Towards Modular Development in Event-B

Thai Son Hoang[1], Hironobu Kuruma[2], and Michael Butler[1]

[1] ECS, University of Southamtpon, U.K.
[2] Research and Development Group, Hitachi Ltd., Japan

Background. Event-B [2] developments are mostly structured around *refinement* and *decomposition* relationships [3]. This top-down development architecture enables system details to be gradually introduced into the formal model. More often, this result in large model with monolithic structures.

Motivation. Various composition approaches have been proposed [6, 4, 7, 5]. This proposal is inspired by these approaches and development methods such as classical-B [1], working towards modular development in Event-B.

Machine inclusion. Our first concept is *machine inclusions*. Machine A that includes machine B inherits B's variables and invariants. Variables of B cannot be modified directly, but only through event's synchronisation [7]. Multiple instance of the included machine can be achieved via prefixing, similar to [5]. The syntactical "flatten" of A can be seen in Fig. 1.

```
machine B
variables y
invariants J(y)
events
 event f
  any u where
   G_B(y, u)
  then
   BAP_B(y, u, y')
  end
```

```
machine A
includes p_B
variables x
invariants I(x, p_y)
events
 event e
  synchronises p_f
  any t where
   G_A(x, t)
   H_AB(x, p_y, t, p_u)
  then
   BAP_A(x, t, x')
  end
```

```
machine (flatten_)A
variables x, p_y
invariants
 I(x, p_y)
 J(y)
events
 event e
  any t, p_u where
   G_A(x, t)
   H_AB(x, p_y, t, p_u)
   G_B(y, u)
  then
   BAP_A(x, t, x')
   BAP_B(p_y, p_u, p_y')
  end
```

Fig. 1: Machine inclusion

Refinement-chain inclusion. While machine-inclusion mechanism gives us a direct reuse of machine, it is often the case that we want to reuse a refinement-chain. Approaches such as [4, 8] allow to incorporate refinement chains in to the development. However, these approaches involve generating of model which is often cumbersome to accomodate changes. We propose to to include the refinement-chain inside the machine itself. Fig. 2 illustrates a situation where A includes a refinement chain from B_1 to B_m. Semantically, A has double "in-

```
machine A
includes B1 ⟶ Bm
. . .
 event e
  synchronises f
. . .
```

Fig. 2: Refinement-chain inclusion

terfaces". To its abstract machine, it acts as a machine with an inclusion of B_1. To its concrete machine, it acts as a machine with an inclusion of B_m. Since the refinement of B_1 by B_m has been proved separately, the refinement of $(\mathsf{A} + \mathsf{B}_1)$ by $(\mathsf{A} + \mathsf{B}_\mathsf{m})$ is almost correct-by-construction. The extra manual work required is the refinement of the extra guard, e.g., H_{AB} in Fig. 1.

Some evaluation. We have applied the idea (manually) to several examples. The result is quite encouraging with reduction of the modelling and proving efforts. In particular, this approach can be used several times, e.g., to have a hierarchy of inclusions. The resulting models also easier to comprehend and in particular, it should incorporate with changes to the imported model without any additional effort.

Conclusion. The concept here is not new and incorporate many existing work. We consider this as an effective way to have modular development in Event-B which will reduce the modelling and proving efforts. We are investigating how to extend Rodin to support the approach in the most efficient way.

References

1. J-R. Abrial. *The B-book: Assigning Programs to Meanings.* Cambridge University Press, 1996.
2. J-R. Abrial. *Modeling in Event-B: System and Software Engineering.* Cambridge University Press, 2010.
3. J-R. Abrial and S. Hallerstede. Refinement, decomposition, and instantiation of discrete models: Application to Event-B. *Fundam. Inform.*, 77(1-2):1–28, 2007.
4. Thai Son Hoang, Andreas Fürst, and Jean-Raymond Abrial. Event-B patterns and their tool support. *Software and Systems Modelling*, 12(2):229–244, May 2013. http://dx.doi.org/10.1007/s10270-010-0183-7.

5. Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Dubravka Ilic, and Timo Latvala. Supporting reuse in event B development: Modularisation approach. In Marc Frappier, Uwe Glässer, Sarfraz Khurshid, Régine Laleau, and Steve Reeves, editors, *Abstract State Machines, Alloy, B and Z, Second International Conference, ABZ 2010, Orford, QC, Canada, February 22-25, 2010. Proceedings*, volume 5977 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2010.
6. Michael Poppleton. The composition of event-b models. In Egon Börger, Michael J. Butler, Jonathan P. Bowen, and Paul Boca, editors, *Abstract State Machines, B and Z, First International Conference, ABZ 2008, London, UK, September 16-18, 2008. Proceedings*, volume 5238 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2008.
7. R. Silva and M. Butler. Parallel composition using Event-B. `http://wiki.event-b.org/index.php/Parallel_Composition_using_Event-B`, 2009.
8. Renato Silva and Michael J. Butler. Supporting reuse of event-b developments through generic instantiation. In Karin Breitman and Ana Cavalcanti, editors, *Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings*, volume 5885 of *Lecture Notes in Computer Science*, pages 466–484. Springer, 2009.