The Use of Industrial-Strength Formal Methods

Jonathan P. Bowen

The University of Reading
Department of Computer Science
Whiteknights, PO Box 225
Reading, Berks RG6 6AY, UK
Email: *J.P.Bowen@reading.ac.uk*

Michael G. Hinchey

New Jersey Institute of Technology
Real-Time Computing Laboratory
CIS Department, University Heights
Newark, NJ 07102, USA
Email: *hinchey@cis.njit.edu*

Abstract

Formal methods are used in a surprisingly wide variety of applications and ways throughout the world. While they may still be considered a niche market, there is growing evidence that they can be used successfully in industry if applied judiciously. This paper discusses some of the issues concerning the successful application of formal methods and surveys a number of examples of industrial usage, with a large bibliography for further reading on the state of the art in this area.

1 Introduction

Formal methods are recommended by many academics but avoided by many industrial practitioners. Despite some successes, formal methods are still little used in industry at large, and are seen as esoteric and irrelevant by many managers. The debate about the applicability of formal methods in practice continues apace [18], perhaps due to over-optimistic expectations by some [24]. There are many myths concerning the use of formal methods [15, 6], although guidance [7]. and standards [5, 3] are readily available; but in order for the techniques to become widely used, technology transfer from theorists to practitioners must be fostered [4].

This paper surveys a selection of industrial applications where formal methods have been used effectively. Following on from an existing book giving examples of formal methods in use [19], a selection of further examples in a forthcoming book [20] are outlined. Finally some issues to be considered for continuance of the technology transfer process are aired.

2 Example Uses of Formal Methods

We present a collection of examples of use of industrial-strength formal methods, performed by experts in the field. More complete information and details may be found in the books [19] and [20]. A excellent survey of a number of industrial applications of formal methods is reported in [10]. Summaries of observations as a result of this large report may be found in [14] (particularly with regard to critical systems) and [11].

Two other recent collections of case studies applying different formal methods to the same case study are a production cell [23] and a steam boiler controller [1]. The latter includes a CD-ROM containing material associated with the various approaches.

• Experience using Promela and Z in the Design of a Storm Surge Barrier Control System
(Computer Management Group, The Netherlands)

As the final phase of a 40 year project to protect the Netherlands against floods due to storm surges, a movable barrier is currently being built near Rotterdam. A fully automated system, called BOS, will operate the barrier. BOS takes the decision to open or close the barrier based on measurements collected through various sensors and networks, and it controls the movements of the barrier. The system has to satisfy very high reliability demands. Not closing in time could result in the flooding of Rotterdam, whereas not opening in time could cause severe damage to the barrier itself. As long as the barrier is closed the important harbor of Rotterdam is cut off from the rest of the world, leading to great economic losses.

Computer Management Group (CMG) was awarded the contract from the Netherlands Ministry of Public Works to build BOS. Because of the high reliability requirements BOS has to meet, CMG established a close collaboration with the Formal Methods group at the University of Twente to transfer knowledge and provide consultancy on the application of formal methods to system design.

As part of an effort to integrate formal methods

with CMG's design trajectory the decision was made to:

1. Use the Spin tool to validate Promela specifications concerning the communication between subsystems of BOS, and communication with components belonging to the 'outside world' from the perspective of BOS.

2. Use Promela combined with Z to formally specify crucial parts of the design.

The Spin tool set (Spin and Xspin) was used to validate parts of the design, in particular the communication interfaces with the outside world. Promela combined with Z was used to specify crucial aspects of the design. The Z specifications were used initially to clarify requirements in the form of dataflow diagrams and natural language texts. In a later stage of the project, the Z specs were used to derive tests for the components of the BOS system.

• Demonstrating the Equivalence of Source Code and PROM Contents

(Nuclear Electric, UK)

The translation of the requirements for a computer based system into the stored code executed by the is an extended process with many opportunities for the introduction of errors. One potential source of errors is the suite of tools used to translate the high level language source program, defining the algorithms to be used, into the binary code, stored in PROM, which actually controls the system.

A highly automated technique has been developed to demonstrate equivalence between the source code and PROM contents for a safety critical programmable protection system which is to be used in the UK nuclear industry [31].

• Formal Top Level Specification of the Honeywell STOP Operating System

(Data Sciences, UK)

A prototype exercise has been carried out to construct, using the Z notation, a formal specification of a secure software system based on a sample set of the elements of the STOP operating system. The specification comprises a mathematical description of a *secure state*, together with a mathematical description of three system *operations* which change the state.

