Adding Functionality to *openOME* for Everyone

Ralf Laue, Arian Storch

Chair of Applied Telematics / e-Business, University of Leipzig, Germany
laue@ebus.informatik.uni-leipzig.de

Abstract. Adding new functionality to graphical editors like *openOME* usually requires to become familiar with the programming environment of the underlying framework.
We present an interface for Eclipse EMF/GMF-based modelling tools that allows to add new functionality very quickly - without the need to be familiar with Eclipse development.

1 Primary Features

The Eclipse Modeling Framework (EMF) and the Eclipse GMF (Graphical Modeling Framework) are well-established frameworks for developing modelling tools. An example for an EMF/GMF-based i^* modelling tool is *openOME* [1].

We know from our own experience that often a lot of knowledge is required before someone can actually start to add functionality to EMF/GMF-based editors. The aim of our Eclipse plugins - called Eclipse Modeling Toolbox - is to make the task of extending editors like *openOME* as easy as possible. In this section, we present the possibilities offered by our plugins.

1.1 User-Defined Attributes

An additional view for user-defined attributes allows to add own attributes to model elements or to a model as a whole. Fig. 1 shows how user defined attributes have been added to a task in an i^* model. If a URL referring to a local or remote file is used as an attribute value, this file can be opened by clicking on the URL. This way, it is possible to associate additional files to a model element (for example business process descriptions that are related to a task).

1.2 Export and Import

By using the Eclipse Modeling Toolbox, it is possible to create export and import functionality for other file formats. This export and import is done by means of XSLT transformations. So far, we have created transformations for exporting an *openOME* model to a set of Prolog facts. We are currently working on an export to the interchange format iStarML [2].

A new format can be supported by simply adding a new XSLT file and inserting information about the export/import format to a configuration file.

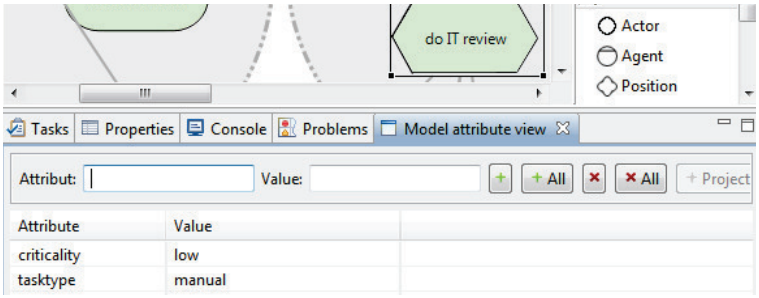

Fig. 1. User-defined attributes added to a task

1.3 Integrating Add-ons

We tried to make the integration of third-party programs into *openOME* as easy as possible. We know from our own experience that often a lot of knowledge is necessary before someone can actually do such integration. At least, the answers to the following questions have to be known:

- How can we access the model data and transform them into the data format expected by the third-party program?
- How can we start the third-party program from within the modeling tool?
- How can we transfer the answers given by the third-party program back into the user interface of the modeling tool?

With our plugins , we provide easy-to-understand interfaces for dealing with the above questions. The already mentioned export scripts can be used for accessing and transforming the model (including its user-defined attributes as described in Sect. 1.2).

The information on how to start the external program can be added to a configuration table at runtime, either manually or by importing an XML file containing the necessary information. This creates a new menu item from which the external program can be started.

Finally, we have to make sure that the results computed by the third-party tool are transferred back into the *openOME* user interface. For this purpose, we provide several interfaces. They abstract away Eclipse implementation details and allow the external program

- to print information into the Eclipse console view,
- to add information about an error, warning or information to the Eclipse problem view
- to add a visual marker to the graphical model element,
- to add, delete or change attributes of the modelling elements (which includes existing attributes such as “name”, graphical attributes such as “element size” and user-defined attributes)

With the described features, it is possible to integrate new functionality into *openOME* without having to learn about Eclipse development. In the most cases, new features can be added even without having to compile the sources.

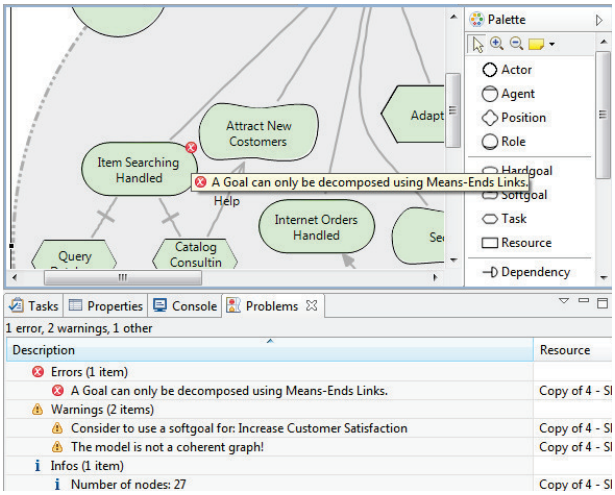

Fig. 2. Results from the add-on "Extended Model Validation" in *openOME*

1.4 Validation

As an example for a useful *openOME* add-on, we have developed an add-on called "Extended Model Validation". It exports the information that is contained in the model into a set of Prolog facts. Afterwards, SWI-Prolog is called for locating modelling problems such as syntactical errors (like "dependency link without dependum"), layout problems and (to some extent) problems with the labels (like labels including the phrase "...to be..." within a task instead of a goal). More information about the model validation approach can be found in [3].

2 Status and Future Plans

It is important to mention that the Eclipse Modeling Toolbox can be integrated within any EMF/GMF-based modelling tool. It has been used successfully within the business process modelling tool *bflow** for the development of some useful add-ons. Future plans related to *openOME* include to provide iStarML import and to add more functinality to the add-on-mechanism. Everyone is invited to use and improve the Eclipse Modeling Toolbox which is available at `http://sourceforge.net/projects/eclipsemodeling/`.

References

1. www.cs.toronto.edu/km/openome/: (Openome, an requirements engineering tool)
2. Cares, C., Franch, X., Perini, A., Susi, A.: Towards interoperability of i* models using iStarML. Computer Standards & Interfaces **33** (2011) 69 – 79
3. Laue, R., Storch, A.: A flexible approach for validating i* models. In: Proceedings of the 5th International i* Workshop, Trento, Italy. (2011)