

# Confessions of a Formal Methodist

**Michael G. Hinchey**

Software Verification Research Centre  
The University of Queensland  
St. Lucia, Queensland 4072

mike@svrc.uq.edu.au

## Abstract

The author, a confirmed formal methodist, while not abandoning his adherence to this “religion,” concedes that formal methods alone will not save the day, and that a number of other approaches have been, must be, and will continue to be, of relevance in the development of safety-critical systems. These, however, are not incongruous or antithetical to the application of formal methods.

*Keywords:* Formal methods, system safety, technology transfer.

## 1 Introduction

The term “formal methods” has now been used (albeit somewhat inappropriately) (Dean and Hinchey, 1996) in the computing and system engineering milieu for over three decades. In this context (and for the purposes of this paper), the term is taken to refer to a class of mathematical techniques, notations, and tools, which have been widely advocated for the development of large-scale complex systems. It is also widely accepted that the use of these techniques can result in high-quality systems that can come in on-time, within budget, and which satisfy user requirements (Bowen and Hinchey, 1995b).

However, these techniques have not achieved the level of acceptance, nor levels of use, that many believe they should. Even in the classes of systems where their application would seem essential -- namely critical systems -- their use is still far from the norm.

The author, with various colleagues, has been advocating the widespread application of formal methods for over a decade (Hinchey and Bowen, 1995; Hinchey and Bowen 1999). While clearly there have been a number of success stories, the uptake of formal methods has been far from speedy. Indeed, some might argue that it has been a complete failure. The author, while not abandoning his adherence to this “religion,” concedes that formal methods alone will not save the day, and that a number of other approaches have been, must be, and will continue to be, of relevance in the development of safety-critical systems. These, however, are not incongruous or antithetical to the application of formal methods.

## 2 What is this magnificent science?

*Why does this magnificent applied science which saves work and makes life easier bring us so little happiness? The simple answer runs: because we have not yet learned to make sensible use of it.*

-- Albert Einstein

For the purposes of this paper, we define formal methods to be those sets of tools and techniques that are used for the unambiguous specification and design of complex computer (and other) systems, as well as for the analysis and subsequent implementation of these specifications and designs.

A key requirement is that the notation used for specification and design has a complete formal semantics. This requirement excludes most conventional programming languages from our definition, but, increasingly, various functional and logic programming languages can be included in this category. Of particular note: our definition excludes languages such as UML from the category of a “formal method”, although many proponents of UML would have us include it.<sup>2</sup> Moreover, our definition includes theorem proving systems, such as HOL, Isabelle and PVS, which have had many great successes in recent years.

## 3 Myths of Formal Methods

At least part of the failure, or perceived failure, of formal methods is attributable to unreasonable and unachievable expectations.

(Hall, 1990) and (Bowen and Hinchey, 1995b) highlight various myths of formal methods held by the system development community. Surprisingly, several of these still hold.

Many still believe that formal methods are difficult to use and require great mathematical expertise. Indeed it is true that mathematical skill is required for more advanced aspects of formal methods, and in particular in the development of critical applications. However, at the Software Verification Research Centre, we have found that we are able to involve end-users in the development process, and we have had great success in encouraging

---

Copyright © 2002, Australian Computer Society, Inc. This paper appeared at the *7th Australian Workshop on Safety Critical Systems and Software (SCS'02)*, Adelaide. Conferences in Research and Practice in Information Technology, Vol. 15. P. Lindsay, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

---

<sup>2</sup> There is currently a move towards adding formalism to UML in both the new UML standard and as a separate language pUML (precise UML).

our clients and collaborators to apply formal methods in areas where traditionally they would not have done so.

Many also still believe that they do not need to apply formal methods. However, we have had several circumstances where clients approached us with problems with systems that they did not consider to be safety-critical. Yet, on a closer examination, these applications (typical involving moving machinery or equipment) were clearly safety-critical, and in at least one circumstance involved the loss of life. Therefore, even if the use of formal methods were not mandated by regulations, their use was considered essential as the ultimate check on the safety of the system.

However, the greatest problem in “selling” formal methods has been that they have failed to live up to the “hype” with which they were introduced. Many were lead to believe that formal methods were the Holy Grail of system development, a panacea for all problems, particularly in developing safety-critical applications.

Clearly this was never the intention of formal methods protagonists. Formal methods were introduced as a means of clearly and unambiguously specifying system requirements. A great majority of system failures can be traced back to errors and misinterpretations at the requirements stage. Formal methods were introduced as a means of overcoming this, and there have been a number of successful industrial applications; see for example those reported in Hinchey and Bowen (1995) and Hinchey and Bowen (1999). There have also been significant projects involving analyzing formal methods and attempting to achieve efficient automatic code-generation, with varying degrees of success. Theorem provers and functional programming languages have been particularly successful in these areas.

