On B and Event-B: Principles, Success and Challenges

Jean-Raymond Abrial(✉)

Marseille, France
jrabrial@neuf.fr

After more than 20 years since the publication of the book on B [1], and almost 10 years since the publication of the book on Event-B [2], the purpose of this short paper is to present some key points of these technologies. Note that this presentation might be incomplete as I am certainly not aware of many developments which have taken place around B and Event-B in recent years.

This paper is organised around four topics. Section 1 is devoted to the listing of the basic principles on which B and Event-B have been developed. Section 2 is devoted to differences and similarities between B and Event-B. Section 3 explains where B and Event-B are spread around the world. Finally, Sect. 4 contains various issues and challenges encountered by these technologies.

1 Basic Principles

In this section, I study the main principles forming the basis of B and Event-B, i.e. developing systems which are intended to be correct by construction, using classical logic and mathematical notations, and developing various tools. Note that many information about B can be found in this website [3], whereas many information about Event-B can be found in this website [4].

1.1 Being Correct by Construction

The main purpose of B and Event-B is to help engineers developing systems that will be *correct by construction*. It means that B and Event-B are not programming languages of any kind. They are modelling systems. In case of B, modelling and developing software systems and in case of Event-B, modelling and developing global complex systems, involving not only software, but also physical environments and even human users.

1.2 Using Refinement

In order to achieve gradually a correct by construction approach, it is fundamental to handle refinements. This means that a development is made of a series of steps starting at a very abstract level and aiming at a final concrete one. This is simply incorporating in B and Event-B a classical approach used in many other engineering disciplines.

M. Butler et al. (Eds.): ABZ 2018, LNCS 10817, pp. 31–35, 2018.
https://doi.org/10.1007/978-3-319-91271-4_3

1.3 Mathematical Notation

The correct by construction approach together with the usage of refinement imply that each constructing steps be guaranteed by some theorems to be successfully proved. The goal of such theorems is to ensure that each step is valid and does not depart from the previous one. For doing so, it is important that statements of these theorems be expressed using a very classical mathematical notation, i.e. that of predicate calculus and of typed set theory.

1.4 Tools for Developing Models

It was necessary to develop tools for analysing and checking models for B (Atelier B, developed and maintained by the software house Clearsy) and for Event-B (Rodin, developed and maintained by the software house Systerel). As a matter of fact, a pen and paper approach is not possible any more as systems are becoming very complex these days.

1.5 Tools for Generating Theorems

It is out of the question that human users of B and Event-B generate directly mathematical statements to be proved at each steps of a development. This is far too much error prone to leave this in the hands of human users. An important tool called the *Proof Obligation Generator* has thus been developed for that purpose for B (in Atelier B) and for Event-B (in Rodin). Such a tool has been strongly influenced by what had been developed before for VDM [5].

1.6 Tools for Proving

Once some mathematical statements have been generated by the Proof Obligation Generator, it is important to be able to prove them in a mechanical way. For doing this, some proving tools have been constructed for B (in Atelier B) and for Event-B (in Rodin). With such tools, both automatic and interactive proofs can be performed.

1.7 More Tools

Other tools were developed in Universities (Southampton, Duesseldorf, Turku) and Industries (Siemens Transport, Clearsy, Systerel). With such tools, it is possible to perform model checking, automatic refinement, model decomposition and structuring, data validation and various translations to classical programming languages, etc.

2 Comparing B and Event-B

The book on B [1] was first published in 1996, whereas that on Event-B [2]was published in 2010. Consequently, Event-B had been able to take advantage of issues encountered in the usage of B. Also note that Event-B has been strongly influenced by the development of *Action systems* [6]. Event-B contains some simplifications over B. This has allowed us to extend the usage of Event-B to the modelling of systems that are not restricted to software. In what follows, I explain differences and similarities between B and Event-B.

2.1 Differences

One of the main differences between B and Event-B concerns operations (in B) and events (in Event-B). Each operation in B is usually defined together with a *pre-condition* containing a predicate that must be true for the operation to be able to be *called*. On the other hand, each event in Event-B is usually defined together with a *guard* containing a predicate that must be true for the event to be able to *occur*.

