Reflections on the teaching of Formal Methods, Requirements & Software Engineering

Ken Robinson

UNSW Australia

ABZ2014 Conference Workshop June 2014

Reflections on talk

Comment

This talk contains assertions that hopefully will provoke discussion. The intention is to explore and provoke some aspects of how reliable software modelling and implementation might be pursued.

Software Models vs Engineering Models

Software Models vs Engineering Models

Observation

Some academic staff seem to regard many software formal methods as equal

Software Models vs Engineering Models

Observation

Some academic staff seem to regard many software formal methods as equal

Example

The formal methods **Z** *and* **B** *are equivalent to* **Event-B**.

Software Models vs Engineering Models

Observation

Some academic staff seem to regard many software formal methods as equal

Example

The formal methods **Z** *and* **B** *are equivalent to* **Event-B**.

Comment

Not so.

Even though *Z* and *B* were very significant contributions to formal methods and *Event-B* is part of the same family, *Event-B* is radically different to both *Z* and *B*.

Software Models vs Engineering Models

Observation

Some academic staff seem to regard many software formal methods as equal

Example

The formal methods **Z** *and* **B** *are equivalent to* **Event-B**.

Comment

Not so.

Even though *Z* and *B* were very significant contributions to formal methods and *Event-B* is part of the same family, *Event-B* is radically different to both *Z* and *B*.

Engineering models must contain *forward* and *backward* traces to the *requirements*.

Software Models vs Engineering Models

Observation

Some academic staff seem to regard many software formal methods as equal

Example

The formal methods **Z** *and* **B** *are equivalent to* **Event-B**.

Comment

Not so.

Even though *Z* and *B* were very significant contributions to formal methods and *Event-B* is part of the same family, *Event-B* is radically different to both *Z* and *B*.

Engineering models must contain *forward* and *backward* traces to the *requirements*.

Something that neither *Z*, nor *B*, can achieve very effectively.

Software Models vs Engineering Models

Observation

Some academic staff seem to regard many software formal methods as equal

Example

The formal methods **Z** *and* **B** *are equivalent to* **Event-B**.

Comment

Not so.

Even though *Z* and *B* were very significant contributions to formal methods and *Event-B* is part of the same family, *Event-B* is radically different to both *Z* and *B*.

Engineering models must contain *forward* and *backward* traces to the *requirements*.

Something that neither *Z*, nor *B*, can achieve very effectively.

Event-B provides the capability of bidirectional tracing of requirements to be incorporated in the model.

Is *Software Engineering*, Engineering?

The birth of the term *Software Engineering*

The term was coined at a NATO conference called in 1968 to discuss the *software crisis*.

Where are we?

Are we any closer to realising the objectives of that event?

Is *Software Engineering*, Engineering?

The birth of the term *Software Engineering*

The term was coined at a NATO conference called in 1968 to discuss the *software crisis*.

Where are we?

Are we any closer to realising the objectives of that event?

If not, why not?

Is *Software Engineering*, Engineering?

The birth of the term *Software Engineering*

The term was coined at a NATO conference called in 1968 to discuss the *software crisis*.

Where are we?

Are we any closer to realising the objectives of that event?

If not, why not?

Last time I looked, something like 85% of large software system projects (multi-million dollars) failed because of failure to satisfy the requirements.

Requirements for Engineering

In general, *engineering* requires:

- Rigorous requirements;
- Diligent implementation of those requirements;
- Legal responsibility for satisfying the contract.

Requirements for Engineering

In general, *engineering* requires:

- Rigorous requirements;
- Diligent implementation of those requirements;
- Legal responsibility for satisfying the contract.

Microsoft can say, *and have done*:
You chose this product, if it doesn't work then it is your fault.

Are there differences between "engineering" disciplines?

Differences from older, conventional engineering disciplines

- Not continuous;

Are there differences between "engineering" disciplines?

Differences from older, conventional engineering disciplines

- Not continuous;
  In general, cannot interpolate or extrapolate behaviour.

Are there differences between "engineering" disciplines?

Differences from older, conventional engineering disciplines

- Not continuous;
  In general, cannot interpolate or extrapolate behaviour.
- Unstable

Are there differences between "engineering" disciplines?

Differences from older, conventional engineering disciplines

- Not continuous;
  In general, cannot interpolate or extrapolate behaviour.
- Unstable
  In general, other engineering implementations can tolerate errors and find a stable configuration.

Are there differences between "engineering" disciplines?

Differences from older, conventional engineering disciplines

- Not continuous;
  In general, cannot interpolate or extrapolate behaviour.
- Unstable
  In general, other engineering implementations can tolerate errors and find a stable configuration.

To those we can add:

- The *implementation* is not visible:
  - With most other engineering disciplines the structure of the implementaton is visible;
  - or at least can be seen in similar terms to the requirements.
  - This is where *Event-B* has important capabilities.

Formal Methods are not completely formal

The idea that formal methods provide absolute precision in implementation because mathematical proof provides certainty is unfortunately not correct.
Unfortunately, even rigorous requirements will be informal to some extent, being written in some form of natural language.
But this puts *Software Engineering* to no greater disadvantage than other engineering disciplines.

Despite evidence to the contrary
Software Engineering is surely possible?

Even if it is 45 years late!

Teaching of Event-B to Software Engineers

This is an addition to the talk given to the workshop.
An outline will be given of the courses given at the *School of Computer Science & Engineering* (CSE) that Peter Ho and I developed and delivered to *Software Engineering* undergraduate students. The courses were designed to give a strong foundation in software system design and implementation. Event-B featured strongly in the requirements gathering and design stages.
The above is in the past tense as neither Peter nor I am now teaching at CSE and the courses have not been maintained, largely due to lack of familiarity of software system methods beyond conventional computer science methods.

Requirements Gathering and Modelling

- **Requirements gathering workshop**
  The first course is concerned with requirements gathering and also the determination of requirements for the system to be designed and implemented in subsequent courses by each student team.
- **System Modelling & Design**
  Event-B course that develops use of Event-B for modelling systems. This course is standalone and is not limited to Software Engineering students.
  The course is based on the book: *System Modelling & Design* a book I am writing and nearing completion. I am happy to send a copy to anyone who is interested.

Developing Event-B Model from Informal Requirements

The second workshop develops an Event-B model based on the requirements developed in the preceeding workshop and the course on Event-B Modelling.
This is a quite difficult course partly due to the lack of familarity with Event-B and also lack of experience in rigorous definitions of behaviour (requirements).
At the end of this course each student team develops a scenario for their model, which they then demonstrate using animation.

Implementation Workshop

In the final workshop of this series the teams decide on what programming language they will use to implement their model and then proceed with the implementation.
No Event-B implementation tool is used.
Despite the fact that no rigorous tool is used, *requirements tracing*. in both directions, is required.
The students are encouraged to annotate their implementation with Event-B event references to enable informal tracing back to the formal model.
Despite the difficulty (lack of experience) with the rigorous requirements followed by semi-rigorous implementation many students gain significant experience from the overall procedure.