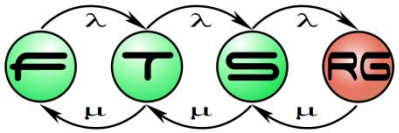
Model Management

Dániel Varró
Ákos Horváth
Gábor Bergmann

also contributed by
M. Brambilla, J. Cabot and M. Wimmer

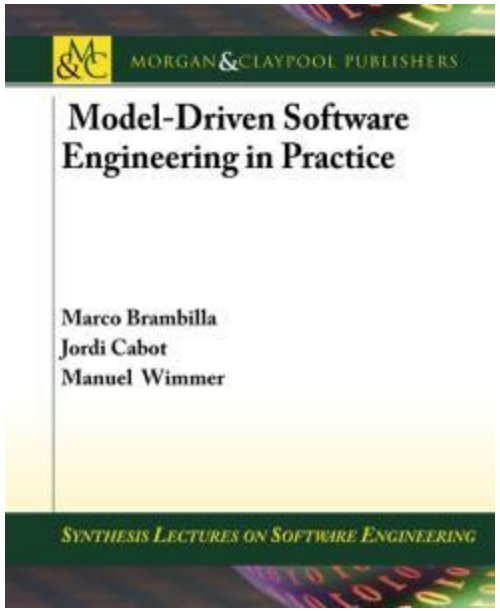
Model Driven Systems Development
Lecture 10



Chapter #10

MANAGING MODELS

Teaching material for the book
Model-Driven Software Engineering in Practice
by Marco Brambilla, Jordi Cabot, Manuel Wimmer.
Morgan & Claypool, USA, 2012.

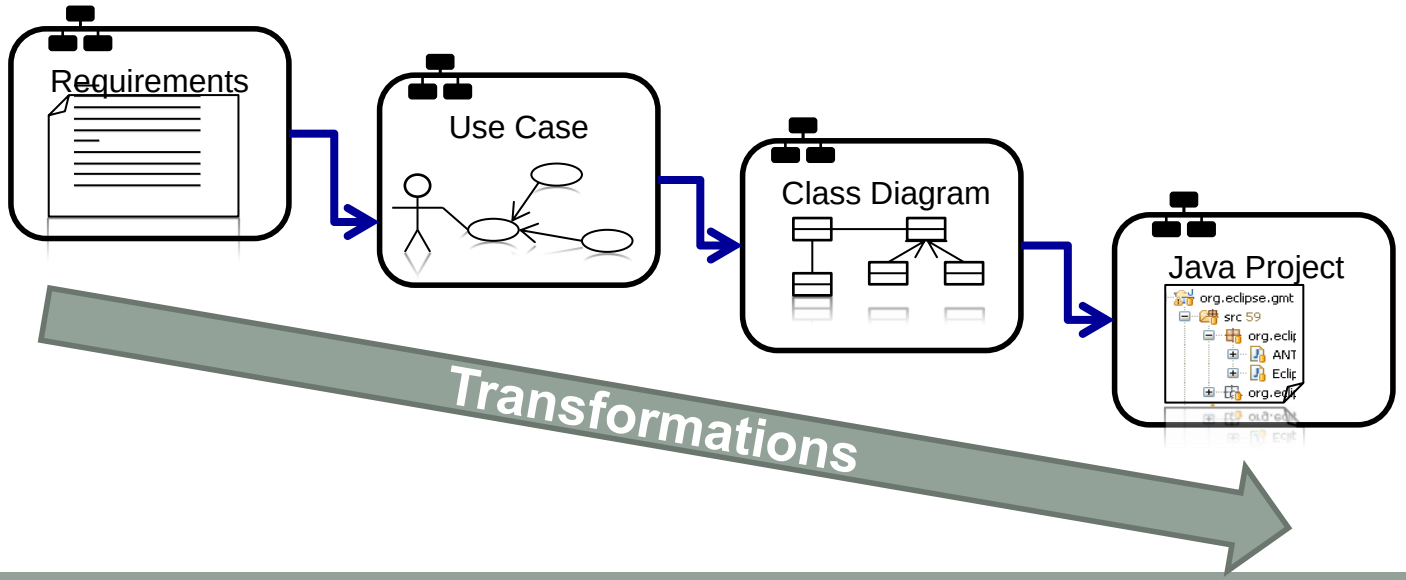


Motivation

Why Model managing?

- In MDE *everything is a model* but as important as that, *no model is an island*

- All modeling artefacts in a MDE project are interrelated. These relationships must be properly managed during the project lifecycle



Content

- Model Interchange & Persistence
  - Persistence to files (XMI, JSON)
  - Persistence to repositories (CDO, EMFStore, NeoEMF)
  - Interchange between tools
- Collaborative Modeling
  - Connectivity
  - Access Control
  - Versioning
  - Conflict Management
- Misc: Model Co-Evolution, Megamodeling

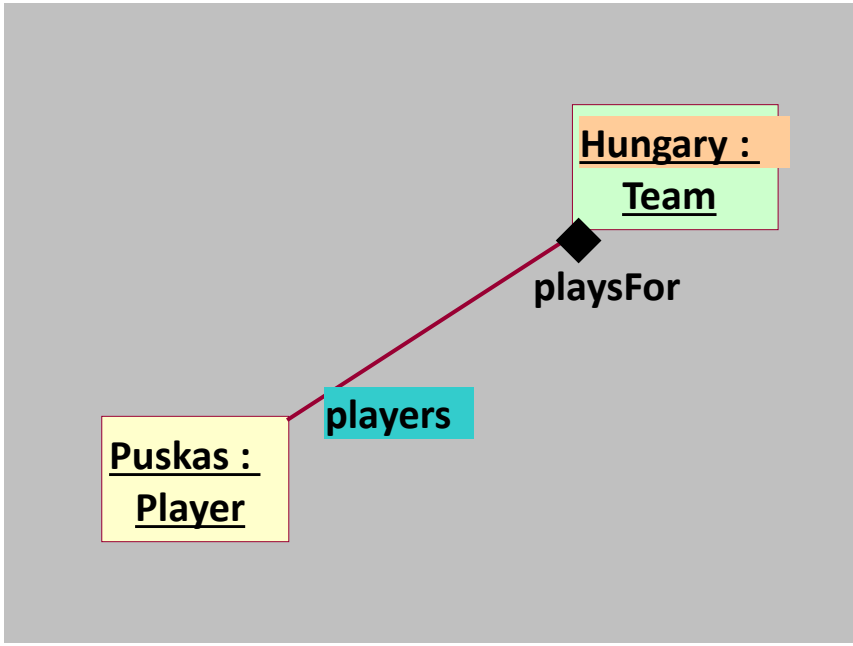
MODEL PERSISTENCE AND INTERCHANGE

- Persistence to files (XMI, JSON)
- Persistence to repositories (CDO, EMFStore, NeoEMF)
- Interchange between tools

Persist to file: XMI 2.0 document

- OMG XMI Standard (XML Metadata Interchange)
  - Supported by EMF out-of-the-box

```
<fb:Model xmlns:fb="…" xmlns:xmi="…"
 <teams xmi.type="Team" xmi.id="t1" name="Hungary">
   <players xmi.id='p1'
    name='Puskas'
    number='10'
    playsFor='t1'/>
  </teams>
</fb:Model>
```



Persist to file: emfjson document

- JSON standard: supported by **emfjson** project
  - Similar to XMI, no substantial benefits

```
{
    "eClass": "http://www.eclipselabs.org/emfjson/junit#//Node",
    "label": "root",
    "target": {
        "$ref": "//@child.0"
    },
    "child": [
        {
            "eClass": "http://www.eclipselabs.org/emfjson/junit#//Node",
            "label": "n1",
            "source" : {
                "$ref": "/"
            }
        }
    ]
}
```

Fundamental Question: Cross-refs

- Models are **graphs**, not trees → cross-references
  - AST not enough, must use linking
  - Fragmentation into smaller files → cross-file refs
- Cross-reference serialization options

	Identifier-based	Positional (fragile!)
Path-based (absolute or relative)	../foo/bar/baz	/child[3]/child[5]
Direct	123e4567-e89b-…	-

XMI standard solutions
- XPath
- XMI ID (resource-relative)
- XMI UUID (globally unique)

emfjson is similar

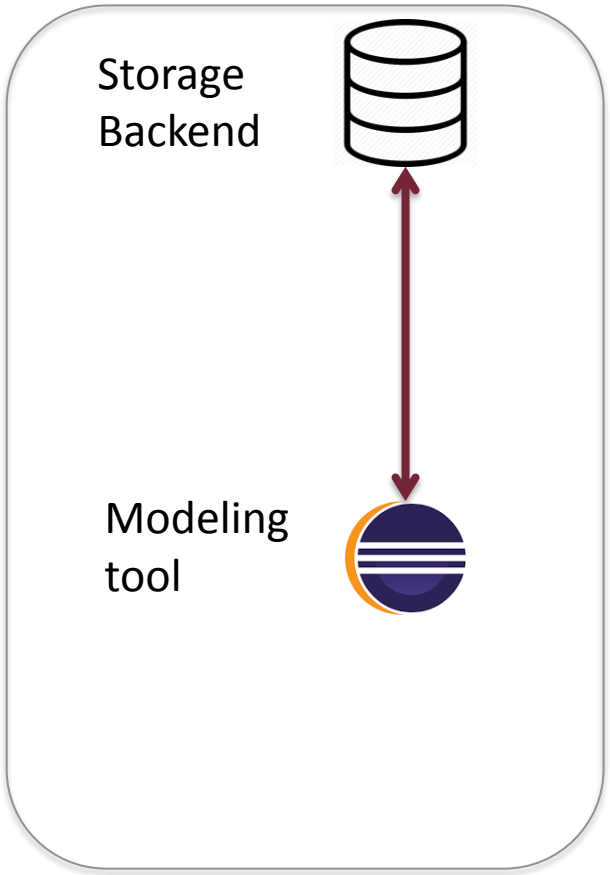
Model Persistence

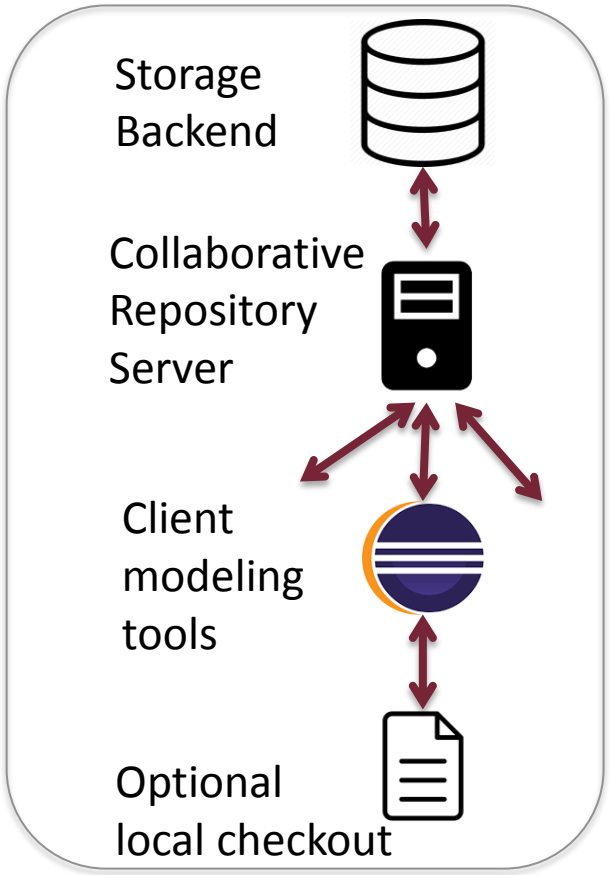
- Typically models are serialized in plain files, following the previous XMI format or any other proprietary XML format

