Seven Myths of Formal Methods

IEEE Software 7(5), pp. 11-19
September 1990

Written by J.A. Hall

About the author:

J. Anthony Hall is a leading British software engineer specializing in the use of formal methods, especially the Z notation. He is a Fellow of the Royal Academy of Engineering.

http://www.anthonyhall.org/

Reason behind the article

"Formal methods are controversial. Their advocates claim they can revolutionize development. Their detractors think they are impossibly difficult. Meanwhile, for most people, formal methods are so unfamiliar that it is difficult to judge the competing claims. There is not much published evidence to support one side or the other, and a lot of what is said about formal methods is based on assertions and not facts. Thus, some of the beliefs about formal methods have been exaggerated and have acquired almost the status of myths." (First paragraph of the article).

Definition

Formal methods are mathematical techniques for developing computer-based software and hardware systems.

An attempt at objectivity

From the third paragraph:

“This article takes a practical look at formal methods, presents some myths – favorable and unfavorable – and explains what we have found to be the truth behind them.”

Myth 1

Formal methods can guarantee that software is perfect.

No method can guarantee perfection. Even if a method was in some sense “perfect”, the above claim still could not be made, due to the imperfection of the human users of the method.

Question: In what sense is this myth favorable?

This myth leads to both unrealistic expectations and the idea that formal methods are somehow all-or-nothing. The reality is that no such guarantee can be given – but the usefulness of formal methods does not depend on such absolute perfection.

If you take the position of this myth, than any problem with formally developed software is a refutation of formal methods' usefulness.

Myth 2

Formal methods are all about program proving.

The fact is that formal methods are all about specifications.

The main activities included in formal methods are:

- Writing a formal specification.
- Proving properties about the specification.
- Constructing a program by mathematically manipulating the specification.
- Verifying a program by mathematical argument.

Program verification is only one aspect of formal methods.

From an economic point of view, the most important part of a formal development is the system specification. For many projects, this is the only part of the development that is formal.

In any case, a formal specification of what a program is to do is a prerequisite for verifying that the program is correct.

Myth 3

Formal methods are only useful for safety-critical systems.

“The fact is that formal specifications help with *any system*.”

Probably the largest practical implementations of formal methods have been in non-critical systems.

Formal methods should be used wherever the cost of failure is high. Such systems include those that are:

- critical in some way.
- replicated many times.
- fixed into hardware.
- dependent on quality for commercial reasons.

Myth 4

Formal methods require highly trained mathematicians.

The fact is that mathematics for specification is easy.

A much higher level of mathematical skill is needed if you intend to go beyond formal specification and carry out a fully formal development that includes proofs.

Therefore, competent people who can cope with the necessary mathematical manipulations are the ones who must carry out safety-critical projects. Of course, the same is true of bridge building.

Myth 5

Formal methods increase the cost of development.

“The fact is that formal methods *decrease* the cost of development.

A completely formal development is indeed very expensive. But because many benefits come from just writing formal specifications, it is important to know if this too is costly.

- From the authors experience working on the “CASE” project for the company Praxis, writing a formal specification led to higher productivity.
- Rolls-Royce and Associates has reported that on a safety-critical project where it used formal specification and planned testing, it achieved better productivity figures than when it used neither.
- IBM has highlighted the cost of learning to use a formal method as an important one-time cost.

Myth 6

Formal methods are unacceptable to users.

The fact is that formal methods help users understand what they are getting. To realize this benefit, you must make the formal specification comprehensible to the user.

This is achieved mainly by paraphrasing the specification into natural language,

Myth 7

Formal methods are not used on real large scale software.

The fact is that formal methods are used daily on industrial projects. For example:

- Transaction processing. IBM's CISC is a large twenty year old transaction-processing-system. It contains more than half a million lines of code. IBM is using Z to respecify key CISC interfaces to improve its maintainability.

- Hardware: there are at least 3 examples of Z being used to specify hardware
  - The Secure Multiprocessing of Information by Type Environment secure computer architecture.
  - SMITE's order code has been specified in Z by the British company Plessey. The floating-point for the transputer was specified in Z.
  - Tetronix has been using Z to specify the functionality of oscilloscope families.
- Compilers. The Danish Datamitik Center has for many years been developing industrial compilers using formal methods.

- Software tools.
  - The CASE project on which the author worked is one example.
  - The interface to the Portable Common Tools Environment, a European standard for software engineering.

- Reactor controls. Rolls-Royce and Associates used a combination of English and formal specification to specify nuclear-reactor software control.

Seven Facts

- Formal methods are very helpful at finding errors early on and can nearly eliminate certain classes of errors.
- They work largely by making you think very hard about the system you propose to build.
- They are useful for almost any application.
- They are based on mathematical specifications, which are much easier to understand than programs.
- They can decrease the cost of development.
- They can help clients understand what they are buying.
- They are being used successfully on practical projects in industry.

