Code Generation with the Event-B Tasking Extension (Tool Development)

Andy Edmunds
University of Southampton

July, 2010

Abstract

In previous work [2, 4] we described an approach for linking Event-B with Java using an intermediate specification language. In the Deploy project, generating Ada and C source code [1, 5, 6] are of particular interest. We learned from our work in [3] that easing the transition from Event-B to the Intermediate Specification would be beneficial. In our current work we therefore propose an approach that provides a specification notation which is closer to Event-B than the target language (previously the specification notation more closely corresponded to the target source code). We provide an extension of Event-B which we call the Event-B Tasking Language. We use the EMF representation of Event-B and make use of its extension mechanism.

In Version 1 of the Tasking Language, we introduce Tasking Machines and Shared Machines. Shared Machines are just Event-B Machines, as previously defined. We simply give it this name to point out the distinction. Tasking Machines are Event-B Machines extended with sequence, loop, branch, and call constructs. Loop, Branch and Call constructs can simultaneously invoke a local (task) event, and a remote (shared) event. In Version 1 of the Tasking Language we do not accommodate parameter passing or nested constructs. In addition to the new tasking operators we allow specification of the task type and task priority. The task type may be Repeating, Triggered and Periodic, and are modelled with continuous looping behaviour; one-shot tasks terminate after a single execution. The task priority, which is used in the mapping to Ada code, is not treated in a formal manner.

The work undertaken includes construction of EMF metamodels representing the Event-B Tasking Language, and a Common Language Metamodel. We also make use of the existing EMF eventbcore metamodel and the EMF formulas metamodel The Common Language Metamodel is an abstraction of some common programming language elements, and can be used with other translator plug-ins for generation of source models/code. An Ada EMF model is being developed, and work on translators is being undertaken in Newcastle by Alexei. The translation is performed, in Eclipse, by the Epsilon suite. Epsilon includes ETL, a model to model

transformation language (that is, EMF model to EMF model). We specify translation rules, written in ETL, to translate source model elements of the tasking model to the target model elements.

We provide two translators; the first is for translating the the tasking model to an Intermediate Model suitable for subsequent translation to an implementation. The second translation is to an Event-B model which describes the behaviour of the implementation. The result of this translation should refine the abstract tasking development. We describe the Tasking Metamodel extension to Event-B, the Intermediate Language Metamodel, and provide an overview of the translations to each.

References

[1] SPARKAda. Available at http://www.praxis-his.com/sparkada/index.asp.

[2] A. Edmunds and M. Butler. Linking Event-B and Concurrent Object-Oriented Programs. In *Refine 2008 - International Refinement Workshop*, May 2008.

[3] Andrew Edmunds. *Providing Concurrent Implementations for Event-B Developments*. PhD thesis, University of Southampton, March 2010.

[4] Andrew Edmunds and Michael Butler. Tool support for event-b code generation. February 2010.

[5] MISRA. *MISRA-C:2004 Guidelines for the Use of the C Language in Vehicle Based Software*. Motor Industry Research Association, Nuneaton CV10 0TU, UK, 2004.

[6] T.S. Taft, R.A. Tucker, R.L. Brukardt, and E. Ploedereder, editors. *Consolidated Ada reference manual: language and standard libraries*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.