# Formal Methods & Traditional Engineering: by Michael Jackson

## Introduction

Formal methods have not been taken up by industry to the extent that their creators and advocates think desirable. Certainly there have been some important successes. The Praxis CDIS development, the IBM Hursley redevelopment of CICS, the TCAS system, the INMOS T9000 transputer and a number of others are well known; but their very notability indicates how few convincing examples there are. And even among the claimed successes some, such as the CICS project, used a formal notation but made no use of the opportunities for formal analysis, calculation or proof.

And yet the need for the increased rigour and precision that formal methods seem to offer is indisputable in many parts of many system developments. Use of formal methods is sometimes likened to the use of mathematical calculations in traditional engineering. The bridge engineer calculates weights and stresses; the software engineer should similarly calculate preconditions, or prove the absence of deadlock in communicating processes, or demonstrate rigorously that the invariant of a data structure is maintained by the operations on the data. Such calculations, and the model-checking that can sometimes take their place, are possible only if the relevant part of the development has been described formally: informal descriptions admit neither calculation nor reliable analysis and checking.

Advocates of formal methods have characterised the slow and reluctant take-up as a technology transfer problem, implying that the technologies of Z and Larch and CSP and VDM and Lotos and many other formal methods are ready and waiting, and that it remains only to convince and train the people whose work will certainly become easier and more effective by the use of such methods.

## Constructive Methods

Yet although the claimed advantages of formal methods are correctness and reliability, the methods are implicitly presented as *constructive* development methods — that is, methods for describing new problems, and inventing and designing novel solution structures within which correctness and reliability can be assured. This constructive development task is very different from the description and analysis of what has already been invented or designed, or the making of detailed design calculations within a tightly constrained framework. Its goal is the creation of new structures for problems and their solutions.

This implicit claim to play a constructive role is firmly within the tradition of commercial structured and object-oriented methods, and of the more academic approaches such as stepwise refinement. Yet neither formal methods nor their precursors in the tradition can offer real constructive help to the developer. Constructive methods can be effective only in narrowly restricted areas, where they can help the developer to create solution structures of a limited class by manipulating and combining similarly limited problem structures. In broader areas, where realistic problems are found, there is little hope of devising or finding general constructive methods: the constructive task itself is too difficult and unstructured.

The inability of formal methods to play a constructive role is not objectively a defect in comparison with most or all of the informal methods they might seek to displace. But nonetheless it works strongly to their subjective disadvantage. By allowing the comparative evaluation to include an assessment of a method’s ability to play a constructive role, advocates of formal methods have relegated their real advantages of precision and

reliability to a secondary place in the estimation of most potential users. Development managers responsible for safety-critical systems are likely to value correctness highly enough to adopt a formal method. But most others are more likely to feel that they have already burnt their fingers in the ‘structured revolution’ and have no zeal for further combustible experiments with a new formal revolution.

## Universal Methods

Formal methods are also presented implicitly as universal, or nearly universal, methods. Of course, it is recognised that some formal methods such as Lotos and Esterel are ‘for real-time systems’; that some others, such as Z and VDM, do not handle concurrency and are best suited to systems with a complex and elaborate global state. But these qualifications are no more than discreetly acknowledged minimal departures from a default presumption of universal applicability. No mainstream formal method carries a maker’s disclaimer that it is useful only for a small and narrowly defined class of problem. And there is little or no provision for combining different formal methods within a single development: that is a research topic, not a common and well-supported practice. Indeed, if the use of a formal method is seen as underpinned by explicit formal semantics, the fact that different methods have different semantic domains seems to make any combined use completely impossible.

Again, formal methods advocates are following in the footsteps of the promoters of the structured, object-oriented and stepwise methods they seek to replace. And again they are being led to accept evaluation criteria that relegate their real advantages to a place of insignificance. A universal method can not be effective, because by its nature it can not take advantage of any particular features of the problem in hand. Worse still, it must abstract from any feature whose universal treatment is simply too hard: this is why most or all formal methods abstract from the control properties of events and state changes.

Universal methods arose initially out of the immaturity of the new field of software development. Later they persisted because the buyers of development methods are most often development managers: beset by problems on every side, these managers are desperately anxious to simplify wherever simplification — or oversimplification — seems possible. The methodology problem, at least, seemed reducible to a multiple choice: whether to buy development method A, B or C. In this tendering process a vendor offering only a sharply focused method of restricted applicability is at a catastrophic disadvantage; understandably, few method vendors have rushed to disqualify themselves in this way.

## What Real Engineers Do

Advocates of formal methods often rightly draw on a comparison with the practitioners of traditional established engineering disciplines. Bridge and aeroplane designers calculate: designers of software should therefore do so too. The argument seems compelling. But the analogy with traditional engineering is not straightforward: it carries a hidden lesson.

Traditional engineers do not use constructive methods for large decisions; nor do they use universal methods. They work on traditional problems for which they have developed traditional solutions. A new bridge or car or aeroplane or TV set is very closely similar to its predecessors, and shares the same structuring of problem and solution. The traditional engineer does not start with a clean sheet of paper. Successful designs evolve over many generations in communities in which many engineers are working on almost identical products.

