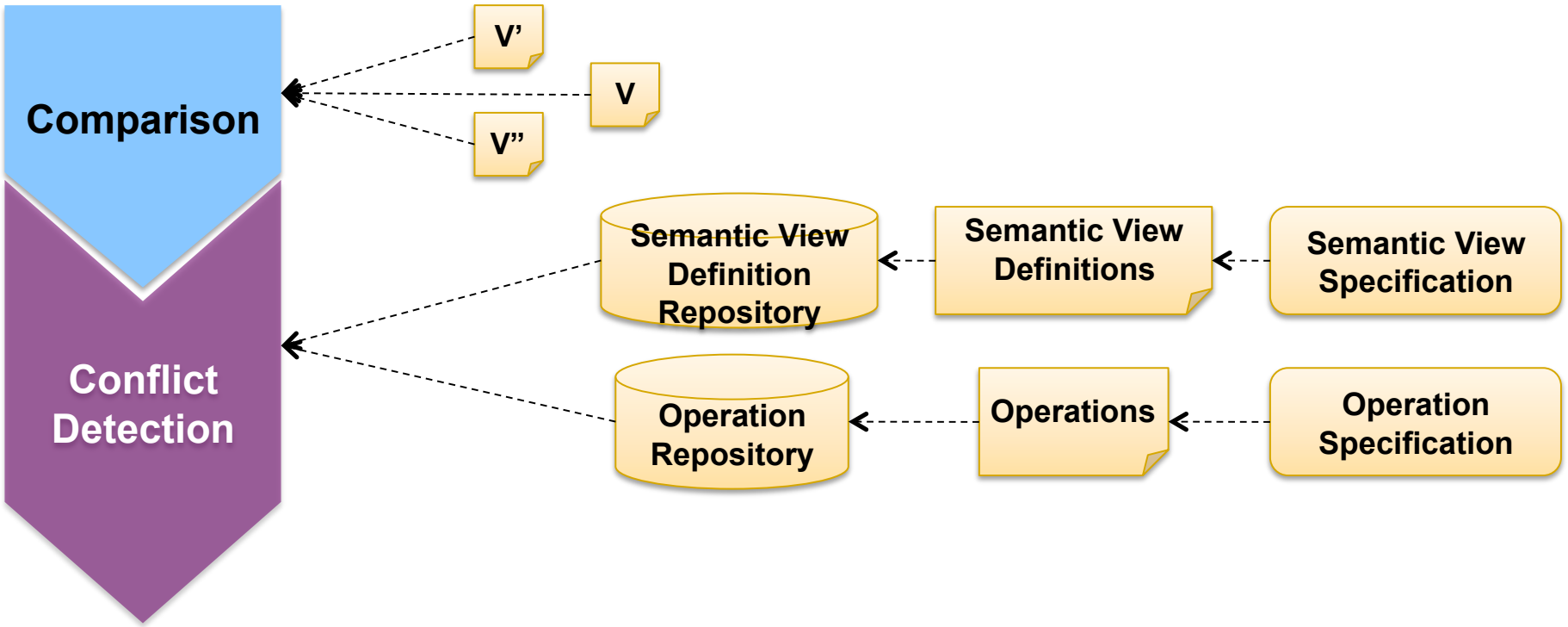


→ Higher semantics
→ Better quality

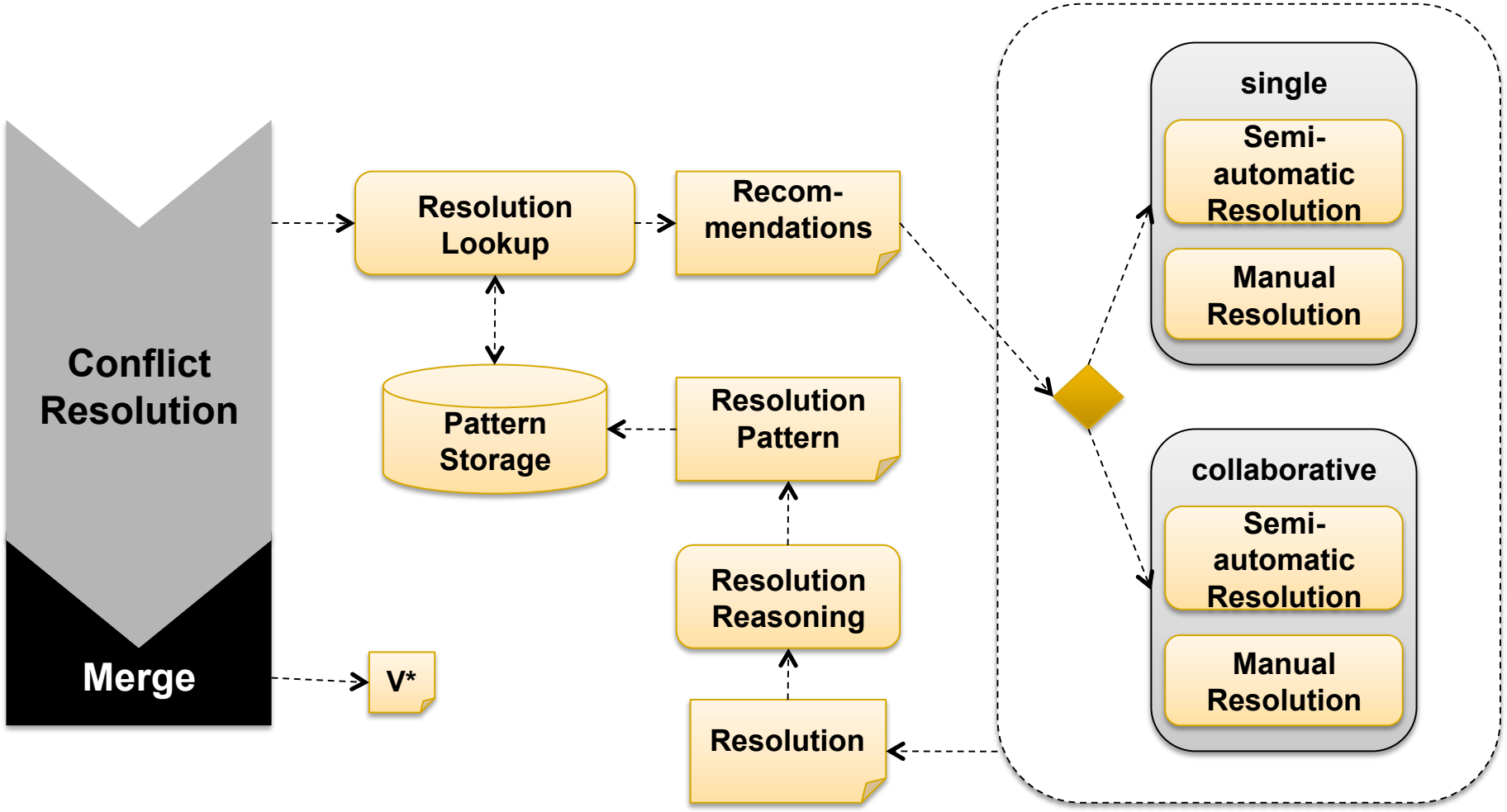
Collaborative Conflict



Summary: AMOR Workflow (1/2)



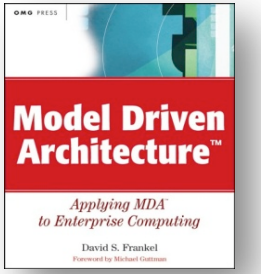
Summary: AMOR Workflow (2/2)

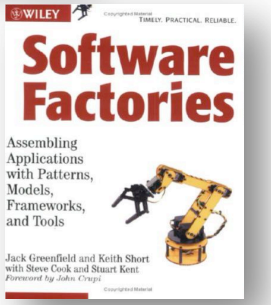


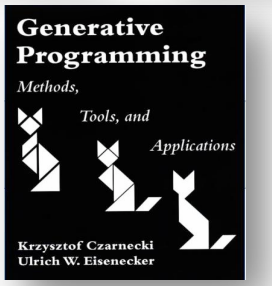
Literature

Model Driven Development

- **Books**
  - K. Czarnecki, U.W. Eisenecker: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 2000
  - D.S. Frankel: Model Driven Architecture – Applying MDA to Enterprise Computing. Wiley, 2003
  - J. Greenfield, K. Short: Software Factories. Wiley, 2004
  - S.J. Mellor, M.J. Balcer: Executable UML: a foundation for model-driven architecture. Addison-Wesley, 2002
  - IEEE Computer, Special Issue on Model Driven Engineering. February 2006
  - Cover Feature: Model Driven Engineering by Douglas Schmidt, Vanderbilt University
- **Websites**
  - www.codegeneration.net/
  - www.metacase.com
  - www.planetmde.org
  - www.omg.org/mda/
  - www.modelware-ist.org







Versioning (1/2)

- **Articles**
  - Altmanninger, K., Seidl, M. and Wimmer, M., “A Survey on Model Versioning Approaches”, International Journal of Web Information Systems (IJWIS), Vol. 5 No. 3, 2009
  - Conradi, R. and Westfechtel, B., “Towards a uniform version model for software configuration management”, *Software Configuration Management*, Springer, 1997
  - Mens, T. “A state-of-the-art survey on software merging”, *IEEE Transactions on Software Engineering,* Vol. 28 No. 5, 2002
  - Estublier, J., Leblang, D., van der Hoek, A., Conradi, R., Clemm, G., Tichy, W. F. and Wiborg-Weber, D., “Impact of software engineering research on the practice of software configuration management”, *ACM Transactions on Software Engineering and Methodology,* Vol. 14 No. 4, 2005

