Event-B Specification Templates for Defining Domain Specific Languages

Ulyana Tikhonova

Technische Universiteit Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
u.tikhonova@tue.nl

Domain-Specific Languages (DSLs) are a central concept of Model Driven Engineering (MDE). They are considered to be very effective in software development and are being widely adopted by industry nowadays. A DSL is a programming language specialized to a specific application domain. It captures domain knowledge and supports reuse of such knowledge via common domain notions and notation. In this way, the DSL raises the abstraction level of solving problems in the domain. A DSL is usually implemented as a translation from the domain concepts to the programming language of an execution platform, such as C/C++ or Java. In our work we investigate how translation of a DSL to the Event-B formalism can facilitate design and development of the DSL, and support understanding and debugging of DSL programs.

The Rodin platform offers various supporting tools for Event-B that can be applied to a DSL specification in order to implement a set of dedicated use cases. For example, the DSL semantics can be prototyped and then analyzed using automatic provers and model checkers (provided by AtelierB and ProB plug-ins); DSL programs can be simulated and debugged using animators and visualization tools (ProB and BMotion Studio). Although providing an extensive tool support, Event-B is not designed for specifying the DSL semantics. Therefore, in order to realize practical benefits of applying Event-B to a DSL, we adopt the specification formalism to the MDE context through the following model-to-model transformations.

- The *DSL-to-Event-B* transformation automatically generates an Event-B specification of an arbitrary DSL program from a set of Event-B specifications that define the DSL semantics on the meta-level [3]. Such a transformation composes the resulting specification out of the DSL semantic specifications using the technique of *shared event composition* [2].
- The *DSL-to-BMotion* transformation automatically generates a visualization for each concrete DSL program following (mimicing) the DSL graphical notation. The resulting visualization runs in the BMotion Studio together with the ProB animator and provides a graphical user interface (GUI) for the Event-B specification being animated.

As a result, DSL end-users do not need to know formal notation of Event-B to create and run specifications of their programs. We validated this approach by means of a *user study* performed in the industrial context with the real-life mature DSL. The interviewed users confirmed that the DSL-based development

process will benefit from having such harnessed specification of the DSL. However, the users indicated that to realize these benefits one needs to keep the Event-B specification of the DSL consistent with the actual implementation of the DSL, following all its changes and updates. The latter requirement might cause a high overhead of applying the DSL-to-Event-B transformations, as they realize rather complex translation of the high-level DSL concepts to the low-level Event-B concepts. In other words, the semantic gap between a DSL and Event-B is quite wide.

To manage the wide semantic gap between the DSL and Event-B and to generalize the proposed approach so that it is applicable to other DSLs, we introduce *Constelle* – an intermediate language of *reusable specification templates*. An arbitrary DSL can be defined in Constelle using specification templates, which have been identified as reusable (successful) design solutions and stored in a library. On one hand, specification templates can be (re)used to define different DSLs. On the other hand, such a definition captures intentions of the DSL developer in a more clear way.

Constelle builds on top of the Event-B formalism. This means that all specification templates are implemented in Event-B. And the *Constelle-to-Event-B* transformation automatically generates Event-B specifications of a DSL from its Constelle definition. For this, the Constelle-to-Event-B transformation specializes the invoked templates using *generic instantiation* [1] and weaves them together using *shared event composition* [2]. In this way we ensure that specification templates are reused together with the proof obligations discharged for them.

The Constelle language is not restricted to the scope of DSLs. Our vision is that a library of Event-B specification templates can be shared and filled by Event-B practitioners working in various domains. Moreover, in future specification templates can be used as a source for automatic generation and/or for semi-automatic configuration of other artifacts. For example if each specification template is coupled with a visualization template for the BMotion Studio, then a specialized visualization for a DSL can be generated from the Constelle definition. If each specification template is coupled with a source code in C/C++, then we might be able to generate the corresponding DSL implementation in source code.

References

1. R. Silva and M. Butler. Supporting Reuse of Event-B Developments through Generic Instantiation. In K. Breitman and A. Cavalcanti, editors, *11th International Conference on Formal Engineering Methods, ICFEM*, volume 5885 of *Lecture Notes in Computer Science*, pages 466–484. Springer, 2009.
2. R. Silva and M. Butler. Shared Event Composition/Decomposition in Event-B. In B. K. Aichernig, F. S. de Boer, and M. M. Bonsangue, editors, *Formal Methods for Components and Objects (FMCO)*, pages 122–141. Springer, 2010.
3. U. Tikhonova, M. Manders, M. van den Brand, S. Andova, and T. Verhoeff. Applying Model Transformation and Event-B for Specifying an Industrial DSL. In *MoDeVVa@MoDELS*, pages 41–50, 2013.