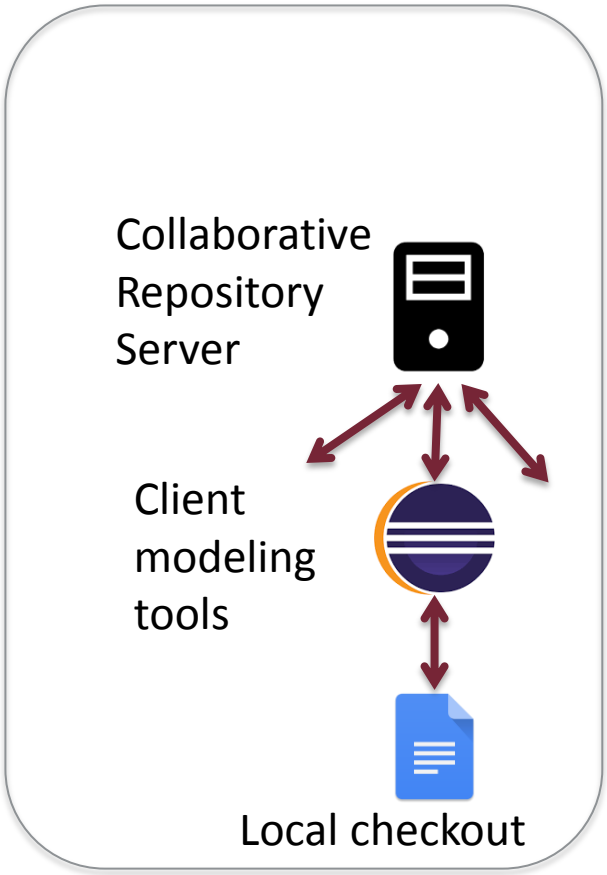
- Doesn’t work well with large models. Scalability issues
  - Loading the whole model in memory may not be an option
  - Random access strategies plus lazy loading (i.e. loading on demand) are needed

NeoEMF vs. CDO vs. EMF Store

- NeoEMF
  - New & simple
  - No collaboration
- Eclipse CDO
  - Most features
  - Most daunting
- EMF Store
  - Compromise
  - Offline checkout





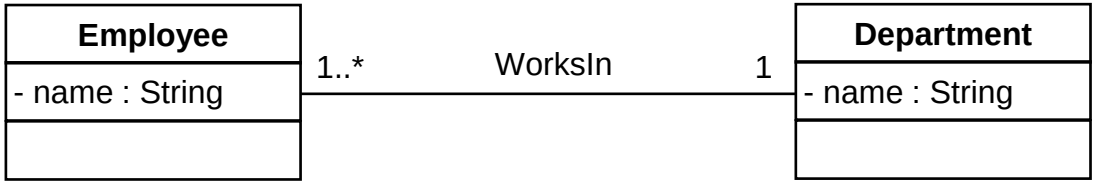


Model Once Open Everywhere

- There's a clear need to be able to exchange models among different modeling tools
  - In a perfect world, you'd be able to choose ToolA for specifying model, ToolB to check its quality, ToolC to execute it….
- We are still far away from this goal
- Solution attempt: XMI (XML Metadata Interchange), a standard adopted by OMG for serializing and exchanging UML and MOF models
- But each tools seems to understand the standard in a different manner

XMI example

(simplified and partial versions of the actual XMI files)




```
<packagedElement xmi:type="uml:Class" xmi:id="c001"
name="Employee">
  <ownedAttribute xmi:id="a001" name="name"/>
</packagedElement>
<packagedElement xmi:type="uml:PrimitiveType"
xmi:id="t001" name="String"/>
<packagedElement xmi:type="uml:Class" xmi:id="c002"
name="Department">
  <ownedAttribute xmi:id="a002" name="name"
type="t001"/>
</packagedElement>
<packagedElement xmi:type="uml:Association"
xmi:id="as001" name="WorksIn" memberEnd="e001e002">
  <ownedEnd xmi:id="e001" type="c002"
association="as001"/>
  <ownedEnd xmi:id="e002" name="" type="c001"
association="as001">
    <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="un001" value=""/>
  </ownedEnd>
</packagedElement>
```

ECLIPSE

```
<UML:Class xmi.id='c001' name='Employee'
visibility='public' isSpecification='false'
isRoot='false' isLeaf='false'
isAbstract='false' isActive='false'>
 <UML:Classifier.feature>
  <UML:Attribute xmi.id='a001' name='name'
visibility='public' isSpecification='false'
ownerScope='instance'
changeability='changeable'
targetScope='instance'>
   <UML:StructuralFeature.multiplicity>
    <UML:Multiplicity xmi.id='m001'>
     <UML:Multiplicity.range>
      <UML:MultiplicityRange xmi.id='mr001'
lower='1'upper='1'/>
     </UML:Multiplicity.range>
    </UML:Multiplicity>
   </UML:StructuralFeature.multiplicity>
…
```

ArgoUML

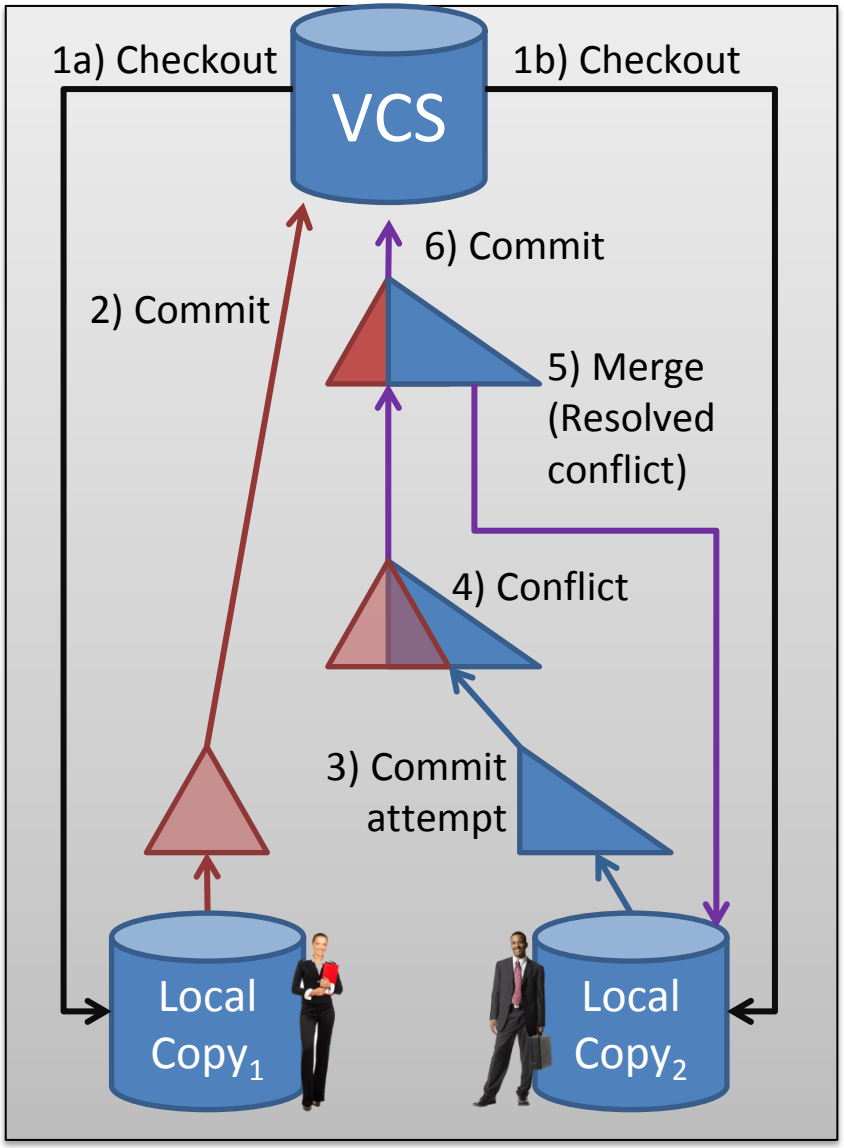
Connectivity → Access Control → Versioning → Conflicts

COLLABORATIVE MODELING

Challenges

- Connectivity (online/offline)
- Access Control
  - Granularity & model fragmentation
  - Read & write permissions, obfuscation, policies
- Versioning
  - Versioned Storage
  - Model Comparison (Matching, Differencing)
- Conflict Management
  - Serialization & Locking to avoid conflict
  - Merging to resolve conflict

