PROGRAMME ON INFORMATION & COMMUNICATION TECHNOLOGIES

Working Paper Series 30

THE EVOLUTION OF DEF STAN 00-55 AND 00-56: AN INTENSIFICATION OF THE 'FORMAL METHODS DEBATE' IN THE UK

Margaret Tierney

UNIVERSITY OF EDINBURGH

The Working Paper Series

The ESRC Programme on Information and Communication Technologies (PICT) has a strong commitment to the dissemination of research findings to a wide audience including academics, technologists, government officials, industrialists, trade unionists and community groups. As part of this commitment, Edinburgh PICT (see back cover) produces various internal publications which it distributes directly. These comprise Series of Research Reports, Working Papers and Student Papers.

The Working Paper Series aims to facilitate the fastest possible dissemination of results emanating from PICT research at Edinburgh. Specifically, the Working Papers cover three types of results

(i) preliminary results of research in progress

(ii) results about specific aspects of the Information and Communication Technologies (ICTs), representing little known information of value by way of 'background briefing'

(iii) conceptual or theoretical contributions to current policy or academic debate about technology in general or ICTs in particular

Not all of the papers in the Series originate directly from PICT projects but all reflect an association with Edinburgh PICT which has either benefited from or contributed to this research effort. The views expressed in individual papers are those of the authors and cannot be attributed to any part of the University or the ESRC.

A full list of other Working Papers in the series to date can be found at the back of this paper, along with an order form. Payments should be sent, with orders, to:

Research Centre for Social Sciences
University of Edinburgh
56 George Square
Edinburgh EH8 9JU

[redacted]

Edinburgh PICT
Working Paper No. 30

THE EVOLUTION OF DEF STAN 00-55 AND 00-56: AN INTENSIFICATION OF THE 'FORMAL METHODS DEBATE' IN THE UK

Margaret Tierney

1991

ISBN 1-872287-33-6

THE EVOLUTION OF DEF STAN 00-55 and 00-56: AN INTENSIFICATION OF THE 'FORMAL METHODS DEBATE' IN THE UK

Margaret Tierney*

This paper traces the evolution of two standards regulating the identification and production of safety-critical software for defence applications, issued by the Ministry of Defence as Interim standards in 1991. Def Stan 00-55 governs *The Procurement of Safety-Critical Software in Defence Equipment* and Def Stan 00-56 governs *Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment.* The standards - 00-55, in particular - have become an important forum for articulating the fears and aspirations of those who work in the UK safety-critical software engineering field - one which has largely revolved around the integral role 00-55 demands for **formal methods** of software development for safety-critical functions or components. In recounting the story of their gestation within the MoD during the early '80s; their controversial release in draft form in 1989; and their subsequent second release as interim standards in 1991, the object has been to illuminate some of the current 'politics' of formal methods of software production, and to consider how the standards gel - or jar - with an emergent sense of the place of formal methods within the discipline of software engineering.

The work reported on here was funded by the ESRC, under the PICT (Phase 2) Initiative, as part of a larger investiation of The Generation of Advanced Computing: Formal Methods of Software Development, being conducted at Edinburgh University by Donald MacKenzie, Eloina Pelaez and Margaret Tierney

I would like to thank all the respondents mentioned in the course of this report for their generous co-operation in allowing themselves to be interviewed, and for their useful comments on an earlier draft of this paper.

---

*Margaret Tierney is a Research Fellow in the Research Centre for Social Sciences, Edinburgh University, 56 George Square, Edinburgh EH8 9JU. Telephone (031) 650 4076. Email: M. Tierney@UK.AC.ED. (JANET)

Introduction

Standards are hardly such stuff as dreams are made on. Yet the Ministry of Defence's interim standards for *The Procurement of Safety Critical Software in Defence Equipment* (Def Stan 00-55) and *Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment* (Def Stan 00-56) embody principles for changing the face of software engineering for the safety-critical field. This paper offers a socio-historical account of the emergence of the standards. The importance of tracing their evolution is obvious in a sense. 00-55, in particular, has been the subject of considerable controversy since its release as a draft in 1989. It mandates the use of formal methods for safety-critical software development in defence applications, and reaction has principally focused on the likely effect of such methods on software engineering work in this, and related, fields.

This paper concerns itself with how the 'formal methods debate' in the UK has intensified as a consequence of the MoD's release of the standards. A socio-historical approach enables us to locate what might seem to be an exclusively technical debate within a wider range of organisational, industrial and occupational dynamics. Why should a defence standard (whatever its contents) generate such excitement and disquiet not only amongst software engineers, but also policy-makers, educationalists and professional and trade associations? In what ways are the 'formal methods community' and the 'safety-critical community' being shaped and intertwined as a consequence of 00-55? What does its focus on formal methods reveal about the direction of change of software engineering as a whole? The question addressed here is why 00-55 has become such an important forum for articulating the fears and aspirations of those who work in the UK safety-critical software engineering field.

The story told is a history of how 00-55 and 00-56 came into being in their present Interim form. I will not embark on a discussion of the detailed contents of the two standards here. In a separate paper, I consider those contents with a view to their implications for the detailed labour practices and skills of software engineers working in the formal methods field

(Tierney:1991). Rather, in recounting the events which have shaped the standards into their Interim form, this paper seeks to illuminate some of the current 'politics' of formal methods. The object is to consider how the standards gel - or jar - with an emergent sense of the place of formal methods within the discipline of software engineering. The paper begins with a consideration of how 00-55, in particular, fits into current discourse on software engineering. A history of their evolution is then traced, concluding with thoughts as to the impact of the MoD's initiative on the take-up of formal methods within the UK.

1. 00-55 as a Document about Software Engineering

Interim 00-55 offers a powerful statement of what the discipline of software engineering entails, or ought to entail. Although 00-55 is usually cited in the context of its commitment to formal methods of software production - and most discussion of the MoD's initiative has centred on the controversies stemming from that commitment - it devotes as much space to regulating the management and organisation of the development labour process as it does to the actual techniques and practices required for realising formally-based designs. To this extent, 00-55 marks a *continuity* with current debates about the scope of software engineering. It is to this issue that we first turn.

The birth of software engineering is generally traced to the identification, at the Garmisch conference of 1968, of the 'software crisis' (Pelaez:1988, Ince and Andrews:1990). This crisis refers to the way in which problems in producing 'well-made' programs have become the dominant constraint on the take-up of computer-based systems across expanding application fields over the last 25 years. That is, the *manner* in which software is written by programmers has increasingly been understood as causing, and then exacerbating, on-going difficulties in maintaining and modifying complex systems (Schlender:1989). In consequence of the identification of this crisis, the techniques and organisation of programming work have come in for sustained analysis and reform over the last two decades. Loosely speaking, software engineering is the emergent discipline which attempts to regulate *how* software is produced.

It has found its niche in formulating methods, tools and techniques which, in combination, aim to re-shape the esoteric and unsystematic 'craft' of programming into the more intellectually visible discipline of 'engineered' software production.[1] 'Engineering' in this context refers to two things: establishing a framework within which software may be developed in a systematic way as part of a total system, and regulating the management and organisation of the people who build such systems. A fundamental assumption of software engineering is that managing technology is inseparable from managing the process through which it is done - though there is huge debate about how that may best be achieved (Macro and Buxton:1987, Pelaez:1988, Friedman:1989, Ince and Andrews:1990). Nonetheless, in this regard, 00-55 is emphatically a product of the discipline which spawned it, for it insists on the careful interweaving of the 'technical' with the 'organisational' for optimising safety in software design. Its formalism is global in the sense that it not only requires adherence to particular formal design techniques, but it also demands a steady stream of documents which itemise the tasks, responsibilities and progress of named actors in defined relationships to each other.

00-55 is entirely orthodox in another sense also. Since the '70s, one of the most notable achievements of the software engineering discipline has been to propagate a new conception of the software life cycle. Ince and Andrews (1990) use the phrase 'the Mongolian Hordes approach' to describe how the software life cycle was conceived of in the pre-'70s 'craft' culture. This approach works on the assumption that finished code is always full of bugs, so the sooner finished code is produced, the quicker those bugs can be found and fixed. Working modules and sub-systems are, in theory, 'assembled up' from there. The life cycle begins with coding: to specify and develop systems without writing code would be an odd thing to do.

Dismantling this design-as-you-code approach was one of the first crusades of the emergent software engineering discipline, for it holds that design is a separate activity from coding; that software production involves more than

---

[1] This discursive opposition between the 'crafting' of software and its 'engineering' rapidly gained currency throughout the '70s.

just programming. The waterfall conception of the software life cycle, which gained orthodoxy in the '70s, is a direct outcome of this central tenet. The waterfall model suggests that the first task is to analyse the problem so as to write out the requirements for the new system; the system is then designed 'in abstract' through progressively more detailed specifications (of the system, of its programs, of the modules within those programs, and so on); and only then is code written down as the concrete downstream outcome of the detailed design specification.

The ideal waterfall model of the life cycle has rarely worked in practice; its rigid linearity from conception through to execution is sent awry by the hiccups of normal programming practice in handling changing user requirements during the design process (Friedman:1989). Nevertheless, the *idea* that programming is driven by the design specification is by now a commonplace (Ince:1990b). 00-55 stands entirely within this software engineering consensus. In its draft form of 1989, a good third of its requirements were devoted to outlining the precise conditions under which a project could move from one life cycle phase to another, with each phase rigidly demarcated from its predecessor. The Interim version of 1991 does not demand the same level of adherence to the waterfall life-cycle. Rather, it distinguishes between the management of the life cycle of a safety-critical *project* - for which it makes specific 'waterfall-like' demands - and the life cycle of software - for which it suggests that "contractors may use their customary life cycle when developing the Safety Plan, but should relate it to the project life cycle" (00-55:1991b, 4.2).

Where 00-55 has parted company from a broad-based software engineering consensus is, of course, in its requirement for the use of formal methods. By 'formal methods' I am referring less to those particular mathematically-based techniques and tools cited in Interim 00-55, but to the status of maths as the central conceptual framework for software engineering. In a sense, the discipline of mathematics has always been the "underlying reality of computer systems" (Abrahams:1987). Computers are both 'arithmetic' number-crunchers and 'logical' symbol manipulators. A defining characteristic of maths is that it is a general and abstracted system of notation: it can be applied across a wide variety of situations. At the same

time, it is a precise system of notation. It is believed to debar ambiguities from itself "with a convincing power that has no equal in any other intellectual discipline" (Pelaez:1988, p.223). However, the explicit incorporation of mathematics into solving computer systems problems has had a checkered history ever since the genesis of 'formal methods' in 1948, with the debate on whether Turing proved that a sub-routine could meet its specification (Ince:1990b).

On the one hand, there are champions, such as Dijkstra or Hoare, who argue "that programs can be derived from their specifications through mathematical insight, calculation, and proof, using algebraic laws as simple and elegant as those of elementary arithemetic" (Hoare:1986). Formal methods, which harness the model-building and reasoning capacities of mathematics, are seen to offer the best way of gaining intellectual control over software. They fall into what we might call an 'analytic' rather than a 'constructive' approach to solving design problems.[2] In the formal approach, the emphasis is on using mathematical notations to model part of the system at one or more stages of its development (specification, refinement etc), and then, to bring mathematical reasoning to bear in validating the correctness of those models. For software developers using formal methods in the procedural programming which characterises much safety-critical work, the task is how to use mathematical notations so as to conceptualise the flow of control through a program's structure (drawing on graph theory, for instance, to reason about the behaviour of program loops); how to deal with the flow of data through that structure (reasoning about the definition and usage of variables within a program module); and, how to deal with the control of information within a program (using ideas from set theory and predicate calculus to establish the semantic relations between statements hypothesised at the start of a program, and statements which make conclusions at its finish).

