Editorial

Rhetoric versus modernism in computing

Inventing a new logic and investigating its semantics or proof theory is a thriving branch of theoretical computer science. Much work is also done on decidability and complexity aspects. But little attention is paid to how these new logics are going to be *used*. Of course one cannot expect such information in the existing publications, as they are concerned with the preliminary, necessarily theoretical phase of research into the new logic.

But the application for funding of the research cannot avoid this question. There the usual justification is the formal, automatic correctness proof of safety-critical software. It is taken for granted that no more needs to be said. Aren't computer programs notoriously error-prone? Isn't it inevitable that we are going to trust our lives more and more to automatic equipment controlled by computer programs? Isn't it then equally inevitable that such software is subjected to automatic proof of correctness which will then eliminate the possibility of malfunction due to software error?

The answer to such questions is that it has been tried for a long time and, so far, has failed to have any practical impact. Even though the first attempts started over twenty years ago, the lack of success so far does not of course mean that automatic program verification is impossible and undesirable. But past experience should be examined. I doubt whether this has been done enough. In this editorial I would like to contribute to such an examination.

In the late 1960s, when the difficulty of building large software systems correctly and within budget became clear,[1] projects were funded that were devoted to formally proving, by means of a computer, that a program meets its specification. I trust that these projects were duly declared a success. But what counts is that so far, over twenty years later, they have not had any impact on the way software is built. This failure has attracted the attention of Richard De Millo, Richard Lipton, and the late Alan Perlis [4], who described the use of an automated verification system in a thought-experiment roughly as follows.

> Imagine that such a proof system has been implemented. It is unrealistic to assume that it is much shorter (if at all) than, say, ten thousand lines of code. Moreover, it is limited to proving correctness of small programs (say, a few hundred lines). You feed the prover such a small program, with its specification, go out for lunch, and find on return an enormous amount of unintelligible output. This may be a proof. You skip to the end, where it says: QED. So now you know it *is* a proof.
>
> The question arises: does this episode raise your confidence in the program subjected to this treatment; does it help you to decide whether it is justified to include it in a safety-critical application such as a controller of an aircraft or of a cardiac pacemaker? To start with, the verification program itself is not verified. Moreover, the specification is a dense piece of formalism, not much shorter than the program it specifies and as much subject to error as the program itself.

[1] The difficulty became known as the 'Software Crisis'; see [5] for the first meeting devoted to this topic.

This was approximately the thought experiment of De Millo, Lipton, and Perlis. Since then increase in speed of computers has had the effect that the running of such a program can be scheduled independently of lunch. And of course, it is the aim of research into new logics to make the specification considerably shorter than the program and to make it less subject to error. But the main question remains unchanged: should the formal, automatically generated proof influence a responsible professional's decision to include the program in a safety-critical application? Such a professional will need to *understand* the proof and will take into account that neither the compiler nor the operating system under which the program is to run have been verified.

This thought experiment could easily have been performed before any automatic verification project was funded. Several were. None has had an impact on the way software is built, so far at least. Indeed, after a quarter of a century, the Software Crisis continues to rage unabated. If this is too strong a claim, it is not because of advances in formal verification of programs, but because of the introduction of *code inspections* in software development. In this approach, programmers are not permitted to run their code. Instead, they are required to revise it until they are confident of its correctness. Then it is subjected to an *inspection*, a formal meeting with a few peers, in which a participant goes through the code, paraphrasing small bits at a time, leading an attempt by all present to detect errors. This practice, though still not widespread, has been shown [1, 7] to be effective in improving the quality of software compared to the still prevalent approach where no inspections are conducted, and where confidence is gained by failure to find faults during testing.

