Tailor-made model comparison: How to customize EMF Compare for your modeling language

Philip Langer <planger@eclipsesource.com>
EMFCompare

- Comparing and merging EMF-based models
  - Obtaining differences among model versions
    - Two-way differencing
    - Three-way differencing
  - Merging concurrently modified model versions
  - Merge viewers
    - Showing differences
    - Merging differences
    - Resolving conflicts
  - Integration with Eclipse team providers, such as EGit
  - Ships with dedicated support for Papyrus UML and GMF
The way how you model...
… should be as close as possible to how you compare/merge them
How does EMF Compare work?

- Compare and merge the **logical structure** of the models
- Raise the unit of comparison to **model elements**

- EMF Compare’s core is built with generic algorithms
  - Make use of the metamodel to obtain knowledge on logical structure
  - **Reflective API** to access and compare values generically
  - Support **every modeling language** that is specified in EMF

---

**EMF Compare**

EMF's reflective API

EOBJECT, EREFERENCE, EATTRIBUTE, ...

Metamodel (UML.ecore)

Model (model.uml)
How does EMF Compare work?

- Model comparison phases
  - Matching
    Finding corresponding model elements
  - Diffing
    Identifying differences among corresponding model elements
  - Analysis of differences
    Equivalences, dependencies, and conflicts among differences
How does EMF Compare work?

- Model comparison phases
  - Matching
    Finding corresponding model elements
  - Difffing
    Identifying differences among corresponding model elements
  - Analysis of differences
    Equivalences, dependencies, and conflicts among differences

![Model comparison diagram](image)
How does EMF Compare work?

- Model comparison phases
  - Matching
    Finding corresponding model elements
  - Differing
    Identifying differences among corresponding model elements
  - Analysis of differences
    Equivalences, dependencies, and conflicts among differences

**Diagram:**

1. **Match** of class Artist of Version 1 with Artist of Version 2
   - Diff: isAbstract = false vs. isAbstract = true
   - Diff: superClasses = [] vs. superClasses = [Actor]

2. **No match** of Actor in Version 1
   - Diff: Class Actor has been added in Version 2
How does EMF Compare work?

- Model comparison phases
  - Matching
    Finding corresponding model elements
  - Diffing
    Identifying differences among corresponding model elements
  - Analysis of differences
    Equivalences, dependencies, and conflicts among differences

**Comparison** of class Artist of Version 1 with Artist of Version 2
- Diff: isAbstract = false vs. isAbstract = true
- Diff: superClasses = [] vs. superClasses = [Actor]

**No match** of Actor in Version 1
- Diff: Class Actor has been added in Version 2

**Dependencies**
- Diff: superClasses = [] vs. superClasses = [Actor]
- Diff: Class Actor has been added in Version 2
How does EMF Compare work?

- Model comparison phases
  - Matching
    Finding corresponding model elements
  - Diffing
    Identifying differences among corresponding model elements
  - Analysis of differences
    Equivalences, dependencies, and conflicts among differences

Comparison Model

Match Model
- Matches of corresponding elements
- Scope of the comparison

Difference Model
- List of differences of each side
- Generic differences
  - Reference change
  - Attribute change
  - ...

Relationships among differences
- Equivalences
- Dependencies
- Implications
- Conflicts
How does EMF Compare work?

**Comparison Model**

**Match Model**
- Matches of corresponding elements
- Scope of the comparison

**Difference Model**
- List of differences of each side
- Generic differences
  - Reference change
  - Attribute change
  - …

**Relationships among differences**
- Equivalences
- Dependencies
- Implications
- Conflicts
When and why is customization needed?

- Gap between what EMF Compare processes and what users see
  - EMF Compare only “knows” the abstract model and the metamodel
  - Users only know how to use the modeling editor
When and why is customization needed?

- Gap between what EMF Compare processes and what users see
  - EMF Compare only “knows” the abstract model and the metamodel
  - Users only know how to use the modeling editor

<table>
<thead>
<tr>
<th>Model (model.uml)</th>
<th>Metamodel (UML.ecore)</th>
<th>EMF’s reflective API</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EObject</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EReference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EAttribute ...</td>
</tr>
</tbody>
</table>

- Provides domain-specific view of the model
- Hides complexity of the modeling language
- Offers composite actions and types

I have never seen the metamodel!
When and why is customization needed?

- Gap between what EMF Compare processes and what users see
  - EMF Compare only “knows” the abstract model and the metamodel
  - Users only know how to use the modeling editor

![Diagram showing EMF's reflective API and its relation to Metamodel (UML.ecore) and Model (model.uml)]
Customizing EMF Compare

- EMF Compare is a highly-customizable framework
  - Enabling us to implement tailor-made model comparison tools
  - Take generic support of EMF Compare and make it domain-specific

- Customization and extension points: EMF Compare Core
Customizing the EMF Compare Merge Viewer
Customizing the EMF Compare Merge Viewer

- **Group providers** specify the content of this tree over a comparison. Default providers can be customized.

- **Additional group providers** can be added for more options.

- **Filters** can be added and auto-activated or just added as option for users.

- **Elements have to adapt to a type**. Double-clicking an element will open the content merge viewer for the respective type.

- **Content merge viewers**. One or more content merge viewers can be provided for a defined type.
Demos of Example Customizations

- Library example
  - Additional filter
  - Additional domain-specific grouping option
  - New difference type

- Other model viewer
  - Custom content merge viewer

- UML-RT Protocol Customization
  - Custom item providers
  - Custom difference type
Future Perspectives

● It isn’t possible to build a one-size-fits all model comparison tool
  ○ Merge viewer should be as close as possible to how users work
  ○ Customized modeling editor → customized model comparison tool

● Build a highly extensible and well-performing basis
  ○ Make extending EMF Compare as easy as possible
  ○ Maximize the reuse of knowledge from the modeling editor

● Build a perfect ready-to-use model comparison tool
  ○ For Papyrus UML, UML-RT, and SysML on top of EMF Compare
  → http://www.collaborative-modeling.org
Thank you very much!

- More information
  - [http://www.collaborative-modeling.org](http://www.collaborative-modeling.org)
  - [EMF Compare Developer Guide](http://www.collaborative-modeling.org/emfcompare)
  - [EMF Compare Forum](http://www.collaborative-modeling.org/emfcompare)

- Contact us
  - planger@eclipsesource.com
Evaluate the Sessions

Sign in and vote at eclipsecon.org