Event driven B: methodology, language, tool support, and experiments

1 author:

Thierry Lecomte
ClearSy System Engineering
54 PUBLICATIONS 418 CITATIONS

SEE PROFILE

Event Driven B : methodology, language, tool support and experiments

Thierry Lecomte

ClearSy
1330, avenue .R. Guilibert Gauthier de la Lauzière
13856 Aix En Provence Cedex 3 France
Thierry.lecomte@clearsy.com

This article presents work related to event driven B that has been undertaken by ClearSy, in close collaboration with J.R. Abrial and Louis Mussat (DCSSI). The main objective has been to verify suitability of B for system modelling. Most of these activities have been funded by MATISSE project (IST-1999-11435).

Introduction

The B-method ([1]) has been successfully applied to software specification and development. A new approach, using at least the same language but with a different point of view, has been designed, in order to address complex system modelling.

The reader may find below a short description of this approach ([4]), the language extensions ([10]) required by this approach, a description of the related tools ([9]) and some references to industrial experiences.

Event driven approach for system studies

The system studies represent all the reflexions and decisions situated before the drawing up of the specification documents of the individual parts of a complete system. They are mainly aimed at elaborating the architectural decomposition of a system and are sometimes “validated” by means of a number of simulations i.e. executions of behavioural models.

Our thesis ([1], [3]) is that these system studies can be greatly enhanced by using the main ingredients that have proved to be quite successful in software specification and design: refinement techniques, decomposition and mechanised proofs.

The main idea of that approach consists in studying our future system by means of one (or better several) model, on which we intend to apply certain correctness criteria. Rather than analysing the results of a number of simulation sessions and conclude that they confirm the selected criteria, we are going to prove directly on the model that it is indeed the case.

For building the model of a future system, the idea is to place oneself mentally above the system in question and try to imagine what we could observe from there. These observations are essentially made by means of two kind of notions: state and events. The latter constitutes what we are able to observe. The properties of the state are the safety properties, whereas the properties of the events are the liveness properties. Proofs might be performed to validate these two kinds of properties.

One you have made some observations and corresponding proofs at a certain level “above” the system, you might envisage to step down a little and thus be able to observe other interesting things (a more precise state and more events), which you were not able to distinguish before because you were too high in the sky. From this

level, you might step down one more time, and so on: this is the so-called parachute paradigm.

In abstract system model, we deal with perfect knowledge of our system. By going down, we have to take into account reality by introducing sensors, actuators and communications (including delays). That way, the “reality”, which is represented in the memory, belongs to a certain past. We are at the heart of system studies. At each step of our overall construction, we have to prove that the control and communication parts in construction ensure the preservation of the laws, only elaborated in terms of the physical state at the very early stages of the development.

Once the model is quite large and difficult to develop further, it is time to envisage to decomposing the model into several sub-models. Once separated from the main body, a sub-model can be developed further from the rest of the system. Those sub models could be software or hardware based, or represent communication channels. Each sub model contains events that may occur within one sub-model. For software based models, a scheduler should be added before going to code, by traditional means or by completing a B software development.

A general algorithm for system studies can be found in [4].

Language extensions

B was initially designed for software development and has been successfully used in many occasions. With the increasing complexity of distributed systems and related industrial needs for validation, the refinement and proof based B method has been adapted to system modelling, not by changing the underlying language at first, but by modifying the way to model (event based approach of closed systems). B can be used, as it is, for system modelling, but some specific features need to be inserted in B models by hand.

As expressed in [3], some language extensions are required to be able to express event-based properties. After some experiments, and to keep compatibility with the tool implementing the B-Method, Atelier B, we decided to extend the B language by adding new features that can be easily translated in “sequential B”, using syntactic sugars.

The new features are: Post conditions, Modalities, Event based syntax and Explicit refinement. The reader will find a complete description of these extensions in [10].

Tool Support

Two kind of applications are being developed. The first one is a translator from event-driven B to “sequential B”, replacing extended features by traditional B specification. A complete description of the tool, coming along with complete examples, can be found in [7].

The second tool that is being developed is able to transform a set of events into a piece of code, by aggregating events. Thus it is able to generate equivalent algorithms. Aggregation rules have been defined ([11]) and are applied, either automatically or interactively, on an event set.

Industrial Experiments

Many examples have been used to tuned both methodology and tools. Most of them are available. In [5], basic algorithms are presented. In [6], telecom problems are treated: simple transmission protocol, leader election on a ring-shaped network, distributed mutual exclusion, termination detection, distributed routing algorithm for mobile agents.

The main applications of event B, at ClearSy, are in Specification expertise and Reverse engineering.

Our most heavy experiment in specification expertise is related to car modelling, representing tens of man years experience. The objective was to be able to produce unambiguous specification of removable components in a car, in order to ease diagnosis and locate defects. 40 functions have been modelled for each car, representing about 63 000 lines of B model per car.

Concerning reverse engineering, event based approach has been used for retrieving initial specification of a real-time software, whose specification was at that moment in the form of a 40 page ADA pseudo code like description. Then, from the abstract specification, an event based development was performed, leading to a fine description of the system. A simple scheduler was then developed to organise software events to obtain final software.

Conclusion

Event driven B approach seems promising as it offers services that would be used for system modelling and validation. Application domains are various, ranging from requirements document expertise to reverse engineering of systems. Language extensions and tools are available, or will be available soon.

Hardware system design is being studied within the framework of the PUSSEE project (IST).

Most resources listed in this article are available for download either on Matisse web site (www.qinetiq.com/matisse) or on Atelier B web site (www.atelierb.societe.com).

References

[1] JR Abrial . *B-Book: Assigning Programs to Meanings)* Cambridge University Press (1996).

[2] JR Abrial . *Extending B Without Changing it (for Developing Distributed Systems)* in Habrias, editor, *1st Conference on the B-Method*. November 1996

[3] JR Abrial and L. Mussat. *Introducing Dynamic Constraints in B.* In D. Bert editor, *B'98: Recent Advances in the Development and Uses of the B-Method*. LNCS vol 1393. 1998

[4] JR Abrial . *Guidelines to Formal System Studies* – November 2000

[5] JR Abrial . *Event Driven Sequential Program Construction* – August 2001

[6] JR Abrial . *Event Driven Distributed Program Construction* – August 2001

[7] JR Abrial . *Formal Construction of Proved Circuits* – August 2001

[8] JR Abrial . *Event Model Decomposition* – (To appear)

[9] ClearSy . *Evt2b translator* – September 2001

[10] ClearSy . *Event B Reference Manual* – June 2001

[11] ClearSy . *Evt2b0 translator* – (To appear)