Modellbasierte Softwareentwicklung (MODSOFT)

Part II

Domain Specific Languages

Semantics

Prof. Joachim Fischer /
Dr. Markus Scheidgen / Dipl.-Inf. Andreas Blunk

{fischer,scheidge,blunk}@informatik.hu-berlin.de
LFE Systemanalyse, III.310

Agenda

prolog (1 VL) — **Introduction:** languages and their aspects, modeling vs. programming, meta-modeling and the 4 layer model

0. (2 VL) — **Eclipse/Plug-ins:** eclipse, plug-in model and plug-in description, features, *p2*-repositories, *RCP*s

1. (2 VL) — **Structure:** *Ecore*, *genmodel*, working with generated code, constraints with *Java* and *OCL*, *XML/XMI*

2. (3 VL) — **Notation:** Customizing the tree-editor, textural with *XText*, graphical with *GEF* and *GMF*

3. (4 VL) — **Semantics:** interpreters with Java, code-generation with *Java* and *XTend*, model-transformations with *Java* and *ATL*

➡ epilog (2 VL) — **Tools:** persisting large models, model versioning and comparison, model evolution and co-adaption, modular languages with *XBase*, *Meta Programming System* (MPS)

Agenda

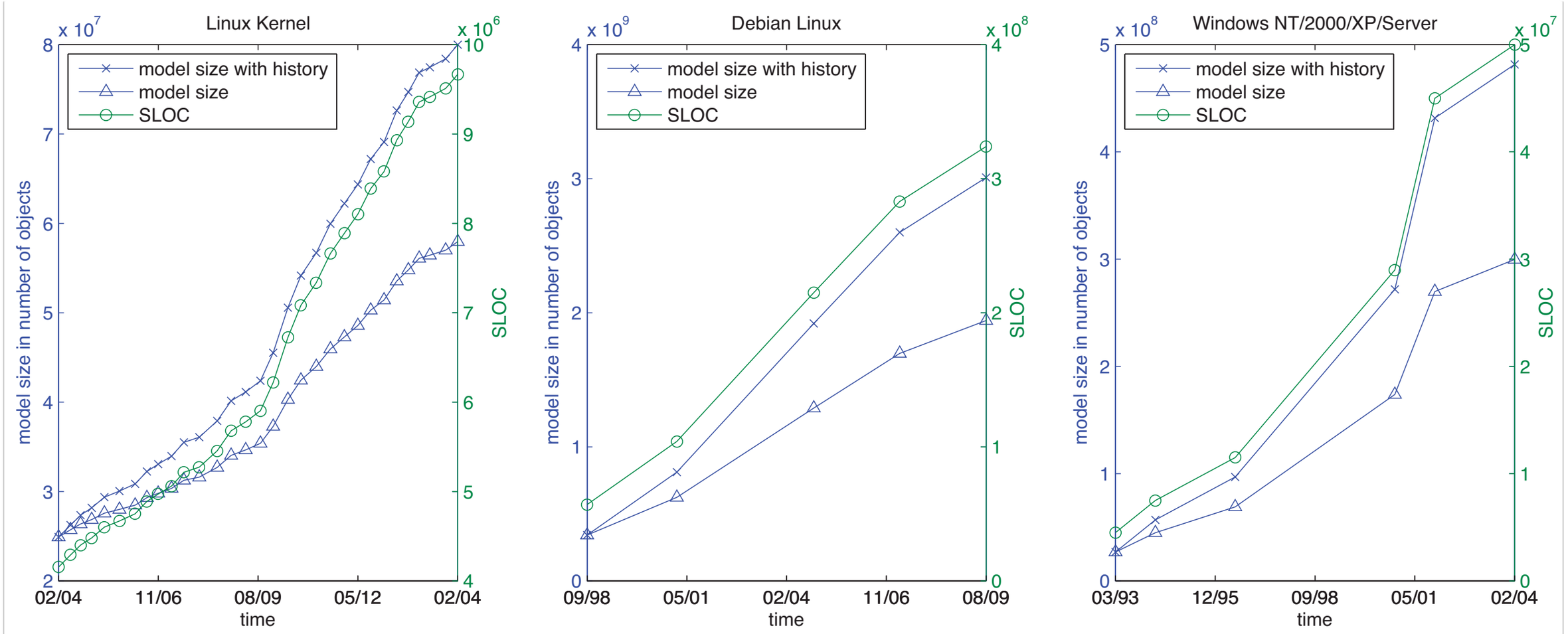
- Persistence of large models (e.g. CDO)
- Mode comparison (EMF Compare)
- Eclipse Client Platform (ECP)
- Version control for models (EMF Store)
- Course summary

Persistence of Large Models

EMF and Databases

Large?

- the size of typical software models is small enough
- versions
- industry scale model-based software projects
  - e.g. factory lines in automotive industry, more variants than sold cars
- reverse engineering of large code bases
- non software-model EMF-models
  - sensor data (usually XML and column/row databases)
  - geo-spacial models (usually XML or relational-databases + geo-spacial indices)



- reverse engineering of large code bases
- non software-model EMF-models
  - sensor data (usually XML and column/row databases)
  - geo-spacial models (usually XML or relational-databases + geo-spacial indices)

Large?

- How to measure model size?
  - number of objects: not all objects have equal size
  - memory: serialized? heap vs. XMI/XML vs. binary? Compression?
  - different heap representations in EMF
    - each feature as a field
    - all features in a dynamic array
  - syntax vs. syntax
    - representations of the same model in different syntaxes can yield different memory requirements
    - e.g. serialized AST's ~ 400*code
  - syntax vs. semantic
    - space needed to represent model in a certain syntax != minimum space needed to express its semantic

too Large?

- Processing of models requires more main memory than available
- Processing of models requires more time than available/ sensible
- Depends on the actual processing task
  - queries
  - editing
  - execution/transformation

Why does EMF not scale?

- Most tools are build under the assumption that models are relatively small
- Models have to be loaded as a whole or have to be spread over multiple resources manually
- EMF's resource unload, does not really unload the contained objects
  - they are logically removed from the resource-set
  - references between objects are not broken, the objects cannot be collected by JVM's GC

Technological Spaces

- Object-oriented meta-models, e.g. EMF, MOF
- formal languages, e.g. context-free grammars
- XML
- databases
  - relational databases
  - NoSQL databases
    - graph databases
    - document, column, key-value databases

Technological Spaces

- Different technological spaces focus on different things
- OO MM: presenting human readable and editable software models
- XML: serialization and interchange of data
- Databases: scalability, safe storage, and optimized queries

Solution – Mapping to other Technological Spaces

- Textual representations: we already mapped EMF to a different technological space: formal languages, i.e. context-free grammars
- Mapping to databases
  - different mappings for different database technologies
  - different database technologies have different properties, correct choice depends on use-case
  - mappings can be more or less natural