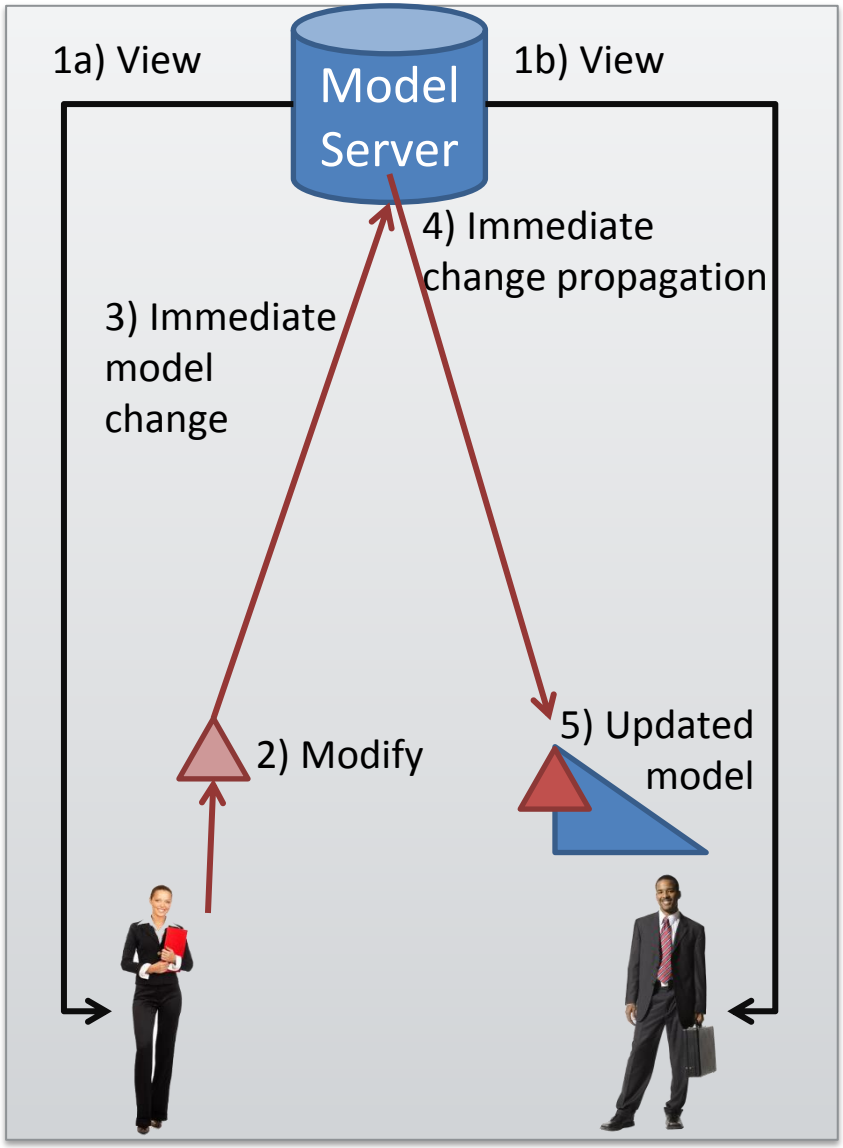
Offline Connectivity



- Workflow
  - „Take home” the model
    - Work on the model separately
    - Use desktop modeling tool
  - Upload updated model
  - VCS-like workflow
- Goal:
  - Offline use of local copies
  - VCS compatibility
  - Pristine modeling tools

Online Connectivity

- Workflow
  - Web client or connected desktop tool
  - Simulataneously by several users
  - ~Google Spreadsheets
- Goal:
  - Efficient change propagation (incrementality)



Model Repositories

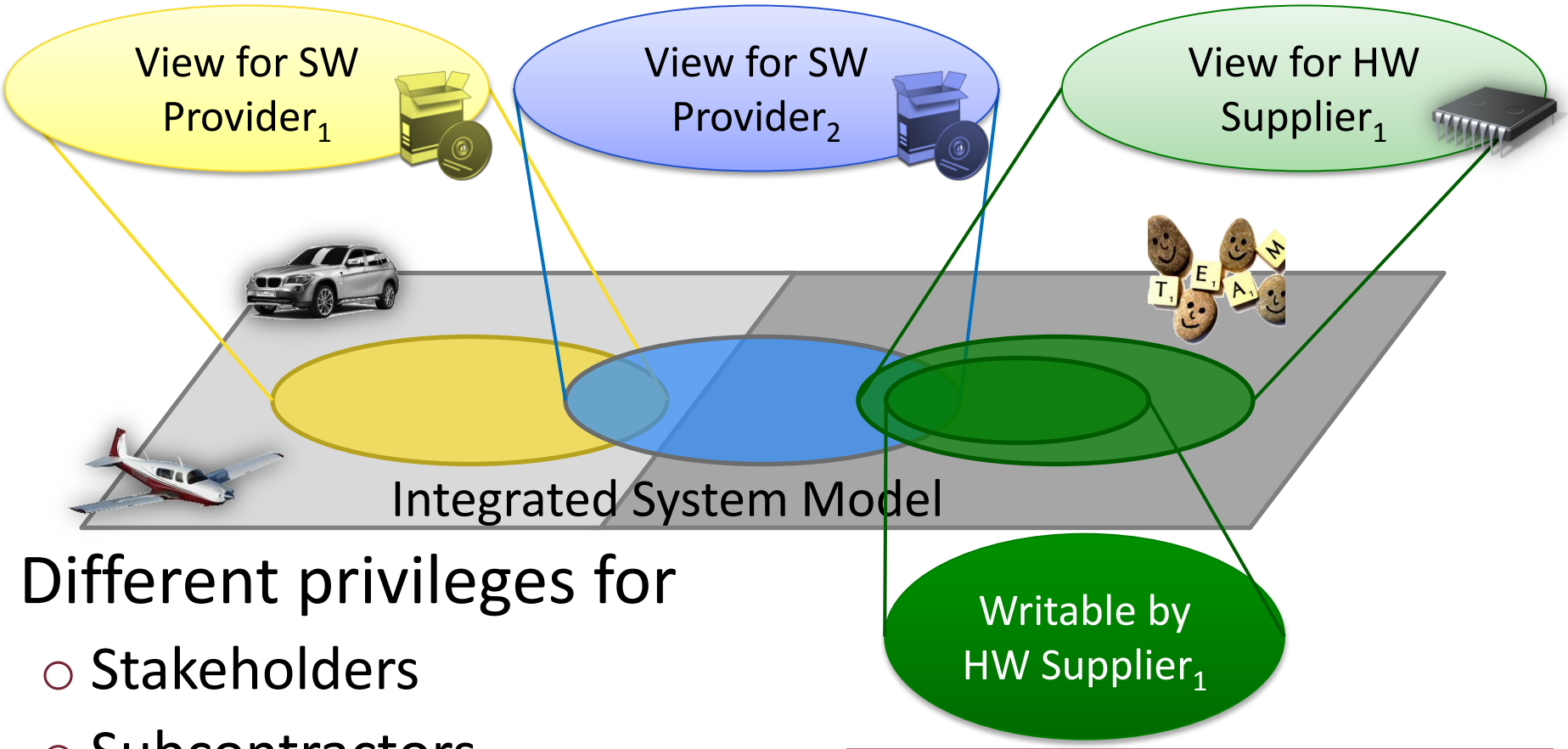
- File-based VCS
- Model-aware repositories
  - **EMFStore**: Eclipse open-source, model-level, offline
  - **CDO**: Eclipse open-source, object-level, online
  - Emerging enterprise solutions
    - E.g. No Magic **Teamwork Cloud**, Obeo **Designer Team**
  - Public cloud-based repositories
    - Axellience **GenMyModel**

OBEO Designer

ACCESS CONTROL

- Granularity & model fragmentation
- Read & write permissions, obfuscation, policies

Access Control in Collaboration



- Different privileges for
  - Stakeholders
  - Subcontractors
  - In-house teams

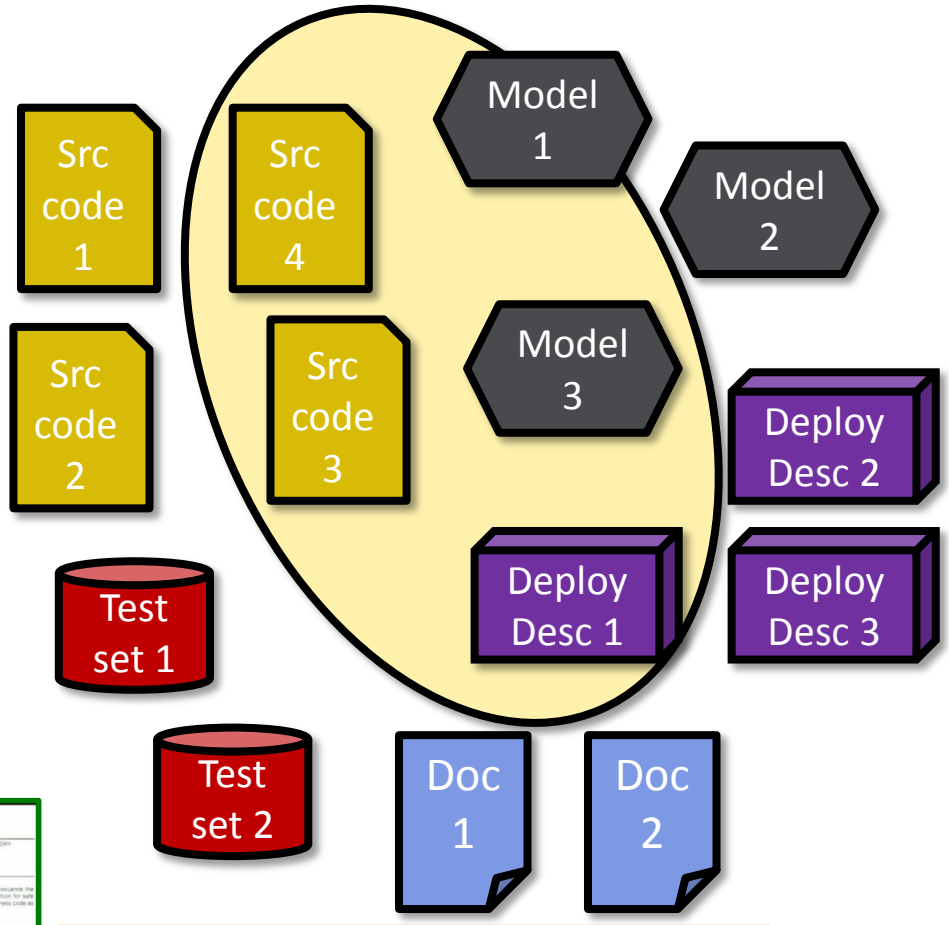
Challenge:
How to provide secure access for collaboration?

File-level Access Control

System Designer

SW Supplier

Platform Provider



Certification Authorities

Problem: How to give partial access to an artifact?

File-level Access Control

System Designer

SW Supplier

Platform Provider

Src code 1

Src code 2

Src code 3

Src code 4

Model 1a

Model 1b

Model 2

Model 3

Deploy Desc 1

Deploy Desc 2

Deploy Desc 3

Test set 1

Test set 2

Doc 1

Doc 2

Certification Authorities

Traditional (Git/SVN) Solution:

- Splitting artifacts
- All-or-nothing access

File-level Access Control

Consequence:
- ~1000 files for large automotive models

System Designer

Src code 1

Src code 2

Src code 3

Src code 4

Test set 1

Test set 2

Model 1a2

Model 1a1

Model 1b

Model 2

Model 3a

Model 3b

Deploy Desc 1

Deploy Desc 2

Deploy Desc 3

SW Supplier

Platform Provider

Certification Authorities

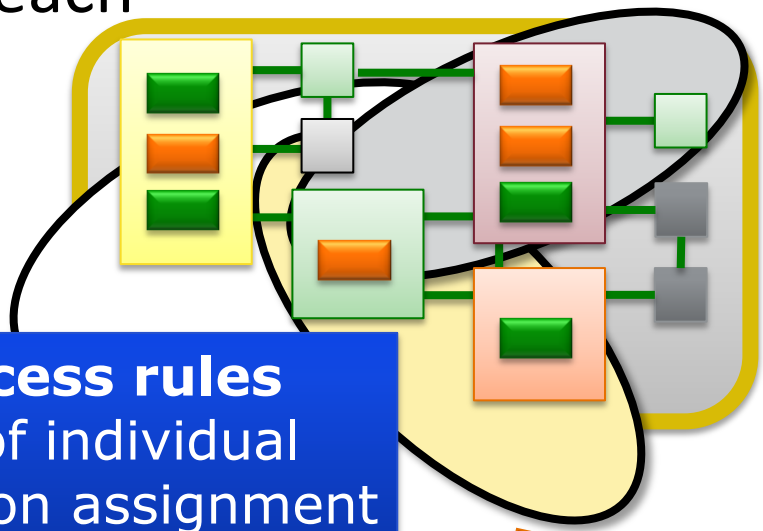
Limits:
- Rigidity – can we change permissions?
- Cyclic dependencies between files?
- Hiding only some attributes of an object?
- Obfuscating an attribute, without hiding it?

