Bridging the Epsilon Wizard Language and the Eclipse Graphical Modeling Framework

Dimitrios S. Kolovos, Richard F. Paige, Louis M. Rose, and Fiona A.C. Polack

Department of Computer Science, University of York,
Heslington, York, YO10 5DD, UK.
{dkolovos,paige,lmr109,fiona}@cs.york.ac.uk

1 Introduction

The Eclipse Graphical Modeling Framework (GMF) provides tooling that enables developers to implement high-quality diagrammatical editors for modelling languages built atop the Eclipse Modeling Framework (EMF). Epsilon is an Eclipse GMT component that aims at providing infrastructure for developing tool support for task-specific model management languages. The Epsilon Wizard Language is a language built atop Epsilon, which targets the task of in-place model transformations. In this paper, we present the results of our effort to provide a bridge between GMF and EWL that enables users to define and execute in-place transformations (wizards) expressed in EWL in the context of GMF-based editors. The rest of the paper is organized as follows. In Sections 2 and 3 we briefly discuss Epsilon and EWL. In Section 4 we present our implementation and in Section 5 we conclude and provide directions to future work on the subject.

2 The Epsilon GMT Component

Epsilon is a component of the Eclipse GMT incubator project. Its aim is to provide infrastructure that enables users to implement task-specific model management languages and supporting tooling with reduced effort. The infrastructure provided by Epsilon includes an imperative OCL-based model management language (EOL [1]), an extensible framework for accessing models of diverse technologies (e.g. EMF, MDR), reusable development tools such as interpreters, editing and launching facilities, ANT tasks, code profiling tools [2], and utilities such as a customizable reflective editor for EMF models (Exeed) [3] and a multi-tab editor (ModeLink) for establishing strongly-typed links between different EMF models.

3 The Epsilon Wizard Language (EWL)

Among the languages developed atop Epsilon is the Epsilon Wizard Language (EWL), which has been designed to support the task of interactive in-place model transformations. In this section, we briefly discuss the syntax and semantics of EWL. For a more detailed discussion on the language, readers can refer to [4].

The basic concept of EWL is that of a *wizard*. A wizard in EWL consists of three parts: a *guard*, a *title* and a *body*. The guard specifies if the wizard is applicable to a specific selection of model elements. The title, specifies a context-aware title for the wizard and the body specifies the effects the wizard will have on the model. An EWL module consists of a number of wizards and helper operations. Upon request, the EWL interpreter can return the subset of wizards that apply to a given selection of model elements. We use the exemplar UML2 wizard of Listing 1.1 to introduce the concrete syntax of EWL wizards:

Listing 1.1. Exemplar UML2 EWL Wizard

```
1  wizard ExtractInterface {
2
3    guard : self.isKindOf(Class)
4
5    title : 'Extract interface from class ' + self.name
6
7    do {
8
9      var newName : String;
10     var message : String := 'Please provide a name for the new interface';
11     newName := UserInput.prompt(message, 'I' + self.name);
12
13     var i : new Interface;
14     self.owner.packagedElement.add(i);
15     i.name := newName;
16
17     var g : new Generalization;
18     self.generalization.add(g);
19     g.general := i;
20
21     if (UserInput.confirm('Update references?', true)) {
22       for (te : TypedElement in
23         TypedElement.allInstances.select(e|e.type = self)) {
24
25         te.type := i;
26       }
27     }
28
29     for (o : Operation in self.ownedOperation.clone()) {
30       i.ownedOperation.add(o);
31     }
32
33   }
34
35 }
```

The *guard* part of this wizard (line 3) specifies that it is only applicable if the selection, accessible through the built-in `self` variable, is a single UML2 class. If the selected class is named `Connection`, the *title* of the wizard (line 5) will evaluate to

Extract interface from class Connection. Finally, when the *do* part is executed (lines 7-35), it will prompt the user for a name for the new interface (line 11), create a new interface and assign it the specified name (lines 13-15) and create a generalization relationship between the class and the interface (lines 17-19). Then it will prompt the user to confirm that (s)he also wants to update any `ETypedElements` (e.g. `Properties`) in the model that point to the class to now point to the new interface (line 21) and move all the operations from the class to the interface (lines 29-31).

4 Bridging GMF and EWL

To provide support for EWL wizards in existing GMF editors, we added a new sub-menu named *Wizards* to the context menu of `IGraphicalEditPart` objects. In GMF, an `IGraphicalEditPart` is the visual representation of an underlying model element (`EObject`), which can be resolved through the `IGraphicalEditPart.resolveSemanticElement()` method. Thus, when a number of `IGraphicalEditParts` are selected and the user selects the wizards sub-menu, the guards and titles of the EWL wizards associated with the filename extension of the diagram (e.g. ecore_diagram, umlclass_diagram etc) are evaluated. The titles of the applicable wizards appear under the Wizards submenu as seen in Figure 1. Then, the user can execute an applicable wizard by selecting its title. To be of practical use, our implementation provides support for undoing/redoing the effects of the executed wizards by using the `ChangeRecorder` EMF facility.