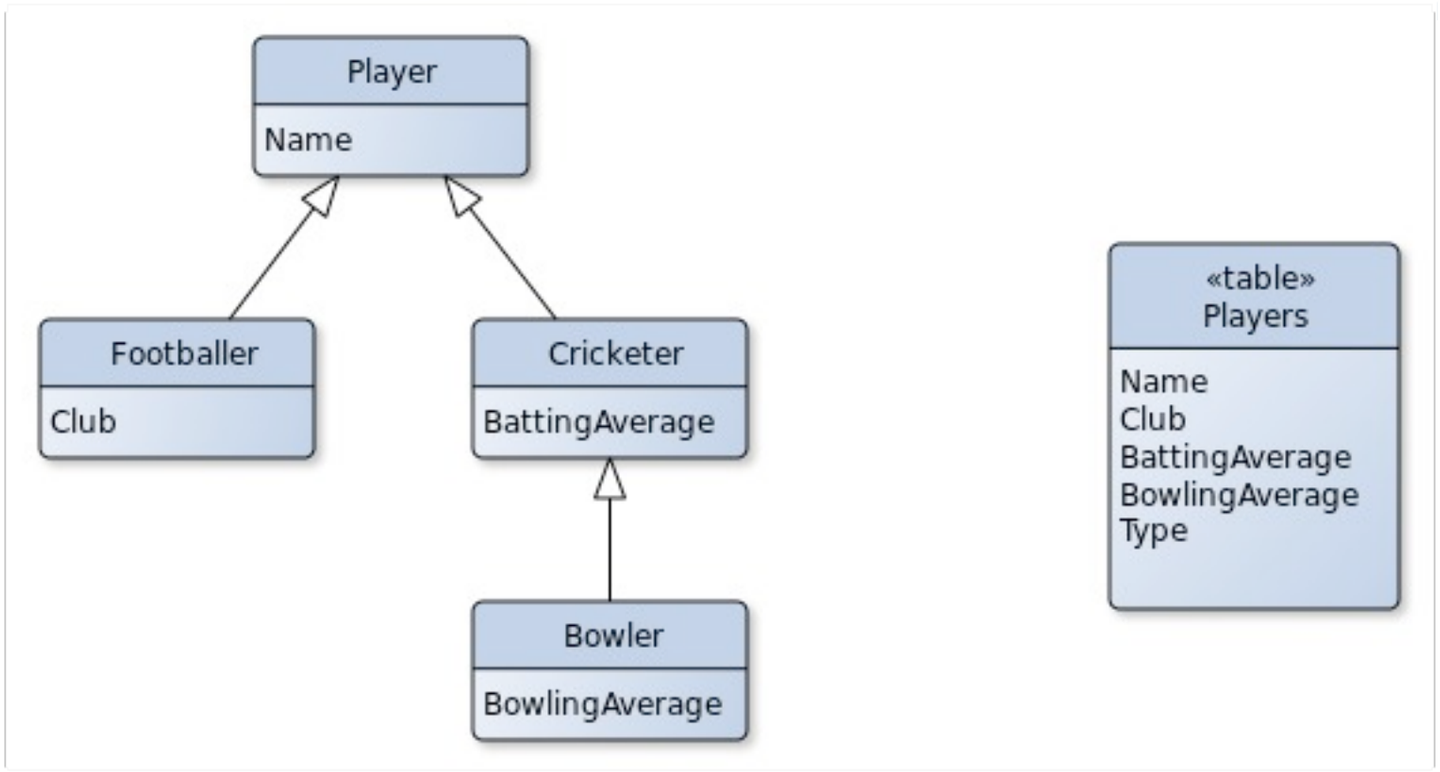
Databases – Relational vs. NoSQL

- Atomicity, Consistency, Isolation, Durability (ACID)
  - reliable database-wide transactions
- Basically Available, Soft state, Eventual consistent (BASE)
  - distributability and availability is more important than immediate consistency
- Consistency, Availability, Partition tolerance (CAP)-theorem, a.k.a Brewer's theorem
  - "only *two* of *three* possible"
  - ACID: not partition tolerant
  - BASE: not consistent

EMF and Relational Databases

- Object-oriented data-models vs. relations/tables
- Object Relational Mappings (ORM)
  - each class gets its own table
  - each attribute its own column
  - an object is represented as row in the corresponding table
  - different solutions for inheritance hierarchies
    - one table per hierarchy
    - one table per sub class
    - one table per concrete class
  - one table per reference

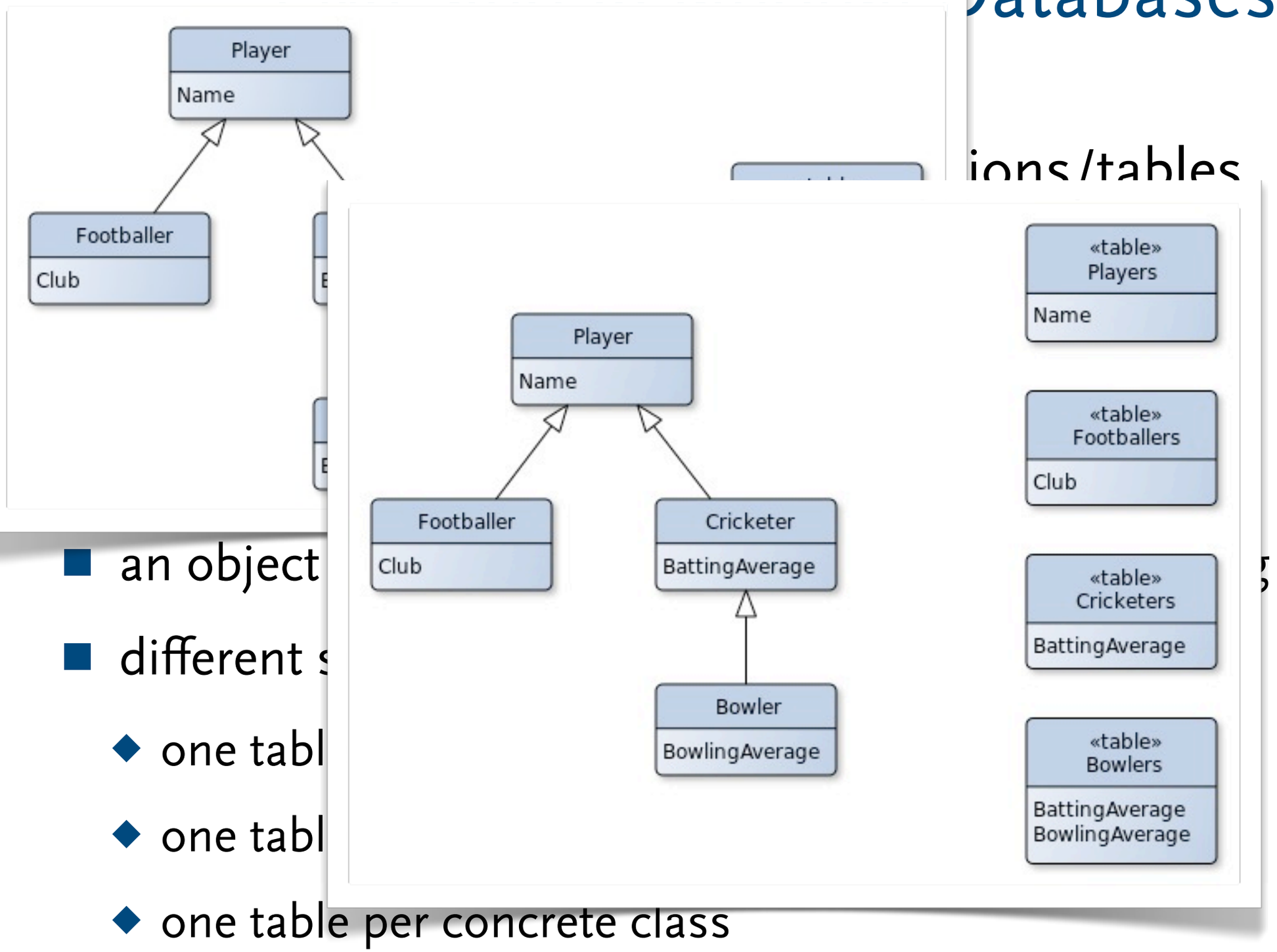
EMF and Relational Databases



ions/tables

- an object is represented as row in the corresponding table
- different solutions for inheritance hierarchies
  - one table per hierarchy
  - one table per sub class
  - one table per concrete class
- one table per reference