This results in having both pre-conditions or guards being assumptions when doing a proof on an operation or on an event (e.g. invariant preservation proofs). So far thus, there are no differences between the two. However, both differs strongly when dealing with refinement: pre-conditions can be weakened only, whereas guards can be strengthened only. This possibility of guard strengthening is particularly important as it allows users to build models starting from very abstract cases down to more realistic ones.

In fact, proof obligations are far simpler for events than for operations. In the case of operations one has always two rules: one for pre-conditions and one for post-conditions. In the case of event, one has always a single rule.

Another important distinction between the two concerns parameters. In the case of an operation, such parameters cannot be refined, whereas event parameters can be freely refined (removed, added or modified).

Basic sets, constants and their properties are handled differently in B and Event-B. In B, basic sets, constants and their properties are defined in the *abstract machine* where operations are defined. In Event-B, sets, constants and their properties are defined in separate structures called *contexts*. This gives users more flexibilities in Event-B than in B.

Event-B does not contain any programming constructs such as conditionals, choices, sequencings or loops as B does. This greatly simplifies proof obligations generated for Event-B with comparison to those generated for B. This simplification is particularly important in the case of sequencing. In fact, all such constructs can be handled in Event-B by using events only. Of course, code generation is simpler in B because of the presence of such constructs. To do the same in Event-B one has to apply some specific rules.

2.2 Similarities

Both B and Event-B use the mathematical notation of predicate calculus and typed set theory. This means that proof obligations stated for both approaches can be handled by similar provers. In fact, the prover of Atelier B (called PP) is used successfully in the Rodin toolset. Other external provers (e.g. SMT provers) are used in both Atelier B and Rodin.

In both cases, there is a notion of *machine* acting as an encapsulation for operations (in B) or events (in Event-B). However there is an important distinction between the two. The list of input-output definitions of the operations of a B machine (its *signature*) is fixed once and for all, although it is not the case for the events of an Event-B machine. Notice that events have no output parameters. But there are more: such a list of events can be further extended with new events in a refinement. This very important feature has been borrowed from Action Systems [6]. It allows users to be very flexible in the gradual construction (with refinement) of an event system.

In fact such similarities allows one to use Event-B within the Atelier B tool which is used for B. Some proof obligations which are specific to Event-B have been developed for that purpose in B.

3 Spreading

3.1 Spreading in Industry

B is extensively developed in Industry by the software house Clearsy, claiming that more than 30% of its business is devoted to B. A very rich and well documented article [7] presents the development of B at Clearsy. The main activity is with train systems in many places around the world: North and South America, Europe, Asia. Some information about the industrial use of B can be found in the Clearsy web site [3].

3.2 Spreading in Academia

Event-B (more than B) is widely spread in Universities in Europe (France, United Kingdom, Finland, Germany, Spain), in America (Canada, Brazil, Columbia), in Asia (China, Japan), etc.

4 Challenges

One of the main challenge of these technologies is the poor spreading of B in Industry: as said in Sect. 3.1, B is essentially used in train systems. There are clearly other industries where such a formal modelling approach could be used successfully, e.g. energy, automotive, aeronautics, space, etc. However, people in charge there claim that it is not possible, essentially because it is too difficult to modify engineering approaches which have been established for many years. The paper cited above [7] makes a very fine analysis of these difficulties.

There has been no development where both Event-B and B are used one after the other. First Event-B for the system analysis, then B for the software development. I think it could be quite possible. However, most of the time, system engineers and software engineers have different cultures.

References

1. Abrial, J.-R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press, Cambridge (1996)
2. Abrial, J.-R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge (2010)
3. Clearsy. www.clearsy.com/en
4. Southampton University. www.Event-B.org
5. Jones, C.B.: Systematic Software Development Using VDM, 2nd edn. Prentice Hall, Upper Saddle River (1986)
6. Back, R.-J., Kurki-Suonio, R.: Decentralization of process nets with centralized control. Distrib. Comput. **3**(2), 73–87 (1989)
7. Lecomte, T., Deharbe, D., Prun, E., Mottin, E.: Applying a formal method in industry: a 25-year trajectory. In: Cavalheiro, S., Fiadeiro, J. (eds.) SBMF 2017. LNCS, vol. 10623, pp. 70–87. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70848-5_6