Formal methods, as a whole, are concerned with 'proving' by means of detailed logical steps, that a function or component is demonstrably correct

---

[2] This is the loose classification used by Ian Currie, an RSRE engineer, to describe broad philosophical differences in conceptualising software design solutions. Interview with Ian Currie, Malvern, 4th June 1991.

with respect to its specification. However, mathematical proof, especially as it is applied to computer-based systems, is a disputed concept (MacKenzie:1991), and most formal methods practitioners are extremely cautious about claiming that proof is what is achieved through formal techniques. Specifically, many are wary that the 'proof question' may set up the whole *field* of formal methods as a strawman which can only stand (or, more likely, fall) on the delivery of categoric proofs of correctness. And indeed, it is on the question of proof that many sceptics have hinged their arguments against formal methods: that building verifiable computer systems is necessarily a different process to building mathematical proofs. Some stress the differences in the social processes of the two activities (eg. DeMillo et al:1979); others argue that the nature of software is inherently contextual rather than abstract (eg. Mellor:1989)[3]; yet others claim that deductive algorithmic reasoning, upon which mathematical proof is built, is of a different order to the inductive reasoning necessary for verifying programs in relation to their particular target machines (eg. Fetzer:1988).

This account of the cleavage within software engineering about transposing mathematical proof to verifying software is academic in the sense that we are not considering here the substantive value of the particular formal methodologies, notations and tools advocated by 00-55 (see Tierney:1991). The point, though, is that - unlike the orthodoxies[4] of software engineering which stress the inter-relationship of the design of software with its management, and the desirability of the specification-led life cycle - we find little consensus as to the appropriateness of pursuing the mathematical route to good software design.[5] On the contrary. The 'radical' camp within

[3] Mellor's discussion cited here is specifically about the manifestation of software faults. He argues that the appearance of faults are necessarily unique rather than 'abstractable' occurrences: "typically the software fault shows itself only under specific and unusual circumstances; a combination of a particular execution path through the code, a certain set of inputs and a given internal state of the system." (Mellor:1989).

[4] These, of course, are 'orthodoxies' only in the sense of being agreed definitions of what 'good practice' should look like. There is considerable debate within software engineering as to the appropriate strategies this discipline should espouse in order to achieve these ends (see Sommerville:1990 for a textbook overview of current models and techniques).

[5] For instance, in 1989, the DTI circulated a questionnaire to companies involved in safety-critical work, asking their reaction to 00-55. Only a third were in favour. A

software engineering (which includes most formal methods champions) is quite clearly differentiated from the 'reformist' camp.[6] The latter argue that if software engineering is to succeed in changing computing practice from its current unsystematic methods and practices, it must remain thoroughly pragmatic about the labour and product markets of 'real world' software production.

First, it points to the existing strengths of mature and historically dependable engineering disciplines: "the main characteristics of any engineering discipline is that there is no single correct solution, and there is the freedom to choose a good solution using some sort of aesthetic appeal" (Ince and Andrews:1990). This image of the engineer as a pragmatic tool-user is contrasted with the utopianism of the mathematical tool-theorists, who lack an empirical track-record and who prescribe just 'one best way' of tackling design problems (Gibbons:1990). In contrast, the radical camp see the strength of engineering as being that the development methods it uses (in any of its specialist branches) are *reproducible*.[7] Software engineering is currently the main exception, precisely because the methods and practices which drive it are not sufficiently intellectually visible. Were software engineering to incorporate mathematical notations as part of its tool-kit, and use formal techniques in appropriate ways, the strengths of existing engineering disciplines could also be felt in the complex field of software.

Second, the reformist camp point to the existing software labour market. The majority of systems designers and practically all systems users are not mathematically trained. To conceive of software design in terms of its algebraic and logical properties requires mathematical sophistication on the part both of the designer herself and the user who is involved in specifying that design (Ince and Andrews:1990). Virtually all software engineers would argue that this shortage in the existing software skill pool cannot be quickly turned about (Youll and Simms:1988, Macro:1990, p.114,

third were completely against it, and a third "although admitting that a standard was necessary, doubted whether a mathematical standard was required" (Ince:1989a)

[6] Pelaez (1988) has documented the historical roots of this intellectual split in the field of software production.

[7] Interview with Martyn Thomas, Praxis, Bath, 9th October 1990.

Gibbons:1990). However, the strategy for changing the skill base splits, roughly, into those who favour the introduction of tools and training which support existing methods of specification and design (eg. the CASE approach), and those who favour a new kind of training, to make use of the tools and notations which support the formal analysis and verification of software (Quintas:1991).

Following from the last point, the reformist camp stress that software engineering is, first and foremost, a commercial activity. As Pelaez notes, "the problems of programming are not just cultural: they have to do with the fact that software and computers are produced for sale on the market..For the computing industry (it is) a colossal *must* to hide the machine's mathematical nature as much as possible" (Pelaez:1988, p.223/4). The trouble is that mathematics, per se, is extremely user-*un*friendly: "the notation (of predicate calculus) itself tends to make it unsuitable for activities in requirements' specification as...the use of abstract languages...is positively manager-bellicose" (Macro:1990, p.46). In commercial terms, therefore, the reformist school argue that maths-based techniques can only ever be used effectively in "very special case" applications and organisations (Macro:1990 p.114), eg. safety-critical applications for defence. Software engineering which is premised on formal specifications, it is argued, will never diffuse *beyond* such special cases into the generality of software systems. Not only are mathematical notations foreign to most users (and hence a poor communication tool)[8] but they increase software quality at the expense of programmers' productivity (Quintas:1991). In contrast, the radical school see enormous commercial advantages in formal methods. Since mathematical notations are not just about correctness but about *visibility*, a radical approach to design clarifies, at an earlier point in the life cycle, where the ambiguities lie in specifying a problem. The quicker such ambiguities are caught, the less maintenance problems will occur later - even if mathematical modeling were then to be abandoned and more established techniques used to actually achieve the design.[9]

[8] Dr Phil Bennett, Centre for Software Engineering Ltd, sees a particular problem here: since mathematical notations and models are difficult even to *read* without special training, client engineers do not know on what basis they should trust that formal notations actually *do* express their specifications satisfactorally. Interview with Phil Bennett, Flixborough, 18th June 1991.

In a similar commercial vein, the maths-based approach to software engineering comes in for criticism from within the discipline's ranks on the grounds that the necessary infrastructural tools to support formal analysis are seriously underdeveloped as purchaseable commodities. The transfer into the market place of innovations which go towards supporting the mathematical verification of software, such as structured editors, syntax checkers, proof checkers and theorem-provers has hardly begun (Sommerville:1989, p.455, Gibbons:1990). Software engineering based on formal methods, it is argued, is disadvantaged by the fragile and nascent technologies it can draw on, relative to more well-developed engineering-based technologies. To which most formal methods champions would respond - give it time.

In the light of this debate within software engineering about formal methods, 00-55 is a landmark document. It endorses and prescribes the 'new order' for software engineering, for it mandates design practices which are amenable (in varying degrees) to formal analysis and verification. In this regard, it nails its colours firmly to the mast of the maths-based approach to software design. To the extent that the MoD is a powerful agency in the UK which can shape the direction of change, 00-55 is more than just a defence standard. It is a regulatory framework which may produce a sea-change in the software engineering field over the next decade. We turn now to describe the evolution of 00-55 and 00-56 into their current Interim versions so as to illustrate some of the pressures towards - and resistance to - the introduction of the 'radical' programme for software engineering.

2. The Impetus for Def Stan 00-55 and 00-56

Even in their origins, 00-55 and 00-56 are unusual standards. In general, commercial computing standards emerge as de facto, or market-defined,

[9] Interview with Martyn Thomas, Bath, 9th October 1990.

documents rather than reflecting the interests of a single organisation (Libetta and Woolgar:1991). The MoD, however, is no ordinary user and the impetus behind developing a unique standard for its safety-critical software has stemmed predominantly from technical and organisational changes within the MoD itself.

On the technological front, computers have moved since the early '70s, from playing a supporting role in military systems' performance, to the late '80s, where they are central to their capabilities (Barrie:1989). Defence software developers must now deal with systems which are both potentially dangerous and technically difficult: defence systems typically have very stringent real time constraints, they need to interface with an array of non-standard hardware, and they are frequently complex (Ince:1990a). The switch from analogue to digital computers over the last 20 years or so, has also signalled a rapid increase in the amount and the complexity of software used in weapons systems.

One effect of the shift from analogue to digital computers is that the locus of systems complexity is increasingly concentrated into software design, rather than into testing for hardware reliability.[10] But all software faults are systematic, rather than random. Thus, where analogue computers are subject to progressively degraded performance, digital computers - when they fail - can fail suddenly, totally and non-randomly. As Air Vice-Marshal Brown, till recently the Chairperson of the MoD Safety-Critical Software Steering Group, has pointed out: "On an engineering level, it is very doubtful whether software errors can be factored into systems reliability assessments in the customary probabilistic sense, since the logical and data conditions that cause them may not arise from random and uncorrelated events" (Brown:1990). Thus the techniques used for dealing with anticipated hardware component failure are not appropriate for the unanticipated

[10] There is no technical inevitability about concentrating systems' complexity in the software. Rather, with the introduction of software into the systems equation, developers can increase the capability of computer systems since software is so malleable a thing. Of course, greater capability increases complexity, thus compounding the problems of controlling for that complexity through the software.

situations which result from software (or, for that matter, hardware) design faults.[11]

Within the MoD, the increased use of safety-critical applications[12] - and the attendant recognition of software design as a crucial area for regulation and control - has emerged, piecemeal, within each specific application sphere. In organisational terms, each MoD division has historically regulated the safety requirements of its own applications. Until the early '80s, this strategy was seen as offering the best chances for ensuring safety, since it enabled systems designers to preserve very direct links with the previous generation of technologies and to capitalise on accumulated application-specific expertise in dealing with particular types of risks. Under this regime, MoD officials, specialising in their own application domains, supervised the quality of enhancements and modifications to each of the systems under their control. For as long as systems development work was either largely an in-house affair, or was conducted by defence contractors (working on variable-price contracts) who were subject to the MoD's overall responsibility for projects, this organisational solution to controlling safety proved satisfactory.

However, throughout the early '80s, pressure to change the existing organisational management of software development (including safety-critical applications) significantly increased. In line with government policy to open up to the private sector work traditionally done by MoD staff, the number of development contracts being run by private contractors, rather than MoD research establishments such as RSRE,[13] increased. In addition, these new styles of contract devolve responsibility for projects onto the contractors themselves. As Ince (1989b) points out, this transfer of

[11] Interview with Air Vice-Marshal Brown, London, 9th November 1990.

[12] A safety-critical function or system is "one in which a failure or design error could cause risk to human life; it is of the highest level of safety integrity" (00-55:1991a, 0.1).

[13] This is the Royal Signals and Radar Establishment, based in Malvern, a leading MoD research laboratory. RSRE has a long history of collaboration with academic and industrial R&D sites. It has now been hived off as part of an independent agency - the Electronics Division of the Defence Research Agency - rather than operating as a direct arm of the MoD itself.

responsibility came in the form of a policy switch towards awarding competitive fixed-price contracts[14]. Under fixed-price contracts, private sector developers are more likely to be made fully responsible for meeting performance and timescale requirements: project management becomes their own internal affair.