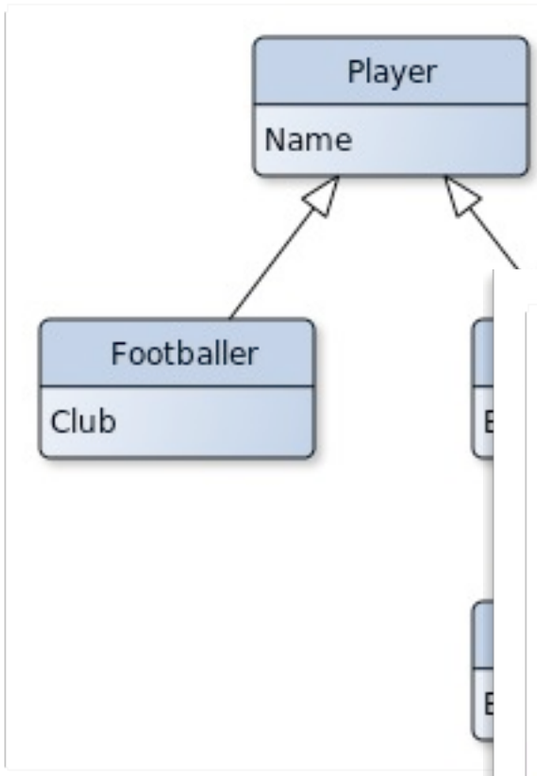
EMF and Relational Databases

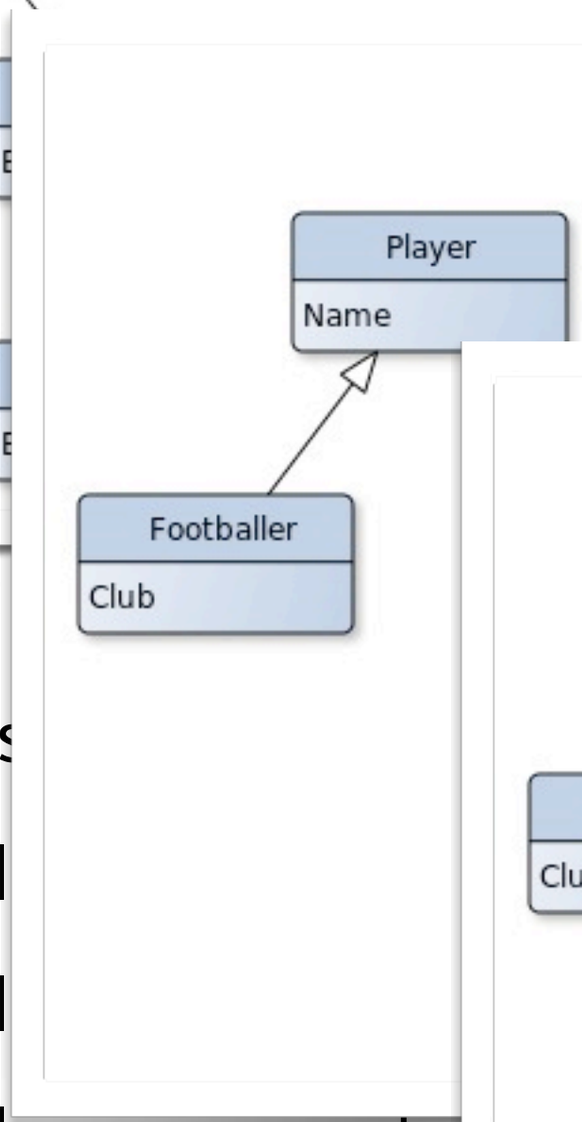


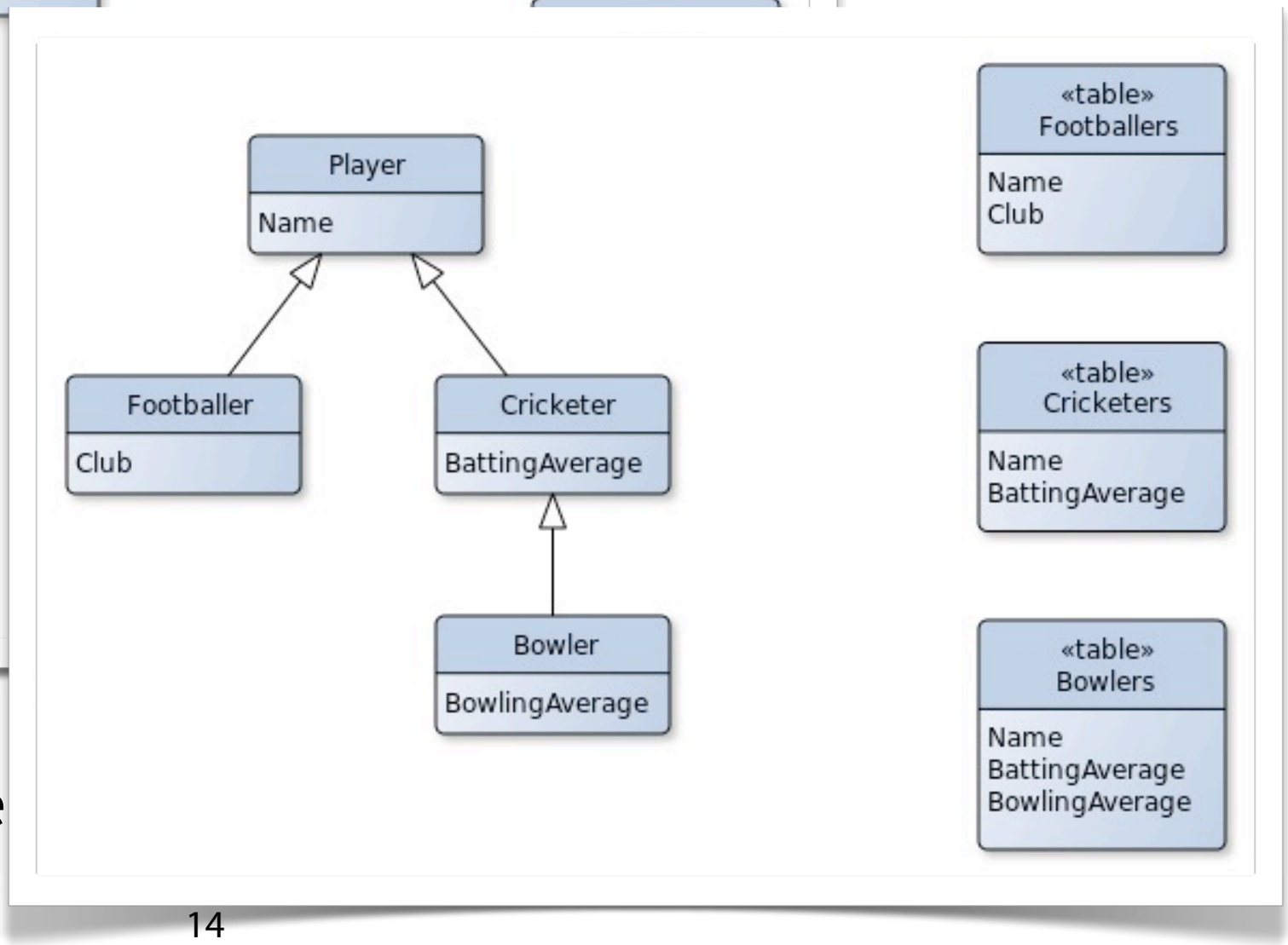
- an object
- different s
  - one tabl
  - one tabl
  - one table per concrete class
- one table per reference

EMF and Relational Databases

- an object
- different s
  - one tabl
  - one tabl
  - one table per concrete
- one table per reference

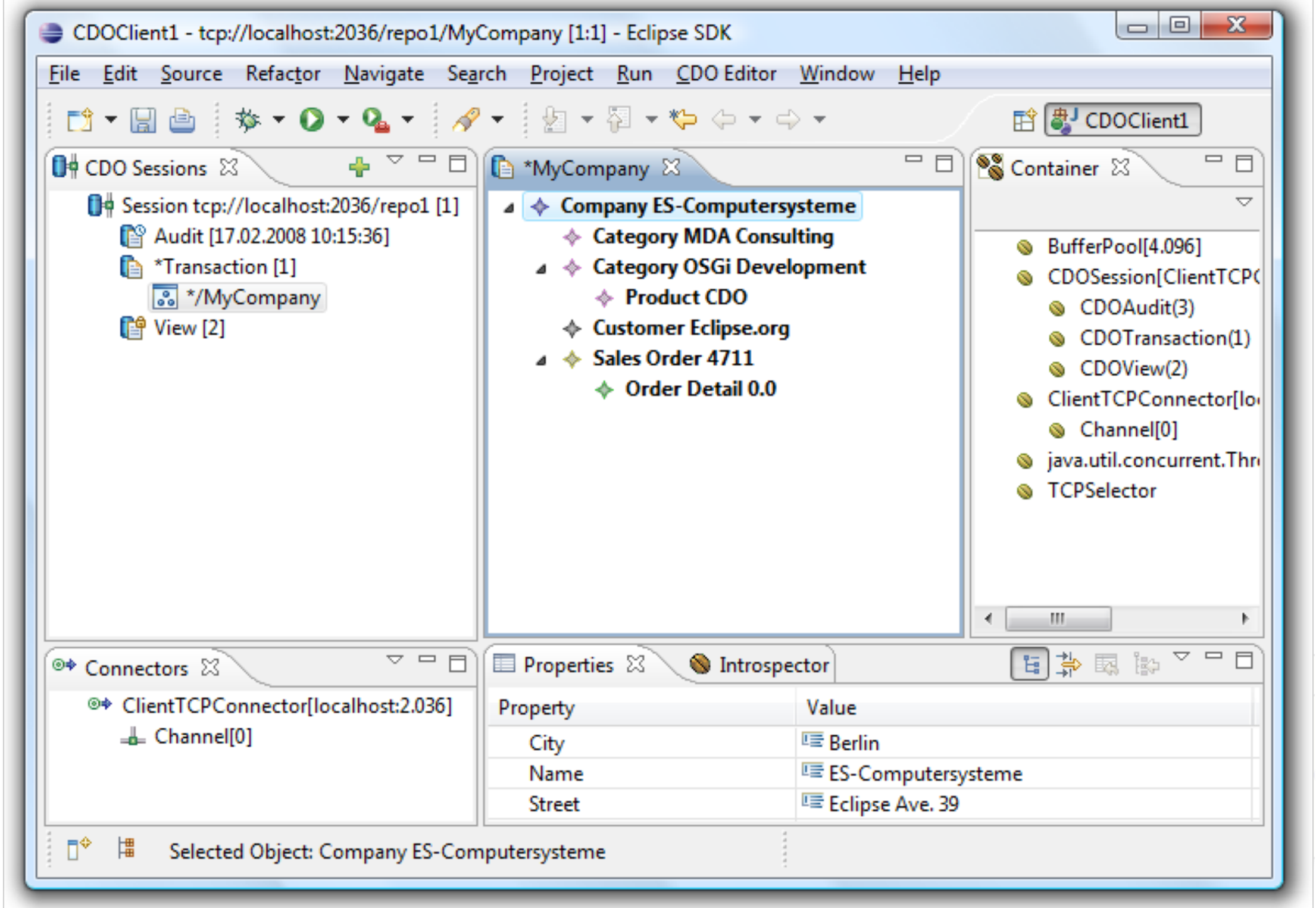






Concrete ORM's and Frameworks

- hibernate, indirect
  - general ORM for Java
  - mapping of generated Java classes
- Connected Data Objects (CDO), direct
  - dedicated EMF-framework
  - special EObject implementation
  - EStore-based
  - Client/server architecture