Model-level Access Control

- Fine-grained access control
  - Additional access restrictions
    - complementing file-based solutions
  - Grant separate permissions on each
    - object (class instance)
    - slot (attribute instance)
    - link (reference instance)

assets

MONDO
SCALABLE MODELLING AND MODEL MANAGEMENT ON THE CLOUD



Challenge:
How to express policy for so many assets?

→ …use **access rules** instead of individual permission assignment

Challenge:
How to identify assets in rule-based policy?

→ …rules may evaluate **model queries** for the model element

Internal (Referential) Consistency

- **Goal**: self-contained models in standard format
  - Compatible with off-the-shelf model tooling

- **Internal consistency** (≠ well-formedness rules)
  - Object invisible → slots, links, contents invisible
  - Opposite references match up
  - etc.

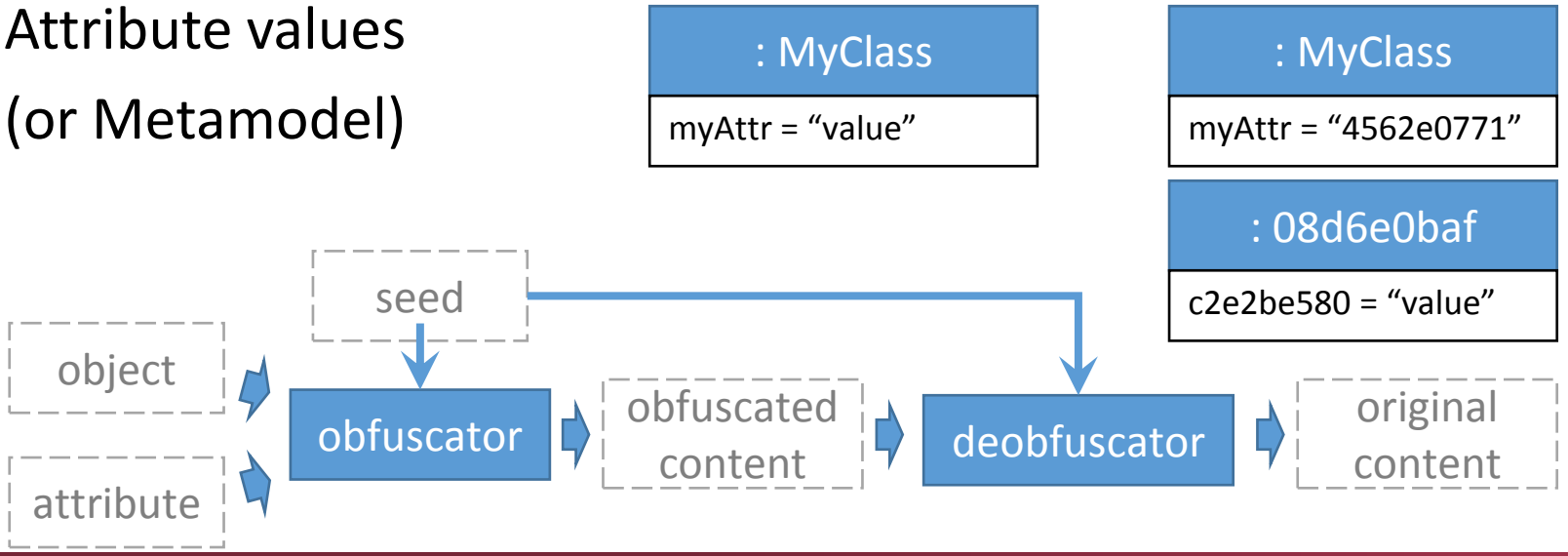
- **Permission dependencies / conflicts**

Deriving Effective Permissions for Modeling Artifacts from Fine-grained Access Control Rules. Csaba Debreceni, Gábor Bergmann, István Ráth and Dániel Varró. First International Workshop on Collaborative Modelling in MDE, Saint Malo, France, Oct 4. 2016

Filtering and Obfuscation

- Read Access Control
  - Hide
    - Objects
    - Reference links
    - Attribute values
  - Obfuscate
    - Attribute values
    - (or Metamodel)

Challenge:
required attributes
(e.g. IDs, names)

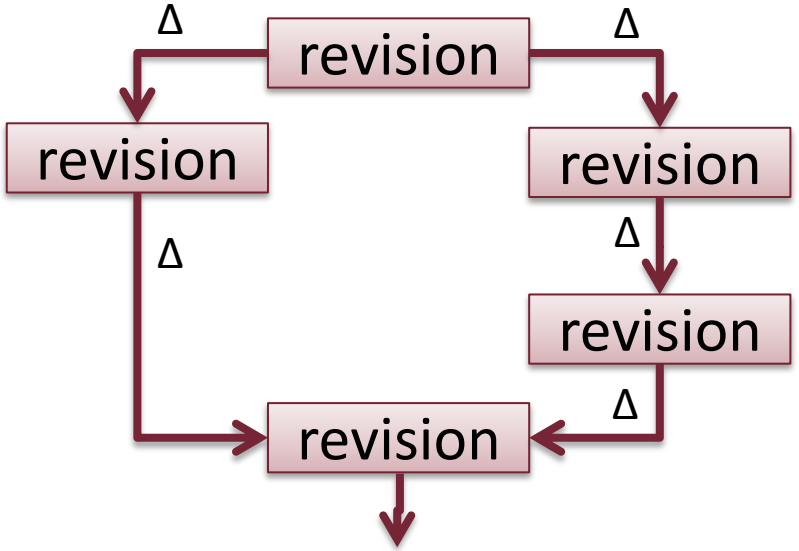


MODEL VERSIONING

- Versioned Storage
- Model Comparison (Matching, Differencing)

Model Versioning & Branch & Merge

- Versioned Storage
  - Store revisions
    - Requires more space
    - Diff operations expensive
  - Store deltas only
    - Requires reliable model differencing & patching
    - History operations expensive



- Version History Structure
  - Linear
  - Branching

In all cases,
model comparison required

Model Comparison

- Comparing two models is a key operation in many model-management operations like model versioning
- Goal of model comparison is to identify the set of differences between two models
- These differences are usually represented as a model themselves, called a *difference model*

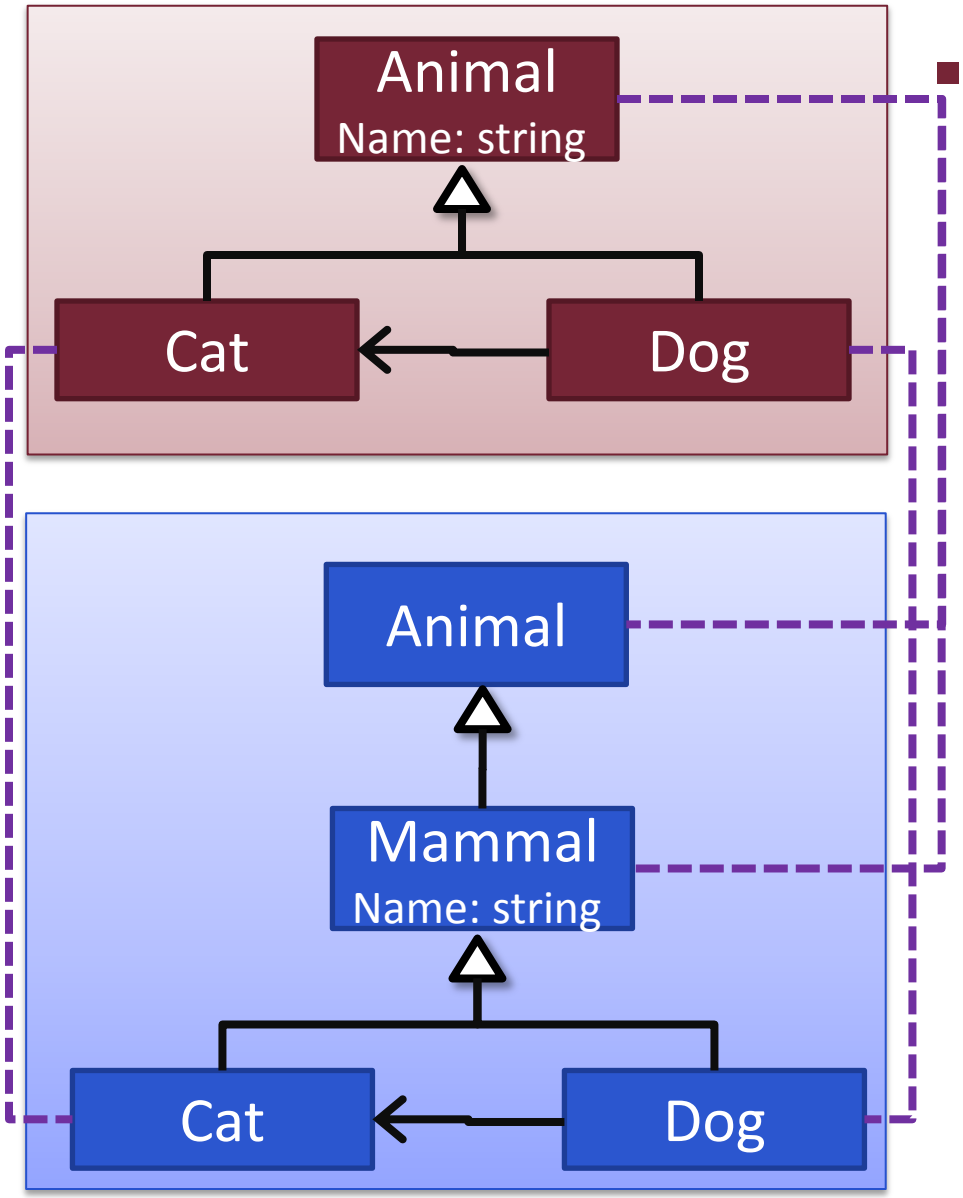
Model Comparison: Model matching

Phase 1 of a model comparison process

- Identify the common elements in the two models
- How do we establish which elements have the same identity?
  - Static identity: explicit id's annotating the elements
  - Signture identity: Identity based on the model element features (i.e. name, contained elements,…)
- Identity can be a probabilistic function (similarity matching)
- Works better if users redefine the concept of matching for specific DSLs (so that their specific semantic can be taken into account)