This new policy direction in the amount and the kind of contracts awarded to external contractors increasingly left the MoD exposed when it came to controlling the quality of their safety-critical applications. On the one hand, the old regime of appointing internal Government Safety Officers for particular application fields was incompatible with the new style of transferring responsibility for projects to the private sector (Brown:1990). In addition, as one American report suggests, a more serious problem facing defence establishments in general in that while the use of software has been expanding exponentially, the number of software-qualified defence staff has remained more or less constant over the past decade (Aviation Week, 17/10/88). One UK observer asserts that, quite simply, there is not enough internal MoD expertise to monitor high integrity software (Ryan:1990).

Thus, from the early '80s, monitoring safety through separate arrangements for each discrete application field began to dissolve as the MoD sought ways to establish an integrated regulatory framework. The opening up of safety-critical (and other) applications to the private sector fueled a need within the MoD to actively shape *how* contractors would fulfil their project commitments. As Martyn Thomas has said, one of the key motivations for 00-55 and 00-56 was "a desire to stimulate the defence supply industry to use and develop better software engineering techniques" (Ryan:1990). But, 'shape' or 'stimulate' in what way exactly? Two other dynamics within MoD may indicate why formal methods should have been chosen as the dominant shaping strategy endorsed in 00-55.

[14] It is interesting, though, that in the discussion session which followed Martyn Thomas' presentation of Draft Def Stan 00-55 at a workshop on formal methods held in Canada in July '89, none of the participants "would admit to having signed a fixed-price contract to develop verified software" (Ryan:1990).

First, there were up-coming problems facing key user divisions within the MoD, which focused attention on *safety-critical* issues. The Ordnance Board are responsible for producing explosive arming and fusing devices. Traditionally, the Ordnance Board had relied on the watch-making industry for supplying them with highly precise electro-mechanical instruments. By the early '80s, however, this type of manufacturing industry was in terminal decline, and the Ordnance Board found themselves turning to electronic devices using software as the means to handle explosive control functions.[15] The software in fusing applications, while being relatively simple in computational terms,[16] is, of course, highly safety-critical. And it was the Ordnance Board, most especially through its Chairperson, Rear-Admiral Hilton, which initially articulated the problems inherent in adopting software logic as an embedded feature of its systems, particularly for devices such as torpedoes and missiles which have to be guided but are not manned. It was they who argued most strongly that all embedded systems must "be presumed dangerous until shown to be safe, and that absence of evidence for the existence of dangerous errors does not amount to evidence for the absence of danger" (Brown:1990).

Other MoD voices were echoing the Ordnance Board's during this period: the navy, the army, but most especially the Flight Certification Teams within the Royal Air Force.[17] In particular, the RAF saw that from the late '70s the NATO nations were moving towards building aircraft which are inherently aero-dynamically unstable, eg. the Eurofighter project. In addition - and posing a more immediate problem - most older aeroplanes were being retrospectively fitted with ever more complex computer systems. While the work of the Ordnance Board is not in the public domain, the safety issues surrounding military aviation were (and are) clearly articulated within MoD as being of public concern.[18] Thus, while the Ordnance Board were arguing the necessity for a intensified MoD effort in regulating the

[15] Interview with Colin White, then Secretary of the MoD Safety-Critical Software Steering Group, London, 9th June 1990.

[16] And, hence, intuitively more amenable to formal techniques.

[17] Interview with Dr Brian Gladman, London, 6th August 1991.

[18] Interview with Prof John Cullyer, Warwick, 5th June 1991. Prof Cullyer was a key engineer employed by RSRE until the late '80s, when he took up his present post in the Engineering Dept of Warwick University.

production of safety-critical applications (that is, generating a powerful *internal* user demand), the RAF - most especially through the efforts of Air Vice-Marshal Brown - were arguing that any such regulation should belong, without fetter, in the *public* domain. From the mid '80s onward, an MoD-wide concern with adopting or building an integrated open framework for managing safety-critical software was firmly on the agenda.

This growing internal concern with safety-critical issues was given an added external push by the introduction of legislation in 1987 - the Consumer Protection Act, Part 1 - which might, in case law, require UK software developers to demonstrate, in accident inquiries, that their software had not contributed to injury (Quintas et al:1990). Through this product liability legislation, the MoD (not uniquely, of course) were now open to the possibility of being prosecuted for failure to comply with the terms of the act. Thus, MoD's concerns about regulating the safety of its systems were not just reflections of internal demand, nor even of a generally-felt moral responsibility within the Ministry. Crucially, it had to be able to *demonstrate* that the pursuit of systems' safety has been followed as assiduously as possible. This concern with being able to demonstrate - should it prove necessary - that safety-critical applications have been built with all possible care, is perhaps one pointer as to why MoD policy-makers have forged such a strong link between *safety-critical* issues and *formal methods* of software production.[19]

However, there is also a second factor influencing the espousal of the formal methods approach which we find in 00-55, namely, the work of the MoD's RSRE (now the DRA). MacKenzie and Spinardi (1991) have documented the considerable role of RSRE within the UK in conducting both basic and experimentally-applied research in a diverse range of fields. A number of these RSRE-led developments of the '70s and '80s explicitly attempted to initiate and extend the MoD's repertoire for using mathematically-based

[19] Dr Brian Gladman, the Chair of the Working Group, has noted however that the possible implications of this legislation only gradually become an 'argument' for formal methods. During the early decision-making period, the upcoming Act had not been a focus of the MoD's internal debates. Interview with Dr Brian Gladman, London, 6th August.

ways of conceptualising and structuring software. Clustered within the formal approach, RSRE have achieved considerable success, though not always without controversy, in applying formal methods to various safety-critical and security-critical application areas.[20] Under Dr Brian Gladman, the head of RSRE's Computer Applications Division and later its Computer Communications Group between 1981 and 1989, it has been active in transferring formal methods research into experimental civil applications, eg. an instrument landing system undertaken in collaboration with the Civil Aviation Authority.[21] Nor has its formal methods work been confined to software design alone - it was this establishment which undertook the formal verification of the VIPER microprocessor (Cullyer and Pygott:1987).

RSRE have also been a key research site for the development of *tools* which support the formal approach to systems building, most notably here - because of their integral role in operationalising 00-55 - tools to support formal static analysis of software. It was Bob Phillips, of RSRE, who wrote the abstract language called FDL (Functional Description Language) as an attempt to impose a coherent structure on the sprawling spaghetti programs he inherited on a radar project in the early '70s. FDL, and a relation IL (Intermediate Language), later became essential intermediary languages in the SPADE and Malpas formal static analysis tools sets developed for commercial use during the '80s.[22]

During their development, these innovative tools were used on certain existing military avionic software to check on the coherence of its code.

[20] Though the specification goals of security and safety are clearly different, the detailed methods and tools for achieving these goals are often similar. Formal methods, for example, have lent themselves well to both. Bloomfield et al (1991) note that "there is a large but generally well-hidden body of experience of the use of formal methods in the security industry", which presumably includes security-critical applications within the MoD. It seems reasonable to assume that, within and between MoD research establishments, there is a flow of both information and technical personnel across the two problem domains,

[21] Prof John Cullyer, then at RSRE, was centrally involved in this work with the CAA.

[22] Malpas stands for 'Malvern Program Analysis Suite', and SPADE, 'Southampton Program Analysis and Development Environment'. Essentially, these tools translate source code into FDL (SPADE) or IL (Malpas). Interposing an intermediate language between the source code and the tools allows the same set of tools to be applied to source code written in a variety of different languages.

Between 1982 and 1985, up to 10% of modules or functions checked were discovered to deviate from their specification, even where they had undergone extensive dynamic testing at the time. While most of the anomalies discovered were minor, about one in twenty of the deviant modules proved to have errors "which would have had direct and observable effects on an aircraft's performance" (Dettmer:1988). Thus, RSRE's explorations into formal techniques and tools looked likely to be able to make an immediate and direct contribution to safety: formal methods had arrived on the MoD's safety-critical agenda. Indeed, 00-55 is not the first MoD standard to call for the use of formal methods - the MoD Naval Engineering Standard NES 620 issued in October '86 recommends them (Jackson:1989).

It is important to stress that the MoD were not living inside any sort of 'formal methods vacuum' at this time. A number of more general government initiatives and policy statements were endorsing the formal approach. For example, the Alvey Programme[23] had a sizable formal methods component within its Software Engineering activities. In addition, the ACARD report of 1986 had made special mention of the need for software development standards, and had stressed the contribution formal methods could make to the design of safety-critical systems. As a consequence of that publication, associations such as the IEE (Institute of Electrical Engineers) and the BCS (British Computer Society) had set up internal committees to assess and comment on, amongst other techniques, the application of formal methods to safety-critical systems (IEE:1989). In addition, the ISO (the International Organisation for Standardisation) had, by 1987, published policy recommendations for how best to incorporate formal methods in the development of software standards.[24]

RSRE's work - especially in the context of this growing UK-wide interest - pointed to formal methods as a technology with a future, one which could be commercially encouraged by its harnessing to safety-critical software.

---

23 This was the outcome of the Committee of Inquiry, set up by the DTI in 1982 under the chairpersonship of John Alvey, to advise the government on the scope for a collaborative research programme in IT.

24 For a discussion of this and other initiatives, see the report produced by the BCS Formal Methods in Standards Working Group (Ruggles:1990).

3. Setting 00-55 and 00-56 in Motion

By 1985, the MoD had set up a Safety Critical Software Steering Group, whose brief was to develop new policies, procedures and technical standards. The Steering Group, chaired by Air Vice-Marshal Brown, was composed of about a dozen senior representatives of the MoD's Controllerates and its major Operations Divisions. One of its first tasks was to establish a larger procedural Working Group, chaired by Gerry Price, the MoD's Director of Standardisation. This Group had about 17 members. It included representatives from the Ordnance Board, RSRE, Royal Aerospace Establishment, Navy and Air Departments, and from the Defence Quality Assurance (DGDQA) area. All members were those with responsibility for safety-critical applications within their discrete fields. Most were engineers, though not all of them champions of the formal approach to systems engineering.[25] RSRE, through Dr Brian Gladman, was nominated as the technical authority within the Working Group, and its management was delegated to Aquila, part of DGDQA.

The first questions facing the Steering Group were contentious. A standard was needed, but on what should it be based?[26] Should an existing civil standard be modified for MoD use, since writing a new standard is a costly business? Or should a new one be devised, built on the back of RSRE's largely successful experimentations in applying formal methods to safety- and security-critical design? It had been known since 1982 that the International Electrotechnical Commission (IEC) standard for safety-critical systems was in the process of being devised.[27] The MoD - in line

---

25 Interviews with Alan Burton, 4th June 1991, and John Cullyer, 5th June 1991.

26 Knowing that discussion about an integrated MoD standard would, by its complexity, be lengthy, the RAF proceeded independently with a domain-specific modification of a civil aviation standard - DO178A - and issued it for the short-term as Def Stan 00-31.