CDOClient1 - tcp://localhost:2036/repo1/MyCompany [1:1] - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run CDO Editor Window Help
CDOClient1
CDO Sessions
Session tcp://localhost:2036/repo1 [1]
Audit [17.02.2008 10:15:36]
*Transaction [1]
*/MyCompany
View [2]
*MyCompany
Company ES-Computersysteme
Category MDA Consulting
Category OSGi Development
Product CDO
Customer Eclipse.org
Sales Order 4711
Order Detail 0.0
Container
BufferPool[4.096]
CDOSession[ClientTCP(
CDOAudit(3)
CDOTransaction(1)
CDOView(2)
ClientTCPConnector[lo
Channel[0]
java.util.concurrent.Thr
TCPSelector
Connectors
ClientTCPConnector[localhost:2.036]
Channel[0]
Properties
Introspector
Property
Value
City
Berlin
Name
ES-Computersysteme
Street
Eclipse Ave. 39
Selected Object: Company ES-Computersysteme

Connected Data Objects (CDO)

Client

- Client adds/modifies CDOObjects
- Client transaction creates temporary IDs for new objects and records change deltas
- Commit() sends new packages, new revisions and revision deltas to the server

Server

- Server passes data to the configured store
- Store remaps temporary IDs and persists the data
- Server sends back ID mappings
- Server notifies other sessions about invalidations

Client

- Client transaction applies ID mappings

Connected Data Objects (CDO)

```
// Open an embedded connection
IConnector connector = JVMUtil.getConnector(container, "default");

// Open a session and register the model
CDOSession session = CDOUtil.openSession(connector, “repo", true);
session.getPackageRegistry().putEPackage(Model1Package.eINSTANCE);

// Start a transaction and create a resource
CDOTransaction transaction = session.openTransaction();
Resource resource = transaction.createResource("/my/big/resource");

// Work normally with the EMF resource
resource.getContents().add(getInputModel());
transaction.commit();

// Cleanup
session.close();
connector.disconnect();
```

ORM – Disadvantages

- *Object-relational Impedance Mismatch*
- fast queries, but depend on SQL on mapped data
- slow traversal/navigation

EMF and Document/Column/Key-Value Databases

- Two strategies
- *one entry per object*
  - object to database entry
  - entry keys as IDs for references
  - each object is serialized in a database-friendly format, e.g. JSON for MongoDB
  - natural index for entry keys
  - secondary indexes for other attributes
- *fragmentation*
  - automated distribution of model object over many resources
  - resources are serialized into database entries, URI as key
  - natural index for URIs
  - no secondary indexes

Disadvantages

- scalability issues with very large value-sets
- limited indices and query capabilities
- fast traversal/navigation

EMF and Graph-Databases

- *index-free adjacency*, constant execution time navigation from one node to another, no index involved
- one-to-one mapping
  - objects to nodes
  - references to edges
  - attributes as node attributes (supported by most graph databases)

Disadvantages

- no indices besides the model itself (and proprietary database query capabilities)
- Very fast traversal via index-free adjacency
- simple mapping
- in reality graph-databases do not actually allow constant time navigation

Summary

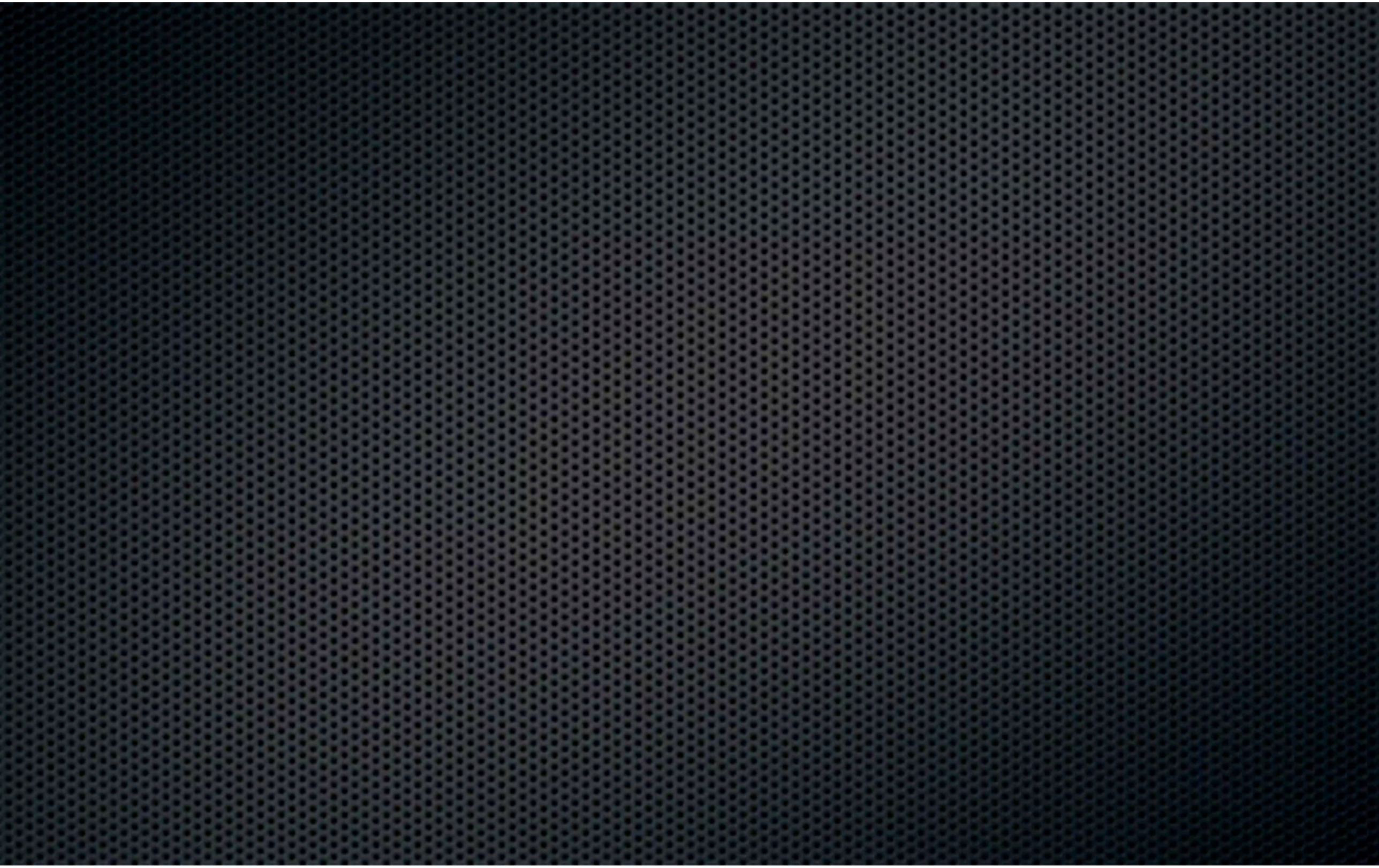
- EMF models can be too large
- Solution mapping to other technological spaces
- Mappings for different database technologies exist

