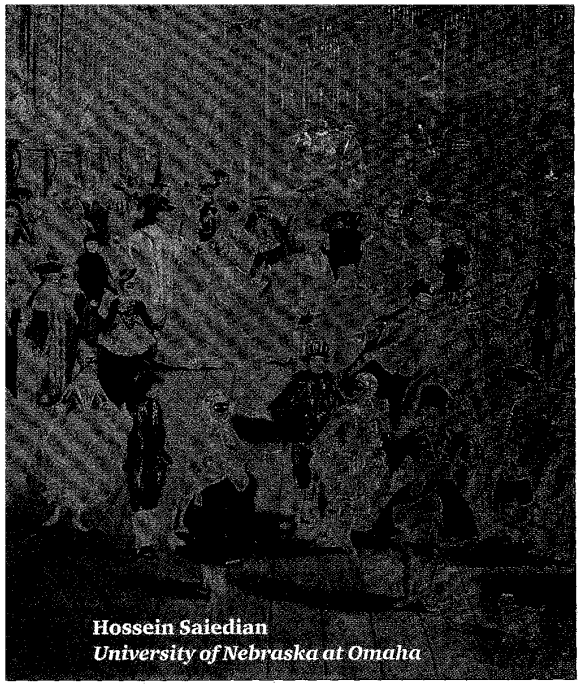

Hossein Saiedian
University of Nebraska at Omaha

An Invitation to Formal Methods

One of the most challenging tasks in software system design is to assure reliability, especially as these systems are increasingly used in sensitive and often life-critical environments such as medical systems, air traffic control, and space applications. It is therefore essential for developers to employ those methods that offer a high degree of assurance that the system's requirements accurately capture the users' critical requirements and that an implementation in software (or hardware) is an accurate realization of the design.

Software reliability, although the primary concern of the computing industry, is not the only one. Another major concern has been the increased cost of developing and maintaining software. Precise, up-to-date figures representing the actual cost of software in industry are neither available nor easy to establish. Boehm suggested that the worldwide cost of software in 1985 was roughly $140 billion ($70 billion in the US alone).[1] Even if we assume a modest growth rate of 10 percent per year (Boehm suggested 12), the software cost in the year 2000 will be close to $600 billion ($300 billion in the US). Much of software's cost stems from testing and maintenance. The absence of rigorous practices that eliminate residual specification and design errors (caused by imprecision, ambiguity, and sometimes, plain mistakes) has translated into a significant portion of this cost.

Many claim that formal methods not only provide added reliability but also have the potential to reduce costs. They provide a notation for the formal specification of a system whereby the desired properties are described in a way that can be reasoned about, either formally in mathematics or informally but rigorously. In essence, formal methods offer the same advantages for software (or even hardware) design that many other engineering disciplines have exploited—namely, mathematical analysis using a mathematically based model. Such models allow the designer to predicate the behavior and validate the accuracy of a system instead of having to rely entirely on nonassuring exhaustive testing.

The clear advantages of a more mathematical approach to software design has certainly been well documented; the literature contains many excellent examples of applications of formal methods for large, critical, or even business transaction systems. Despite the evidence, however, a large percentage of practitioners see formal methods as irrelevant to their daily work.

To be sure, practitioners can't simply be blamed for their skepticism. What is hindering the adoption of formal methods by industry? Is the notation too complex? Or are existing formal methods unable to contribute to, benefit from, or be integrated with traditional methodologies? Is there still a perception of high complexity and costs? Has the barrier been the lack of easy-to-use support tools? Or would an increase of formalism in computer science or software engineering education translate

Featuring:

Jonathan P. Bowen, *University of Reading*

Ricky W. Butler, *NASA Langley Research Center*

David L. Dill, *Stanford University*

Robert L. Glass, *The Software Practitioner*

David Gries, *Cornell University*

Anthony Hall, *Praxis*

Michael G. Hinchey, *New Jersey Institute of Technology*

C. Michael Holloway, *NASA Langley Research Center*

Daniel Jackson, *Carnegie Mellon University*

Cliff B. Jones, *University of Manchester*

Michael J. Lutz, *Rochester Institute of Technology*

David Lorge Parnas, *McMaster University*

John Rushby, *SRI International*

Jeannette Wing, *Carnegie Mellon University*

Pamela Zave, *AT&T Research*

0018-9162/96/$5.00 © 1996 IEEE

into a more formal approach in industry by the next generation of software engineers?

