Tooling: Code Generation Update

A. Edmunds, C. J. Lovell, R. Silva, I. Maamria and M. Butler

The Event-B method, and supporting tools have been developed during the DEPLOY project. A number of the industrial partners, are interested in the formal development of multi-tasking, embedded control systems. During the project, work has been undertaken to investigate automatic generation, from Event-B models, for these type of systems. Initially, we chose to translate to the Ada programming language, and use it as a basis for the abstractions used in our approach. The first version of the code generator supported translation to Ada, and the current version also has limited support for C.

We released a new version of the code generator on 14-12-2011. The presentation will give details of the the enhancements that have been made. We have made changes to the methodology, user interface, and tooling. The code generators have been completely re-written. The translators are now implemented using Java only. In our previous work we attempted to make use of the latest model-to-model transformation technology, available in the Epsilon tool set, but we decided to revert to Java since Epsilon lacked the debugging and productivity features of the Eclipse Java editor. We have also updated the documentation, including the Tasking Event-B Overview, and Tutorial materials.

We described our previous code generation feature as a demonstrator tool; chiefly a tool designed as a proof of concept, used by us to validate the approach. In this sense, the tool as it stands now, is the first prototype intended to be used by developers. However, we can use the demonstrator as a baseline, and describe the new features as follows:

- Tasking Event-B is now integrated with the Event-B explorer. It uses the extensibility mechanism of Event-B EMF (In the previous version it was a separate model).
- We have the ability to translate to C and Ada source code, and the source code is placed in appropriate files within the project.
- We use theories to define translations of the Event-B mathematical language (Theories for Ada and C are supplied).
- We use the theory plug-in as a mechanism for defining new data types , and the translations to target data types.
- The Tasking Event-B to Event-B translator is fully integrated. The previous tool generated a copy of the project, but this is no longer the case.
- The translator is extensible.
- The Rose Editor is used for editing the Tasking Event-B. A text-based editor is provided, using the Rose extension, for editing the TaskBody.
- The composed machine component is used to store event 'synchronizations'.
- Minimal use is made of the EMF tree editor in Rose.

A text-based task body editor was added, to minimize the amount of editing required with the EMF tree editor. The task body editor is associated with a parser-builder; after the text is entered in the editor the EMF representation is generated (by clicking a button) that is, assuming parsing is successful. If the parser detects an error, information about the parse error is displayed in an adjoining text box. When specifying events in the task body, there is no longer a need to specify two events involved in a

synchronization. The code generator automatically finds the corresponding event of a synchronization, based on the event name, and using the composed machine component. Composed machines are used to store event 'synchronizations', and are generated automatically during the decomposition process. This reduces the amount of typing in the TaskBody editor, since we no longer need to specify both local and remote (synchronizing) events. The new feature also overcomes the 'problem' that we previously experienced, with duplicate event names in a development, and event selection, when specifying the task body. The EMF tree editor in Rose is now only used minimally; to add annotations for Tasking, Shared and Environ Machines; typing annotations, and parameter direction information; and sensing/actuating annotations, where necessary. Further work is under way to integrate the code generation feature with the new Rodin editor.

The code generation approach is now extensible; in that, new target language constructs can be added using the Eclipse extension mechanism. The translation of target's mathematical language is now specified in the theory plug-in. This improves clarity since the the translation from source to target is achieved by specifying pattern matching rules. The theory plug-in is used to specify new data-types, and how they are implemented. Translated code is deposited in a directory in the appropriate files. An Ada project file is generated for use with AdaCore's GPS workbench. Eventually this could be enabled/disabled in a preferences dialog box. The Tasking Event-B to Event-B translator is now properly integrated. Control variable updates to the Event-B model are made in a similar way to the equivalent updates in the state-machine plug-in. The additional elements are added to the Event-B model and marked as 'generated'. This prevents users from manually modifying them, and allows them to be removed through a menu choice.

The DEPLOY project is funded by the European Commission ICT DEPLOY contract / grant number: 214158.