27 The IEC standard is the outcome of a collaborative international endeavour, chaired by the UK through Dr Phil Bennett, the Centre for Software Engineering Ltd in Flixborough, Scunthorpe. The standard governs the safety-criticality of whole systems across a range of sectors, rather than focusing exclusively either on software or on defence. Formal methods are one - though only one - technique recommended by the IEC for achieving high-integrity software. The IEC standard eventually emerged in draft form in 1989 - the same year as 00-55 and 00-56.

with its normal policy to use existing commercial standards where possible[28] - might have chosen to row in behind this activity. Or it might have chosen to adopt or expand one or more of its existing quality assurance standards, such as 05-21, and its associated interpretation for software quality, 00-16, or for software procurement, 00-31.

In the event, the forthcoming IEC standard was rejected as a possibility. As a sectorally generic standard, it was thought to be unlikely to yield something sufficiently focused on defence needs. In addition, as it was then at a very early stage of development, there was little available material upon which any thorough assessment could be made[29] - although Dr Phil Bennett, the Chair of the IEC Working Group, has suggested that there was not, in fact, much communication between the MoD and the IEC from '85 and '89 as to the potential for co-ordinating the IEC work with MoD's specific interests.[30] With this emergent civil standard found wanting, what then of the possibilities for adapting a current MoD standard? To investigate this, the Ministry commissioned Logica, a UK software consultancy firm, to review MoD's existing standards for procuring software, including 00-31, the standard to which most defence contractors were bound at that time. Logica's conclusions were critical: it saw major deficiencies in the MoD's ability to monitor safety features (Ernest-Jones:1988).

Within the Steering and Working Groups there were a number of active champions for wedding the formal approach of software engineering to safety-critical developments. Air Vice-Marshal Brown, Dr Brian Gladman, and Rear-Admiral Hilton were not only sufficiently enthusiastic (an enthusiam they held in common with, for instance, John Cullyer on the Working Group), but they held sufficient organisational clout to ensure that the formal methods option remained a central debating point throughout this decision-making period. Notwithstanding their status as relatively untried techniques, formal methods promised a degree of assurance in

---

28 Personal communication, June 1991, from Colin White, Secretary of the MoD's Safety-Critical Software Steering Group till 1990, in response to an earlier draft of this paper.

29 Interview with Dr Brian Gladman, London, 6th August 1991

30 Interview with Phil Bennett, Flixborough, 18th June 1991

pursuing design correctness unmatched by any single other software engineering technique. A decision to build a new standard around formal methods of software development would not only address the organisational, legislative and ethical pressures to demonstrate that safety-critical defence applications were being built to the highest possible assurance of safety. Such a standard would also give an important kick-start to the commercial take-up of formal methods in the UK. Any sustained commercialisation would assist the establishment of an infrastructure (of tools and expertise) which would not only benefit safety-critical work, but perhaps also subsequent security-critical applications which are not in the public domain.[31]

The formal methods argument won the day. By mid 1987, the Steering Group had commissioned Praxis, another UK software engineering firm, to review the requirements for a new standard for safety-critical software based on formal methods.[32] This work was contracted out because RSRE staff had not the resources to take it on themselves.[33] Praxis, in turn, sub-contracted most of the writing of the initial requirements specification to two people, Robin Bloomfield and Peter Froome, who were partners of a newly-formed independent consultancy company called Adelard. Froome and Bloomfield were software engineers who had worked in the Central Electricity Generating Board as safety specialists on nuclear power plants, and were practitioners of the formal approach. Between them, Adelard, Praxis, and the MoD Working Group wrote the technical, organisational and managerial requirements which later emerged as draft 00-55 (Ernest-Jones:1988). This work was finished by Spring 1988.[34]

---

31 In this regard, one respondent (not an MoD employee) has speculated that the Steering Group must have had the interests of the UK and US defence security-critical communities strongly in mind at this time. MoD respondents, however, have never suggested any such direct linkage of security- and safety-critical aims, in formulating 00-55.

32 Both Logica and Praxis are well-known within the UK for their commitment to formal methods as the appropriate direction in which software engineering should be moving. Praxis in particular - through its chairperson, Martyn Thomas - has generated a high public profile as an advocate of this approach.

33 Interview with John Cullyer, Warwick, 5th June 1991.

34 Interview with Robin Bloomfield, London, 5th November 1990.

A preliminary discussion version of the new standard was circulated for editorial comment to the Working Group and others drawn from the MoD's Computer Policy Consultative Committee, which included representatives of the Computing Services Association (CSA) and the Electronic Engineering Association (EEA)[35]. Many of these early recipients of 00-55 were themselves pioneers in the nascent field of applying formal methods to software design. As such, their reaction to the first draft was not primarily a disquiet about the mandating of formal techniques. That reaction was to surface later amongst the wider software engineering community. Rather, these early comments took issue with one specific economic implication of the draft.

The draft was not sufficiently explicit as the *scope* of development work which would be bound to the terms of the 00-55 standard. It was a single document, which was not at that point separated into a code for establishing how any safety-critical components were to be identified (00-56), and a code for specifying how the building of such safety-critical components was to be pursued (00-55). At this stage, it was open to interpretation whether *all* software could be deemed safety-critical, implying the application of formal methods to complete systems.[36] The MoD Working Group readily acknowledged, and set about correcting, the unwieldiness of the document. It was not just a problem of bulk. It was a recognition that hazard analysis techniques are entirely distinct - in intellectual approach, and in the problems they aim to address - from formal design techniques.[37] The next task was to establish the *boundedness* of what was intended, in mandating formal methods for safety-critical software.

By the start of 1989, Adelard had again been commissioned by the MoD to write the first draft of 00-56 - the standard covering the MoD's requirements

---

35 This consultative committee had been established for a number of years, meeting on a 3-monthly basis, including the period of 00-55's gestation. Interview with Dr Brian Gladman, London, 6th August 1991.

36 This is how Prof Patrick Hall, Computing Dept, The Open University, recollected the debate as it was discussed when the IEE/BCS Standards Committee were later considering the first issues of 00-55 and 00-56. Personal communication from Prof Hall, May 1991.

37 Interview with Prof John Cullyer, Warwick, 5th June 1991.

for subjecting all proposed systems developments to initial hazard analysis so as to map out which parts of the system must be regarded as safety-critical. This work was commissioned by the Policy Coordinating Committee for Guided Weapons and was produced, in the first instance, to satisfy an Admiralty specification. By April '89, this work was completed, and was reviewed and edited by a separate Working Group, which had some members in common with the 00-55 Group. Thus, by now, a document existed which defined "the scope of 00-55" (Ryan:1990).

In its draft form and subsequently, 00-56 has never become enmeshed in the controversies surrounding its 'revolutionary' sibling, 00-55. The techniques it endorses for identifying, and controlling for, the susceptibility to hazards of a proposed system are well-understood by systems engineers. Fault Tree Analysis, Failure Modes and Effects Analysis, and so on, fall within the established skill base of systems engineering, and have been used extensively in the aerospace and nuclear industries. Indeed, 00-56 was intended as a UK instantiation of an existing US meta-standard, MilStan 882B.[38] 00-56 - though now codifying those techniques into a single framework - means, in effect, 'business as usual'. Not so, in the case of 00-55.

By May 1989, the Draft Interim versions of 00-55 and 00-56 were circulated by the MoD to interested parties. This is normal MoD practice: all Def Stans have a interim status for the first few years of their life to enable relevant actors to use, review and provide feedback on their operation.[39] The Def Stans, in this case, were sent to thousands of systems development practitioners, academics and professional and trade associations. Feedback to 00-55 was not slow in coming.[40] Much of the reaction was critical, and

---

38 Personal Communication from Robin Bloomfield, July 1991.

39 The launch of 00-55 and 00-56 as drafts in 1989, prior to their second release as interim in 1991, is a slight departure from this practice, which has enabled the MoD to gain a longer period for commentary while the Def Stans are still provisional. Personal communication from Robin Bloomfield, July 1991, in response to an earlier draft of this paper.

40 By June 1990, the MoD had received about 1,370 written comments on 00-55, later rising to over 1,500. Interview with Air Vice-Marshal Brown, London, 9th November 1990.

centred around the integral role which 00-55 gives to formal methods of software development. But why formal methods should constitute a barrier for industry is a topic for later sections of the paper. For the moment, I want to pursue another question, which may throw light on the subsequent evolution of the standards: why should 00-55, a draft document specific to defence, come to *matter* so much?

A month after the release of Draft Interim 00-55, RSRE hosted a conference attended by 240 people. The very fact that this conference was held underscores the MoD's intentions for 00-55. It is a public domain document, which the MoD's Steering Group were at pains to emphasise should receive the widest circulation possible. 00-55 offers one emphatic response to the question: what if there were a disaster - in any sector, not just defence - and in the accident inquiry, software was indicted as a possible culprit? Could civil sectors such as avionics, railway signalling, off-shore oil and nuclear energy ignore 00-55 as 'not invented here'? In issuing 00-55 as an open document, MoD had set up the challenge that they could not. In this sense, 00-55 is not just another military standard. On the one hand, it is a "meta standard for the defence applications sector" (Brown:1990), which is intended to be compatible with the more specific requirements of particular MoD application fields.[41] On the other hand, it sets the terms of a UK-wide debate on the principles and practices of software engineering in *all* safety-critical fields.

If trouble it is, the 'trouble' with the 00-55 draft was that it demanded - and has received - the attention of software engineers who come to the safety-critical problem from a plethora of directions. Thus, the passionate reactions (both for and against) to 00-55's mandating of formal methods, expressed at the June '89 conference and in subsequent forums on the standard, are not just animated technical debates about safe ways of designing software. Rather, they are reactions which also express the political, commercial and occupational interests of the players who have a stake in shaping the software engineering field. 00-55 has set up the

[41] 00-55 and 00-56 gain their initial prefix of '00' within the MoD's classification system since they *are* generic defence standards.

'formal methods debate' as the forum through which those interests are currently being articulated. We turn now to look at who those players are, and on their fears about the implications of the MoD's initiative.

4. Reactions to Draft 00-55

Responding on an individual company basis to the MoD's feedback sheet on 00-55, the largest - though not, in that individualist guise, the loudest - group to react, were the defence contractors who would be most immediately affected by 00-55's demands. Since about half of all UK electronics design production is for the military,[42] these companies by no means occupy a minority niche within the software industry. The suppliers did not, by and large, query the suitability of formal methods to the task of safety-critical software design.[43] Rather, the problem was that many of these software and systems houses were unlikely to be able to meet the technical and organisational requirements of 00-55, were it to become mandatory (Ince:1989b). After releasing 00-55 for comment, the National Physical Laboratory circulated a questionnaire to its software suppliers and found that about 70% of its respondents were opposed to 00-55 as it stood in 1989. Their objections centred around the commercial infancy of formal methods tools; the lack of industrial experience in using formal techniques; and the cost of re-training staff to become competent to handle Def Stan-bound applications (Ernest-Jones:1988, Ince:1989a,b, 1990a,b, Lowe:1989, Financial Times:13/6/89). It is also interesting that regardless of whether suppliers were pro- or anti-formal methods, the proposed timing of 00-55 was seen as a problem.

Amongst the 'anti' lobby, there were those in favour of instituting *some* kind of standard for monitoring software reliability.[44] The argument against 00-

[42] Interview with John Cullyer, Warwick, 5th June 1991.

[43] Interview with Alan Burton, Malvern, 4th June 1991.