In seeking answers to these questions, I contacted some of the best minds in academia and industry, asking them to share their insights.

ROUNDTABLE CONTRIBUTIONS

We open with a point/counterpoint on the viability of formal methods in industrial practice. Two long-time advocates of formal methods, Jonathan Bowen and Michael Hinchey, identify four areas they believe are important in the industrial adoption of formal methods: standards, tools, training, and correcting the misconceptions. Robert Glass speaks for practitioners who perceive a chasm between researchers and practitioners. He argues that it will be impossible to advance formal methods in industry until the chasm is effectively bridged.

Formal methods light

Formal methods can be applied to all or selected components of a system in varying degrees, depending on the criticality and nature of that system. Of course, increasing the level of formality may imply increased costs (for example, in formal analysis and verification). Is it possible to maintain acceptable levels of rigor without complete formalization, or to use partial formalism during modeling and analysis? Commentaries by Cliff Jones and by Daniel Jackson and Jeannette Wing address these issues. While emphasizing rigor and the importance of a formal basis, Jones suggests using a less-than-completely formal approach in most development cases. Jackson and Wing, on the other hand, advocate a lightweight approach to formal methods in which cost-effectiveness is achieved by developing partial specifications (with a focused application) that can be analyzed mechanically. The authors discuss partiality in terms of language, modeling, analysis, and composition.

Formal methods in practice

The experts from industry examine formal methods from the vantage point of their direct experiences. Anthony Hall elaborates on their economic benefits. His industrial experiences in using formal methods for relatively large projects (see his recent article in the March issue of *IEEE Software*) suggest that such methods may indeed make software development cheaper if used as part of an overall engineering approach. David Dill and John Rushby discuss the successful application of formal verification in hardware design. Formal methods have substantially more practical impact on hardware design, they claim, not only because hardware is easier to verify, but also because the problem has been approached with an eye to maximizing ROI. Here, formalists target areas of high payoff, use highly automated techniques, and apply formal verification more to debugging than assurance. Michael Holloway and Ricky Butler discuss three primary impediments to the wide-scale industrial use of formal methods: inadequate tools, inadequate examples, and the "build it and they will come" expectations. Finally, Pamela Zave discusses the conceptual gaps between the mathematical models offered by formalists and actual application domains.

Resources

For a comprehensive list of formal methods resources, try this URL: http:// www.comlab.ox.ac.uk/archive/formalmethods.html.

Engineering mathematics

If software engineering is in fact an engineering discipline, then application of formal methods should parallel the use of mathematics in other engineering disciplines. Michael Lutz believes that the gap between researchers and industrial practitioners is largely due to a misunderstanding of the differences between research mathematics and engineering mathematics. He outlines these differences and suggests some approaches that may accelerate the use of formal methods in industry. David Parnas, who was the first winner of the Norbert Wiener Award for Professional and Social Responsibility, also believes that formal methods should be more like the mathematics used by other engineering disciplines. While emphasizing the importance of routinely using mathematics in software engineering, Parnas expresses concerns about the notation purveyed by most formal methods researchers.

Education

The term *formal* in formal methods derives from formal logic, a powerful tool intended for reasoning and certifying certain properties. But has this topic been adequately presented as a useful tool? David Gries, who is known for significant contributions to computing education and who won the 1995 Karl V. Karlstrom Outstanding Educator Award, reflects on the role of logic as the foundation of most formal methods. He argues that until people are comfortable using formal logic, they won't be comfortable with most formal methods.

THIS ROUNDTABLE WILL, it is hoped, serve as a beginning for a more serious forum in which academics and practitioners can sit down together to discuss their needs, expectations, and goals in an effort to close the gap between them. I am open to suggestions about how to proceed from here. ❙

Acknowledgment

I thank Scott Hamilton for helping organize and coordinate this roundtable.

Reference

1. B. Boehm, "Improving Software Productivity," *IEEE Computer*, Vol. 20, No. 10, Sept. 1987, pp. 43-58.

Hossein Saiedian *is an associate professor in the Department of Computer Science at the University of Nebraska at Omaha. His research interests include software engineering models, formal methods, and object-oriented computing. E-mail hossein@cs.unomaha.edu.*