Model comparison = Graph similarity problem

Example: Model Comparison



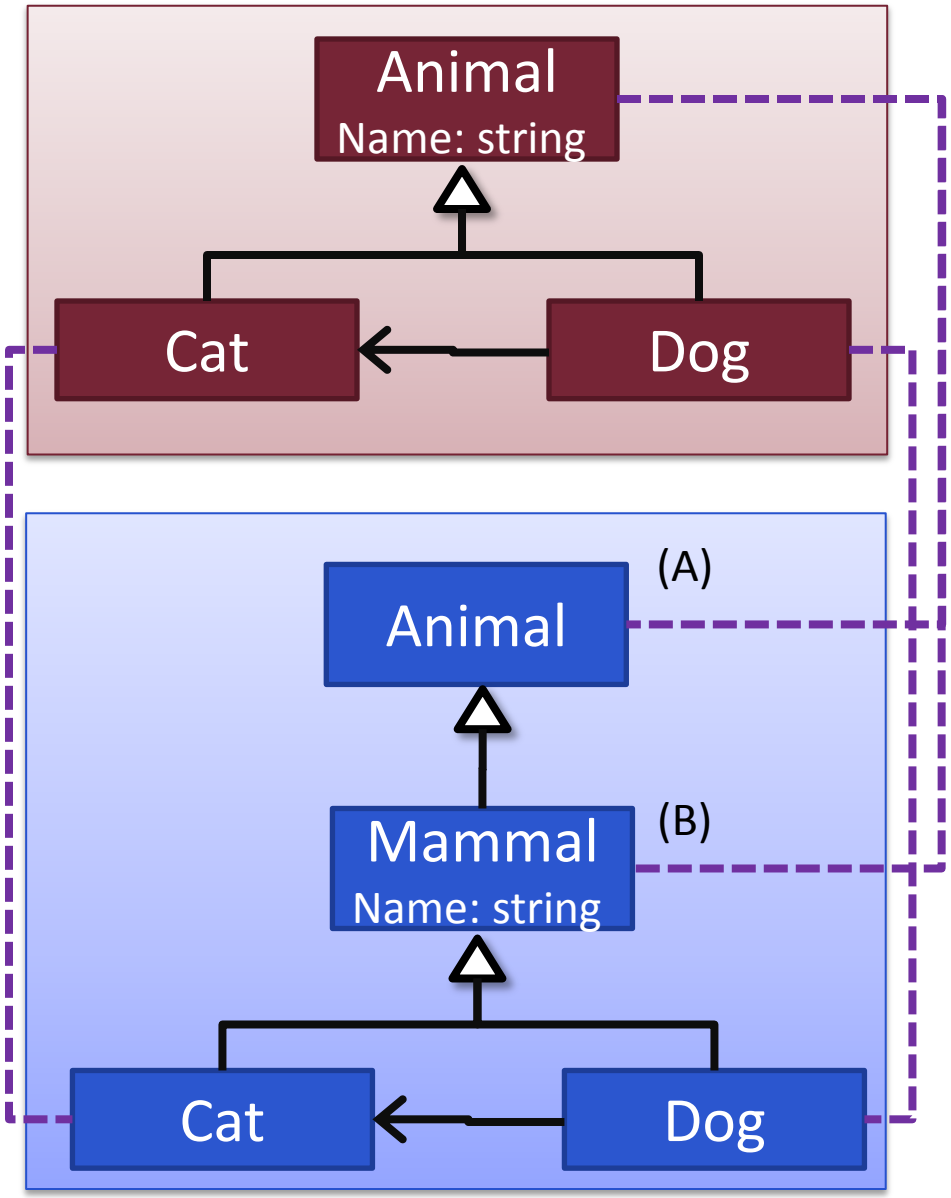
- What is the best matching?

Model Comparison: Model differencing

Phase 2 of a model comparison process

- Matched elements are searched for differences
- A difference corresponds to an atomic add / delete / update / move operation executed on one of the elements
- These differences are collected and stored in the difference model

Example: Model Difference



- What is the difference?
- Matching (A)
  - Del Gen: Cat → Animal
  - Del Gen: Dog → Animal
  - Add Cls: Mammal
  - Add Gen: Mammal → Animal
  - Add Gen: Cat → Mammal
  - Add Gen: Dog → Mammal
  - Move Att:
    Name: Animal → Mammal
- Matching (B)
  - Rename: Animal → Mammal
  - Add Cls: Animal
  - Add Gen: Mammal → Animal

Best Practices to Help Model Matching

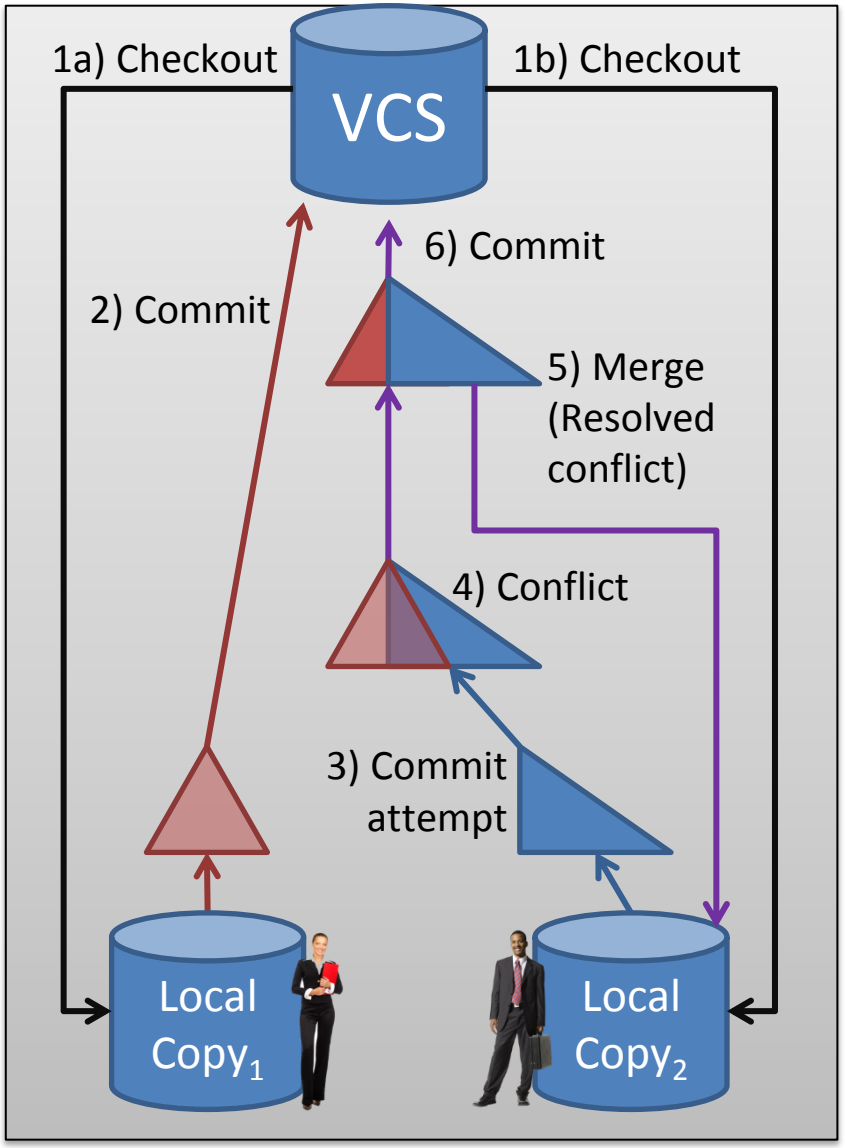
- If possible, use **element identifiers** that are
  - **Unique**
    - Can be local (qualified), broken by *moving elements*
    - Preferably **globally unique** (move-resitant)
  - Stable (across reloading&saving)
- How?
  - **Intrinsic**: part of the domain, available in metamodel
    - E.g. book ISBN number
  - **Extrinsic**: only provided by modeling tool / persistence
    - Use UUID/GUID → randomly generated, collisions unlikely

CONFLICT MANAGEMENT

- Serialization & Locking to avoid conflict
- Merging to resolve conflict

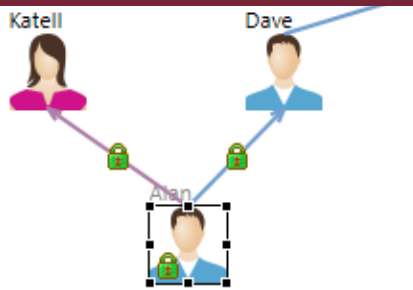
Conflict Management

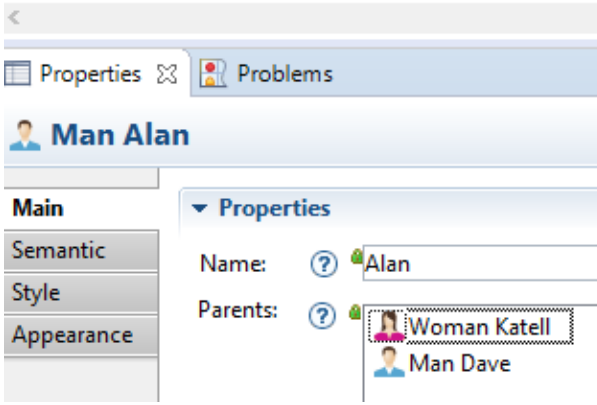
- Can we avoid conflicts?
  - Global serialization
    - Changes are sequenced
    - Online mode only
  - **Locking**
    - Temporary write ban
    - Not for security, but coordination



Locking Challenges

- Granularity (similar to Access Control)
  - File-based (inflexible) by VCS
  - Fine-grained by model-aware repos
- Lock compatibility (e.g. R/W)
- Incident/accidental changes
  - E.g. move on diagram → conflicts?
- What initiates a lock?