[44] The publication of the IEC draft in 1989 was an obvious 'alternative' candidate in this respect, though the picture is not that simple. We have seen that the MoD knew of the evolving IEC draft, when building 00-55 and 00-56. In many respects - most especially the procedural rather than the technical recommendations - the drafts are similar. Thus, the IEC draft was not so much a contrast to 00-55, as that 00-55 was

55 was that it seemed to equate 'reliability' with 'formal methods': an equation both commercially unpragmatic, and not demonstrated - through experience of use - as achieving safety any better than alternative existing software engineering techniques could. That is, draft 00-55 was pitched too far beyond current software engineering practices. Since its requirements were unobtainable without an extensive national support programme which was not yet in place, 00-55 was, in effect, unenforceable (Ince:1990a, 1990b). Thus, some of the disquiet that has been expressed has focused on the problems which the 'idealism' of 00-55 could bring in its wake (see, for example, Craigen:1990). As John Buxton, the Director of Systems Engineering in the DTI, has noted, there was concern that the issue of software reliability had been "somewhat taken over by the formal enthusiasts".[45] Through 00-55, it seemed as if the MoD had offered formal methods the moral highground on questions of reliability.

This kind of reaction to the 00-55 draft also surfaced in the Risks Forum of the ACM. For example, Philipson commented that the strict enforcement of a formal-methods-based standard (though he is not talking specifically of 00-55 in this case) "simply makes the developer liable for errors, but doesn't do much for actually improving software reliability" (Philipson:1989). Seen from the perspective of defence contractors employing alternative techniques to formal methods in pursuit of reliability, 00-55 seemed to diminish the value of those alternatives, a value based on a solid and extensive body of existing skill and experience.

Even amongst the pro-formal methods lobby, the timing of 00-55 has received critical comment over the last two years. Ince, for example, has argued that the endorsement of formal methods "makes sense", on grounds of both safety and cost efficiency. However, to introduce the standard now when formal methods skills are still so thin on the ground, is to give formal techniques "a bad name" - one they do not deserve (Ince:1989b). On a slightly different tack, SPRU's[46] evaluation of the Alvey Programme on formal

niched in defence, and targetted on software. The biggest contrast, of course, was that 00-55 mandated formal methods for safety-critical software, where the IEC did not.

45 Interview with John Buxton, London, 14th June 1990.

46 SPRU is the Science Policy Research Unit, based in Sussex University, Brighton,

methods, suggests that too early an introduction of standards such as 00-55 may stifle innovation in such a commercially underdeveloped technical field (Quintas et al:1990, p.68).

Finally, a problematic feature of 00-55 (as with any standard of this kind) is that it demands *certification* of the software as an output. Certification, in relation to safety-critical software, is a particularly onerous undertaking. Normally, it involves the formal approval of a product. Software, however, is not something which is 'produced' as much as 'designed' and it cannot, in general, be black-box tested as fully as electro-mechanical systems due to its complexity. Thus, the certification of software is concerned as much with the design process as with the finished product (Jesty et al:1990). For software subject to 00-55, only certain suppliers will be approved as possessing sufficient skills to underwrite certificates of safety, which show that the application has passed the Def Stan test. At this early point in the commercial development of formal methods, very few suppliers are currently competent to do this.[47] The requirement for certification buttresses the barriers to entry into this market niche - an effect which may truncate the diffusion of formal methods techniques to a widening spread of companies.

Both the MoD and their defence contractors were mindful of another audience for a standard on safety-critical software - Europe and the US. This brings us to a second type of critical reaction to draft 00-55. From the MoD's point of view, 00-55 was to act as a "benchmark for judging whether any emerging UK national or international standard can be adopted for MoD use" since MoD policy is "for preferring international standards to those of purely national application"(Brown:1990). However, since the emergent IEC draft - or, what little existed of it at the time - had been rejected by the Steering Group in 1985, the task now was to 'sell' 00-55 to standards-making bodies as a better, safer, more focused alternative for the

47 Interview with Robin Bloomfield, London, 5th November 1990. This issue is not just a question of the MoD's assessment of competence, but also of suppliers' willingness to accept responsibility. In 00-55, the designers signing the Certificate are accepting liability both for the product, and for the process by which it was built. This liability is 'for life': it does not become void until the software is no longer in use - perhaps 20 years later. Interview with John Cullyer, Warwick, 5th June 1991.

international defence sector. Undoubtedly, the MoD's European partners had been held in mind in formulating 00-55: as a British endeavour, 00-55 could be presented directly to the EC nations, through the British Standards Institute, as a possible European standard. Less directly, 00-55 was hoped to influence the US defence sector. Members of the MoD Working Group visited the US Dept of Defense (DoD) in Washington to generate awareness of 00-55, an activity aided by Air Vice-Marshal Brown's fine presentation of the rationale for the two Def Stans, which he delivered at the international COMPASS conference in June 1990 (Brown:1990).[48]

To date, neither the Europeans nor the Americans have been keen to adopt 00-55 as their own. I am not privy to the debates within and between NATO partners on the suitability of 00-55 as an international standard for the defence sector. However, we may find some explanations for reluctance by looking at the status of formal methods in the US and in Europe.

First, formal methods R&D in the US is characterised by its concentration in security-critical, rather than safety-critical, application fields.[49] For example, the DoD, in 1986, issued the 'Orange Book' - Trusted Computer System Evaluation Criteria - which recommended formal methods for high integrity defence software (see SSSC:1991). The Orange Book - unlike the MoD's broader intentions for 00-55 - does not claim any common cause between the use of formal methods for both security- and safety-critical software.[50] More so than is the case in Britain, it appears that the two domains of activity are held conceptually and pragmatically separate. On that account, we could suspect that 00-55's promotion of a general 'culture of

[48] Interview with John Cullyer, Warwick, 5th June 1991.

[49] The vast majority of formal methods practitioners in the US are employed either by the DoD, or by houses which engage substantially in security-critical work, eg. Odyssey in NY State, Computational Logic Inc, in Austin Texas, and SRI International in San Francisco. In general, large US corporations do not have the equivalent industrial interest in formal methods for safety-critical systems as we find in Britain, eg. GEC Avionics, Rolls Royce, British Rail, Nuclear Electric, Marconi, British Aerospace etc.

[50] In Brown's COMPASS presentation, he noted that "access to the broadest possible intellectual base" is essential if technologies to support formal methods for safety-critical software are to become readily available. In this respect, "safety-critical software engineering tools and methods should recognise and draw upon the work done to establish high-integrity design principles for secure computing systems" (Brown:1990).

safety' may not easily dovetail with the sector-specific interests of the US military in formal methods; that the informational networks which enable some flow of papers or technical personnel between the two domains are not in place.

Second, it would seem that beyond a number of European universities and industrial research sites (most notably, in France and Denmark), formal methods do not enjoy the same status in Europe as in Britain: they generate less excitement amongst software engineers, and fewer organisations have undertaken applied formal methods research for safety-critical activities.[51] One explanation for this European diffidence might be found in the low status of British engineers, relative to their European counterparts. In the UK, engineering covers a multitude of diverse skills; engineers span a range of class positions and affiliations; and the professional status of engineering is not assured (Whalley and Crawford:1984, Whalley:1986, Jackson:1989). In this context, the UK software engineering discipline has perhaps not been as well equipped either to *resist* the 'scientific' basis underlying the rise of formal methods in the UK, or to *impose* uniquely 'engineering' alternatives, on that science.[52] If it is the case that European engineers constitute a more powerful occupational group than their UK counterparts, it may explain the observation, by Jesty et al (1990), "that the 'must do this, must not do that' approach taken by (draft) 00-55 is not at all well received by engineers in the European road transport industry".

00-55 has not been taken up by the IEC as a potential international standard for safety-critical software. And this poses a particular problem for UK defence contractors who are responding to 00-55 - the stringency of 00-55 could hinder the chances of winning contracts abroad, as we now consider.

[51] In the late '80s, the EC launched a major research program, EURET, on safety in the road, railways, air and maritime fields. The UK had 131 people at the bidders' meeting (many of whom were formal methods practitioners), a number far in excess of any other European nation. Interview with John Cullyer, Warwick, 11th October 1990.

[52] Both Dr Phil Bennett and Prof John Cullyer have pointed to the 'scientific' origins of formal methods within UK universities and research laboratories: mathematicians, rather than practising engineers, dominated the early seminal work on formal methods in the UK - a factor which has contributed to the UK's reputation as a world leader in formal methods research.

A development process centred on formal methods pushs its costs up-stream: the specification and design stages cost more, while maintenance costs are presumably reduced. Indeed, this very change of emphasis from down-stream to up-stream is one of the major promised benefits of formal methods, for they are better able to catch those ambiguities of specification which become increasingly costly to correct the further down the life-cycle the software goes. However, there is a snag. For this benefit to be felt, client organisations must change the way projects are budgeted for. As Quintas has pointed out, budgeting for formal methods runs against normal practice. The Project Managers responsible for commissioning or building software are not normally the same people - and sometimes not even the same departments - responsible for maintaining the software later. On this account, Project Managers in client organisations have a problem justifying the higher initial cost of formal methods developments, when the benefits are difficult to assess in advance, and anyway, they probably fall into somebody else's patch (Quintas:1991).

The radicalism of 00-55 *presupposes* a more fundamental revolution in how cost-benefit analysis is done in budgeting new projects. For contractors with a foot in both domestic and international markets, this is a real sticking point. The MoD may be fully committed to the budgeting changes demanded by 00-55. But what would happen if, as is common, the supplier attempted to 'sell on' parts of its 00-55-bound work to other organisations? UK bids for international contracts, which draw on work done to 00-55 standards, may be just too expensive to win out against competitors who are not bound to comply with anything approaching the stringency of 00-55. Thus, for some contractors, the threat of draft 00-55 was that this very condition of entry into trade with the MoD could, at the same moment, become a barrier to trade in a similar market niche internationally.

While suppliers - in their capacity as individual respondents - expressed their concerns directly to the MoD, a complementary strategy for gaining the MoD's attention was to use their trade and professional associations to present a *collective* industrial response. And this brings us to a third cluster of reaction to 00-55. Institutions such as the British Computer Society (BCS), the Institution of Electrical Engineers (IEE) and the

Computing Services Association (CSA), were as a whole reluctant to endorse 00-55. Indeed, united under the IEE, it was this lobby group which fired the first serious public shots at the draft in October 1989, an action which escalated 00-55's status as the reference point for much subsequent UK discussion on formal methods. The implications of 00-55 were becoming not merely noteworthy, but newsworthy.

The professional and trade associations embody, with different emphases, the status quo of software engineering. Some of their members were already involved in building up a UK safety-critical systems community, largely independently of 00-55.[53] Phil Bennett, Chair of the IEE Safety-Critical Working Group, has argued that this community had not been wedded to formal methods as a matter of principle; that 00-55 (dangerously, in his view) promulgated formal methods in the UK on the back of the politically hot topic of safety-critical systems.[54] Less contentiously, other IEE and BCS members felt that 00-55's endorsement of formal methods was, in itself, a wholly praiseworthy move towards achieving greater assurance of the reliability of the software components of safety-critical systems. However, it was felt that the MoD had not actively sought any contribution from this wider safety-critical community in formulating 00-55. Here, indeed, was an issue: if 00-55 was actively targetting the broader safety-critical community, why had the MoD ignored that community till now? Why had it not sought advice from external sources, beyond the contractors it had commissioned to write the drafts?[55]

The various associations embodied safety-critical expertise which the MoD might have solicited in drafting 00-55. In Section Two, we saw that these institutions had entered the safety-critical debate through the Alvey Programme and post-ACARD Report discussions. For example, the IEE, through Prof Bob Malcolm, invited the BCS, through Martyn Thomas, to

---

53 There are, of course, some overlaps between the two. Martyn Thomas of Praxis, for instance, not only played a key role in formulating the 00-55 draft, but was (and is) an active member of the BCS and the CSA.