Model Comparison and Merging

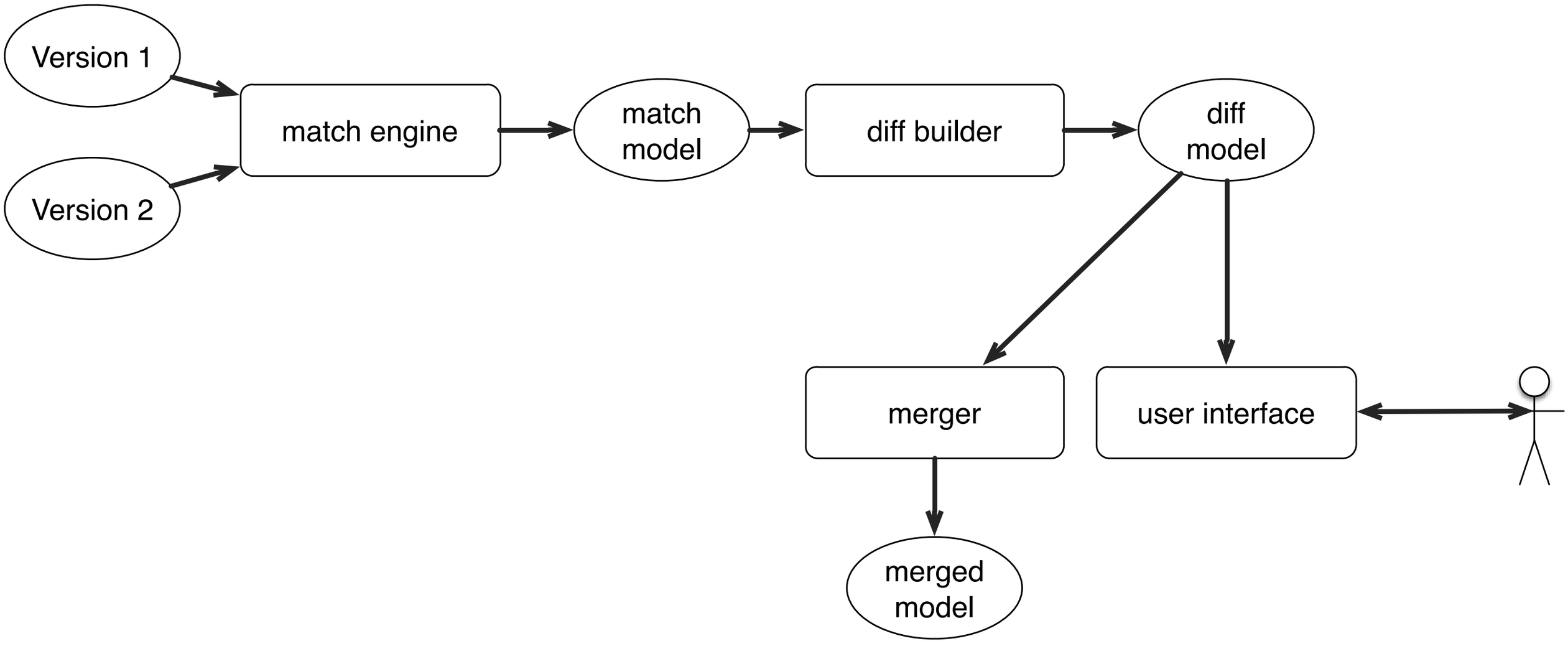
EMF Compare

EMF Compare

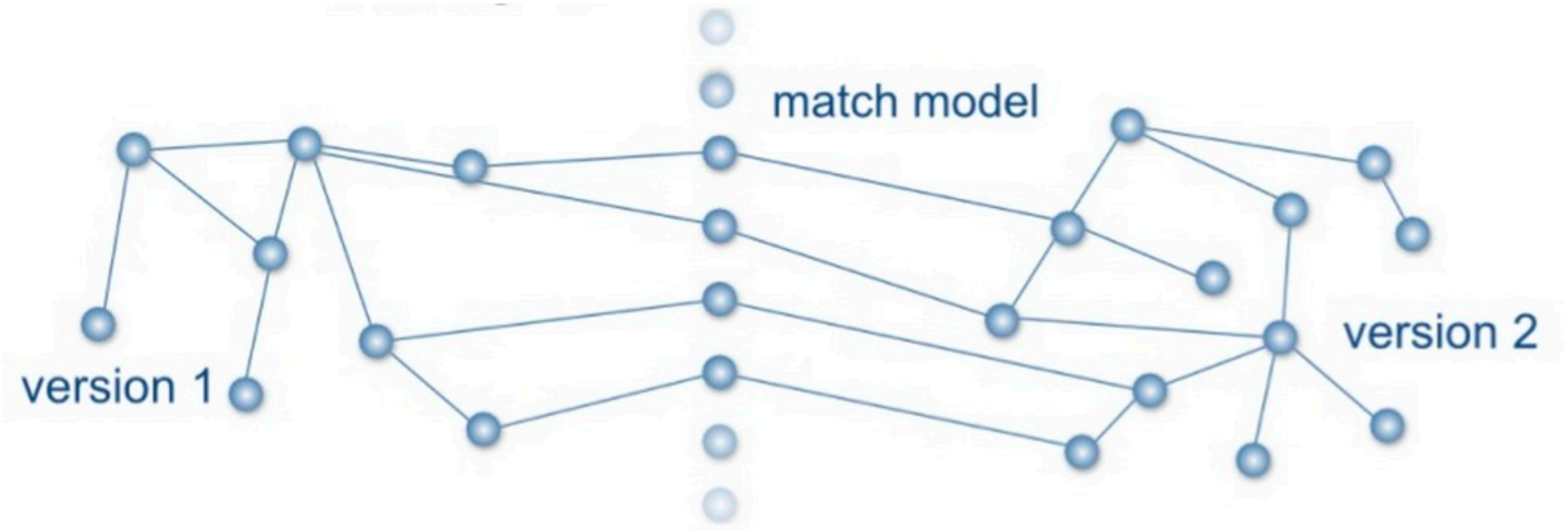
- API and UI
- Allows you to compare two (or more) models
  - generate matches
  - generate differences
  - compare differences similar to textual diff (e.g. in SVN, GIT clients)
    - based on model-elements not based on LOC
  - merge models interactively



