Formal Methodologies

Build it and they will come

THE UNIVERSITY of York

Chris Kimble
February 2008

Overview

- A review of the theory
- Types of Formal Methods
- Success ...
- ... and lack of success
  - “Myths”
  - Views from Industry
  - Experimental evidence
- An example from life

Review

- Last week
  - A classification of software design methods based on philosophical theory
    - Rationalism and Empiricism (Epistemologies)
    - Realism and Anti Realism (Ontologies)
  - An indication (assertion) of what sort of methods might fit into each category
  - An examination of the practical implications of these viewpoints
  - An explanation of what comes next

Preview

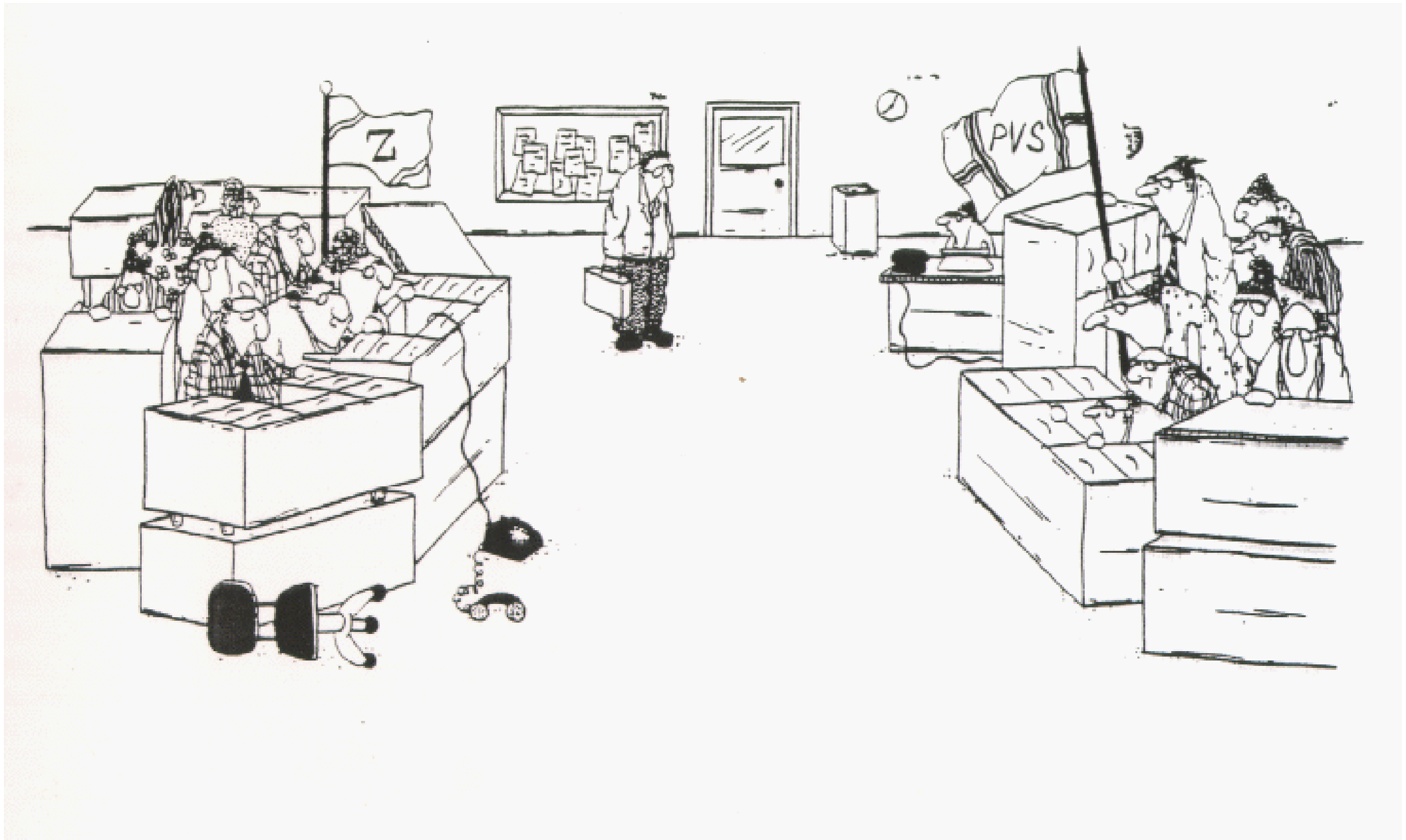
- What are the key features of formal methods?
  - Assumes software and program descriptions to be equivalent (i.e. both are complete and closed)
  - There is a seamless equivalence between the software description, the representation in the designers mind and the underlying aspects of reality that are being modelled
  - By removing any distinction between software and programs, the formal strand seeks to introduce mathematical rigour into both program and software design
  - There is a correct answer that can be discovered by the application of logic and reason = realist ontology and rationalist epistemology