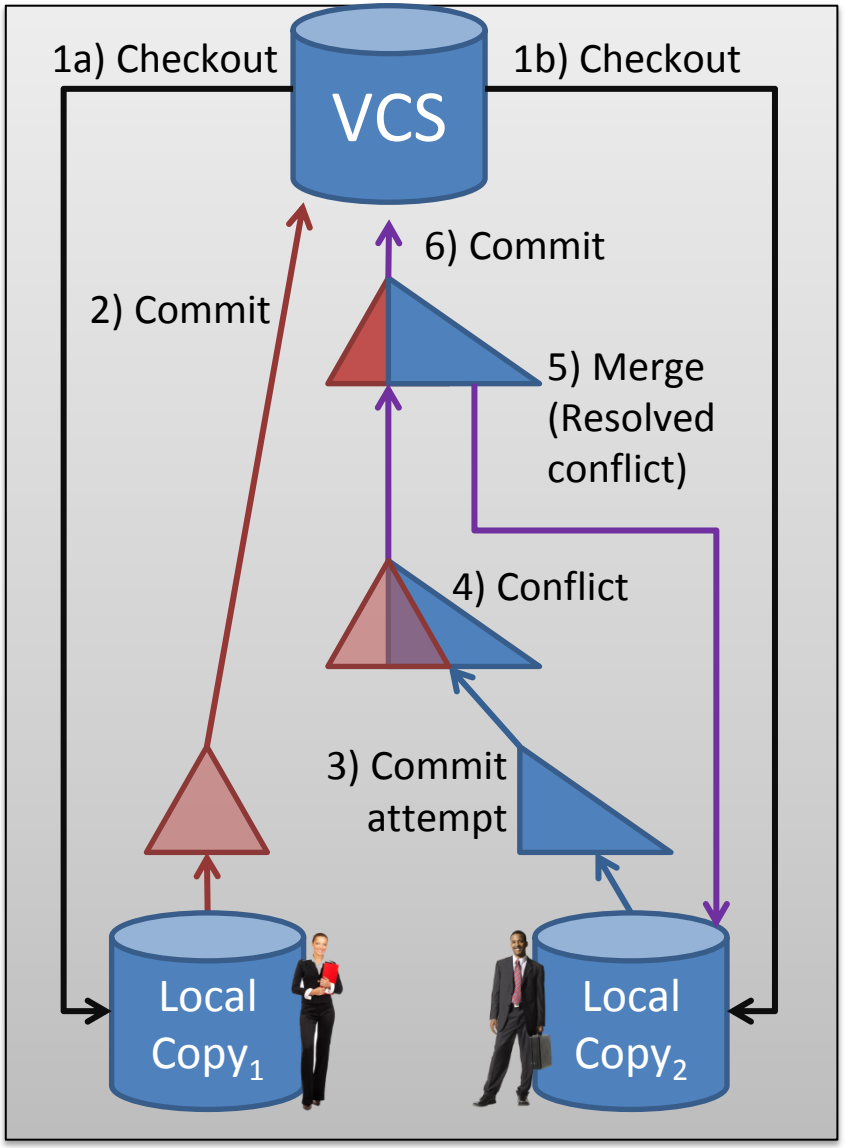




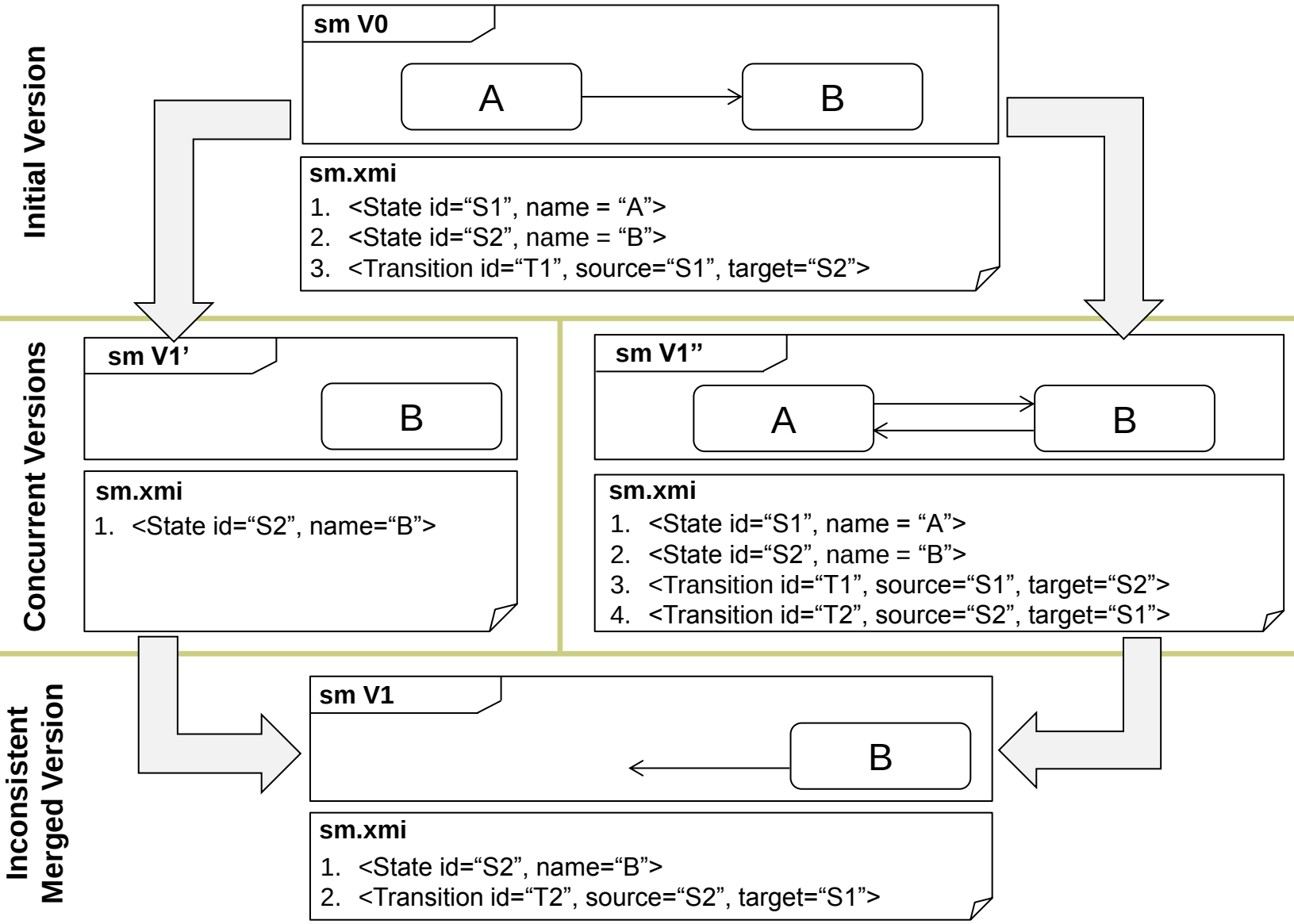
Manually initiated	View-driven locking	Property-based locking
• Explicit locks • Model regions are manually locked by users	• Derived locks • Locks are placed based on the focus of the user	• Protecting preconditions of complex refactoring • Changes violating a property are disallowed

Conflict Management

- Can we avoid conflicts?
  - Global serialization
    - Changes are sequenced
    - Online mode only
  - Locking
    - Temporary write ban
    - Not for security, but coordination
- If conflict: **merging**
  - Based on 3-way difference
  - Lot of work, error-prone



Model Versioning

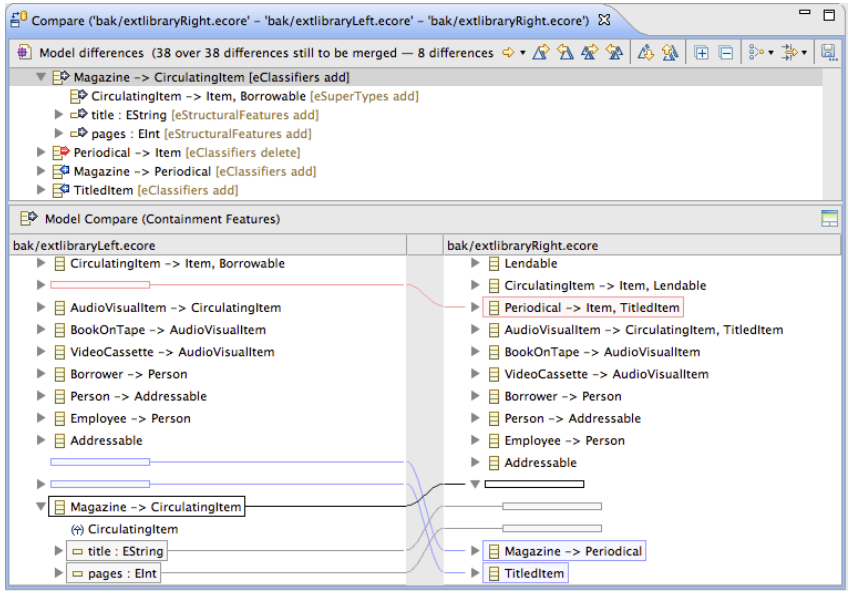


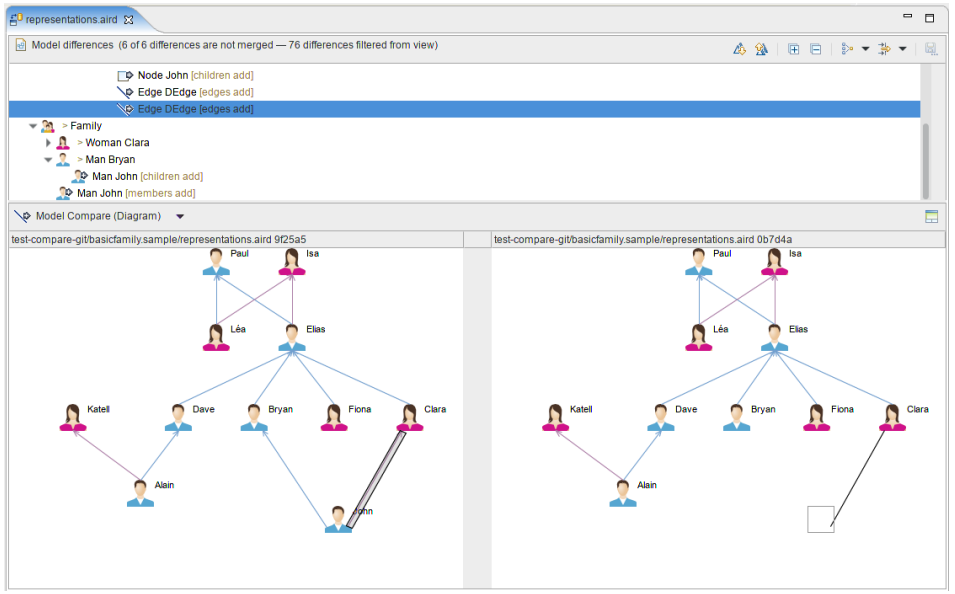
Model Merge Solutions

- File-based merging
  - Challenge: referential integrity
  - Automated: 💣
  - Manual: XMI not really human-readable ☹
  - When it works: textual concrete syntax
- Model-aware merging
  - Challenges:
    - Referential integrity 👍
    - Incidental (non-essential) changes, e.g. diagram move
    - High-level well-formedness