Versioning (2/2)

- **Conference & Workshop Papers**

  - Alanen, M. and Porres, I.: *Difference and Union of Models*. In UML 2003 – The Unified Modeling Language, Springer, 2003.
  - Altmanninger, K.: *Models in Conflict – Towards a Semantically Enhanced Version Control System for Models.* In Models in Software Engineering; Workshops and Symposia at MoDELS 2007, Springer, 2008.
  - Cicchetti, A., Ruscio, D. D. and Pierantonio, A.: *A Metamodel Independent Approach to Difference Representation.* In Journal of Object Technology, Vol. 6 No. 9, 2007.
  - Kögel, M.: *Towards Software Configuration Management for Unified Models.* In Proceedings of the International Workshop on Comparison and Versioning of Software Models (CVSM), Leipzig, Germany, ACM, 2007.
  - Murta, L., Corrêa, C., Jo a. G. P. and Werner, C.: *Towards Odyssey-VCS 2: Improvements over a UML-based Version Control System.* In Proceedings of the International Workshop on Comparison and Versioning of Software Models (CVSM) in conjunction with the International Conference on Software Engineering, Leipzig, Germany, ACM, 2008.
  - Oda, T. and Saeki, M.: *Meta-modeling based Version Control System for Software Diagrams*. In IEICE Transactions on Information and Systems, Vol. E89-D No. 4, 2006.
  - Schneider, C., Zündorf, A. and Niere, J.: *CoObRA – A small step for development tools to collaborative environments*. In Proceedings of the Workshop on Directions in Software Engineering Environments in conjunction with the 26th International Conference on Software Engineering (ICSE), 2004.

AMOR Project

- Altmanninger, K., Kappel, G., Kusel, A., Retschitzegger, W., Schwinger, W., Seidl, M. and Wimmer, M.: *AMOR – towards adaptable model versioning*. In 1st International Workshop on Model Co-Evolution and Consistency Management (MCCM) in conjunction with the 11th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2008.
- Brosch, P.: *Improving conflict resolution in model versioning systems*. In Proceedings of the Doctoral Symposium at the 31st International Conference on Software Engineering (ICSE), Companion Volume, pp. 355–358, 2009.
- Brosch, P., Langer, P., Seidl, M. and Wimmer, M.: *Towards end-user adaptable model versioning: The by-example operation recorder*. In Proceedings of the International Workshop on Comparison and Versioning of Software Models (CVSM) in conjunction with the 31st International Conference on Software Engineering (ICSE), IEEE Computer Society, pp. 55–60, 2009.
- Altmanninger, K., Seidl, M. and Wimmer, M.: *A survey on model versioning approaches*. In International Journal of Web Information Systems (IJWIS), vol. 5, no. 3, pp. 271-304, 2009.
- Brosch, P., Langer, P., Seidl, M. and Wimmer, M.: *We can work it out: Collaborative Conflict Resolution in Model Versioning*. Accepted for the European Conference on Computer Supported Cooperative Work, 2009.
- Brosch, P., Langer, P., Seidl, M., Wieland, K. and Wimmer, M.: *Mining of Model Repositories for Decision Support in Model Versioning*. Accepted for the Second Model-Driven Tool & Process Integration Workshop, 2009.
- Brosch, P., Langer, P., Seidl, M., Wieland, K., Wimmer, M., Kappel, G., Retschitzegger, W. and Schwinger, W.: *An Example is Worth a Thousand Words: Composite Operation Modeling By-Example*. In Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems, Denver, Colorado, USA, October 4-9, 2009.
- Wieland, K.: *Advanced Conflict Resolution Support for Model Versioning Systems*. In Proceedings of the Doctoral Symposium at the 12th International Conference on Model Driven Engineering Languages and Systems (MODELS), Denver, Colorado, USA, October 2009.
- Altmanninger, K., Brosch, P., Kappel, G., Langer, P., Seidl, M., Wieland, K. and Wimmer, M.: *Why Model Versioning is Needed!? An Experience Report*. In Proceedings of the Models and Evolution Workshop, joint MODELS Workshop on Model-Driven Software Evolution (MoDSE) and Model Co-Evolution and Consistency Management (MCCM) at the 12th International Conference on Model Driven Engineering Languages and Systems (MoDELS), Denver, CO, USA, October 2009.

Thank you for your attention!

http://smover.tk.uni-linz.ac.at

http://www.modelversioning.org