This is the contrast I want to consider here: on the one hand the approach via verification, relying on formal methods, which turned out to be costly and fruitless. On the other hand the approach via inspections, requiring no science, no new methods, only common sense. The latter turns out to be effective. How is it possible that the effective, easy, cheap and obvious was overlooked in favour of the difficult, expensive, and possibly impotent? This can only be explained by a mindset which turns the obvious into the opposite. If this is indeed so, the mindset must influence affairs other than those related to computing. In the following I quote evidence that this is indeed the case, that indeed the mindset is so pervasive as to have been identified elsewhere well before the first attempts at formal verification of programs, that the belated identification in computing brands this field as a backwater in the world of ideas. The mindset is called 'modernism'; its sane alternative goes by the name of 'rhetoric'. The success of inspections is a triumph in computing of rhetoric over its modernist alternative.

Of rhetoric I will say no more than the obvious: that 'specious eloquence' is not its primary meaning. In one of C. P. Snow's Two Cultures it is not necessary to make this point. In the other, it very much needs to be made. Rhetoric, then, is the *art of persuasion*. This is the case independently of whether the motive for persuasion is honourable. The primary tool of rhetoric is a natural language. In certain specialized and limited contexts graphs, tables and algebraic expressions are supporting tools of rhetoric. It is the modernist delusion to believe that such specialized and limited contexts are strongholds from which the entire range of rational discourse is to be conquered.

More needs to be said about modernism. My starting point has been a paper by

Donald McCloskey entitled 'The Rhetoric of Economics' [3]. McCloskey argues that rhetoric should be the main intellectual tool of economists, but that this central position has been usurped by a complex of attitudes and techniques for which McCloskey uses the term 'modernism', which he introduces in the following way:

> (It) is an amalgam of logical positivism, behaviourism, operationalism, and the hypothetico-deductive model of science. Its leading idea is that all sure knowledge is modeled on the early 20th century's understanding of certain pieces of 19th-century physics. To emphasize its pervasiveness in modern thinking well beyond scholarship it is best labeled simply 'modernism', that is, the notion ... that we know only what we cannot doubt and cannot really know what we can merely assent to.

By now modernism has achieved a venerable old age. Here are modernist quotes; one each from the 17th, 18th, 19th and 20th centuries.

In the 17th century, symbolic logic did not exist. Leibniz seems to have been the first to see its possibility. He was optimistic about its range of applicability, writing (quoted in [6]):

> If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, to sit down to their slates, and to say to each other (with a friend as witness, if they liked): Let us calculate.

Moving to the 18th century, I have the following from David Hume (quoted in [3]):

> When we run over libraries, persuaded of these principles, what havoc must we make? If we take in our hand any volume — of divinity or school metaphysics, for instance — let us ask, *Does it contain any abstract reasoning concerning quantity or number?* No. *Does it contain any experimental reasoning concerning matter of fact and existence?* No. Commit it then to the flames, for it can contain nothing but sophistry and illusion. (Italics Hume's.)

Lord Kelvin: 'When you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind' (cited in [3]). As representative of the 20th century, I select Wittgenstein who wrote: 'Anything that can be said at all, can be said clearly. And of what cannot be said clearly, one must not speak'. A postmodernist retort (Michael Polyani in 1962, as reported by McCloskey [3]): the methodology of modernism sets up 'quixotic standards of valid meaning which, if rigorously practiced, would reduce us all to voluntary imbecility'.

McCloskey was concerned with the ill effects of modernism in economics. Susanne Langer [2] makes a similar observation for psychology:

> Psychologists have probably spent almost as much time and type avowing their empiricism, their factual premises, their experimental techniques, as recording experiments and making general inductions.

A few pages later:

> The physicists' scheme, so faithfully emulated by generations of psychologists, epistemologists, and aestheticians, is probably blocking their progress, defeating possible insights by its prejudicial force. The scheme is not false ... but

it is bootless for the study of mental phenomena ... Instead of a method, it inspires a militant methodology.

And it is this militant methodology that goes under the name of 'modernism'.