Rationalism and Realism

- *Rationalist* arguments deal principally with epistemology claiming that reason is source of all knowledge and that everything that can be known, must be intelligible and rationally explicable

- *Realist* arguments deal principally with ontology claiming that there is such a thing as truth and that all beliefs can be tested against a reality that is knowable

Formal methods



Choosing a formal method can be a fearful thing.

THE UNIVERSITY *of York*

Chris Kimble
February 2008

Formal methods …

- Have been around for a long time but have never gained wide acceptance except in certain niche applications (typically secure and/or safety critical systems)
- Have their own (passionate) advocates
- Have a list of “success stories” ...
- ... and also have a list of reasons why they are not widely used

Formal Methods

Two types of formal methods:

- Type I
  - Based on set theory and first order predicate calculus
  - Based on model-oriented approach where the system's behaviour is specified as a mathematical model of the underlying state (data) and a collection of operations on that state.
- Type II
  - Based on temporal logic (an extension of propositional logic to show how the truth values change with the time)
  - Based on property-oriented approach where the system's behaviour is specified indirectly by stating a set of properties (axioms) that the system must satisfy

Examples: Z, VDM, UNITY

- Z and VDM
  - Type I
  - Can be used to create an explicit model of the system state which can be used as the basis for implementation
- UNITY
  - Type II
  - supports the specification of both synchronous and asynchronous behaviours

THE UNIVERSITY *of York*

Z

- The Z notation was originally developed by the Programming Research Group at Oxford University in the late 1970s and later developed to include standards, tool-support, extensions, etc
- It is based on first-order predicate logic and contains a catalogue (toolkit) of commonly used functions and predicates.
- Strictly speaking Z is not a method but a notation but examples of refining “Z style” abstract models into more concrete ones do exist

VDM

- The Vienna Development Method (VDM) began as specification language called Meta-IV in the Vienna IBM Laboratory in 1978
- VDM is claimed to be a complete program development method based on the model-oriented specification language (VDM-SL)
- VDM-SL is similar to the Z notation, but has syntax for the description of software modules and algorithms

UNITY

- UNITY is “a programming notation and logic to reason about parallel and distributed programs”
- It was created in 1988 and attempts to focus on what, instead of where, when or how
- An execution starts from any state satisfying the initial condition, every statement is run in a random order until a state is reached where further execution do not change it

Success Stories

- Achieving clearance to carry sensitive information through an Internet gateway
- Assuring safety in the development of programmable logic controllers
- Certifying the Darlington Nuclear Generating Station plant shutdown system
- Designing the software to reduce train separation in the Paris Metro
- Developing a collision avoidance system for United States airspace
- Developing a transaction processing system for IBM
- Developing an air traffic control system
- The design and verification of a RISC processor

THE UNIVERSITY of York

Why are they not more widely used?

- The reason for the lack of acceptance is usually attributed to unfounded “myths” or a simple “misunderstanding” of what formal methods are *really* about.

The Formal Methods Blues

- “I'm just a soul whose intentions are good, Oh Lord, please don't let me be misunderstood”
  - (Bennie Benjamin, Sal Marcus, Gloria Caldwell)

Why are they not more widely used?

Some examples:

- Seven Myths of Formal Methods
  - (Hall,1990)
- Seven More Myths of Formal Methods
  - (Bowen & Hinchey, 1994)
- Revisiting Seven Myths of Formal Methods
  - (Tretmans, Wijbrans & Chaudron, 2001)

The “Myths” of Formal Methods

(Hall, 1990)

- Myth 1: formal methods guarantee perfect software and eliminate the need for testing.
- Myth 2: formal methods are all about proving programs correct.
- Myth 3: formal methods are only useful in safety-critical systems.
- Myth 4: application of formal methods requires highly trained mathematicians.
- Myth 5: applications of formal methods increases development costs.
- Myth 6: formal methods are unacceptable to users.
- Myth 7: formal methods are not used on real large-scale systems.