It appeared that a formal specification of STOP, written in the Z notation, would be a useful entity from the system developer perspective. In addition, it would provide further evidence of the power of the notation, e.g., in this case, as a technique for the a posteriori description of existing products as distinct from its use in development. It was judged that, with the aid of judicious choice, a 'prototype sample' of the system could be selected which would enable some kind of feel to be gained of the difficulty involved in formally specifying its behavior, and of the usefulness (in terms of understanding, etc.) of the results.

• Scheduling and Rescheduling of Trains

(Chinese railway and UNU/IIST, Macau)

Work has been undertaken by UNU/IIST (at the United Nations University in Macau) involving staff from the Chinese railways. The immediate aim was to develop software that can improve the efficiency of the Chinese railway system, particularly in the scheduling of trains. In the longer term it is hoped to develop the capacity of the Chinese railways to produce and even market their own software. The part of the project described includes the development of formal models of some basic components of railways (networks of stations and lines and timetables) and the activities needed to schedule and reschedule trains using a 'running map'. This part finished with the delivery of a prototype running map tool, and some conclusions are drawn about the formal development of this tool. The work used the RAISE specification language, method and tools [27].

• Formalization of Automatic Protection Switching

(AT&T, USA)

Some of the most important requirements that software developers must satisfy come from international standards. Unfortunately, standards are not always expressed in formal notations, and may be ambiguous or incomplete. Standards for automatic protection switching of telecommunication devices have been expressed in several formalisms, including basic LOTOS (Language Of Temporal Ordering Specifications) [29] and Z. In the process several ambiguities in the standards have been discoved, together with some difficulties in applying the formalisms.

• The French Population Census for 1990

(INSEE, France)

This is an example of formal techniques which have proven beneficial in the design and development of the system supporting the 1990 French Population Census. The techniques covering the whole development process, from specification through to implementation and subsequent verification and validation, are generic enough to be of wide applicability. The B-Method was employed in the development of the system. Refinement was exploited in constructing the specification.

The system supporting the census is so large that it had to be built incrementally. More precisely, the system had to grow without modifications to its kernel. This is why a structure for accessing data was defined at the outset. The structure was based on the administrative geography of France at the date of the census. During the development of the system additional information was introduced. This was performed smoothly by storing the new information in databases possessing the same structure.

Implementation was carried out by refinement up to a point where it was system dependent. The implementation of low level B abstract machines is generated by a tool. This tool can generate implementations for a machine where the prototype has been tested and for the mainframe where the system is now running. The system is of a significant size and reliably handles a huge amount of data in a very efficient way; it has been validated by inspection, testing, and last but not least, through five years of constant use.

• Rigorously Reviewing Structured Specifications using Z

(BT, UK)

The Rigorous Review Technique (RRT) was developed during a collaboration between BT (UK) and Leeds Metropolitan University. The technique involves a systematic transformation of a structured specification (minimally Entity-Relationship Model and Data-Flow Diagrams together with the related Data Dictionary entries) into the Z formal specification notation.

RRT has been demonstrated to reveal errors and inconsistencies not normally found using conventional review techniques. The results reveal not only problems with the specification, but also problems with the development process.

• Analyzing Z Specifications with Z/EVES

(ORA, Canada)

The capabilities of ORA Canada's recently released Z/EVES system [28] have been demonstrated by analyzing a formalization of the sliding window protocol. They show how Z/EVES can be used in four different styles for analyzing Z specifications: type checking, schema expansion, domain checking, and proof of general conjectures.

• Using Formal Methods to Develop an ATC Information System

(Praxis Critical Systems, UK)

CDIS is a display information system supporting the new terminal control room at the London area and Terminal Control Centre [16]. It is a distributed real-time information system supporting air traffic controllers. CDIS was developed using advanced software engineering techniques including the extensive use of formal methods for specification and design. The application showed that formal methods can improve quality at no extra cost, and that they are applicable to a large, demanding project.

• Formal Development of a Radiation Therapy Machine Control Program

(University of Washington, USA)

The control program and graphical user interface for a radiation therapy machine at University of Washington, in Seattle, have been developed using formal methods over a long period [21]. The control program of the machine, whose correct operation is clearly safety-critical, was developed using the Z notation for specification, design, and subsequent verification [22].

