



Formal Methods start to add up once again

Anthony Hall

Publication notes

Copyright © Computing

Published in the 8th January 2004 issue

Formal methods start

In our third *Déjà Vu* piece, **Richard Sharpe** looks at using maths as a basis for software

déjà vu

THE chances are high that before I have finished writing this article, the dreaded 'illegal operation' window will open.

I will swear at the screen and mumble: 'If they built roads, bridges, cars, planes and ships like this software, the human race would be doomed.'

Yet software is now at the heart of many of these products. Why are Airbuses, with their fly-by-wire technology, not falling out of the sky as regularly as Windows crashes? Why don't the unmanned Paris Metro trains crash into each other every day?

Part of the reason is that such safety-critical software is developed using a very old system whose time may be coming again: formal methods. Using our rear-view mirror of *déjà vu*, we see that the technique may be about to undergo one of its periodic revivals.

Theory and Praxis

Aircraft designers use mathematics to model the complex systems of lift and thrust needed to keep an Airbus in the sky. Bridge designers use mathematics to assess the stresses on the materials from which they can build bridges. Regulators of these industries demand that designers use rigorous methods.

Formal methods is the equivalent mathematical foundation for software. It is the use of mathematics to specify, model, develop and reason about computing systems. That is both its strength and its weakness.

The weakness is because there are far too few programmers with a background in mathematics who are comfortable with the notation at the heart of formal methods. Show any self-respecting programmer an example of formal notation and they may run for the hills (*see right: an example of Z code formal notation*).

'There is big psychological resistance – people don't like mathematics. I don't know how to solve that problem,' says Anthony Hall, formal methods advocate and principal consultant at Praxis Critical Systems.

Professor Bernie Cohen, professor of computing at City University, is another long-term supporter of formal methods, but warns that the technique requires 'considerable mathematical consideration'.

All formal methods advocates accept that it delays coding. After all, there's the

old joke about the programming manager announcing to the programming team: 'They want a new system. You start coding and I'll see what they want.'

Speed of coding – programmer productivity – has often overshadowed the need for software to really do what it is specified to do.

Yet the pioneering giants of the computer world were often steeped in mathematics and the formality of notation. You can't get more *déjà vu* on the development of electronically-based computing than the work of Alan Turing and John von Neumann, honoured in the terms 'the Turing machine' and 'the von Neumann architecture'. Both were world-class mathematicians. And both used or advocated mathematically-based software development.

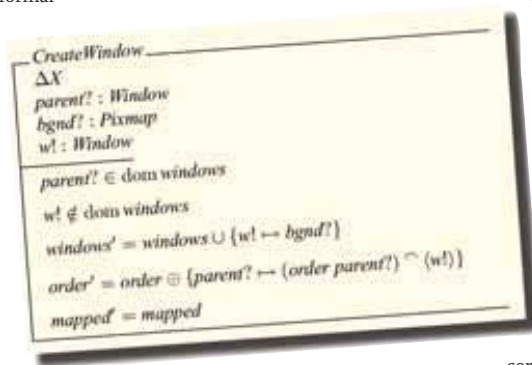
Where did we go wrong?

Many of the early programmers of the 1940s and 1950s were women recruited into the industry because of their mathematical abilities: pioneers such as Adele Goldstine and Grace Hopper.

Indeed, the very first programmer, Ada Augusta Lovelace, wrote proofs of her programmes which she wrote for Charles Babbage's revolutionary engines in the 1830s.

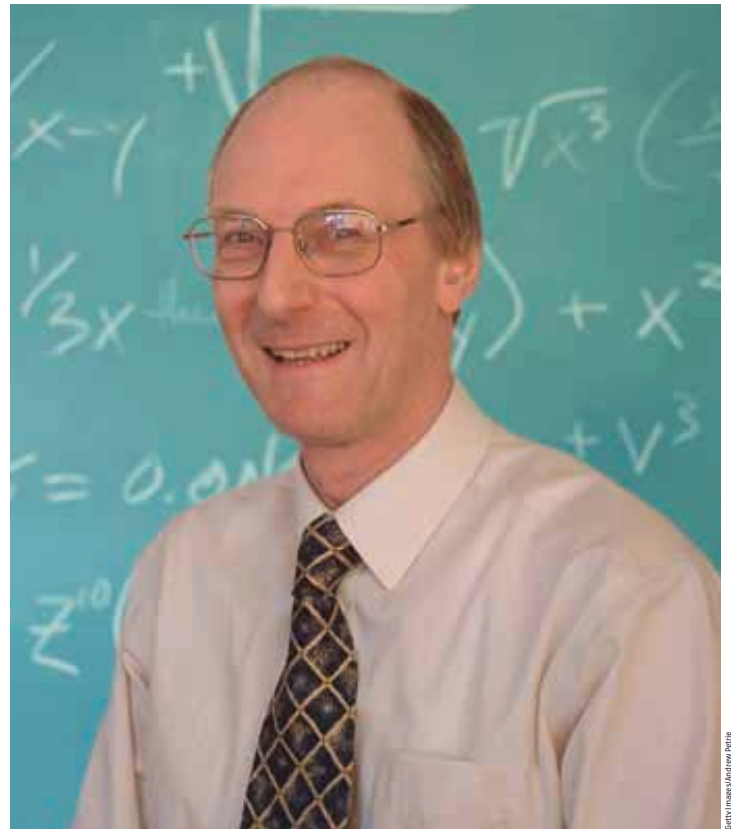
Why, then, did such an obvious approach to proving software can do what it is designed to do fail to catch on in mainstream programming? Why, after 55 years of electronic computer technology, are we writing that formal methods may be in for a *revival*?