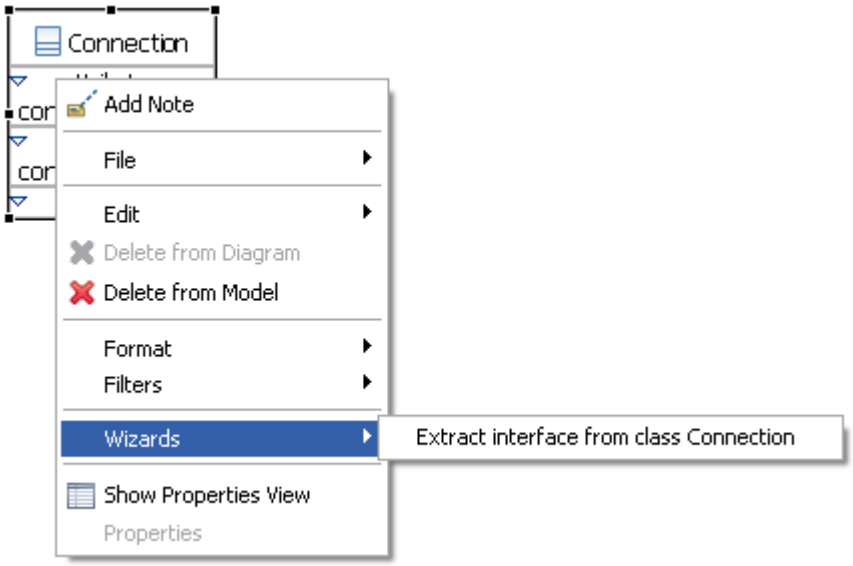

Fig. 1. Screenshot of the additional sub-menu that hosts applicable EWL wizards

Binding specific EWL wizards to diagram file extensions can be achieved in two ways: either by using the `org.epsilon.ewl.eclipse.gmf.wizards` extension point (defined in the `org.epsilon.ewl.eclipse.gmf` plugin) or by using the graphical interface available under Window→Preferences→Epsilon→GMF Wizards.

Since EWL supports calls to methods of Java objects (a feature inherited from EOL), users can also define and use custom Java objects (tools in Epsilon terminology [5]) to perform computationally-demanding tasks at which Java is more efficient. This technique can also be used to perform GMF-specific tasks such as displaying Eclipse-based dialogs to prompt the user to select additional model elements or highlight particular elements in the editor.

The complete source code of our implementation is available on the Eclipse CVS server under the path `/technology/gmt/epsilon/org.epsilon.ewl.eclipse.gmf`. Also, an animated screencast of the example presented in this paper is available at http://www.eclipse.org/gmt/epsilon/cinema/.

5 Conclusions and Further Work

In our view, the ability to specify custom wizards in existing GMF-based editors at the abstract syntax level using a high-level language such as EWL is beneficial both to editor vendors and for end-users. By using this technique, vendors can provide built-in wizards for automating commonplace model refactoring activities. Additionally, end-users can automate repetitive tasks - often unanticipated by the vendor - using a simple language that abstracts over the technical complexity of the underlying EMF and GMF frameworks.

We are working towards using the presented approach to implement interactive fault analysis in a GMF-based editor constructed for a respective DSL and expect to report back on this soon. In the future, we plan to investigate the potential of specifying wizards with macro-recording techniques by generating EWL wizards from EMF `ChangeDescriptors`.

6 Acknowledgements

The work in this paper was supported by the European Commission via the MODELPLEX project, co-funded by the European Commission under the "Information Society Technologies" Sixth Framework Programme (2006-2009).

References

1. Dimitrios S. Kolovos, Richard F.Paige and Fiona A.C. Polack. The Epsilon Object Language. In *Proc. European Conference in Model Driven Architecture (EC-MDA) 2006*, volume 4066 of *LNCS*, pages 128–142, Bilbao, Spain, July 2006.
2. Dimitrios S. Kolovos. An Overview of the Epsilon Profiling Tools, July 2007. http://www.eclipse.org/gmt/epsilon/doc/EpsilonProfilingTools.pdf.

3. Dimitrios S. Kolovos. Exeed: EXtended Emf EDitor - User Manual, 2007. www.eclipse.org/gmt/epsilon/doc/Exeed.pdf.
4. Dimitrios S. Kolovos, Richard F. Paige, Louis M. Rose and Fiona A.C. Polack. Update Transformations in the Small with the Epsilon Wizard Language. *Journal of Object Technology (JOT), Special Issue for TOOLS Europe 2007*, 2007. http://www.cs.york.ac.uk/~dkolovos/publications/EWL.pdf.
5. Dimitrios S. Kolovos. An Overview of the Epsilon Tools, July 2007. http://www.eclipse.org/gmt/epsilon/doc/EpsilonTools.pdf.