• Experience Developing a Trustworthy Network Security Device

(Naval Research Laboratories, USA)

The successful application of formal methods to engineer systems of interest to industry or the military requires their balanced integration with less formal methods (e.g., see [2]). Competing factors such as system safety, security, performance, mission and development time and cost must be weighed and prioritized when defining the process to be used to develop the system. The development of a fieldable network security device, called the External COMSEC Adaptor (ECA), used formal methods in the specification and verification of its most critical requirements and testing and simulation in the verification of its overall functional requirements.

The development process integrates the formal specifications and proofs with structured software documentation to clarify the relationship between the refinement of ECA functionality and the formal argument that the ECA satisfies its critical requirements. Although the process enabled the successful construction of the ECA by properly balancing competing requirements, the experience suggested a number of improvements that could be made to the process.

• Formal Analysis of the UEPS Second Generation Electronic Wallet

(University of Cambridge, UK)

UEPS, the Universal Electronic Payment System, is an electronic funds transfer product which is well suited to developing country environments, where poor telecommunications make off-line operation necessary. It is designed around an electronic wallet

with smart-card and check-book functions; money is loaded from the the bank, via bank cards, to customer cards, to merchant cards, and finally back to the bank through a clearing system. This architecture is uniquely demanding from the point-of-view of security.

As far as the authors are aware, UEPS is the first live financial system whose authentication protocol was designed and verified using formal analysis techniques. This was achieved using an extension of the Burrows–Abadi–Needham (BAN) logic, and raises some interesting questions: firstly, such formal logics had been thought limited in scope to verifying mutual authentication or key sharing; secondly, this work has found hidden assumptions in BAN, and a problem with the postulates of the Gong–Needham–Yahalom logic, both concerning freshness; thirdly, the author highlights the need for a formalism to deal with cryptographic chaining; and fourthly, this type of formal analysis turns out to be so useful that its applications should be routine for financial and security-critical systems.

- **Cleanroom Software Engineering: Theory and Practice**

(Software Engineering Institute, USA)

Cleanroom Software Engineering is a theory-based, team-oriented process for developing and certifying high-reliability software with high productivity. The Cleanroom process puts software development and testing under statistical quality control. In this process, mathematics-based techniques for specification, design, and correctness verification are used to create software approaching zero faults prior to first execution. Statistical usage-based testing is then applied to provide objective statistical estimates of software reliability and fitness for use. The Cleanroom process begins with a specification that not only defines functional requirements, but also identifies the statistical usage of the software and a nested set of usable function subsets that can be developed and certified as increments which accumulate into the final system. Disciplined software engineering techniques for design and correctness verification create provably correct software components for statistical testing and reliability certification.

The Cleanroom process has been applied in a variety of environments, ranging from transaction oriented systems to real-life embedded software. Cleanroom provides a disciplined implementation of key elements of the Software Engineering Institute's Capability Maturity Model (CMM) for Software. The use of Cleanroom continues to grow as societal needs for reliable software continue to increase.

3 Future Developments

The future of formal methods is a subject of continued discussion [17]. To secure a successful future, a number of developments are desirable. These include:

- **An engineering approach.** Formal methods must be integrated smoothly into existing industrial best practice in a manner which causes as little disruption as possible [25].
- **Improved tools.** Most formal methods tools so far have resulted from formal methods research projects, and associated spin-off companies, rather than mainstream tools developers. As a result, their usability, and sometimes robustness, can often leave a lot to be desired. Unfortunately the formal methods tools market is still fairly small and raising capital to invest in serious production quality tools may be difficult. Raising commercial venture capital is likely to be difficult because the banks will be more interested in the size of the market rather than the potential improvement in software quality.
- **Technology transfer investment.** The transfer of technology like formal methods is a time consuming and costly business [30]. The effects and benefits of formal methods are less palpable than some of the other more popular techniques that come and go with fashion. The investment in learning and using formal methods is large, but the returns in the long term can be commensurate with this. Most people who have made the investment have not regretted it afterwards, and would not go back to their old ways.
- **Harmonization of engineering practices.** While the use of formal methods may seem to run perpendicular and even counter to some other concerns on software engineering projects, such friction should be minimized. It is important that all those involved, be it managers or engineers, and whether the personnel involved fully understand the techniques or not, at least understand the way the techniques slot into the overall framework. It can be galling to some managers that the use of formal methods considerably delays the start of production of code in the lifecycle. However it considerably speeds up and improves its production when it is generated.
- **Practical experience.** A number of significant projects have now been undertaken using formal

methods, a number of which have been outlined in this paper, but more are needed to gain a better insight into the general applicability of such techniques. Most successful formal methods project have had the help of an expert on call in case of difficulty. It remains to be seen if formal methods can be successfully applied when less expert help is at hand. Fortunately computer science undergraduate courses (in Europe at least) do now provide some suitable grounding for many software engineers who are now entering the profession. However, the effects will take some time to filter through in practice.