Myth 1

- Argument:
  - Formal methods can guarantee that software is perfect
- Counter argument:
  - Formal methods are very helpful at finding errors early on and can nearly eliminate certain classes of error

Myth 2

- Argument:
  - Formal methods are all about program proving
- Counter argument:
  - They work largely by making you think very hard about the system you propose to build

Myth 3

- Argument:
  - Formal methods are only useful for safety-critical systems
- Counter argument:
  - They are useful for almost any application

Myth 4

- Argument:
  - Formal methods require highly trained mathematicians
- Counter argument:
  - They are based on mathematical specifications, which are much easier to understand than programs

Myth 5

- Argument:
  - Formal methods increase the cost of development
- Counter argument:
  - They can decrease the cost of development

Myth 6

- Argument:
  - Formal methods are unacceptable to users
- Counter argument:
  - They can help clients understand what they are buying

Myth 7

- Argument:
  - Formal methods are not used on real, large-scale software
- Counter argument:
  - They are being used successfully on practical projects in industry

7 more “Myths” of Formal Methods

(Bowen & Hinchey, 1994)

- Myth 1: formal methods delay the development process.
- Myth 2: formal methods are not supported by tools.
- Myth 3: formal methods mean forsaking traditional engineering design methods.
- Myth 4: formal methods only apply to software.
- Myth 5: formal methods are not required.
- Myth 6: formal methods are not supported.
- Myth 7: formal methods people always use formal methods.

Myth 1:

- Argument:
  - Formal Methods delay the development process
- Counter argument:
  - Any model of cost and time estimation is based on historical data and experience, so lack of such data/experience can lead to underestimation

THE UNIVERSITY of York

Myth 2:

- Argument:
  - Formal Methods are not supported by tools
- Counter argument:
  - A large number of tools including theorem provers are Available

THE UNIVERSITY *of York*

Myth 3:

- Argument:
  - Formal Methods means forsaking traditional engineering methods
- Counter argument:
  - A large number of ‘traditional’ methods have been successfully integrated with formal methods

Myth 4:

- Argument:
  - Formal Methods only apply to software
- Counter argument:
  - They have been successfully applied to both hardware and software design

Myth 5:

- Argument:
  - Formal Methods are not required
- Counter argument:
  - The use of formal methods is recommended in any system where the issue of correctness is of concern and in some cases, formal methods are mandated, e.g. UK MoD, Atomic Energy Control Board in Canada, …

Myth 6:

- Argument:
  - Formal Methods are not supported
- Counter argument:
  - Many Formal methods are standardized and have large user communities as well as extensive literature, conferences, industrial training courses, etc

THE UNIVERSITY *of York*

Myth 7:

- Argument:
  - Formal Methods people always use Formal Methods
- Counter argument:
  - UI design is recognised as hard to formalise. It is also recognised that total formalisation is often impractical from a resource, time or financial aspect

Why are they not more widely used?

- Can we learn from industrial experience?
- Observations on industrial practice using formal method
  - (Gerhart, 1993)
- An experience in the formal verification of industrial software
  - (Staskauskas, 1996)
- From formal models to formally based methods: an industrial experience
  - (Ciapessoni et al, 1999)

The industrial experience?

– It turned out to be impossible to assess any cost effectiveness trade-off ... all cases involved so many interwoven factors that it is impossible to allocate pay off from formal methods usage versus other factors (Gerhart, 1993)
– The need to deal with requirements that change while the software is being designed seems to be a fact of life *[however]* modifying a formal specification ... is an extremely difficult task (Staskauskas , 1996)
– The ... use of formal methods helped to improve the quality of the system. Since this was the main goal of the use of formal methods, it can be concluded that their application was a success (Tretmans, 2001)

Why are they not more widely used?

- Can we find out through experiments?
- Why Are Formal Methods Not Used More Widely? (Knight et al, 1997)
  - Experiment = develop a formal specification for a nuclear reactor using Z, PVS and statecharts, and then assess the results using (a) developers, (b) engineers and (c) computer scientists
  - Conclusion = there are many practical barriers to *[formal methods]* routine use in industrial software development projects

A practical example

- Staskauskas, M. G. (1996). An experience in the formal verification of industrial software. *Commun. ACM,* **39**(12es), 256.