54 Interview with Dr Phil Bennett, Flixborough, 18th June 1991.

55 It needs to be recalled, however, that the MoD *had* engaged in some external consultation through the mechanism of their Computer Policy Consultative Committee, as we saw in Section Three.

collaborate in writing a joint report investigating the applicability of a number of techniques, including formal methods, to safety-critical problems in software (IEE:1989).[56] This report clearly endorsed formal methods within its proposals. However, it also looked for a commitment to a *non-sector-specific* approach in building up capability in specifying and designing safety-critical software (IEE:1989, p.11). The problem articulated now was that the MoD, in issuing 00-55, had jumped the gun. As a major - but very singular - organisational and regulatory force, it was not the most appropriate authority for setting the terms of 'best practice' for safety-critical software engineering.

Yet, in an important sense, 00-55 *had* set the terms for a general code of practice. It was not a policy paper, but a 'fact': it stated that safety-critical software for defence applications must be achieved through formal methods. Its status was as a draft, but the MoD's intention (in 1989) was that it would, after due amendment, be made fully operational by 1995. As a standard it had teeth, for it demanded certification of safety-critical software as an output. 00-55 had a tangibility which was alarming. Thus, in the months following the RSRE conference of June '89, the trade and professional associations found themselves attempting to formulate a response to a draft their members were affected by, but which, by and large, they had not had a hand in building. The 'suppliers' debate' - which might, in a different history, have been conducted within the quieter walls of the MoD - was just about to happen more publicly.

If industry in the form of its trade and professional associations wanted to say 'no' to 00-55, why exactly - and what exactly - was it refusing? And how should that protest be made? As to the latter, a very early decision was reached by the IEE (in fact, during an informal gathering of delegates at the end of the first day of the RSRE conference in June '89) to invite the other associations to join with it in formulating a collective response to 00-55.[57] The 'problem of 00-55' was debated over the succeeding months, through

---

56 Interview with Prof Bob Malcolm, Chair of the IEE Software Engineering Group from '86 to '88, London, 31st May 1991.

57 Interview with Prof Bob Malcolm, London, 31st May 1991.

phones, faxes, letters and meetings. Given the innovativeness, the complexity and the sheer power of the 00-55 draft, the associations' responses to it were highly diverse. The problem with formal methods as against other more established techniques is one which has dimensions of costs, skills, education, training and qualifications; of international and domestic markets; of users' requirements and software development labour practices; of techniques, languages and tools. The list in the 'formal methods debate' is long indeed. In formulating a collective response to 00-55, the linkages between these dimensions were not only aired within[58] and between the associations, but the questions themselves began to snowball. 00-55 was becoming more than just itself. It was - depending on the viewpoint - a strawman to be kicked against, or a wedge to jerk software engineering into a better disciplinary paradigm.

The MoD were, of course, fully cognisant that the terms of the debate were escalating. By October 1989, when the IEE called for a meeting with the MoD to respond to the specific queries the associations had found common agreement on, the MoD had its arguments ready. This particular meeting was important - not least because it had the air of a 'showdown' between the MoD and its suppliers, which was later reported to the press by members of the IEE. With its tense atmosphere, it was not the most successful - nor perhaps even the most important - of the many series of meetings the MoD has had with its suppliers.[59] However, as a collective statement of industry's discontent with 00-55 which received press coverage, it was a significant landmark in the debate about 00-55, and has since assumed a legendary status within the technical community. The difficulties raised there were ones to which the MoD were obliged to provide either immediate or longer-term responses, and which have subsequently shaped the evolution of 00-55 and 00-56 into their current interim forms of 1991.

---

58 Bob Malcolm has noted that each separate association did not hold a unanimous position on 00-55. The BCS and CSA, in particular, were deeply divided within their own ranks into pro- and anti-00-55 lobbies. This is hardly surprising, since these associations span the full range from manufacturing to service companies.

59 MoD respondents have, in general, played down its significance, stressing the value of the more measured separate discussions which they have held with their suppliers between '89 and '91.

5. The MoD's response to criticism of 00-55, and the re-writing of 00-55 and 00-56 as Interim Standards

Undoubtedly, the MoD took this meeting seriously. In some contrast to the represented sent by the IEE and its fellow associations, the MoD's most senior policy-makers on 00-55 were in attendance,[60] and they outnumbered the associations' representatives by about 3 to 1.[61] The meeting addressed a number of issues, which we will look at in turn, all of which informed the subsequent re-writing of the two Def Stans between 1990 and 1991. As to the re-writing, the MoD again commissioned Adelard to sift through all the written responses to the drafts and to collaborate with the MoD's Working Group in re-working the technical requirements of the standards in the light of comments received.

First, the MoD responded to the question of why the broader safety-critical community had not been invited to participate in the design of draft 00-55. From the MoD's point of view, it was important to establish a position on this. If, through the anxiety it engendered, 00-55 had become the vehicle for arguing how standards on safety-critical software engineering should best be formulated *in general*, the immediate task was to deflate that debate. Thus, the response was that 00-55 was a standard unique to MoD; that it had been formulated to answer the particular requirements of safety-critical software for defence; that it was normal policy for the MoD to consult and deliberate within its own ranks before submitting a draft to a wider audience for feedback; that suppliers and associations had not been involved in its initial formulation because it was necessary to get the draft standard issued as quickly as possible.[62] The MoD's response was presumably aimed at *containing* the significance of 00-55, in that the standard cannot prescribe any general change in practice, only the practice of MoD safety-critical defence contractors.

[60] Interview with Bob Malcolm, London, 31st May 1991
[61] Interview with Phil Bennett, Flixborough, 18th June 1991.
[62] This is as Phil Bennett and Bob Malcolm recollect the MoD's position.

However, the wider implications of 00-55 meant that the IEE regarded this immediate response as somewhat disingenuous,[63] and that the question of broader industrial involvement demanded a longer-term response than that offered at the October '89 meeting. Indeed, the MoD did subsequently invite comment on the progress of 00-55 from a more general body - the Inter-Departmental Committee on Software Engineering. This committee was convened by the DTI in 1986 and was the sponsor of the joint IEE/BCS report mentioned earlier (IEE:1989). By 1988, it had established a Safety-Related Software Working Group to co-ordinate a government approach to the subject, resulting in the launch of the SafeIT Standards Framework in May 1990 (Bloomfield and Brazendale:1990).[64] The MoD was a member of both the Committee and its Working Group - the membership list is reproduced here from the SafeIT Overall Approach document (Bloomfield:1990). During the re-writing of 00-55 and 00-56, the MoD did not so much invite its SafeIT partners to provide direct input, as to consult them "to let them know what was happening".[65]

In one specific area, at least, this Committee did contribute to a significant change in focus between draft and Interim 00-56.[66] Draft 00-56 had proposed that initial hazard analysis would serve to determine a software function or component as being either safety-critical, or not: if it was safety-critical it would then be subject to development by formal methods as specified in 00-55. Interim 00-56, in contrast, draws on a classification system of the Health and Safety Executive (which is also recommended in the IEC draft) which sets out four degrees of likely accident severity which range from 'Catastrophic' through to 'Negligible', and whose failure probability is graded into five levels, from 'Frequent' through to 'Incredible'. On the basis of drawing up matrices between these two (and I have reproduced the relevant tables from Interim 00-56), the new version proposes four Safety Integrity Levels - from Class A to Class D. Interim 00-56 states that it is *only*

[63] Interview with Phil Bennett, Flixborough, 18th June 1991.
[64] Robin Bloomfield, as we have seen, was a key architect of 00-55 (Interim as well as Draft) as well as being the author of the SafeIT documents. Over this period, he provided the MoD with an important sense of continuity between the two areas of activity. Interview with Alan Burton, Malvern, 4th June 1991.
[65] Interview with Alan Burton, Malvern, 4th June 1991.
[66] Interview with Alan Burton, Malvern, 4th June 1991.

Class A risks which must now be classified as safety-critical functions or components (00-56:1991, 6.6.3.3) and thereby must be subject to the formal techniques mandated in 00-55. Thus, the boundaries on which software must be subject to Interim 00-55 are clearer as compared to its draft form - a change in emphasis which goes some way towards addressing the fears about 00-55 expressed by the IEE at the October '89 meeting.

At that same meeting, the MoD was tackled on a clause of draft 00-55 which proposed that the *MoD* retain full crown copyright over all development work undertaken on future safety-critical applications.[67] This brings us to a second bone of contention expressed at that time. Since formal methods is such a young field, without an infrastructure of established languages, methodologies or commercially-available tools, the trade and professional associations were concerned to protect the intellectual property rights of their members over the R&D needed if the demands of 00-55 were to be met. As a result of the meeting, the MoD withdrew the crown copyright clause (Lowe:1989). As Brown was emphasising by 1990, "there is no intention to place any...limitations on the existence of private intellectual property rights in the software tools used" for 00-55 applications, since this "will encourage the independent development of progressively better tools and methods for the design and implementation of high-integrity software" (Brown:1990).

If draft 00-55 used the device of full crown copyright in order to be able to control the visibility of the software development environment in its entirety, the Interim standard retains that right (00-55:1991a, 5.2) but achieves it more contextually. That is, 00-55 now states that if existing software is being used as part of a safety-critical development, regardless of its ownership, it may not be incorporated into the design "without the prior written approval" of the MoD (Procurement Executive) Project Manager (00-55:1991a, 5.3). In Interim 00-55, the question is explicitly one of visibility, rather than of ownership, and that visibility is now to be negotiated on a case-by-case basis.

---

67 Martyn Thomas, of Praxis, has noted that the reason for including the Crown Copyright clause was to ensure that developers would not be able to hide errors behind a veil of 'commercial confidentiality' in the event of an accident inquiry. Personal communication from Martyn Thomas, May 1991, in response to an earlier draft of this paper.

A third contentious issue raised by the IEE at the October '89 meeting concerned the requirement of draft 00-55 for particular educational qualifications for those involved in auditing and certifying the safety-critical software. Draft 00-55 had drawn heavily on a report the MoD had commissioned from Cranfield IT Institute to assess the training and education of software engineers needed to support 00-55 (Youll and Simms:1988). Following the recommendations of that study, draft 00-55 had stipulated that the Independent Software Safety Assessor (ISSA) appointed to audit the progress of 00-55 developments must be a Chartered Engineer with at least 5 years experience of using formal methods for safety-critical software engineering.

From the associations' point of view, the problems with meeting this requirement were enormous. There are simply not that many qualified software engineers with this level of experience in formal methods, and only a small pool of recently qualified safety software engineers from which experienced ISSAs might eventually be drawn (Ince:1989b). How did the MoD propose to demand that level of attainment, if the appropriate educational infrastructure was not yet in place?[68] In addition, the requirement that the ISSA be a Chartered Engineer was seen by the IEE as missing the point: engineering has so many specialist branches, that being a Chartered Engineer is no guarantee of competence in *software* engineering,[69] and that this credentials barrier screened out those with an acknowledged proficiency in software assessment without having paper qualifications.[70] Finally, 00-55 stipulated that the ISSA must be independent of the Design Authority (ie. the firm contracted to build the software), yet the definition of 'independence' was not fully addressed, and its implications were not certain. At one extreme, suppliers who are clearly independent of

---

68 At a separate meeting of the BCS Education and Training Working Group, the key issue raised was who was to pay for, or administer, any such educational programme for 00-55. The CSA, in particular, stated that it would only embark on such training if required to do so by law, such were the problems and costs of setting up a coherent national training programme. Interview with John Cullyer, 5th June 1991.