- **Assessment and measurement.** Metrics are a problematic area. It would obviously be helpful and commercially advantageous to know the effect of the use of formal methods on the productivity, error rates, etc., in the development process. However these can be hard and expensive to obtain, and even if actual figures are available, these may not measure the aspect that is of real interest. It is also difficult to obtain such commercially sensitive data in the public domain, which slows both academic study and potential solutions to the problems. Metrics should be treated with caution, but improvements in such techniques would be worthwhile.

- **Education and certification.** Most modern comprehensive standard textbooks on software engineering now include a section on formal methods. Many computing science courses, especially in Europe, are now including a significant portion of basic relevant mathematical training (e.g., discrete mathematics [12]). In this respect, education in the US seems to be lagging behind, although there are some notable exceptions (e.g., see [13]). It is particularly important that the techniques, once assimilated, are used in practice as part of an integrated course, but this has not always been the case in the past.

  Once a software engineer is educated with the relevant background and techniques, accreditation by professional institutions may be an important indicator of a certain level of competence, and of continuing training to remain up to date. Checks on software engineers are often very lax compares to those for other types of engineer, even in the development of highly critical systems.

- **Standards and legislation.** Standards are increasingly important as a driving force for the use of formal methods by industry [9]. Some standards strongly recommend or even mandate a formal approach, especially for high integrity applications such as safety-critical systems. Legislation can also insist that appropriate techniques are used when human lives are at risk. The case can arise whereby techniques used to develop software must be defended and justified in a court of law [26]. Given the exponentially increasing use of computers in such systems because of their flexibility, it is likely that formal methods will at least attain a niche market in this area.

4 Conclusion

This paper has surveyed some potential misunderstandings concerning the industrial use of formal methods, some suggestions for guidance for successful application, and state of the art use of formal methods in a number of different areas. It is hoped that this overview will be of benefit to those considering the use of formal methods for either software or hardware development by highlighting the ways in which formal methods are being used. A large bibliography is included for those who wish to read about the issues, experiences and technical details further.

The actual formal methods, etc., available at any given time, can and will of course vary, and hopefully improve. Further up-to-date on-line information concerning formal methods, including companies involved in their use, can be found on-line under the World Wide Web Virtual Library formal methods section, maintained by one of the authors of this paper. Information on the book *Industrial-Strength Formal Methods* [20], including a link to the above information, may be found under the following URL (Uniform Resource Locator):

http://www.cs.reading.ac.uk/archive/isfm/

Acknowledgement

Some of the information included here has been provided by contributors the forthcoming book mentioned above [20], edited by the authors of this paper. Without their contributions this paper would not have been possible.

References

[1] J.-R. Abrial, E. Börger, and H. Langmaack, editors. *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, volume 1165 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[2] J. Bicarregui, J. Dick, and E. Woods. Supporting the length of formal development: From diagrams to

VDM to B to C. In H. Habrias, editor, *Z Twenty Years on – What is its Future?*, pages 63–75, Université de Nantes, France, 1995. IRIN (Institut de Recherche en Informatique de Nantes).

[3] J. P. Bowen. Formal methods in safety-critical standards. In *Proc. 1993 Software Engineering Standards Symposium*, pages 168–177. IEEE Computer Society Press, 1993.

[4] J. P. Bowen, R. W. Butler, D. L. Dill, R. L. Glass, D. Gries, J. A. Hall, M. G. Hinchey, C. M. Holloway, D. Jackson, C. B. Jones, M. J. Lutz, D. L. Parnas, J. Rushby, H. Saiedian, J. Wing, and P. Zave. An invitation to formal methods. *IEEE Computer*, 29(4):16–30, April 1996.

[5] J. P. Bowen and M. G. Hinchey. Formal methods and safety-critical standards. *IEEE Computer*, 27(8):68–71, August 1994.