Unfortunately, many have made baseless claims that formal methods would revolutionize system development. The aim of formal methods is to provide abstraction mechanisms that would enable us to reduce complexity and tackle specific problem areas in system development. This they have done well; but there are other issues to address.

Formal methods have been particularly useful in elucidating requirements, in providing abstraction mechanisms so that specific problem areas can be highlighted and dealt with, and in raising consciousness regarding this potential problem areas.

The myth that they are not “required”, still prevails. Indeed, the mandating of formal methods in various standards in the safety-critical arena has not continued in the manner that had been anticipated (Bowen and Stavriou, 1993; Bowen and Hinchey, 1994). However, a formal approach still remains the only way in which we can meaningfully talk about *correct* systems.

#### 4 Commandments of Formal Methods

Bowen and Hinchey (1995a) identify ten “commandments” or rules of best practice for the

application of formal methods, these being mostly based on personal experience and/or the results of the projects described in (Hinchey and Bowen, 1995).

By and large, the authors have received feedback indicating agreement on the importance of these maxims. However, several more years of experience has highlighted the particular importance of several of these commandments:

- *Thou shalt formalize but not overformalize.* There has been a surge in interest in lightweight formal methods, or “formal methods light”. That is, the realization that formal methods do not apply to all aspects of a system. This has been our experience at the SVRC also (and was advocated in Bowen and Hinchey, 1995a): key components should be formalized and the necessary level for the application of formal methods should be determined. Our experience at the SVRC has been that higher levels (even as far as machine-checked proofs) have been necessary for certain critical applications, but for many applications even the lowest level (merely formally specifying the application at hand) can prove to be useful, and sufficient.
- *Thou shalt have a formal methods guru on call.* While this commandment has proved to be accurate, we have found it equally important to have domain expertise on call. This is true whether that expertise is from our own staff (where we have expertise in transport, defence, mining, and many other areas) or whether supplied by the customer. We have also found that having both domain and formal methods expertise *from the outset* is important.
- *Thou shalt not be dogmatic.* The author has always been keen to emphasize that formal methods are not a guarantee of success, nor a panacea. He has always conceded that there are many other techniques, particularly in the area of safety-critical systems, that have much to offer, and that have demonstrated that they can produce highly reliable, and correct, systems. In recent years, the formal methods community has begun to recognize more that formal methods are not for everyone, and even the most fervent advocates must agree that there are substantial opportunities for error when using formal methods, although these can be considerably reduced when appropriate tool support is available.

#### 5 Sins of Formal Methods

There are several “sins” that the formal methods community has committed over the years (Bowen and Hinchey, 1999) and which we must recognize and rectify:

- *Epidectic:* Formal methods should not be applied merely as a means of demonstrating one's ability, nor to satisfy company whim, or as a result of peer-pressure. Just as the scare of the so-called “millenium bug” saw many firms incur needless expense as they unnecessarily converted their

systems, there is a danger that formal methods could be “forced” upon projects where they are not needed. Realistically, the first thing that must be determined before a formal development is undertaken is that one *really* does need to use formal methods -whether it is for increased confidence in the system, to satisfy a particular standard required by procurers, or to aid in conquering complexity, etc.

- *Hyperbole*: Formal methods are not a panacea; they are just one of a number of techniques that, when applied correctly, have been demonstrated to result in systems of the highest integrity, and particularly in the area of safety-critical systems, other techniques *must* also be used.
- *Pistic* : One must be careful not to place too much trust in techniques and tools that have not been demonstrated to be “correct”. While the use of formal methods should certainly reduce the number of anomalies in system development, inexperience and carelessness can result in specifications that reduce to false, and are as ambiguous as natural language specifications.

Similarly, no proof should be taken as definitive. Hand-proofs are notorious in not only admitting errors as one moves from one line to the next, but also at making gigantic leaps which are unfounded. From experience of machine checking proofs, most theorems are correct, but most hand proofs are wrong (Rushby, 1995). Even the use of a proof checker does not guarantee the correctness of a proof. While it does aid in highlighting unsubstantiated jumps, and avoidable errors, proof should never be considered an alternative to testing, although it may reduce the required amount of testing, and in particular unit testing.

- *Oligarchy*: The formal methods community tends to be very introspective. However, there has been considerable investment in existing software development techniques and it would be foolhardy to replace these en masse with formal methods. Instead it is desirable to integrate formal methods into the design process in a cost-effective manner.

