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

Philip Langer <planger@eclipsesource.com>
Laurent Delaigue <laurent.delaigue@obeo.fr>
Maximilian Koegel <mkoegel@eclipsesource.com>
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?

- Raise the unit of comparison from text line to **model elements**
- Compare and merge the **logical structure** of the models
- 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
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
  - Differing
    Identifying differences among corresponding model elements
  - Analysis of differences
    Equivalences, dependencies, and conflicts among differences

![Diagram showing model comparison phases](image)
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

![Diagram showing model comparison phases](image)

1. **Comparison** 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
  ○ Differing
    Identifying differences among corresponding model elements
  ○ Analysis of differences
    Equivalences, dependencies, and conflicts among differences

![Model Comparison Phases Diagram]

- **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
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

EMF’s reflective API

- EObject
- EReference
- EAttribute

Metamodel (UML.ecore)

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

Model (model.uml)

- 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

The relative customization distance
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** can be provided for a defined type, and 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
Future Perspectives

- It isn’t possible to build a one-size-fits all model comparison tool
  - Difference 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**
  - [EMF Compare Developer Guide](http://www.collaborative-modeling.org)
  - [EMF Compare Forum](http://www.collaborative-modeling.org)
  - Contact us

- **Follow us on** [http://www.collaborative-modeling.org](http://www.collaborative-modeling.org)
Evaluate the sessions at www.eclipsecon.org