[6] J. P. Bowen and M. G. Hinchey. Seven more myths of formal methods. *IEEE Software*, 12(4):34–41, 1995.

[7] J. P. Bowen and M. G. Hinchey. Ten commandments of formal methods. *IEEE Computer*, 28(4):56–63, April 1995.

[8] J. P. Bowen, M. G. Hinchey, and D. Till, editors. *ZUM '97: The Z Formal Specification Notation, 10th International Conference of Z Users, Reading, UK, 3–4 April 1997, Proceedings*, volume 1212 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[9] J. P. Bowen and V. Stavridou. Safety-critical systems, formal methods and standards. *IEE/BCS Software Engineering Journal*, 8(4):189–209, July 1993.

[10] D. Craigen, S. L. Gerhart, and T. J. Ralston. An international survey of industrial applications of formal methods. Technical Report NIST GCR 93/626-V1 & 2, Atomic Energy Control Board of Canada, US National Institute of Standards and Technology, and US Naval Research Laboratories, 1993.

[11] D. Craigen, S. L. Gerhart, and T. J. Ralston. Formal methods reality check: Industrial usage. *IEEE Transactions on Software Engineering*, 21(2):90–98, 1995.

[12] N. Dean. *The Essence of Discrete Mathematics*. The Essence of Computing Series. Prentice Hall, 1997.

[13] D. Garlan. Integrating formal methods into a professional master of software engineering program. In J. P. Bowen and J. A. Hall, editors, *Z User Workshop, Cambridge 1994*, Workshops in Computing, pages 71–85. Springer-Verlag, 1994.

[14] S. L. Gerhart, D. Craigen, and T. J. Ralston. Experience with formal methods in critical systems. *IEEE Software*, 11(1):21–28, January 1994.

[15] J. A. Hall. Seven myths of formal methods. *IEEE Software*, 7(5):11–19, September 1990.

[16] J. A. Hall. Using formal methods to develop an ATC information system. *IEEE Software*, 13(2):66–76, March 1996.

[17] J. A. Hall, D. L. Parnas, N. Plat, J. Rushby, and C. T. Sennett. The future of industrial formal methods. In J. P. Bowen and M. G. Hinchey, editors, *ZUM '95: The Z Formal Specification Notation*, volume 967 of *Lecture Notes in Computer Science*, pages 238–242. Springer-Verlag, 1995.

[18] C. Heitmeyer. Formal methods: A panacea or academic poppycock? In Bowen et al. [8], pages 3–9.

[19] M. G. Hinchey and J. P. Bowen, editors. *Applications of Formal Methods*. Prentice Hall International Series in Computer Science, 1995.

[20] M. G. Hinchey and J. P. Bowen, editors. *Industrial-Strength Formal Methods*. International Series in Formal Methods. Academic Press, 1997. In preparation.

[21] J. Jacky. Specifying a safety-critical control system in Z. *IEEE Transactions on Software Engineering*, 21(2):99–106, 1995.

[22] J. Jacky. *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, 1997.

[23] C. Lewerentz and T. Lindner, editors. *Formal Development of Reactive Systems: Case Study Production Cell*, volume 891 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[24] Luqi and J. A. Goguen. Formal methods: Promises and problems. *IEEE Software*, 14(1):73–85, January 1997.

[25] K. Kumar M. D. Faser and V. K. Vaishnavi. Strategies for incorporating formal specifications in software development. *Communications of the ACM*, 37(10):74–86, October 1994.

[26] D. MacKenzie. Computers, formal proof, and the law courts. *Notices of the American Mathematical Society*, 39(9):1066–1069, November 1992.

[27] RAISE Language Group. *The RAISE Specification Language*. BCS Practitioner Series. Prentice Hall International, 1992.

[28] M. Saaltink. The Z/EVES system. In Bowen et al. [8], pages 72–85.

[29] K. J. Turner, editor. *Using Formal Description Techniques: An Introduction to Estelle, LOTOS and SDL*. John Wiley & Sons, 1993.

[30] D. Weber-Wulff. Selling formal methods to industry. In J. C. P. Woodcock and P. G. Larsen, editors, *FME '93: Industrial-Strength Formal Methods*, volume 670 of *Lecture Notes in Computer Science*, pages 671–678. Formal Methods Europe, Springer-Verlag, 1993.

[31] L. A. Winsborrow and D. J. Pavey. Assuring correctness in a safety critical software application. *High Integrity Systems*, 1(5), 1996.