Simulating Rhapsody SysML Blocks in Hybrid Models with FMI

Yishai A. Feldman[1] Lev Greenberg[1] Eldad Palachi[2]
[1]IBM Research – Haifa, Israel [2]IBM Rational, Rehovot, Israel
{yishai,levg,eldadpal}@il.ibm.com

The Functional Mockup Interface (FMI) [1] standard enables hybrid simulation of models from different tools. Such tools can have different underlying behavioral semantics, creating challenges when models are combined. A case in point is the combination of the Rhapsody® tool [3, 2], widely used to describe and implement discrete control behavior in SysML, and Modelica®, widely used to model multi-domain physical systems.

This paper describes a plugin we developed for exporting Functional Mockup Units (FMUs) from Rhapsody, and the results of combining generated FMUs with models from other tools. The plugin is demonstrated in action through an example of a hybrid model, in which a controller specified as a SysML statechart is exported as an FMU and imported into a Modelica model in the SimulationX® tool [4].

When a Rhapsody FMU is used in a different environment, some basic assumptions on its behavior are challenged. As a result, modelers need to be aware of the fact that their models will be exported as FMUs and follow the guidelines described in the paper in order to ensure for the exported FMU to express their intentions.

For example, there are constraints on how continuous input variables can be used in Rhapsody; they must not be used to trigger Rhapsody events, otherwise they will cause thrashing of the simulation. Another issue is the fact that events in Rhapsody are sent asynchronously, while FMI semantics is synchronous. There are various strategies for the generated FMU to choose how many events are reported at each step, and these effect the semantics of the communication of the FMU with other units; the paper discusses several strategies and their implications.

Other issues include the fact that simulation time is counted in different units in FMI and in Rhapsody; this can also cause unexpected behavior of the FMU. Similarly, Rhapsody models can employ the full type system of the target language, while FMI defines a restricted set of types. The export plugin will choose the best FMI type corresponding to the Rhapsody type, to the extent possible.

Since different tools come with their own semantics, we expect that such mismatches are common, especially when connecting continuous-time with discrete models, and that our guidelines will generalize to many such cases.

References

[1] Functional Mockup Interface. `https://www.fmi-standard.org`.

[2] D. Harel and H. Kugler. The Rhapsody semantics of statecharts (or, on the executable core of the UML). In *Integration of Software Specification Techniques for Applications in Engineering*, pages 325–354. Springer, 2004.

[3] Rhapsody. `http://www-03.ibm.com/software/products/en/ratirhapfami`.

[4] SimulationX. `http://www.simulationx.com`.