69 See, Tierney (1991) for a discussion of the drift of many kinds of engineers into software engineering - the newest and loosest of the engineering disciplines.

70 Interview with Phil Bennett, Flixborough, 18th June 1991.

each other (eg. they are separately owned) could have good competitive reasons for being nervous of an outside competitor "crawling all over their software".[71] At the other extreme, many supplier companies belong to larger conglomorates: would someone from a different division of such a conglomorate count as being 'independent'?

The architects of 00-55 readily acknowledged in 1989 that the education and training problems of supporting 00-55 are substantial, but that that is not a sufficient reason to avoid stating what is required.[72] However, in re-writing 00-55, the MoD decided *not* to include a general statement on qualifications and training, claiming that it is beyond the remit of a standard to set this kind of national training agenda. Thus, an educational infrastructure for both 00-55 in particular, and formal methods in general, are now seen by the MoD as a matter for the universities and the professional institutions.[73] As to the exact immediate requirements for qualified staff, and clarification of what 'independence' means for the ISSA, Interim 00-55 is considerably less prescriptive than its draft predecessor, as we now see.

The new version of 00-55 is split into two documents - and, incidentally, is immeasurably easier to read than its draft form. The first document, Part 1, is a statement of *requirements* only; the second document, Part 2, is a longer piece which offers detailed *guidelines* for achieving those requirements. Thus, in the sections of Part 1 dealing with staff qualifications (both for design staff and the ISSA), what is demanded is considerably more flexible than draft 00-55: the Design Authority must simply "demonstrate to the MoD Safety Assurance Authority...that the seniority, authority, qualifications and experience of the staff to be employed on the project are satisfactory for the tasks assigned to them" (00-55:1991a, 13) and that the Design Authority and the MoD "shall be satisfied with the experience and qualifications of the Independent Safety Auditor" (00-

[71] Interview with Phil Bennett, Flixborough, 18th June 1991.
[72] Personal communication from Martyn Thomas, May 1991.
[73] Interview with Alan Burton, 4th June 1991.

55:1991a, 16.1).[74] The Guidelines document, however, still retains the preference that the ISSA be a Chartered Engineer with 5 years of experience in the field (00-55:1991b, 16.1.2).

In addition, Interim 00-55 offers a clearer statement as to what is intended by the 'independence' of the ISSA: s/he is to be somebody "commercially and managerially independent from the Design Authority" such that s/he "can assess the safety of the safety-critical software without fetter and free from any possible conflicts of interest" (00-55:1991a, 16.4). The Guidelines suggest that the best solution is to appoint someone from an independent company, "but an independent division of the prime contractor may be acceptable if adequate technical and managerial independence can be shown at Director or Board level" and the ISSA role "may be taken by several people" (00-55:1991b, 16.1.1). In clarifying what independent status means, it is apparent that the MoD have *not* relaxed 00-55's requirement in respect to the external auditing of work. Since possible 'conflicts of interest' are bound to express themselves differently in each organisational context, it seems likely that it will be experience in implementing 00-55 which will gradually expose where the difficulties and sensitivities of this requirement lie.

A fourth area of concern expressed by the IEE at the October '89 meeting concerned the focus of 00-55 on *software* as the key area requiring regulation in safety-critical systems (and much of this discussion was conducted in terms of contrasting 00-55 with the newly released IEC draft). Even if the software is built with the greatest care possible, that need not mean that the system as a whole is any safer.[75] From the MoD's point of view, this criticism must surely have been hard to swallow: 00-55 was under attack for what it was not intended to cover. The 00-55 architects had been aware of the implications of concentrating this standard on safety-critical software and, in formulating 00-55, had claimed it as only the first in a series of MoD initiatives to regulate the safety-criticality of systems (Brown:1990, White:1990, Ryan:1990).

[74] In a separate paper I consider how this very flexibility in the informal recruitment of staff to these posts may work to consolidate 'insider tracks' in the formal methods field (Tierney:1991).
[75] Interview with Dr Phil Bennett, Flixborough, 18th June 1991.

While we have already seen that the MoD had its reasons for opting to go it alone in building 00-55, the comparison of 00-55 with the IEC draft at this meeting highlighted that some *convergence* between standards for safety-critical software would be desirable as a next move. Indeed, Interim 00-55 is claimed as an instantiation of the highest level of the IEC standard, and is less 'isolationist' than its predecessor, in that the focus has shifted away from the finer points in the technical requirements of draft 00-55. Thus, although the MoD's mandating of formal methods has remained constant between the draft and interim versions, the *manner* in which formal design may proceed in Interim 00-55 has changed somewhat. This brings us, finally, to an account of one of the most contested issues raised at the October '89 meeting (and subsequently) about draft 00-55: the so-called 'banned practices'.

In mandating formal methods in draft 00-55, its architects had to contend with the fact that formal methods - no less than any other software engineering techniques - are limited in what they can achieve. While they offer, by means of formal proofs and rigorous arguments, greater assurance that the safety-critical software is *correct* with respect to its specification, in general they are not good at assessing how a piece of software will *perform*, on a given target machine, and within a given time (Joseph and Goswami:1990). For this, other checks need to be instituted - most notably, extensive dynamic testing of all software, after it has been put through the process of formal static analysis. Given the limits of current formal techniques and tools - both for establishing proofs of correctness and for controlling the relationship between correctness and performance assessment criteria - draft 00-55 invoked 'the banned practices' to minimise the likelihood of uncontrollable occurrences cropping up once the safety-critical software was implemented.

The banned practices were, in effect, a list of 'don'ts'. Draft 00-55 prohibited the use of any implementation techniques that are difficult to analyse formally. Amongst the design practices banned were the use of dynamic memory allocation, recursion and floating point arithmetic. Programming in low-level assembly languages was prohibited, as was the use of

interrupts. All the safety-critical software had to be designed so as to be thoroughly isolated from all other parts of the system. Thus, potentially 'leaky' techniques such as multi-processing on a single processor, or spreading the intended execution of separate elements of the software over a number of processors, were also disallowed. This ban - though motivated by attempts to control for what formal methods *cannot* achieve - was highly controversial.

For many suppliers and associations, 00-55 was seen as unduly encroaching on the expertise of software engineers: the MoD had taken upon itself to tell skilled practitioners not just what work to do, but how to go about doing it.[76] In this sense, draft 00-55 seemed to question the competence of the occupation by refusing to allow skilled designers leeway for exercising their judgement, their common sense, and their accumulated experience in safety-critical work. The 'principle of analysability' which guided the banned list (Brown:1990, Ryan:1990) operates at the expense of those tricks-of-the-trade which are well-known to skilled practitioners: the ban insisted that designers stick to the road, even where their skill and experience in using techniques which are formally unanalysable may make it more efficient to cut across the fields.

Even amongst those designers who bend their expertise towards writing software which is amenable to formal analysis, the rigidity of the MoD ban was seen as compromising the production of *efficient* software.[77] For instance, for some, the ban on floating point arithmetic was contentious: it was constraining and its use need not affect the rigour of the software (Ravn and Stavridou:1990). Colin White has suggested that though floating point is inherently unanalysable,[78] the designer has to write considerably more code

---

[76] Interview with Colin White, Secretary of the Safety-Critical Steering Group till 1990, London, 9th November 1990.

[77] The RSRE representatives on the Working Group, for instance, were keen not to completely sacrifice efficiency, in the pursuit of quality, and were opposed to the banned practices as too stringent a set of limits on a software engineer's good judgement. Interviews with Alan Burton, 4th June 1991, and John Cullyer, 5th June 1991.

[78] Interview with Colin White, London, 9th November 1990. White also added that his view of floating point is a minority one, both within the MoD and the formal methods community more generally. That is, there are formal definitions of floating point

if s/he is to avoid using it and this, in itself, raises rather than reduces the likelihood of unpredictable errors occurring in the system. He has suggested that the problem with floating point is not that its use is unsafe, but that it cannot be *demonstrated* to be safe. Hence the ban, and hence the resulting controversy.

As a result of the debate over the last two years, the banned practices in draft 00-55 have been dropped from the Requirements section of Interim 00-55 (ie, Part One) and a requirement for analysability substituted, with extensive supplementary comments in the Guidelines section (Part Two). Thus, the Interim standard now endorses the skills involved in formal methods work, rather than prescribing just 'one best way' of going about such work. Although formal methods remain mandatory - and that, in itself, limits the choice of a designer in picking and choosing amongst techniques - the standard offers greater scope for the exercise of judgement in balancing questions of quality against efficiency, whilst remaining under the formal methods umbrella.[79]

Given the differing interests in draft 00-55 - interests which perhaps were most nearly 'summarised' at the October '89 meeting, but which have been expressed separately and severally since then - it is hardly surprising that the circulation of 00-55 and 00-56 for comment in May 1989 prompted strong and sustained critical reaction. If the IEE meeting was intended as a kind of 'denouncement' of 00-55, we have seen in this last section how the MoD have attempted to shift the emphasis of 00-55, in the intervening years, towards being a standard which is more flexible and contextual, in requiring formal methods for safety-critical software. Nonetheless, in issuing Interim 00-55 and 00-56 in May of this year, the MoD has been careful to avoid a recurrence of the manner in which the IEE spoke to the press about 00-55 in October '89.[80]

---

currently available. A problem still lies with the hardware because few implementations conform to a recognised formal definition of expected operation.

79 Personal communication from Colin White, May 1991, in response to an earlier version of this paper.

80 For instance, the MoD called meetings with various interest groups to present Interim 00-55 during June and July 1991, with the stipulation that the technical or

Certainly, the suppliers, government policy-makers, educational institutions and the trade and professional associations have grown used to the existence of 00-55, and the quieter general reception of the Interim documents would appear to reflect that. In addition, however, the launch of the interim standards differ in one crucial respect from the launch of the drafts: there is no mention made of when the standards will be made *mandatory* for all safety-critical defence applications.[81] In 1989, the feeling had been that after the feedback period was up, the standards would become fully operational by 1995. Indeed, we saw in Section Four that it was fear of the *speed* at which 00-55 might be enforced which underpinned many of the anxieties expressed about the use of formal methods, without there being an adequate infrastructure in place. In contrast, the Interim standard notes in its preface that it is "provisional in order to obtain information and experience of its use"; that "it is recognised that many of the procedures and technical practices needed to generate safety-critical software are still being developed"; and that, although the standard is operational, "if any difficulty arises which prevents application of the Def Stan, the Directorate of Standardisation shall be informed so that a remedy may be sought" (00-55:1991a, p.1).

6. Some Conclusions: The Effect of 00-55 on Formal Methods Software Engineering in the UK

00-55 is a standard which regulates the design and management of safety-critical software for defence applications, by means of formalising the production process and of mandating the use of formal methods for software development. The standard is partially a codification of current agreed principles for software engineering, but also a framework for "good software engineering practice" (Ryan:1990) moulded into the new image of formal methods of specification and design. This paper has described the emergence of 00-55 as an interim standard, with a view to exploring why,

---

national press will not be invited to attend directly, and that audiences will not seek press attention for 14 days following the MoD's presentations.