Moreover, the safety-critical systems community has developed a whole plethora of useful techniques and approaches that have been highly successful and which formal methods could never hope to supplant. Rather, the formal methods community would do well to see how it might exploit these techniques.

- *Ephemeral*: The application of formal methods to the development process should not be seen as “re-inventing the wheel”. Realizing that formal methods cannot replace other techniques, *and should not seek to*, the formal methods community should seek to integrate its techniques with those of various other successful communities, e.g., the real-time systems community.

- *Epexegetis*: Formal specifications *can* be made intelligible and acceptable to non-technical personnel (Saiedian and Hinchey, 1996) but only provided that they are augmented with sufficient amounts of informal explanation, diagrams, etc.

The formal specifier must not use a formal specification as a means of demonstrating his/her own ability, but rather must be willing to rework specifications to make them more intelligible to others. Similarly, abstraction should be used to hide unnecessary detail, making a specification suitable for different audiences.

- *Maiandros*: Most of the successful applications of formal methods have employed formal methods from the early stages of development, replacing natural language and informal specification. Indeed, most projects have reported greatest savings at these early stages, where formal specification techniques have highlighted errors that could be corrected before the programming phase. Therefore, formal methods should be taken on-board from the outset, along with other techniques.

## 6 A New Religion (Conclusion)

The author was a convert to Formal Methodism in the mid-1980s.

Although plagued by a number of misconceptions, myths, and downright mistruths, the application of formal methods can indeed result in the production of high quality software (and other) systems than are on-time, within budget, and satisfy user requirements.

However, their widespread uptake has not as yet occurred. Despite a large number of success stories, their use is still not commonplace, not even in critical applications where their usage is most warranted. This can in part be attributed to the misconceptions described, but more so to the fact that there are a wide range of techniques in the safety-critical arena that are at least as important to the development of such systems.

The author views the future of formal methods as a support and clarification technique to be used in conjunction with existing safety techniques, and not as a substitute for these.

## 7 Acknowledgements

The author would like to extend his thanks to all those colleagues with whom he has debated and discussed the various merits and demerits of formal methods over the years, and collaborated on various. These are too numerous to list, but worthy of particular mention are: Jonathan Bowen, Bob Boyer, Neville Dean, Edsger Dijkstra, Colin Fidge, Tim Flanagan, Tony Hoare, Peter Lindsay, J Moore, and Jim Rash.

## 8 References

BOWEN, J.P. and HINCHEY, M. G. (1999): *High-Integrity System Specification and Design*. London, Springer Verlag.

BOWEN, J.P. and HINCHEY, M.G. (1994): Formal methods and safety-critical standards. *IEEE Computer* **27**(8): 68—71, August.

BOWEN, J.P. and HINCHEY, M.G. (1995a): Ten commandments of formal methods. *IEEE Computer* **28**(4):56—63, April.

BOWEN, J.P. and HINCHEY, M.G. (1995b): Seven more myths of formal methods. *IEEE Software* **12**(4):34—41, July.

BOWEN, J.P. and HINCHEY, M.G. (1999): It's Greek to Me! Method in the Madness. In Hinchey and Bowen (1999).

BOWEN, J.P. and STAVRIDOU, V. (1993): Safety-Critical systems, formal methods and standards. *Software Engineering Journal* **8**(4):189-209, July.

BROOKS, Jr., F.P. (1987) No Silver Bullet: Essence and accidents of software engineering. *IEEE Computer* **20**(4):10--19, April.

DEAN, C.N and HINCHEY, M. G. (1996): Formal methods and modeling in context. In *Teaching and Learning Formal Methods*, 99-116. DEAN, C.N and HINCHEY, M.G. (eds.). Academic Press, London.

HEITMEYER, C. and MANDRIOLI, D. (eds.) (1996): *Formal Methods for Real-Time Computing*. London, Wiley.

HINCHEY, M.G. and BOWEN, J.P. (eds.) (1995): *Applications of Formal Methods*. Hemel Hempstead, Prentice Hall.

HINCHEY, M.G. and BOWEN, J.P. (eds.) (1999): *Industrial-Strength Formal Methods in Practice*. London, Springer Verlag.

RUSHBY, J. (1995): *Formal Methods and their Role in the Certification of Critical Systems*, Technical Report SRI-CSL-95-1, SRI International, Menlo Park, CA, USA.

SAIEDIAN, H. and HINCHEY, M.G. (1996) Challenges in the successful transfer of formal methods technology into industrial applications. *Information and Software Technology* **38**(5):313—322, May.