EMF Compare – Process



Match Model



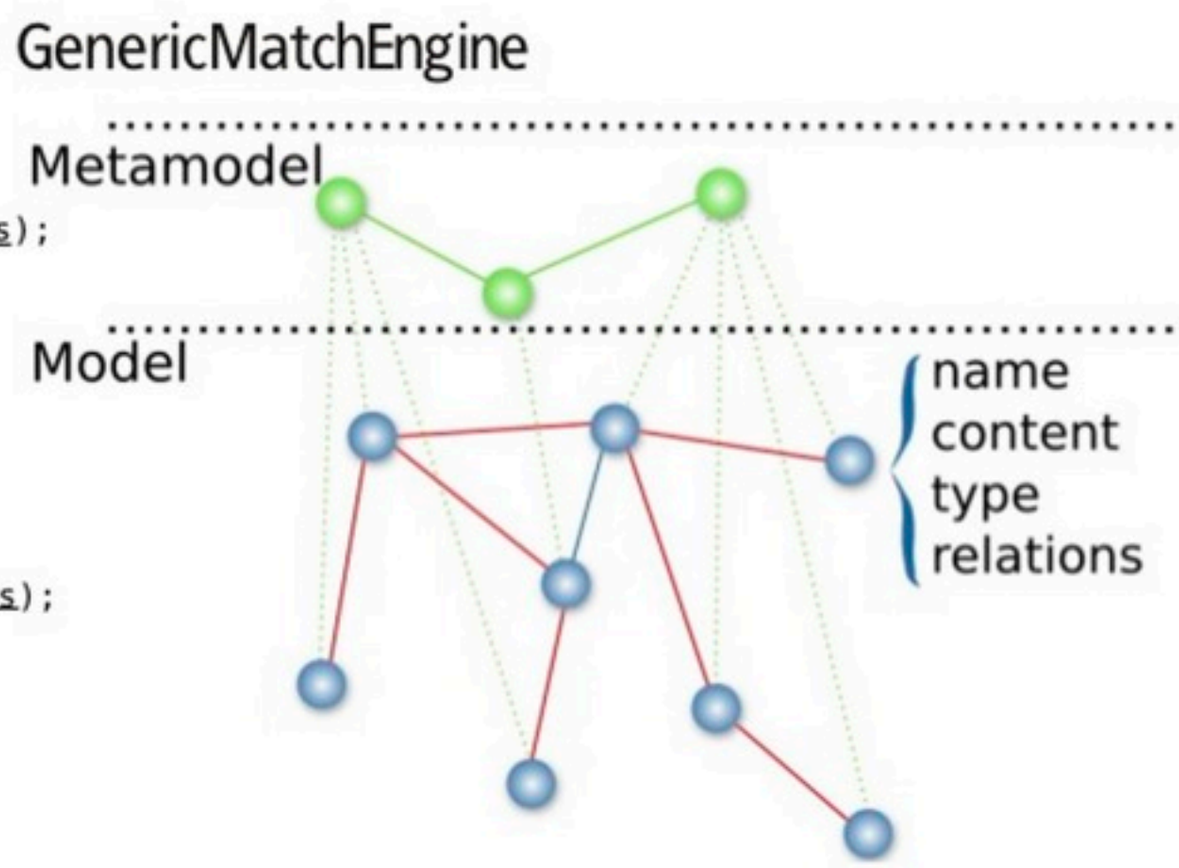
Generic Match Engine

XMI-ID

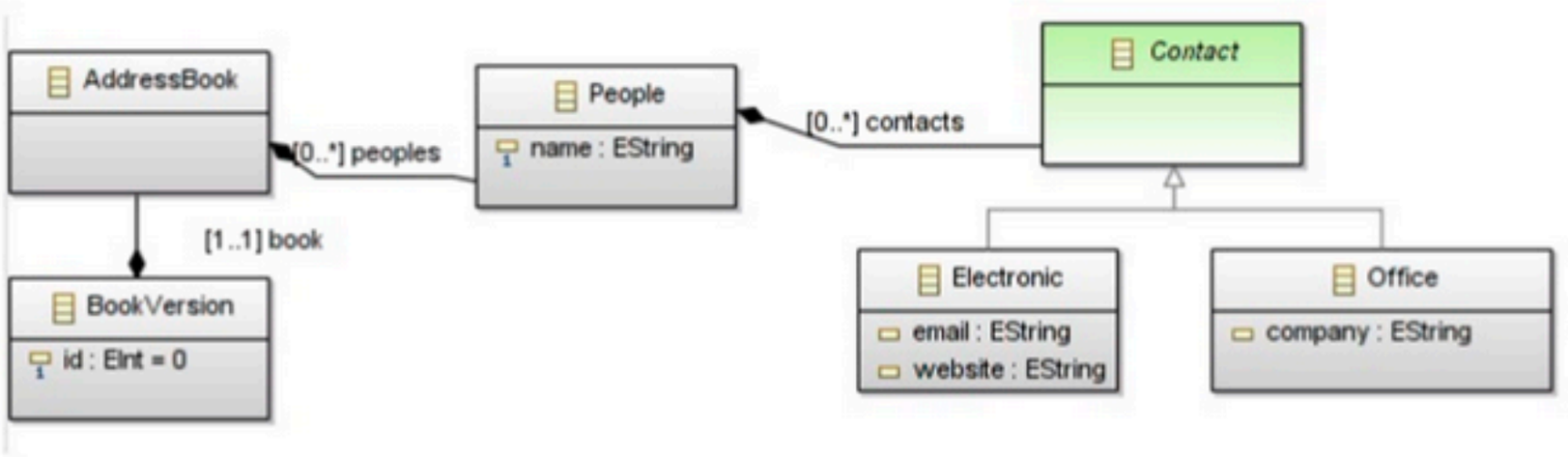
```
options.put(MatchOptions.OPTION_IGNORE_XMI_ID, Boolean.FALSE);
MatchModel match = MatchService.doContentMatch(left,right options);
```

ID-Attribute

```
options.put(MatchOptions.OPTION_IGNORE_ID, Boolean.FALSE);
MatchModel match = MatchService.doContentMatch(left,right options);
```



Custom Match Engines



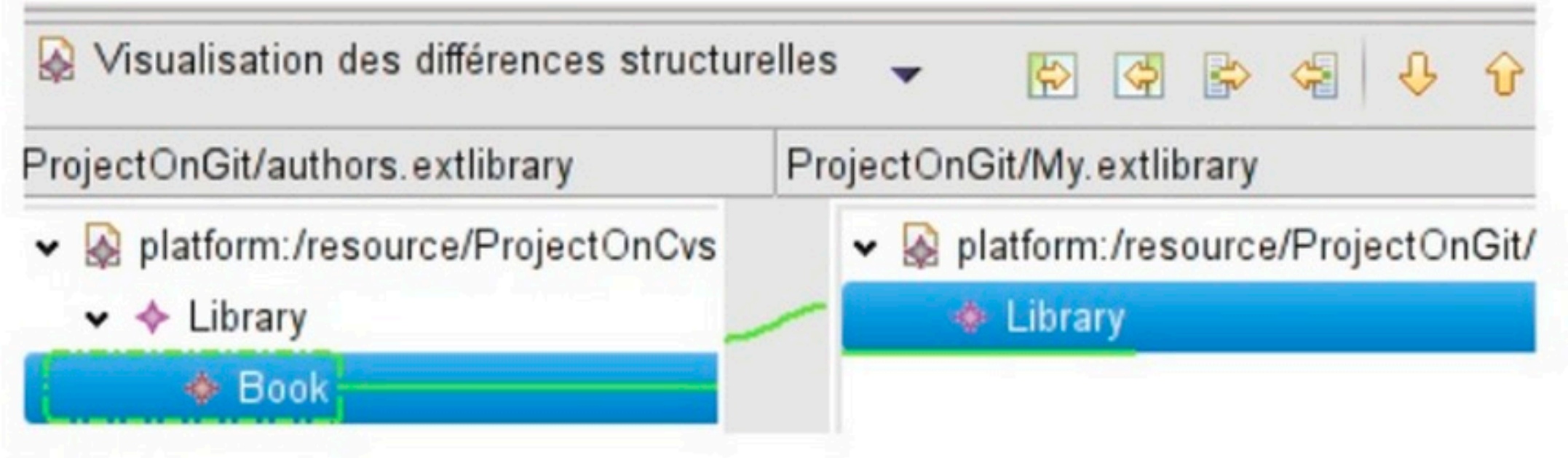

```
public class AddressBookMatcher extends GenericMatchEngine {

/**
 * {@inheritDoc}
 */
@Override
protected boolean isSimilar(EObject obj1, EObject obj2) throws FactoryException {
/*
 * If we've got a People, only check the name similarity.
 */
if (obj1 instanceof People || obj2 instanceof People)
return nameDistance(obj1, obj2) > 0.8;
/*
 * Contacts are similar if : the associated people is similar + their content is quite the same.
 */
if (obj1 instanceof Contact && obj2 instanceof Contact) {
EObject obj1Parent = obj1.eContainer();
EObject obj2Parent = obj2.eContainer();
if (obj1Parent instanceof People && obj2Parent instanceof People)
return isSimilar(obj1Parent, obj2Parent) && contentDistance(obj1, obj2) > 0.5;
}

/*
 * If it's something we don't know about, then use the generic behavior.
 */
return super.isSimilar(obj1, obj2);
}
```

Merging UI


1 change(s) in model
1 change(s) in Library
Book has been added
Removed dependency on resource authors.extlibrary


Visualisation des différences structurelles
ProjectOnGit/authors.extlibrary
ProjectOnGit/My.extlibrary
platform:/resource/ProjectOnCvs
Library
Book
platform:/resource/ProjectOnGit/
Library

Eclipse Client Platform (ECP)

GUI applications based on EMF data

Eclipse Client Platform (ECP)

- One-click build GUI application based on EMF-data models
- generates editors and forms based on a meta-model
- Suited for simple data entry and manipulation applications
- Works well with persistence backends

