The Role of Formalism in Method

Michael Jackson

101 Hamilton Terrace, London NW8 9QY, England
AT&T Research, 180 Park Avenue, Florham Park NJ 07932, USA
jacksonma@acm.org | mj@doc.ic.ac.uk

Wider use of formal methods in the development of computing systems promises better quality in general, and in particular safer and more reliable systems. But we must recognise that formalisation and formal reasoning are not goals in themselves: they are partial means to our goals.

Any useful computing system interacts with the world outside the computer. In problems of a mathematical nature, such as problems in graph theory or cryptography, interaction with the world is a negligible part of the problem: the gravamen lies in the manipulation of the formal mathematical abstractions. But in many problems the balance is different: the quality of the solution must be judged by its effects in the physical world of people, chemical plants, lifts, libraries, telephone networks, aeroplanes—or wherever the problem is located. Unlike the abstract world of mathematics, these real-world domains are not formal; it is not always obvious how—or even whether—formal methods can and should be applied to them.

One possible response is to exclude such domains from the scope of development method. Let the domain experts determine what is required in the domain, what domain properties can be exploited, and what computer behaviour at the interface with the world will satisfy the requirements. We, as software developers, will start from a specification of that computer behaviour, carefully avoiding the messy and irritating informality of the world outside the computer.

But this response is ineffective for two related reasons. First, a specification strictly confined to the computer behaviour at the interface is unintelligible. Such a specification of software to control a lift would be expressed purely in terms of values of shared registers and signals on lines connecting the computer to the electro-mechanical equipment of the lift. No-one could understand such a specification. Second, the domain experts, unaided, are scarcely ever able to produce such a specification to the required degree of formality and precision. As software developers we are compelled to involve ourselves in their work.

We therefore cannot escape a major share of responsibility for determining and bounding the problem that the system is to solve in the world. Inevitably we are led into classifying and structuring real-world problems, and into formalising real-world domains and reasoning about them within the large structures appropriate to the class of problem in hand. We must understand and practise effective techniques of formalising informal domains, and recognise the approximate nature of the resulting formalisations and the limitations of formal reasoning based upon them. And in dealing with the world outside the computer, we must accept that formalisation and formal methods can show the presence of errors, but never their absence.

J. Wing, J. Woodcock, J. Davies (Eds.): FM'99, Vol. I, LNCS 1708, pp. 56–56, 1999.
© Springer-Verlag Berlin Heidelberg 1999