Conclusions, Part 1

- Many of the claims against formal methods are simply not true, and others are at the very least, exaggerated.
- Whether you are an advocate or a detractor, you may find yourself in a situation which requires the use or formal methods. Therefore it is probably wise to at least be familiar with the common principles of formal methods.
- Don't be afraid of formal methods. Working with them might not be as bad as you imagine.

Conclusions, Part 1

- Keep in mind that Anthony Hall is clearly an advocate of formal methods, (he is a software engineer that specializes in the use of formal methods, especially the Z notation. His status as a specialist is only relevant as long as his specialty maintains relevance).

- The seven claims mentioned (and refuted) in this article may indeed not be true. However, this does not indicate that other claims against formal methods are also false.

Seven More Myths of Formal Methods

IEEE Software 12(4), pp. 34-41
Jul 1995

Written by Jonathan P. Bowen
and Michael G. Hinchey

About the authors:

Jonathan P. Bowen FBCS (Fellow of the British Computer Society) is a British computer scientist.

He is Chairman of Museophile Limited, an Emeritus Professor at London South Bank University where he has headed the Centre for Applied Formal Methods.

Michael G. Hinchey is an Irish computer scientist and Director at the Irish Software Engineering Research Centre Lero, University of Limerick, Ireland.

Mike Hinchey studied at the University of Limerick as an undergraduate (was the leading student in his graduating year), Oxford University (at Wolfson College) for his MSc and Cambridge University (at St John's College) for his PhD.

(I couldn't find a picture)

Reason behind the article

“Many of Hall’s myths were - and we believe to a certain extent still are - propagated by the media. Fortunately, today these myths are held more by the public and the computer-science community at large than by system developers. It is our concern, however, that new myths are being propagated, and more alarmingly, are receiving a certain tacit from the system-development community.” (Fourth paragraph of the article).

Myth 8

Formal methods delay the development process.

Estimating development time is a difficult matter. A number of models have been developed to cover cost and development estimation. These models must be based on historical information.

Some formal methods projects were delayed. However, this is associated with the developers lack of experience in determining how long development should take, not with a lack of ability.

Myth 9

Formal methods lack tools.

This claim might have been true in the past, but this is no longer the case.

Today, many projects place great emphasis on tool support.

Myth 10

Formal methods replace traditional engineering design methods.

One of the major criticisms against formal methods is that they fail to support many of the methodological aspects of the more traditional structured-development methods. This statement is *partially* true of *some* types of formal methods. Structured-development methods using a model such as Bohem's spiral model generally support all stages of the system.

Furthermore, today, a major area of research is integration of structured and formal methods. The result is that two views of the system are presented.

Approaches to method integration vary from running structured and formal methods in parallel, to formally specifying transformations from structured-method notations to formal specification languages.

Myth 11

Formal methods only apply to software.

Though not explicitly mentioned as a myth in the first article, it was also refuted by (counter) examples for myth 7: “Formal methods are not used on real large scale software.”

Additional Examples

- The HOL theorem prover was used to verify parts of the Viper microprocessor.
- Other theorem provers that have been used to verify hardware are: Boyer-Moore, Esterel, Nuprl, 2OBJ, Occam Transformation System, and Veritas proof tools.
- Computational Logic's FM9001 has been verified down to gate-level netlist representation using the Boyer-Moore theorem prover

Myth 12

Formal methods are unnecessary.

There are occasions in which formal methods are “overkill”, however, sometimes they are desirable.

Sometimes, they are not just desirable, but required. Many standard bodies have not only used formal specification languages in making their own standards unambiguous, but have mandated or strongly recommended the use of formal methods in certain classes of applications.

Myth 13

Formal methods are not supported.

This claim may have been true in the past, however, today, support for formal methods is indisputable. If media attention is anything to go by, interest in formal methods has grown phenomenally.

“Along with object orientation, formal methods have quickly become great buzzwords in the computer industry.”

Myth 14

Formal-methods people always use formal methods.

Even the most fervent supporters of formal methods recognize that other approaches are sometimes better. In user-interface design, for example,it is very difficult for the developer to determine, and thus formalize, the exact requirements of human-computer interaction at the outset of a project.

“There are many other areas in which, although possible, formalization is impractical because of resources, time, or money. Most successful formal-methods projects involve the application of formal methods to critical portions of system development. Only rarely are formal methods alone applied to all aspects of system development.

Even within IBM’s-CISC project -which is often cited as a major successful application of formal methods-only about one-tenth of the entire system was actually subjected to formal techniques (although this still involved hundreds of thousands of lines of code and thousands of pages of specifications).”

Conclusions, Part 2

- The authors conclude that in order to facilitate the technology-transfer process from formal-methods research to practice, more *real* links between the industry and the academia are required.
- Also, the successful use of formal methods must be better publicized
- More research is required to further develop the use of formal methods.
- Formal methods are not a panacea, but one method among many.

Questions