# Reflections on the Teaching of System Modelling and Design

Ken Robinson
School of Computer Science & Engineering
The University of New South Wales
Sydney NSW Australia

Presented at AVOCS'10
Rodin User and Developer Workshop
September 21 2010

# Overview

This talk is going to reflect on some aspects of my use of Event-B and Rodin in teaching undergraduate software engineering students at UNSW (Sydney Australia).

I have also about half completed a book entitled *System Modelling and Design* that presents much of the material that I use in that teaching. The book may be mentioned in the talk and in any case I will say more at the end of the talk.

# Software Engineering

My concern is with *Software Engineering* and in order to take that professional classification seriously I am promoting the following:

Software Engineers should aspire to *fault-free software*

If they can't, or if the so-called profession can't, then it's about time we dropped what would be the false concept *software engineering*.

No other engineering profession would deny a corresponding aspiration.

It's not an argument that is easy to win.

The last time I mentioned the concept at a teaching committee meeting in our school, I got the retort (from an academic member):

you might as well believe in the Tooth Fairy

# What is Software Engineering and what does it need?

I don't treat Event-B as a *Formal Method*, a term I try not to use.

Software engineers need design methods that mimic those in other engineering disciplines.

Importantly they need to use *quantitative* methods for *verifying* that their designs satisfy their own design objectives

In SE that role is given to logic with a key parallel role to theorem provers.

# Formal Methods

Interestingly, I notice that some of those who teach formal methods or something like it, often use difficult or tricky problems to justify or motivate the use of the formal methods.

I have a much more pedestrian, or mundane view than that. Most of my examples are constructed from simple, mundane examples. I think they more clearly provide focus on the Event-B modelling for Software Engineering (SE).

Does anyone believe that software projects fail because of difficult or tricky aspects of the project?

I seriously doubt it.

Of course, it is easy to imagine that there are, perhaps, some software designers who make projects difficult and have no way of alerting themselves to the faults in their designs. 

# Behaviour

What is of prime interest in software design is *behaviour* and it seems to me that Event-B allows us to model behaviour very closely, without distractions that are present in many other methods, including Classical-B.

Also we don't need to be able translate between the specification formalism and the behaviour we require, as we need to do with many other methods.

# Where is Event-B taught/used?

Let me tell you a little bit about the classes I teach:

System Modelling and Design: learning about producing and verifying system models using Event-B. This year about 50 students.

Software Engineering workshop: concurrent with the above course, the same students work in teams to produce designs for some chosen system.

Following Software Engineering workshop: the students take their models and implement a prototype. This is done by mapping the Event-B model into a class diagram.

Current system: T-Card: a general transport card system.

# The role of animation

It is easy to confuse the roles of rigorous verification (proof) and animation:

- those who think that they both do the same job and animation is easier, (the poor man's prover)
- those who thing that proof is the stronger and therefore animation is not required

Both are wrong.

| | |
|---|---|
| Proof | can show that your model is internally consistent, but not that the model is consistent with requirements, and |
| animation | can help verify that the model behaviour is consistent with the required behaviour, but not, of course, that the model is internally consistent. |

Both are essential tools and we use both.

It is very important to get students to understand the differences: animation looks very attractive!

# Difficulties

The students:

**lack of experience with the relational calculus:** (our) students have a lack of experience with the using relations. interestingly logic does not seem to be so much of a problem;

**inadequate use of invariants:** it is difficult to get students to use invariants:] getting students to use them to express anything beyond variable typing;

**factoring behaviour:** this is generally difficult.

Rodin:

**Strange problems:** Bugs? Don't encourage confidence. At times we have fought against very strange problems.

**Prover performance** Leads to conclusion (probably correct) that Rodin is still immature.

# Student Acceptance

There is some acceptance by some students,

for example, in the project work there is some recognition that the Event-B guards and actions gave them valuable cluse to the code required in the implementation.

Some feeling that Event-B might be necessary for safety critical applications, but unnecessary otherwise.

Probably some feeling that they can do just as well with *test and fix*.

# An interesting question

Is there an Event-B model of Rodin?

If the answer is *No* then this is a difficult question to answer.

My general answer is that Rodin is a tool that is used by many people on essentially similar tasks. Over time it will become very reliable. On the other hand their applicaiton is a one-off.

There is some truth to that answer, if not the *complete truth*.

# Post talk elaboration

At the talk there was some discussion on the preceding slide: some felt that my *general answer* is never correct or satisfactory. It is hard to decide on this issue, but there certainly is a difference between a tool that is going to be used by many different users in different environments and a one-off single application system.

Since the workshop I have talked to Laurent Voisin about some —and more— of the issues raised in the preceding slide. An accompanying short article reports that discussion.

# The Book

*System Modelling and Design* (SMD-KAR.pdf) The book is only about half finished, but may be of help to some people, and any feedback to me would be very gratefully received.

My proposal, if acceptable, is to install it somewhere on the Rodin/Event-B website.

The book is developed as a book that can be read within a PDF viewer.

The book is heavily linked to enable reasonably easy navigation.

*Notation* and *Concepts* are highlighted in the order in which they are encountered in the text of the book. They are also linked in the index.

The book contains the 4-page *Event-B Concise Summary*

The book is now available on the EventB wiki.

# Thank You

Questions?