A Framework for Developing Distributed Protocols with Event-B/Rodin

Paulius Stankaitis[1], Alexei Iliasov[1], Tsutomu Kobayashi[2], Alexander Romanovsky[1], Fuyuki Ishikawa[2]

[1] Newcastle University, Newcastle upon Tyne, United Kingdom
{p.stankaitis|alexei.iliasov|alexander.romanovsky}@ncl.ac.uk

[2] National Institute of Informatics, Tokyo, Japan
{t-kobayashi,f-ishikawa@nii.ac.jp}@nii.ac.jp

Abstract. Model based notations such as Event-B are understood to be well suited for formalisation and verification of various protocols. In general the design of a protocol in Event-B follows a number of principles. To make efficient use of refinement as abstraction and proof structuring techniques it common to use Dijkstra backward elicitation style where an abstract model summarises protocol effect and following refinement steps gradually introduce preceding steps. A model must follow certain design principles in order to be considered an adequate rendering of a protocol. This requires a faithful model of communication, a convincing argument of localisation and, finally, a formal decomposition of a model into independent communicating parts. From our experience developing a protocol requires a often remake of the model which makes the process rather challenging and tedious. Furthermore effectively utilizing the refinement of the Event-B method is a challenge which often results in a poor readability/maintainability of the model (verification automation can be reduced too). We propose to design a toolkit to generate protocol models from a high-level description. Such a toolkit would take a semi-formal, graphical definition of a protocol and automatically generate Event-B refinement chain.

1 Introduction

Problem. From our experience developing a protocol requires a often remake of the model which makes the process rather challenging and tedious. Furthermore effectively utilizing the refinement of the Event-B method is a challenge which often results in a poor readability/maintainability of the model (verification automation can be reduced too).

Proposed Solution. We propose to design a toolkit to generate protocol models from a high-level description. Such a toolkit would take a semi-formal, graphical definition of a protocol and automatically generate Event-B refinement chain.

Actions Required.

– Need to decide on the class of algorithms we want to verify (i.e. distributed-consensus, distributed resource allocation).
– Need to decide on a high-level specification language.

2 Background

3 Framework Requirements

REQ1 The toolkit should be generic enough to cover most forms of communication and consensus protocols;
REQ2 The toolkit should integrate with Rodin;
REQ3 Refinement chain must be constructed automatically from a high-level description;
REQ4 A user should be able to define message types, message payload (fields);
REQ5 A user should be able to define behaviour logic driving message generation, specifically:
  REQ5.a use of Event-B mathematical language to define predicates and expressions;
  REQ5.b **if/then/else** statement to define branching (alternatively, a graphical notation can be employed to capture branching).
REQ6 There should be a facility to express what each protocol party knows at different stages. (Note that this is separate from model of a party state: state is an underlying, implementation concern). This is the basis for expressing verification goals and auxiliary invariants. It is not clear whether there should be explicit definition of a model separately or it can be inferred.
REQ7 Communication mechanism should be implicit however there should be a way to define communication properties: message loss, out of order, duplicates;
REQ8 The protocol should be able to generate Event-B model and refinement steps completely automatically. It is not clear whether a user would be expected to manually change the model afterwards or all annotations can be provided in the toolkit.
REQ9 Verification is to happen at the level Event-B model in Rodin.

4 Utilizing Refinement for Distributed Protocol Class

– Refactoring Refinement Structure of Event-B Machines [3];
– Formal Derivation of a Distributed Program in Event B [2];
– Systematic Translation Rules from astd to Event-B [4];
– An Event-B Development Process for the Distributed BIP Framework [5];

5 Proof Rules for Distributed Protocol Class

– Use Case Scenarios as Verification Conditions: Event-B/Flow Approach; [1]

6 Case Study?

7 Discussions and Conclusions

References

1. Alexei Iliasov. Use case scenarios as verification conditions: Event-b/flow approach. In Elena A. Troubitsyna, editor, *Software Engineering for Resilient Systems*, pages 9–23, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
2. Alexei Iliasov, Linas Laibinis, Elena Troubitsyna, and Alexander Romanovsky. Formal derivation of a distributed program in event b. In *Proceedings of the 13th International Conference on Formal Methods and Software Engineering*, ICFEM'11, pages 420–436, Berlin, Heidelberg, 2011. Springer-Verlag.
3. Tsutomu Kobayashi, Fuyuki Ishikawa, and Shinichi Honiden. Refactoring refinement structure of event-b machines. In John Fitzgerald, Constance Heitmeyer, Stefania Gnesi, and Anna Philippou, editors, *FM 2016: Formal Methods*, pages 444–459, Cham, 2016. Springer International Publishing.
4. Jérémy Milhau, Marc Frappier, Frédéric Gervais, and Régine Laleau. Systematic translation rules from astd to event-b. In Dominique Méry and Stephan Merz, editors, *Integrated Formal Methods*, pages 245–259, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
5. Badr Siala, Mohamed Tahar Bhiri, Jean-Paul Bodeveix, and Mamoun Filali. An event-b development process for the distributed bip framework. In Kazuhiro Ogata, Mark Lawford, and Shaoying Liu, editors, *Formal Methods and Software Engineering*, pages 313–328, Cham, 2016. Springer International Publishing.