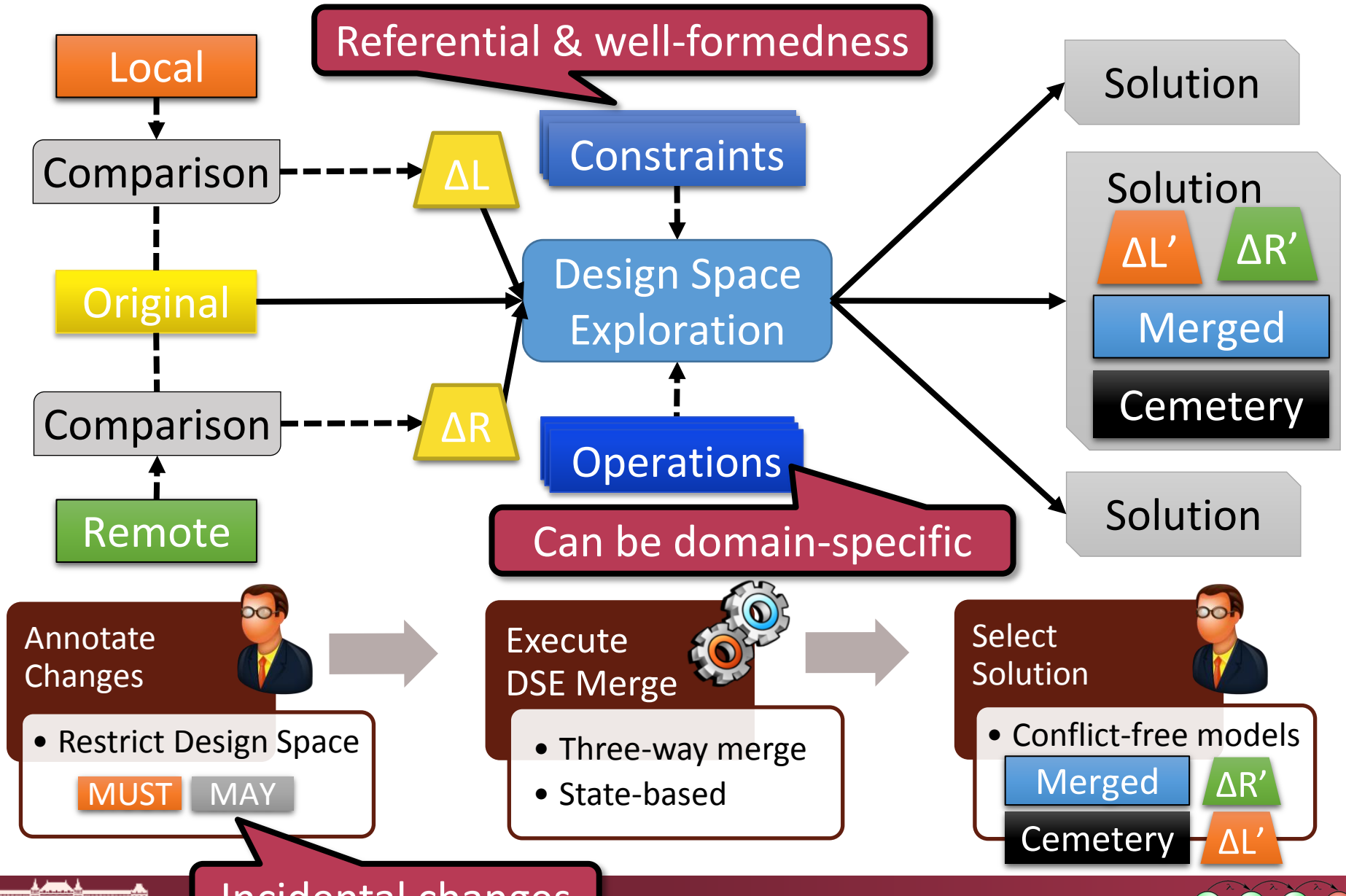
Model-aware Merging UI

- Generic Merge on Abstract Syntax
  - EMF Diff/Merge
  - EMF Compare
- Domain-specific Merge on Concrete Syntax
  - Sirius support in EMF Compare





Merging with DSE



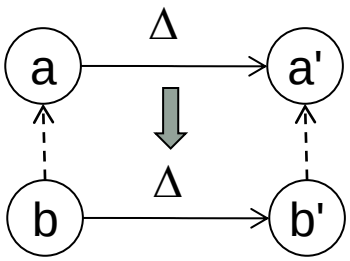
MODEL CO-EVOLUTION

Model Co-Evolution

Tools

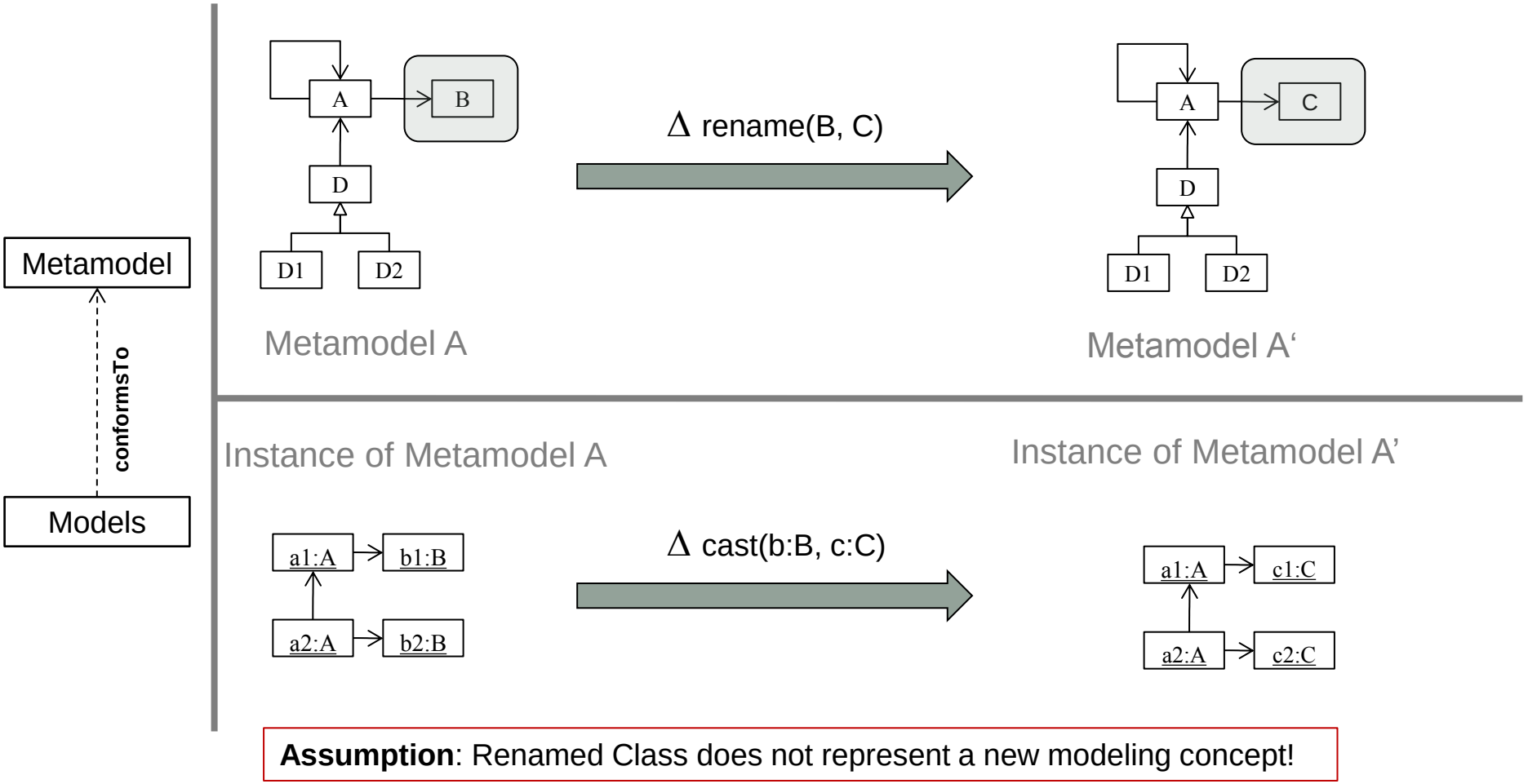
- Model versioning keeps track of the changes in a single modeling artefact but each change may affect many other related artefacts

- 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)
    - $a \rightarrow a'$
    - $b \rightarrow b' \mid r(a',b')$
    - **Challenge: Relationship Reconciliation**
  - Current research focus is on one-to-one relationships:
    - Model / Metamodel evolution
    - Metamodel / Transformation evolution
    - …



