An EMF Framework for Event-B

Colin Snook, University of Southampton,
Fabian Fritz, Heinrich-Heine Universität Düsseldorf, and
Alexei Iliasov, Newcastle University

cfs@ecs.soton.ac.uk, fritz.fabian@googlemail.com, alexei.Iliasov@ncl.ac.uk

Abstract. The Rodin platform for Event-B formal modelling is based on a bespoke model repository. While this approach has some advantages it also means that the platform does not benefit from the host of emerging technologies and supporting packages that are freely available for open source modelling frameworks. The Eclipse Modelling Framework (EMF) has emerged as the de-facto standard package upon which to build any Eclipse based modelling tool. In order to make this support available for our Rodin tools we have now engineered a front-end EMF based representation of Event-B.

1 Introduction

The Eclipse Modelling Framework (EMF)[1] provides supporting infrastructure for building tools and other applications based on a structured data model. The EMF contains generative tools and runtime support for the implementation of a model repository based on a meta-model of the domain. Also provided are a set of adapter classes that enable viewing and command-based editing of the model. Further support for other model manipulation tasks has been added to the EMF. For example, support for model transformation, comparison and merging is provided.

During its initial development, the Rodin platform[2] for Event-B[3] did not utilise EMF technology because of fears about performance[1] and maturity. This prevents integration with other EMF based modelling tools. To completely re-base the Rodin platform on the EMF retrospectively would require extensive re-work of the Rodin verification tools which are tightly integrated with Rodin's bespoke repository format. Our solution, which is being developed as part of the Deploy project[2], is to provide an EMF based 'front-end' to the Rodin platform and retain the current repository and verification tool set. To do this we have implemented an EMF based repository for Event-B models that overrides the default serialisation to persist models via the Rodin API. There were significant challenges to provide a flexible EMF implementation. For example, some tools need to work at the project level manipulating a collection of Event-B

[1] Whether EMF adversely affects performance remains untested since we only provide EMF support for front-end interfaces with low performance demands.
[2] DEPLOY: EU Project IP-214158, www.deploy-project.eu

components (machines and contexts) whereas others are scoped within a single component. A more detailed description of the EMF Framework for Event-B is given on the Event-B wikipedia[4].

The Event-B meta-model defines the structure of Event-B projects. The model is contained within the repository plugin, *org.eventb.emf.core*. It is structured into three packages for clarity. The core package contains a structure of abstract meta-classes so that models can be treated generically as far as possible. The core package also contains mechanisms to handle extensions provided by other plug-ins and a meta-class to model entire projects. There are two sub-packages, contained with the core package, for machine and context.

The persistence plug-in *org.eventb.emf.persistence* overrides EMF's default XMI serialisation so that models are persisted in the Rodin Database. The serialisation uses the Rodin API so that it is decoupled from the actual serialisation of the Rodin database. Our EMF Persistence API provides methods to load and unload components, save changes, and register listeners to projects and components. An extension point is provided for offering synchronisers for new element kinds (extensions). Persistence takes a rewriting approach. That is, it is more efficient to clear the contents of a model component (machine or context) and regenerate it than to calculate what has been changed and only save the changes. Currently, the EMF framework for Event-B is reliant on the Rodin tool but, if in the future alternative verification and validation tools are provided, it could easily be adapted to be independent of Rodin by deleting its persistence plug-in.

Rodin is intended to be extensible so that only a basic core notation is mandatory and extensions to the modelling notation can be added depending on the requirements and preferences of particular users. Rodin's approach to extensibility derives from the mechanisms of the Eclipse IDE and normal Java coding techniques. New kinds of elements are defined by writing Java classes. The Rodin database is made aware of them via extension points. The hierarchical structure is simple and liberal in that each element has a single containment for all its children. An appearance of structure is provided by interfaces that provide wrapper methods to return children of a particular type.

