Challenges in Applying Formal Methods
An SME View

Mathieu Clabaut

Systerel, Aix-en-Provence, France
mathieu.clabaut@systerel.fr

Abstract. This paper outlines past and foreseen challenges in applying both *classical B* and *event B* to design safety related systems in an SME.

1 Activities

Systerel is an SME doing mainly fixed-price activities in the domain of real-time systems and safety-critical software ranging from on-board train speed controller to track-side automatic train protection. Systerel uses formal methods for some of these developments, most of which have been done with *classical B*.

Nowadays, we are doing some analysis and design of safety-critical systems with the help of *event B* while still developing safety-critical real-time embedded software.

2 Past Challenges: Classical B

2.1 At the Beginning

The earlier challenges met when first using the *classical B* method were mainly the following ones:

Planning proof workload was impossible to estimate, which was not in favour of formal method, since the workload of tests used in classical methods was far more easy to estimate.

Monitoring progress was very difficult. How to estimate the proof progress when a single unprovable proof obligation may require a break down of the whole architecture of your software?

Customer evaluation of formal method contribution was not always positive, notably with respect to the development costs: *Where is the return on investment?*

2.2 Now

Planning and monitoring is a bit easier as we have gained experience and domain knowledge. We thus do know where to apply formal methods and where not to.

We are now able to precisely draw the limits of software responsibility with respect to safety and then narrow down the formal properties which are to be modelled with B.

We also have collated somewhat usable metrics and rules of thumbs, and while still having to work until there is no more proof obligations left, we have a better confidence on our modelling principles for a given domain.

We claim better model designs which allow for simpler proofs and better reuse. The fact that a *model is designed to be proved* is now the core of our process.

Customer evaluation is now backed by some years of experience and nowadays, most of our customers reckon quality gains (albeit still internally disputed for some others). For those who master the formal process, using *classical B* is *less expensive than traditional safety-critical developments*.

2.3 Process

Our *classical B* process is composed of:

- Software requirements document, written by *small teams*.
- Formal design and proof done within *small teams*.
- Formal coding and proof done within *large teams*.
- Translation and compilation (automated).
- Integration and functional testing done within *large teams*.

This process is suitable for big industrial software with large teams of developers.

3 Today and Future Challenges: Event B

Nowadays, we are going a step further with the use of formal methods, by using *event B* to help the design of safety-critical systems.

3.1 Process

The starting point of our process, backed by our *classical B* experience, is that *the model has to be designed for proving it* which also implies that *event B* may not be convincing for analysis of existing systems (so called retro-modelling)[1] and may only show its usefulness for ab initio design, where one is allowed to tweak the design in order to *reject complexity* (with respect to safety proof).

The process in use today is made of the following steps:

- System requirement document (*small team*)
- Refinement plan (*small team*)

[1] But still more flexible than *classical B* where architectural constraints make retro-design in B even more difficult.

– Modelling / proving (*small team — basically, one person*)
– Testing (*To be defined...*)

The scalability of such a process is poor. New tools and concepts will definitely be needed to manage complexity and to allow team work on big models.

It is to be noted that despite these difficulties, *event B* proves to be very useful in designing a safety system.

3.2 Project Management

Our main challenge with the use of *event B* and the companion tool *Rodin* are about project management and convey the fact that:

– the desired system design is not known at the beginning of the process,
– heavy model refactoring are thus common.

Planning and monitoring is then a difficult task. How one can estimate the design and proof effort? How can one devise measurement to monitor design and proof progress?

Relation with customers on such a base are also not easy: reporting is no convincing given the lack of monitoring capabilities. The customer evaluation is then done on a poor basis: *"a model has been done, and then? ..."*

The ROI justification of the modelling work is still difficult, even if we are utterly convinced of its usefulness.

4 What's Needed

Roughly stated, the main needs are the following one:

1. Master the modelling process.
2. Reduce costs and delays.
3. Convince customers.

For this, we definitely need tools and methods for tackling model complexity (team based development, decomposition, mathematical extensions, patterns or automated refinements,...), for improving planning and monitoring and for improving customer evaluation and appreciation.

Some interest was also expressed by the space industry in being able to formalize requirements and the companion engineering process with the help of refinement-based formal methods. Maybe an issue to be tackled in the coming years?