Model / Meta-model Co-evolution

Example



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

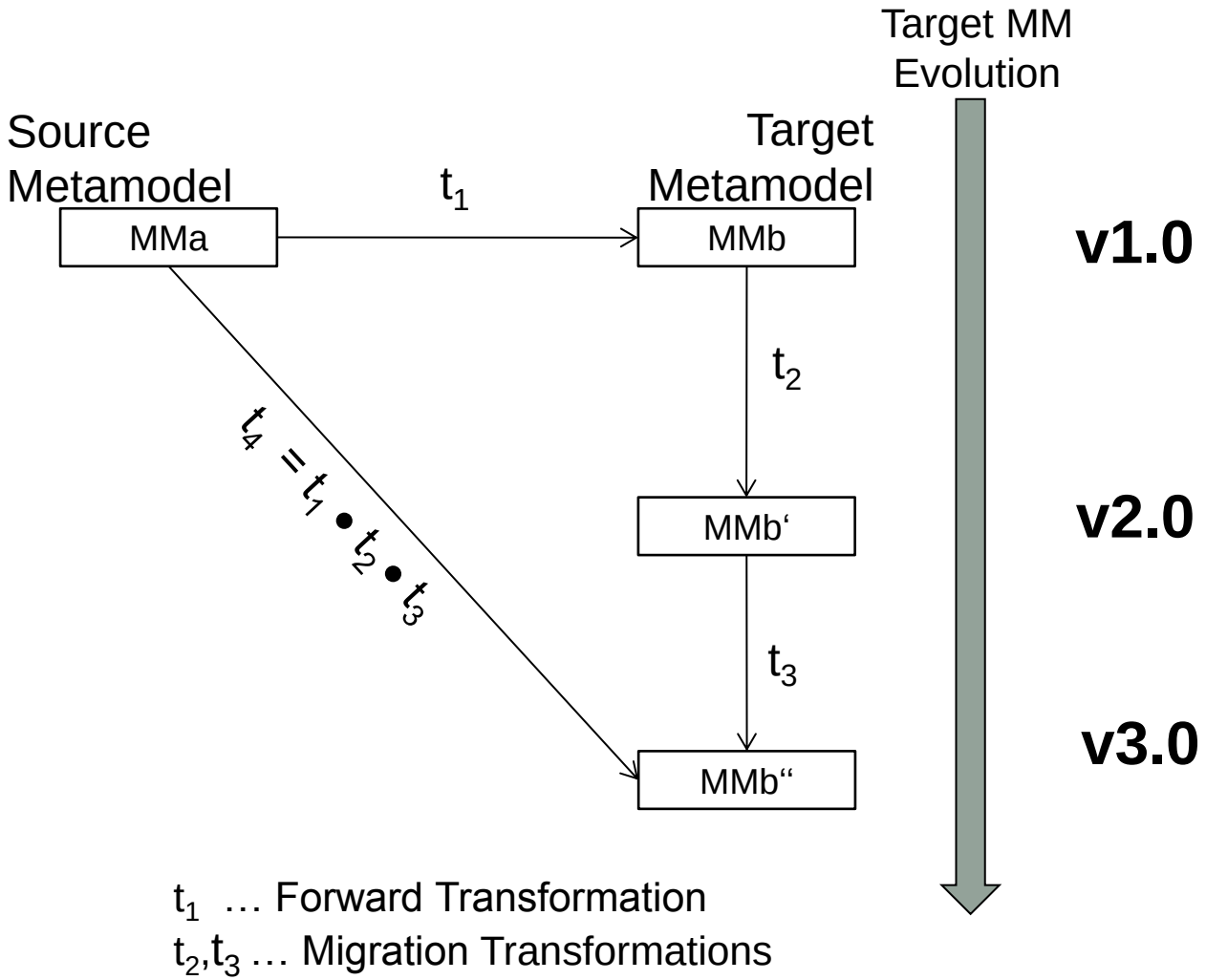
Model / Meta-model Co-Evolution

Process

- Classification of meta-model changes
  - Non-breaking operations: No need to migrate the models
  - Breaking and resolvable: Automatic migration of existing models is possible
  - Breaking and unresolvable: User intervention is necessary
- Tools like Edapt and Epsilon Flock can derive a migration transformation to adapt current models to the new metamodel structure when possible

Meta-model / Transformation co-evolution

Other co-evolution scenarios



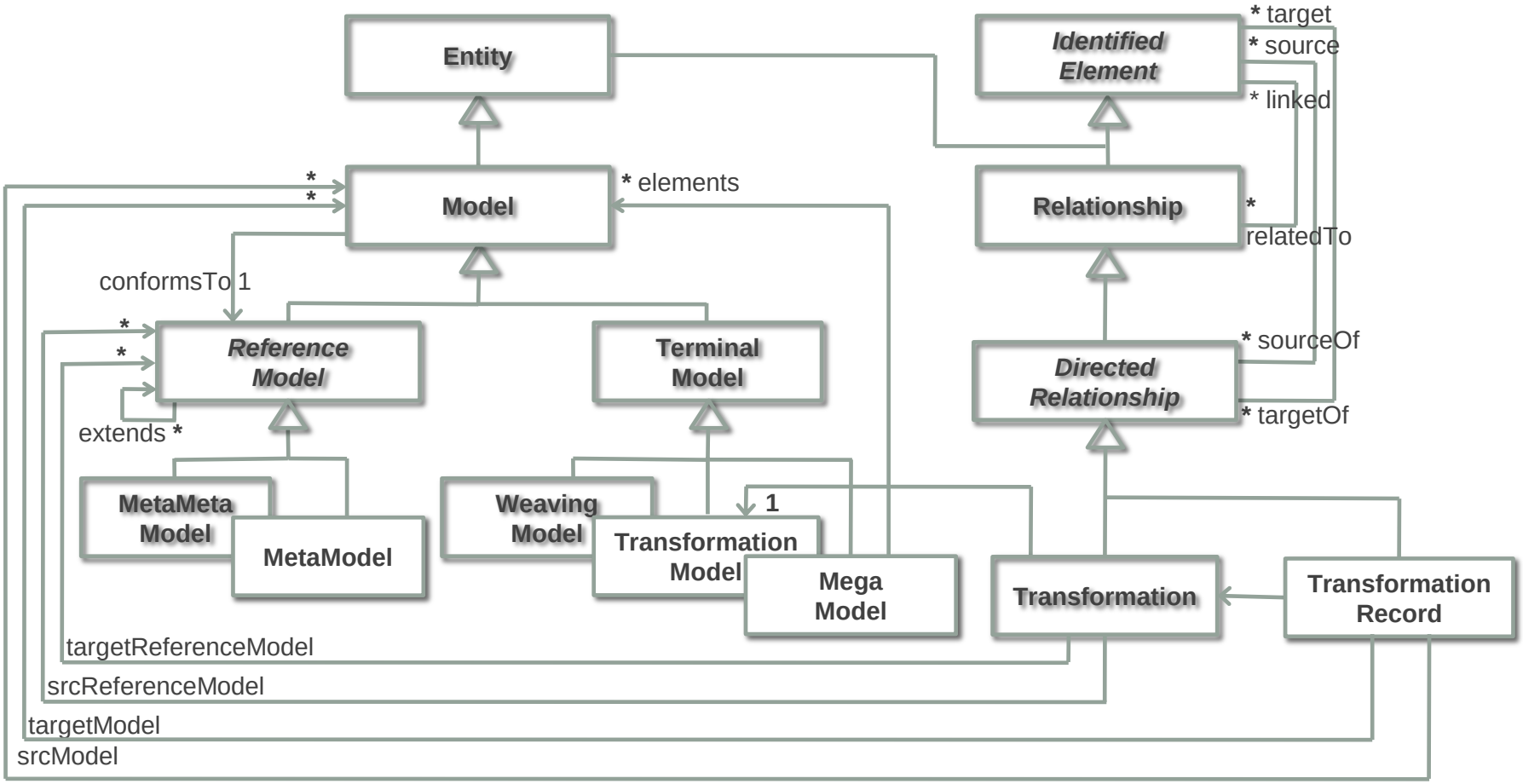
GLOBAL MODEL MANAGEMENT

Global Model Management

- Model-based solution to the problem of managing all this *model ecosystem* appearing in any MDE project
- We represent with a model, the *megamodel*, all the models (and related artefacts like configuration files) and relationships in the ecosystem
- A megamodel can be viewed as a metadata repository for the project
- A megamodel is a model whose elements are in fact other models
- As a model, a megamodel can be directly manipulated using the same tools employed to manipulate “normal” models

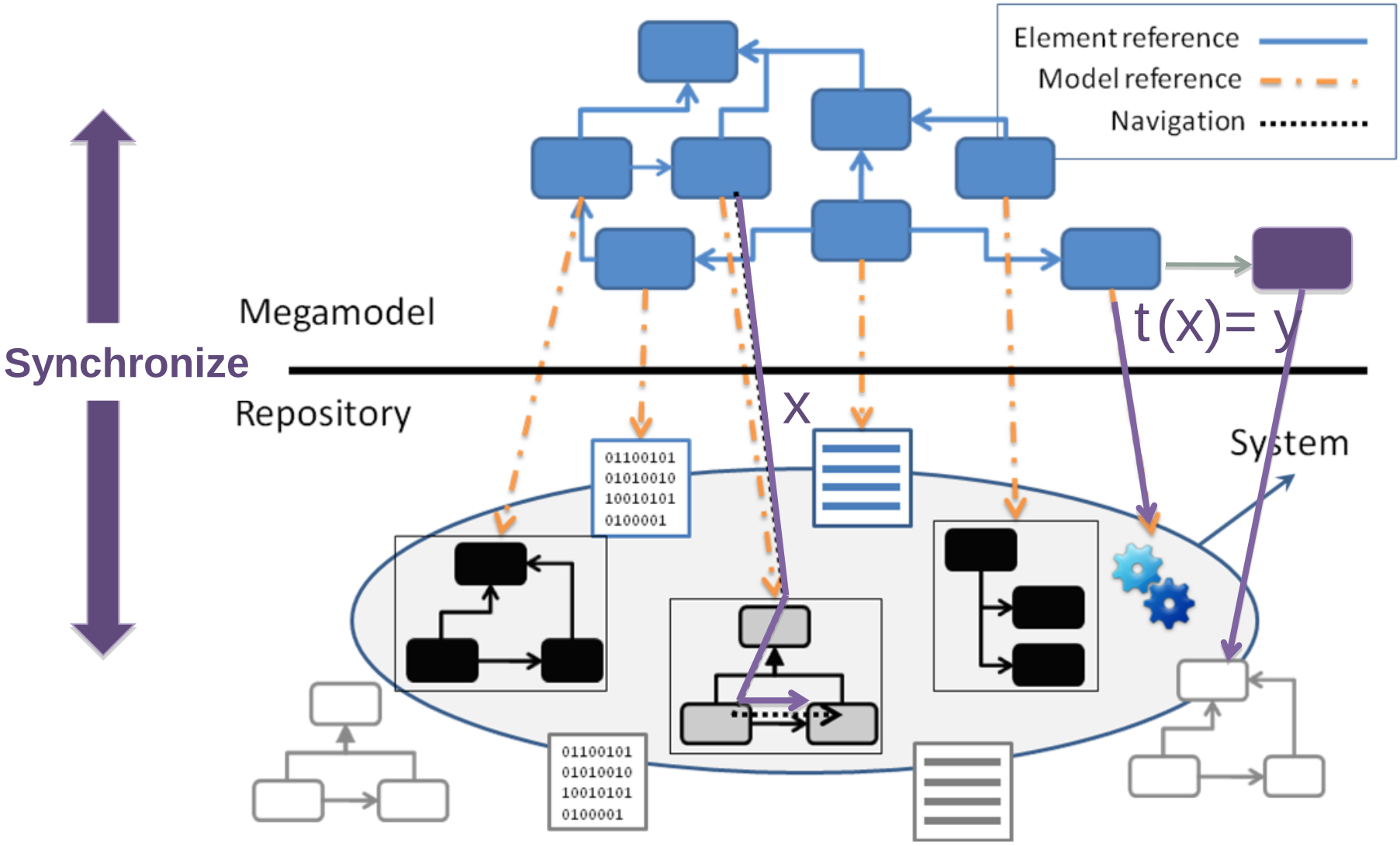
Global Model Management

The metamodel of a megamodel



Global Model Management

Using megamodels



MORGAN & CLAYPOOL PUBLISHERS

MODEL-DRIVEN SOFTWARE ENGINEERING IN PRACTICE

Marco Brambilla,
Jordi Cabot,
Manuel Wimmer.
Morgan & Claypool, USA, 2012.

www.mdse-book.com
www.morganclaypool.com
or buy it at: www.amazon.com