In order to maintain the extensibility of the Rodin platform the EMF framework for Event-B needs to be equally extensible. However, we wish to provide a more structured containment hierarchy in order to utilise EMF generative and dynamic capabilities. To achieve this we have incorporated a mechanism for extensions. The extension mechanism can also be used to store temporary volatile modelling data that will not be persisted. The extensions will only be persisted if valid Rodin identifiers are provided. Two mechanisms are provided, one for simple attributes (corresponding to Rodin attribute extensions) and one for extension elements (corresponding to Rodin element extensions). An extensions containment has been built into the base meta-class (i.e. the abstract class *EventBElement* which is the basis of the inheritance tree). The target meta-class, *Extension*, can be subclassed by extenders so that new elements can be added into the extensions containment. This mechanism has already been used successfully but is still being enhanced in the light of experience. We suggest that an intermediary subclass is introduced which will control the placement of such extensions, otherwise the new elements will be available on all existing and future elements. The content providers of an extension can be customised to bypass these intermediate classes, taking information they provide in order to correctly construct and return the containments they define.

Currently, any extensions made to the EMF framework are not automatically reflected within the Rodin database. Extensions must also be implemented in the Rodin database and persistence synchronisers must be provided to perform load and save functionality. In the longer term it may be possible to provide development tools that automatically generate Rodin database extensions from EMF ones.

2 Tools based on the EMF framework for Event-B

Camille text editor Based on the EMF framework, a state-of-the-art text editor has been created that provides features, such as syntactic and semantic highlighting, code completion, quick navigation and outline view. The text editor is already published and in use by users of the Rodin platform. The Camille editor was developed in parallel to the EMF framework and was therefore a driver for much of its implementation. For example, a mechanism to lazily resolve inter-resource URI references was developed from the need of the text editor to edit reference names without loading the referenced component. The Camille text editor has also contributed an extension to the EMF framework using the extensibility mechanism described above. The extension provides an EMF model of the full syntax tree of Event-B formulas (i.e. predicates, expressions, etc.) which is populated using the existing Rodin parser.

Compare/Merge Editor In several situations conflicts between different versions of an Event-B model can occur and a compare and merge editor is essential for team-based development where it can be linked to a version control system such as SVN or CVS. A prototype team-working plug-in is now available and includes a structure-aware compare and merge editor based on the EMF Compare sub-project.

Structured Editor EMF provides support to generate structured (e.g. tree, list, table) editors for models. An adapted version of this editor allows users to edit machine and context elements within a structure using menu-guided selections. An important feature of this editor is that it should be easily extensible so that extension plug-ins can contribute new containments to existing parents and this is automatically supported without modifying the editor. This feature is driving the ongoing development of the extensibility mechanism.

UML-B [5] is currently being re-implemented as an extension to the Event-B meta-model. The UML-B meta-classes will extend and add to the meta-classes of Event-B. This will provide greater integration between the EMF based Event-B editors and the UML-B diagrammatic editors.

3 Conclusion

An EMF based interface to Rodin provides access to important peripheral support such as model transformation technologies and model management tools. On the other hand the benefits of an efficient bespoke repository and verification tool set have been retained. An important issue is that of extensibility. Mechanisms have been incorporated but are not yet fully finalised. It is likely that improved tool support for Rodin plug-in developers will be required to alleviate the burden of a growing number of facilities that require configuration to support a particular extension.

References

1. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: Eclipse Modeling Framework. 2nd edn. The Eclipse Series. Addison-Wesley Professional (December 2008)
2. Butler, M., Hallerstede, S.: The Rodin Formal Modelling Tool. BCS-FACS Christmas 2007 Meeting - Formal Methods In Industry, London. (December 2007)
3. Metayer, C., Abrial, J.R., Voisin, L.: Event-B Language. Rodin deliverable 3.2, EU Project IST-511599 -RODIN (May 2005)
4. Snook, C., Fritz, F., Iliasov, A.: Event-B and Rodin Documentation Wiki: EMF Framework for Event-B. `http://wiki.event-b.org/index.php/EMF_framework_for_Event-B` (2009) Accessed January 2010.
5. Snook, C., Butler, M.: UML-B and Event-B: an integration of languages and tools. In: The IASTED International Conference on Software Engineering - SE2008. (February 2008)