



A Rodin plugin for quantitative timed models

Joris Rehm

► **To cite this version:**

Joris Rehm. A Rodin plugin for quantitative timed models. Michael Butler and Stefan Hallerstede and Laurent Voisin. Rodin User and Developer Workshop, Jul 2009, Southampton, United Kingdom. 2009. <inria-00431246>

HAL Id: inria-00431246

<https://hal.inria.fr/inria-00431246>

Submitted on 11 Nov 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Rodin plugin for quantitative timed models*

Joris Rehm

joris.rehm@loria.fr - LORIA - Nancy Université

We propose to develop a Rodin¹ plug-in that experiments a systematic use of a refinement pattern. The goal of this pattern is to help in modeling of timed system in Event-B. By timed system we mean system with quantitative temporal constraints and properties.

The user (of the plug-in) will see and modify an Event-B machine augmented with a new operator S . This unary operator over an event name e gives the delay elapsed since the latest triggering of the event e . By using this operator in the invariant, the user will be able to write and prove temporal properties over the events. By using this operator in the guard of an event f , the user will be able to specify temporal constraints over the duration $S(e)$ (which becomes here the delay between the event e and f). For the possible constraints, we plan to consider lower time bounds ($l \leq S(e)$ in guard of f) and upper time bound ($S(e) \leq u$ in guard of f). The upper time bound is a bound within the event f must obligatory occurs, it is not just a possibility. In this text, the usages of the operator S is called the annotations, it is an extension of the syntaxe of the B models. The annotations can only appear in invariant or guards.

Our pattern is an Event-B model that encodes the behaviour of the operator S (we call this model the pattern model). The annotations given by the user define how to refine the pattern model and how to obtain the behaviour of S needed for a particular augmented model. The goal of our plug-in is to generate a normal Event-B model that is the studied (augmented) model where the annotations are replaced with the superposition of the refined pattern model.

To explain briefly what is the idea of the pattern, we show below an augmented model (on the left) and the generated result (on the right). This small model define a light that can be on ($lo = TRUE$) or off ($lo = FALSE$); the light can be switch on by a button (the event on) and goes automatically off (event off) after a delay between $c - d$ and $c + d$. You can see the temporal annotations in the elements named lb_off (Lower Bounds) and up_off (Upper Bounds). The S operator can also appear in the invariant, for example we can write $c + d < S(on) \Rightarrow lo = FALSE$.

EVENTS on $\hat{=}$ Begin act1: $lo := TRUE$ End	EVENTS on $\hat{=}$ Begin act1: $lo := TRUE$ act2: $s_on := 0$ End
---	---

*This work was supported by grant No. ANR-06-SETI-015-03 awarded by the Agence Nationale de la Recherche.

¹<http://www.event-b.org>

<pre> off ≐ When grd1: lo = TRUE lb_off: c - d ≤ S(on) ub_off: S(on) ≤ c + d Then act1: lo := FALSE End </pre>	<pre> off ≐ When grd1: lo = TRUE lb_off: c - d ≤ s_on Then act1: lo := FALSE End tic ≐ Any s Where grd1: 0 < s ub_off: lo = TRUE ⇒ s_on + s ≤ c + d Then act1: s_on := s_on + s End </pre>
---	---

To generate the model the plug-in will have to: declare a new variable s_e for all events which appears in the operator S ; reset this variable s_e to zero in the event e ; replace all $S(e)$ by s_e for all e ; generate a tic event that increments the s_e clocks (for all e) and add in the guard of tic the predicate : $GUARD(f) \Rightarrow s_e + s \leq u$ for all upper bound $S(e) \leq u$ in a event f , with $GUARD(f)$ being the guard of the event e (without temporal annotations).

For the user interface, we plan to specialize the standard “edit” editor of Rodin. The user will be able to add his annotations and when the machine is saved, the plugin will generate a normal machine. The proof will be done over the proof obligations of this generated machine.

For more information about our pattern see [3], another example of pattern for time can be see in [2]. Our notion of (refinement) pattern is the same than the Action/Reaction pattern (see the chapter 3 of [1]). As related work, we can also cite the works about automatic refinement with tools like Bart². There are some common elements between this tool and our work, the major difference is that Bart transforms a B machine to a B machine (more close to an implementation), where our augmented model is not a normal B machine.

References

- [1] Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2009.
- [2] Dominique Cansell, Dominique Méry, and Joris Rehm. Time constraint patterns for event B development. In *B 2007: Formal Specification and Development in B*, volume 4355/2006, pages 140–154. Springer, January 17-19 2007.
- [3] Joris Rehm. Pattern Based Integration of Time applied to the 2-Slots Simpson Algorithm. In *Integration of Model-based Formal Methods and Tools - IM.FMT'2009 - in IFM'2009*, Düsseldorf Allemagne, 02 2009.

²http://www.tools.clearsy.com/index.php5?title=BART_Project