Specialisation is the inevitable precondition and accompaniment of this evolution of

successful designs. In the earliest stages of development of a new field it is accessible to people of solid intellect and a deep knowledge of fundamental principles. But as the field grows and its products are refined through successive generations of design, the body of knowledge implicit in the current versions of the products becomes too large to be mastered by anyone who is not dedicated to the task. In the first half of the nineteenth century, a great engineer of the calibre of Brunel could design railways, tunnels, bridges and ships. Today no engineer could hope to work successfully in so many different fields: each one has simply become too rich to be mastered by any but a dedicated specialist. And within each specialised field further specialisations arise concerned with particular parts or aspects of its already highly specialised products.

The same process of specialisation can be seen in software too. Expert systems, compilers, operating systems and telephone switching systems are all fields that have developed specialised structures and solutions, and a repertoire of specialised techniques for carrying out the tasks and overcoming the difficulties that arise within them. It becomes increasingly difficult for outsiders not schooled in the field to understand the work, let alone to engage in any part of it.

## The General and the Particular

So formalism in traditional engineering is applied locally to well-understood components of well-understood characteristics. The context for applying the formalism is almost fixed, and the calculations to be made are almost standardised.

This does not mean that successful specialised fields discard the general principles of science or engineering: on the contrary, they embody them in their specialised techniques and design procedures. An engineering handbook is not a compendium of fundamental principles; but it does contain a corpus of rules and procedures by which it has been found that those principles can be most easily and effectively applied to the particular design tasks established in the field. The outline design is already given, determined by the established needs and products. For a small electrical power transformer, for example, it is already determined that there will be a laminated iron core in the shape of a figure 8, with primary and secondary windings of enamelled copper wire. The particular task facing the design engineer is to use a set of tables or a calculation procedure to determine the exact shape and size of the core and the wire gauge and number of turns for each winding.

## Where the Design Task Lies

In this context, design innovation is exceptional. Only once in a thousand car designs does the designer depart from the accepted structures by an innovation like front-wheel drive or a transversely positioned engine. True, when a radical innovation proves successful it becomes a standard design choice for later engineers. But these design choices are then made at a higher level than that of the working engineer: the product characteristics they imply soon become well understood, and their selection becomes as much a matter of marketing as of design technology. Unsuccessful innovations — like the rotary internal combustion engine — never become established as possible design choices.

The effect of this deep and inevitable conservatism in design is to divert attention from the large structural questions. The car designer does not spend time wondering whether to power the car by steam or to integrate the fuel tank into the engine block. The working engineer’s design problems are all problems of detail, and are addressed by methods focused at that level. There is no place for constructive or universal methods. The methods of value are micro-methods, closely tailored to the tasks of developing particular well-

understood parts of particular well-understood products.

## Formal Micro-Methods

The clear implication for formal method advocates is that in software development too the most useful context for the precision and reliability that formality can offer is in sharply focused micro-methods, supporting specialised small-scale tasks of analysis and detailed design. These small-scale tasks need not always be concerned with small-scale software components: they will often be concerned with a property abstracted from a large group of components, selected precisely because the property captures an important aspect of their interaction or combined behaviour, just as the weight of a car is made up of the weights of its many individual components.

Each existing formal method may be expected to generate many micro-methods, each using only a part of the formal apparatus of its parent method. It will be accompanied by a clear statement of the context in which it may be applied, most being applicable only in tightly restricted contexts. There is an analogy here with the patterns approach to object-oriented development. A pattern is a ready-made solution scheme to a recurrent design problem. It is typically small-scale, and is selected for use by recognising that it provides a solution to some aspect of a problem facing the developer. A possible objection to the work on patterns in its present form is that a typical pattern is weak in identifying the problem class to which it applies. This weakness may be tolerable in patterns, but must certainly be avoided in the development and presentation of micro-methods.

Developers and presenters of micro-methods must pay serious attention to the relationship between the problem as it appears in the environment and its description or model as it is to be used by the developer. This relationship is a crucial concern in the development of safety-critical systems and others in which correctness and reliability are vital goals. It is well known that many serious failures can be traced to errors in system requirements: often this means to the mapping between the formalisation and the reality that it purports to describe. The use of micro-methods is an applied not a pure discipline. Their effectiveness, like that of applied mathematics, depends entirely on the fidelity of the basic representation of reality to which the calculation or analysis techniques are to be applied.

## Promoting Formal Methods

When formal methods are embodied in collections of derived micro-methods there will be no scope for misunderstanding their utility and purpose. Teaching must focus on small parts or compact abstractions of real applications set in real contexts, rather than on the unrealistically simple complete systems that are the usual examples. Because the individual applications are small-scale, it will be much easier for teachers and researchers to devote enough time to working with practising developers for a real understanding of the practical problem to emerge. It will be similarly possible for a consultancy practice to become common in which senior developers working on real projects can draw on the skills of formal methods experts to help with a particular localised task.

The thrust of formal method advocacy will be the high value of a specialised micro-method in the restricted contexts to which it is applicable; large, overarching claims will come to seem absurd. In this kind of way we may hope that the real benefits that formal methods can bring will be more clearly understood and realised, and that the methods, in a more palatable and practical form, will be brought closer to the widespread use and acceptance that their advocates have for so long desired.

03/06/97