Only one Click to an EMF Application

EMF Client Platform

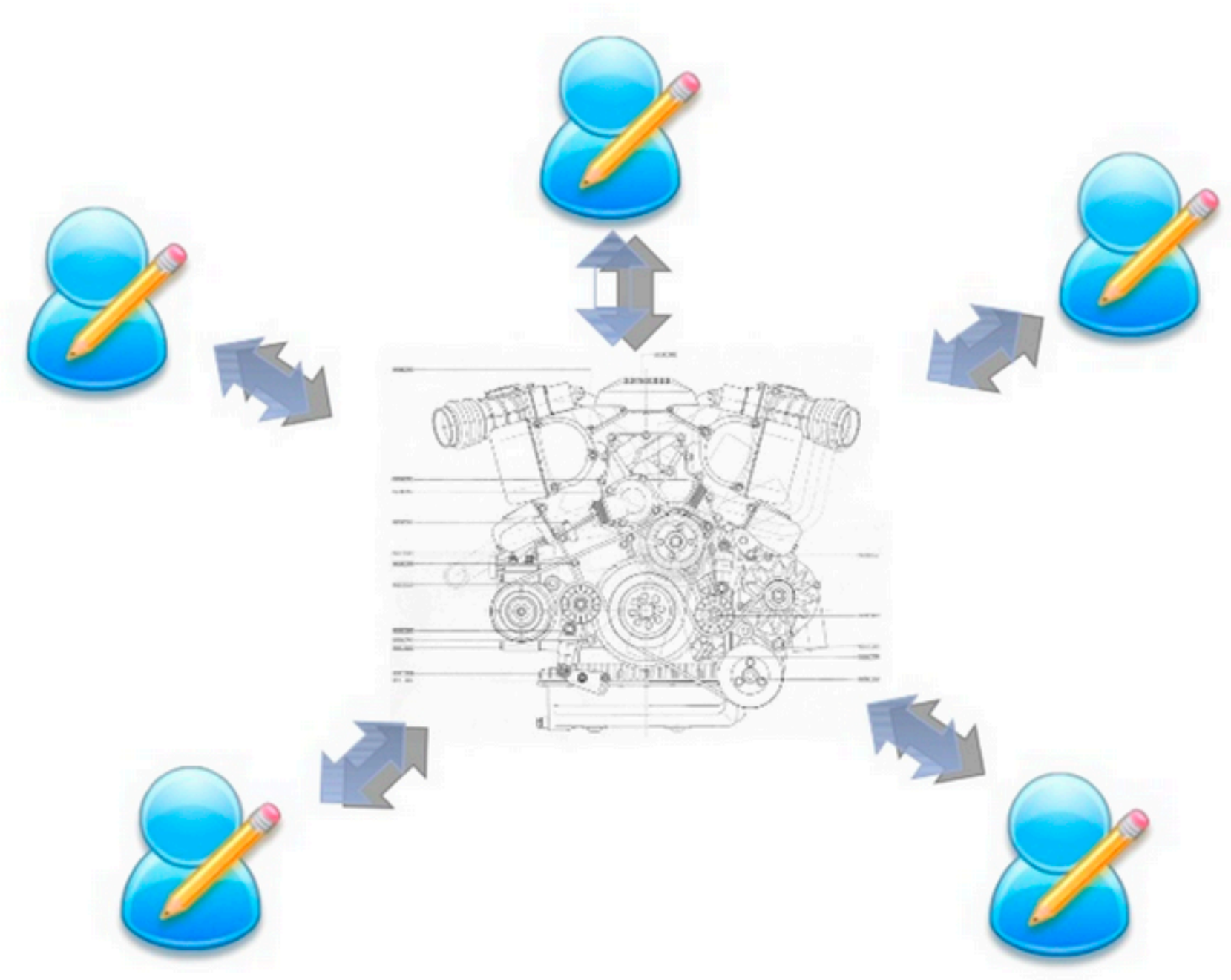
Jonas Helming, Maximilian Koegel
{helming, koegel}@in.tum.de

Chair for Applied Software Engineering
Institut für Informatik
Technische Universität München