And not only psychologists, epistemologists, and aestheticians suffer self-inflicted debilitation by adopting modernist trappings. The modernist urge to derive respectability from the formal, the abstract, and the numerical has given, for example, matrix and graph theory an unwarranted prominence in economics, sociology, psychology, management science, and computer science. These crypto-mathematicians follow a modified form of Kelvin's dictum: 'When you cannot express it as a mathematical theorem, your knowledge is of a meagre and unsatisfactory kind'. without realizing that, from the point of view of their nominal discipline, if it *can* be expressed as a mathematical theorem, the knowledge is of a meagre and unsatisfactory kind.[2]

Mathematics plays a crucial role in the understanding of the tension between modernism and rhetoric, but is poorly understood. On the one hand the modernist mind has seen mathematics as an example to be followed: surely here is the ultimate in precise and secure knowledge, an example to be emulated by all scholarship. On the other hand, it has escaped the modernist that mathematics is still done by rather old-fashioned rhetoric. Where the psychologist is a heavy user of computers (for computation), the mathematician has no such need. Hilbert's program, a typically modernist proposal, has never been followed in mathematics. However, it is the direct precursor of the program verification enterprise in computer science.

I do not want to claim that the only possible language of rhetoric is a purely natural one. Algebraic notation, introduced centuries ago, is now very much accepted in the rich, unmodernist rhetoric of mathematics. It is important to realize that algebraic notation can be used as extension to natural language to clarify what would be more obscure otherwise. But such use need not be part of a formal method. Algebraic notation can be used in rigorous, informal rhetoric. At one time, then, a successful shift has been made to incorporate algebraic notation into the language of rhetoric. On the one hand, the modernist excesses of the past century must be rectified. On the other hand this must not imply that rhetoric should forever be shielded from further enrichment by modest incorporation of additional algebraic notation. But we must keep in mind that *modest* steps are called for: so far anything beyond English enriched by formulas, graphs, and tables has proved to be sterile.

To return to the success of rhetoric in computing in the form of the code inspection, I note that in its reported form, inspections are traditional rhetoric, unaided by any symbolic logic. I believe that an informal version of some program verification techniques, such as Floyd's method of assertions, can make inspections more effective [8]. My guess is that the optimum consists of including assertions into code, which consist of English fortified by logic formulas. This is far from a formal system and much like the traditional rhetoric of mathematics.

In conclusion, let me summarize by saying that logic, formal or not, is only justified as a tool of rhetoric, that is, as a tool in the art of persuasion. So far, automatic program verification has been an example of a use of formal logic that fails to persuade and hence misses any use it could possibly have. The serious consideration of such a bootless exercise is a manifestation in computing of the ubiquitous mindset that is

[2]This retort against the modernists is inspired by Frank Knight, quoted in [3].

called 'modernism'. Its ancient and forever modern antidote is rhetoric, which has recently penetrated into computing in the form of code inspections.

If formal logic is ever to have a use in improving the quality of software by means of verification, it will have to be as an adjunct in a process of rhetoric that persuades a skeptical and knowledgeable expert, who is personally accountable as a professional, that the use of such software is justified.

References

[1] Michael Dyer. *The Cleanroom Approach to Quality Software Development.* John Wiley and Sons, 1992.

[2] Susanne Langer. *Philosophy in a New Key.* Harvard University Press, 1948.

[3] Donald N. McCloskey. The rhetoric of economics. *The Journal of Economic Literature*, **21**, 481–515, 1983.

[4] Richard A. De Millo, Richard J. Lipton, and Alan J. Perlis. Social processes and proofs of theorems. *Communications of the ACM*, **22**, 271–280, 1979.

[5] Peter Naur and Brian Randell, editors. *Software Engineering.* NATO Scientific Affairs Division, 1969.

[6] Bertrand Russell. *A History of Western Philosophy.* George Allen and Unwin, 1946.

[7] Glen W. Russell. Experience with inspection in ultralarge-scale developments. *IEEE Software*, pp. 25–31, 1991.

[8] M.H. van Emden. Structured inspections of code. Technical Report DCS-165-IR, Department of Computer Science,University of Victoria, 1991.

M. VAN EMDEN
University of Victoria, Canada