The expanding fields of computing applications outpaced the predictions of the pioneers. Few computers became many computers, which became ubiquitous computers.



There are not enough people with the mathematical expertise to provide the code needed to exploit the success delivered by Moore's Law in hardware.

New software developments and architectural changes have also deflected



Hall: 'Formal methods are being introduced by stealth'

attention from formal methods.

Highly concurrent systems are a real challenge to formal methods. It's not easy to scale up from modelling one task to modelling many tasks.

Perhaps the single-programming operating system DOS could be modelled with formal methods – but could Windows, with its multiple, unpredictable and concurrent applications? A formal method need not be used for the whole of an operating system or piece of software. It could be used for vital parts, such as Windows' memory management.

Another obstacle to establishing formal methods in the mainstream may be the rise of object-oriented programming.

A pioneer of the definition of objects, Dave Parnas, is something of an opponent of formal systems. The set theory within a software object should be encapsulated, hidden from the outside world.

Software parading its mathematical correctness in front of the world is open code, not the sealed shell of object software.

Formal methods is heavily used in the development of military systems.

Practitioners don't talk in public about it, so the message does not get out to the

wider world of software development.

A further obstacle is the rise of Microsoft's development approach. There is no mention of formal methods in the history of Microsoft's development as a

'There is psychological resistance – people don't like mathematics. I don't know how to solve that problem'

company. Bill Gates took a mathematics course at university, but much of his energy was directed towards computers – the tool itself, not the formal foundation for its application.

A big name steps in

Gates' belief in the heroic programmer, the pure hacker cracking code, lies at the heart of a strong programming tradition today. Yet Microsoft is starting to use formal methods.

'Microsoft gets to code very quickly,' says Tom Ball, a senior researcher at Microsoft who is researching testing and verification.

'It is a problem. But formal methods tools are starting to have an impact. When

to add up once again

people see the tools used, they see a precedent created: programmers are now more willing to annotate the source code so that formal methods tools can find classes of problems faster.'

Ball has developed SLAM, a bug-finding tool based on formal methods now in use at Microsoft. Another researcher at the company has

A formal method need not be used for the whole of an operating system or piece of software

developed a tool called Abstract State Machines Language.

Gates has clearly stated that the main focus of Microsoft development is now security.

This emphasis may make Microsoft take formal methods more seriously in the future. IBM was under similar pressure to prove security when it used Z, a formal method, to formally define the Cics teleprocessing monitor in the late 1980s.

A whole lot of history

But the adoption of formal methods may not be as formal as its advocates would like.

Gates made his first pile of dollars developing compilers and interpreters for Basic. No formal methods there.

But City University's Cohen says that no compiler writer worth his or her salt would write a compiler today without using Bachus Naur Form, a formal method of describing language constructs devised in the late 1950s.

Without being told they are using mathematics, many software developers are using formal notation to describe the software they are trying to build. 'Formal methods are being introduced by stealth,' says Praxis's Hall.

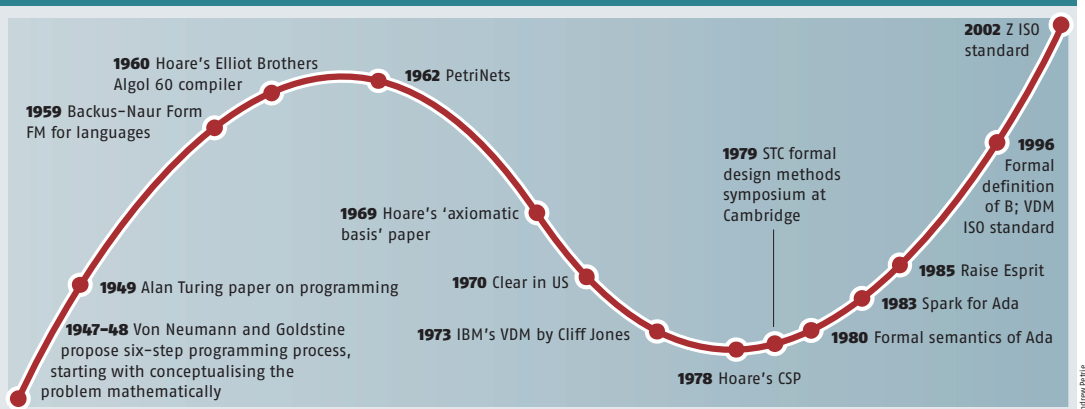
For example, developers devising relational databases use relational algebra conceived by Ted Codd, himself a mathematician and subject of a previous Déjà Vu (*Computing*, 14 August 2003).