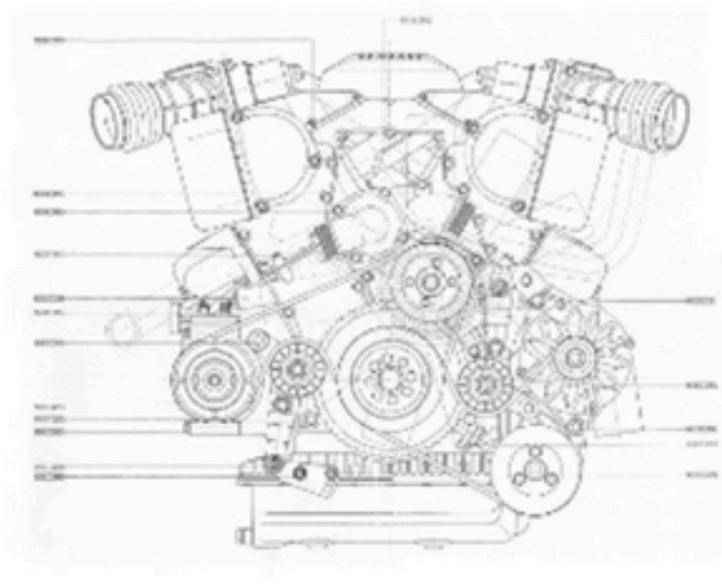
EMF Store

Version control for models based on CDO, ECP, and EMF Compare

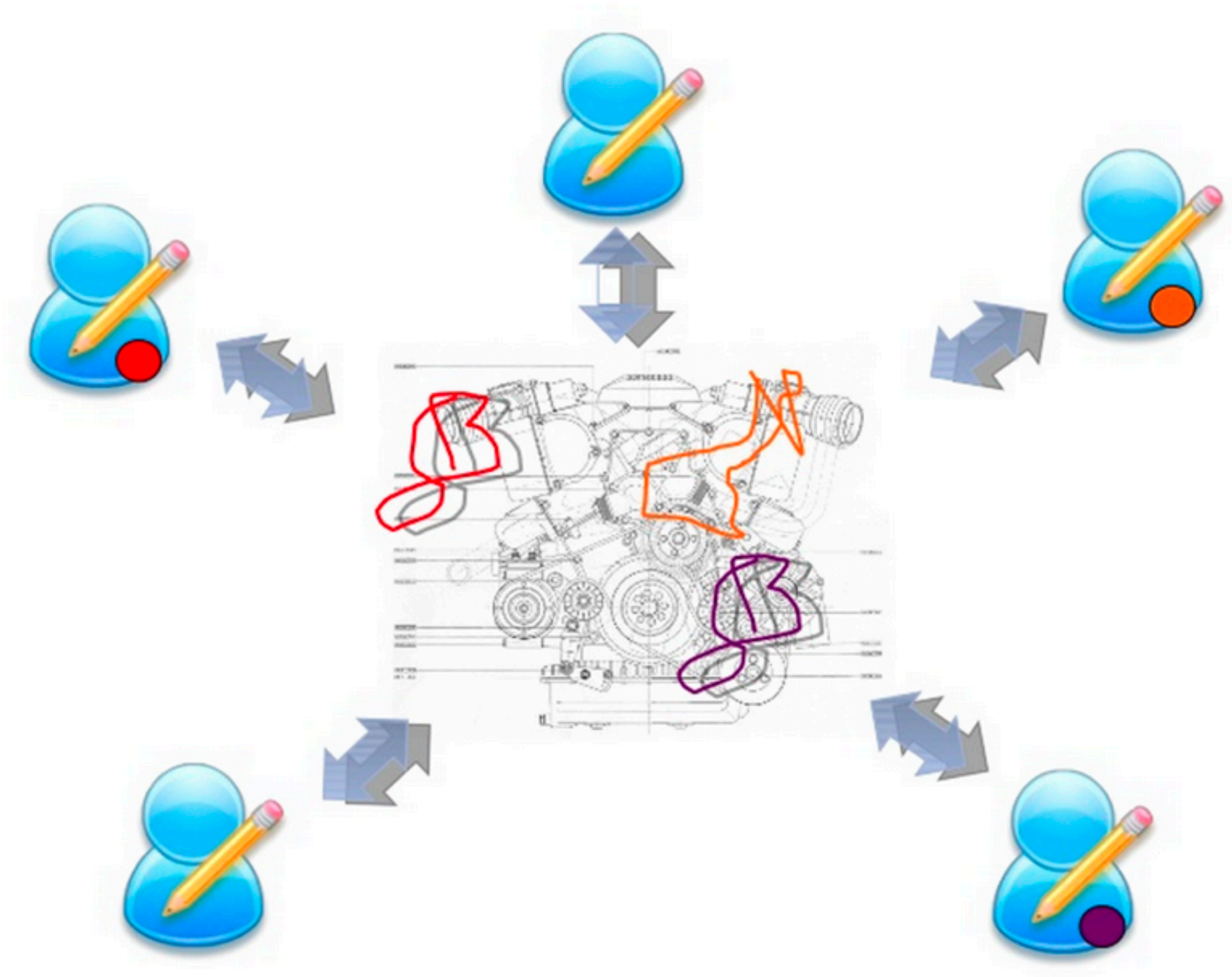
Version Control



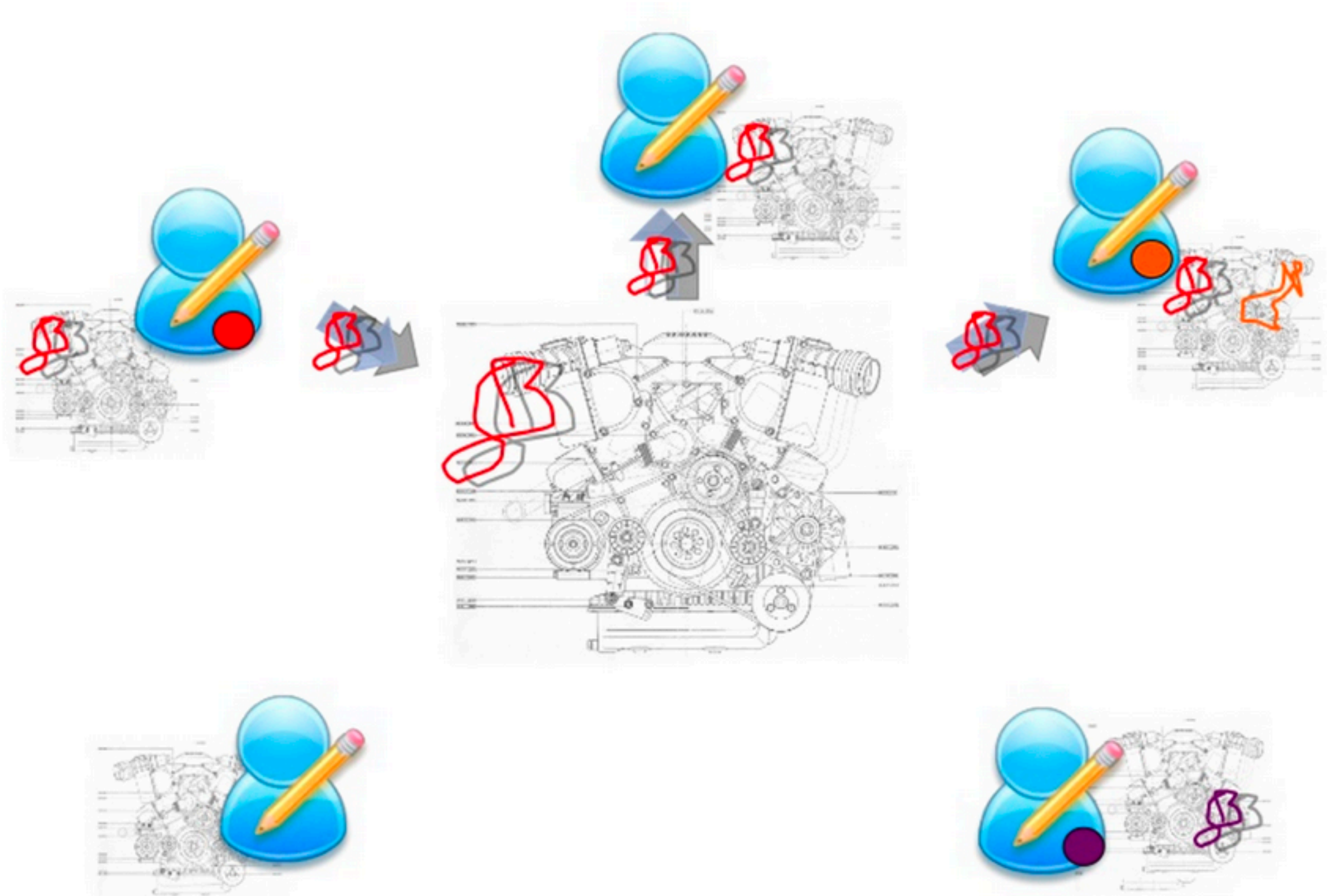
Version Control



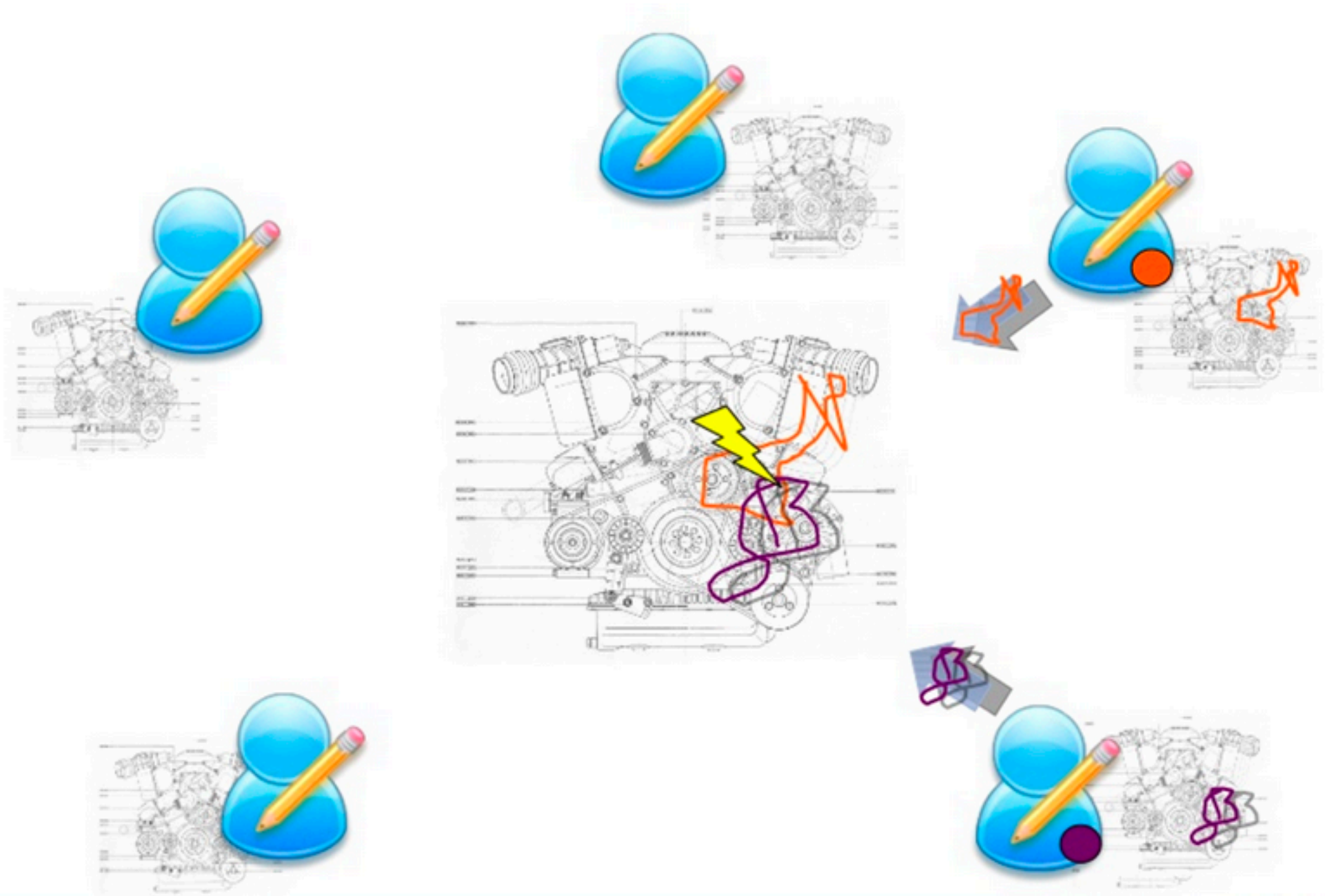
Version Control



Version Control



Version Control



EMF Store

- Client/Server architecture
- Server
  - Versioning and persistence (CDO)
  - Access control
- Client
  - Offline operation
  - Commit/update models
  - Interactive model merging (EMF Compare)
  - Views
    - Repository browser
    - History Browser

Eclipse File Edit Navigate Search Project Run Window Help (Finishing Charge...) Tue 15:00

Unicase – Eclipse – /Users/Shared/workspaces/unicase

Unicase Debug Java

Unicase Navigator

EmfStore Browser | Task View | Error Log | Status View | Status View | Status View | Status View

unicase Developer Server [unicase-internal.informatik.tu-muenchen.de]

117M of 284M

Model Versioning Problems

- No generic match, merge strategy for models, like for text
- Custom merge UI required for models with graphical notation
- Models with textual notation can be managed with traditional text/code based technologies

Course Summary

Course Summary

- Classification of computer languages
- Language aspects (notation, structure, semantics)
- Language descriptions, tools, instances
- Object-oriented meta-modeling, **4-layer**, multi-level-metamodeling, problems with multi-level-metamodeling
- **Ecore,** differences to **UML class diagrams**, **Java-mapping**
- Serialization, notification, MVC
- Validation, **OCL**
- Textual notations, strategies, **grammar-to-metamodel-mapping**, scoping
- Types of semantics/descriptions, interpreter vs. code-generation, code-generation vs. model-to-model, **elaboration**