'Formal modelling is exactly as important as it used to be,' says Cohen. He came from industry and transferred into academia.

Profiled in *Computing* way back in 1984 (see right), his advocacy for formal methods was such that the article's writer, Tony Durham, said: 'No one states the case for formal methods better than Bernie Cohen.'

Today Cohen recognises that the use of formal methods was oversold. Its advocates claimed that programs could be automatically generated from the formal definition. 'And then they discovered to their horror they could not do it,' he says.

Formal methods: a history



'It starts up again when some new generation of tool support comes along to make it a bit easier'

THERE is a rich history of links between industry and academia on formal methods, a history which undermines the claim that formal methods are just an academic fascination.

One of the first to make that transition, well before Bernie Cohen, was Professor Sir Tony Hoare. He worked from 1960 to 1968 at Elliot Brothers, a pioneering UK company which developed computers and was later, by several steps, merged into ICL.

A high point for formal methods was Hoare's development of an Algol compiler implementing the Algol 60 definition. Hoare left Elliot Brothers in 1968 and eventually arrived at Oxford's Computing Laboratory, a consistent centre of expertise in formal methods.

Professor Jonathan Bowen, one of the world's leading advocates of formal methods, is from this Oxford stable. Bowen chairs the BCS group on the formal aspects of computing science, and is chairman of the Z user group. Z is a formal method

which was most used in the 1980s.

Z, developed in the late 1970s, contends with the other formal methods available. Today they include B in variations from the UK and France; VDM, from IBM's Vienna Research Laboratory in the mid 1970s; Raise, developed as part of an Esprit project in the mid-1980s; and extensions to the Ada programming language to provide it with formal semantics in the 1980s. There are many more.

Each is supported by a devoted band of adherents who keep their faith through internet connections. Some even work on the similarities between different approaches. Two of these – Z and VDM – have risen to the level of ISO standards.

Perhaps the very variety of approaches has fragmented the effect formal methods could have had. But Cohen warns that the demand to fix on one standard does not work. Formal standards, such as mandating the use of Coral 66 or Ada, all failed, he says.

Bowen adds, perhaps with regret but with his extensive knowledge of the subject, that the use of formal methods is fairly level.

'There is no dramatic move in any direction. But I'm happy it's not decreasing,' he says.

Bowen holds to the belief that it would be better to develop software without bugs rather than write software with lots of bugs and spend lots of time getting it out.

Hall, at Praxis, agrees. 'A formal methods proof can be more economical than certain types of testing,' he says.

There are more advantages. 'Programs will be several orders of magnitude smaller if people use formal methods,' says Bowen.

There are commercial successes. Praxis claims a string of its own developments using formal methods which were more economical than other approaches.

Susan Stepney, formerly of Logica and

now professor of computer science at York University, says that formal methods were used in Logica in a few key critical systems projects; for example, the Mondex and Multos smartcard projects. So why has it not been taken up yet again?

'I think it periodically stalls, as it is found by each new generation of practitioners to be "difficult",' says Stepney. 'Then it starts up again when some new generation of tool support comes along to make it a bit easier. The B-tool helped. Modern model checkers are helping again.'

The latest help may come from raw computing power. To 'prove' a piece of software requires the technical skills of a mathematician, a daunting task for those programmers interested in hacking out code. But increasingly powerful software can allow a computer to work through the proofs fast enough to be useful. This is the approach of David Crocker, founder of Escher Technologies.

Escher has just launched Version 2.10 of its Perfect Developer Tool. Developers can import UML models, add the detailed specification, and verify that it performs in the way the developer wants. Crocker has used contemporary automated reasoning research to automate the proofing process.

This cuts the human effort required to get to the proof that the software will perform as specified. Escher's tool may prove to be one of the regular historic high points of interest in formal methods.

There is still a lot of ground to be made up before formal methods can have another chance to capture the mainstream of software development. Tom Ball at Microsoft says It has been more popular in Europe than the US because the European software industry is more focused on different types of applications rather than the general development of commercial software.



Formal approach: *Computing*, 1984