81 The only MoD development work for which 00-55 is currently mandatory is the European Fighter Aircraft project (Quintas et al:1990).

and how, its release has intensified the 'formal methods debate' in the UK. To conclude, we look at how the framework may affect the promotion of formal methods nationally.

00-55 has undoubtedly raised the profile of formal methods within UK software engineering. Milan Kuchta has noted that it is "extremely helpful for motivating advances in the state of the practice of critical systems development. By mandating techniques that are extensions of current practice, such standards strongly motivate industry to improve and enhance their capabilities" (Craigen:1990). Since the take-up of formal methods in the UK is closely linked with 00-55, in so far as many suppliers gain some proportion of their business from the MoD, the standards can powerfully shape a national agenda for what formal methods come to mean in Britain.

For one thing, we have seen that 00-55 emphatically locates itself in a common 'culture of safety': it values formal methods primarily for its contribution to the production of *safer* software. This stance does not conflict with the application of formal methods to, say, writing security-critical software or - perhaps more significantly - to writing 'ordinary' commercial software which has lower error rates and requires less maintenance.

However, it seems likely that, even if it is the case that the UK's safety-critical community previously held itself at some distance from the formal methods community, the two are now inextricably inter-twined. As a result of 00-55, formal methods cannot now be ignored as an option in the safety engineering tool-kit, an option moreover whose exclusion from - rather than inclusion in - the design process, must now be justified. In addition, it seems likely that 00-55's emphasis on safety-critical software will mean that any emergent body of UK expertise in the application of formal methods to substantial projects (as distinct from the more limited projects of basic research) will not remain the preserve of the more hidden, sealed-off, informational networks of the security-critical defence community.

In addition, 00-55 may shape the form in which formal methods are taken up in the UK by virtue of its heavy reliance on *tools*, particularly those which assist in the production of proof. It is widely acknowledged that the

technology base for formal methods is underdeveloped (Brown:1990, Ince:1990a, White:1990), and that much current work on formal methods is confined to academic and industrial research sites. The tools that are currently available - assuming that such artefacts offer the speediest option for the transfer of formal methods into commercial use - are inadequate in so far as each is designed for a limited purpose and, collectively, they do not mesh well with each other (Craigen:1990, Jesty et al:1990).

The existence of a framework such as 00-55 may have the effect of aiding the consolidation of a market for *particular* tools (eg. tools to support formal static analysis) rather than encouraging the piecemeal emergence of divergent or duplicated ones (Craigen:1990). That said, however, many of those involved in either building, or responding to, 00-55 have noted that the process of developing reliable and robust tools for diverse safety-critical markets is a slow business indeed. Thus, an outstanding problem remains with operationalising 00-55: though it requires that the tools used for safety-critical software must themselves be verified, it must make-do with the fact that most tools which support formal analysis and verification can currently only partially satisfy this requirement.

This brings us to a final thought. A standard may state that a 'revolution' is required, and yet, in and of itself, be unable to obtain the conditions which ensure that that revolution happens. We have seen in the evolution of 00-55 (for instance, in its dropping of a proposed national agenda for the education and training of safety software engineers) that it does not - and cannot - control the complex evolution of UK software engineering on its own. What the MoD has done is state the direction in which it believes the discipline should be moving, one which demands a central role for formal methods.

Alan Burton, of RSRE, has noted that the single most innovative feature of 00-55 is *not* that it is built around formal methods - every single technique, notation, and tool mentioned in 00-55 has precedents of use elsewhere - but that it puts all these *together* into a single framework.[82] This total framework is still untested on practical defence projects. And, indeed, it

82 Interview with Alan Burton, Malvern, 4th June 1991.

may remain so. The MoD see the next move in the evolution of 00-55 as coming from 'demonstrator projects' which are to be closely monitored in situ, to assess whether this framework is workable in practice. Interim 00-55 is to be tried out on a case-by-case basis, and it is acknowledged by the MoD that 00-55 cannot yet be applied literally, or universally.[83] Thus, the crucial issue of whether a contractor has conformed to 00-55 becomes a matter of fine judgement: 'conformance' is a more relative concept than was evident, say, in 1989. To this extent, 00-55 is more a statement of intent than a categoric code of practice.

Thus, it seems likely that the take-up of formal methods in the UK, whether in specific safety-critical applications or in the building up an infrastructure of tools, will not be 'driven' by the sheer existence of 00-55. As with other innovative advanced process technologies, the take-up of formal methods may be more thoroughly established by the piece-meal emergence of tools for specific purposes; by developers trying out just some aspects of formal techniques in particular application domains; perhaps even by contested cases which are adjudicated by a court of law or an accident inquiry team. In this context, it may turn out that the outstanding achievement of the MoD's initiative has been to create a great debate which will surely continue to roll and rumble, shaping software engineering in the next decade both directly and by default. If this were the eventual epitaph given by industry to 00-55, a very good one it would be.

83 Interview with Alan Burton, Malvern, 4th June 1991.

References

Abrahams, P (1987) 'What is Computer Science?' Communications of the ACM Vol 30 No 6 pp.472-3

ACARD (1986) Software: A Vital Key to UK Competitiveness , Advisory Council for Applied Research and Development, HMSO, London

Andrews, D (1990) 'The Vienna Development Method' in Ince and Andrews (eds) The Software Life Cycle Butterworths

Barrie, D (1989) 'New Model Army?' Computing 17th August

Bloomfield, R (1990) SafeIT 1: Overall Approach ICSE Secretariat, Dept of Trade and Industry, London

Bloomfield, R and Brazendale, J (1990) SafeIT 2: Standards Framework ICSE Secretariat, Dept of Trade and Industry, London

Bloomfield, R, Froome, P and Monahan, B (1991) 'Formal Methods in the Production and Assessment of Safety Critical Software' in Reliability Engineering and System Safety Vol 31 No 1-2 pp.51-66

Brown, MJD (1990) 'Rationale for the Development of the UK Defence Standards for Safety-Critical Computer Software' COMPASS Conference. 7, June, Ministry of Defence

Craigen, D (reporteur) (1990) 'Government Forum' in Dan Craigen (ed) Formal Methods for Trustworthy Computer Systems (FM89) Report from FM89: A Workshop on the Assessment of Formal Methods for Trustworthy Computer Systems 23-27th July 1989, Halifax, Canada. Springer-Verlag

Cullyer, W and Pygott, C (1987) 'Application of Formal Methods to the Viper Microprocessor' in IEEE Proceedings 134 Part E, No 3, May

DeMillo, Lipton and Perlis (1979) 'Social Processes and Proofs of Theorems and Programs' in Communications of the ACM Vol 22 No 4 April

Dettmer, R (1988) 'Making Software Safer' in IEE Review September

Ernest-Jones, T (1988) 'MoD Insists upon Formal Methods' Computer Weekly 9th June

Fetzer, JH (1988) 'Program Verification: the Very Idea' in Communications of the ACM Vol 31 No 9 September

Friedman, A (1989) Computer Systems Development: History, Organisation and Implementation John Wiley & Sons

Gibbons, P (1990) 'What are Formal Methods?' in Ince and Andrews (eds) The Software Life Cycle Butterworths

Hoare, CAR (1986) 'Mathematics of Programming' Byte August, pp.115-149

Ince, D (1989a) 'A Standard for Formal Education' Computing 25th May

Ince, D (1989b) 'Troubled Standard Bearers' The Guardian July 20th
Ince, D (1990a) 'Defending New Values' Computing April 19th
Ince, D (1990b) 'Building Upon a Formal Design' Computing June 21st
Ince, D and D Andrews (1990) 'Software Engineering and Software Development' in Ince and Andrews (eds) The Software Life Cycle Butterworths
Institute of Electrical Engineers (1989) Software in Safety-Related Systems A Joint Report of the IEE/BCS, IEE: London
Jackson, M (1989) 'Formal Methods and Critical Software Development' SafetyNet '89 Conference Proceedings, RSRE, on Industrial Experience of Formal Methods SafetyNet, Worcester
Jesty, P, Buckley, T and West, M (1990) 'Critical Software - A Pre-Standard and its Certification' SafetyNet '90 Conference Proceedings, Royal Aeronautical Society, on Formal Methods for Critical Systems Development SafetyNet, Worcester
Joseph, M and Goswami, A (1990) 'A Formal Description of Realtime Systems' in Ince and Andrews (eds) The Software Life Cycle Butterworths
Libetta, L and Woolgar, S (1991) 'Some Aspects of Standards Discourse' paper to PICT National Network Conference, Yorkshire, 20-22 March
Lowe, K (1989) 'MoD Shifts Position on Defence Standard' Computing October 26th
MacKenzie, D (1991) 'Formal Methods and the Sociology of Proof' British Computer Society Formal Aspects of Computer Science 4th Annual Workshop on Refinement, Cambridge 9th-11th January
MacKenzie, D and Spinardi, G (1991) 'The Technological Impact of a Defence Research Establishment' Dept of Sociology, Edinburgh University
Macro, A and Buxton, J (1987) The Craft of Software Engineering Addison-Wesley
Macro, A (1990) Software Engineering: Concepts and Management Prentice Hall
Mellor, P (1989) 'Can you Count on Computers?' New Scientist 11th February
00-55 (Part 1)/Issue 1 (1991a) Interim Defence Standard The Procurement of Safety Critical Software in Defence Equipment Part 1: Requirements, Ministry of Defence
00-55 (Part 2)/Issue 1 (1991b) Interim Defence Standard The Procurement of Safety Critical Software in Defence Equipment Part 2: Guidance, Ministry of Defence
00-56/Issue 1 (1991) Interim Defence Standard Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment, Ministry of Defence
Pelaez, E (1988) A Gift from Pandora's Box: The Software Crisis PhD Thesis, Edinburgh University
Philipson, S (1989) 'More on Proper British Programs' in ACM SIGSOFT Software Engineering Notes Vol 14 No 1, January
Quintas P, Hobday M, and Guy K (1990) Evaluation of the Alvey Software Engineering Programme SPRU, University of Sussex, Brighton, January
Quintas P (1991) 'Engineering Solutions to Software Problems: Some Institutional and Social Factors Shaping Change' forthcoming in Technology Analysis and Strategic Management
Ravn, A and Stavridou, V (1990) 'Specification and Development of Safety Critical Software: An Assessment of MoD Draft Standard 00-55' Proceedings of ICSE 12 Workshop on Industrial Use of Formal Methods, Nice, March
Ruggles, C (1990) ed. Formal Methods in Standards A Report from the BCS Working Group, Springer-Verlag
Ryan, P (reporteur for Thomas, M) (1990) "Defence Standard 00-55 and other Standards: What is Proposed and Why" in Dan Craigen (ed) Formal Methods for Trustworthy Computer Systems (FM89) Report from FM89: A Workshop on the Assessment of Formal Methods for Trustworthy Computer Systems 23-27th July 1989, Halifax, Canada. Springer-Verlag
Schlender, BR (1989) 'How to Break the Software Logjam' Fortune 25th September
Sommerville, I (1989) Software Engineering Addison-Wesley
SSSC (1991) Computers at Risk: Safe Computing in the Information Age System Security Study Committee, National Academy Press: Washington
Tierney, M (1991) 'Some Implications of Def Stan 00-55 on the Software Engineering Labour Process in Safety-Critical Developments' Research Centre for Social Sciences, Edinburgh University
Whalley, P and Crawford, S (1984) 'Locating Technical Workers in the Class Structure' in Politics and Society Vol 13 No 3 pp.239-252
Whalley, P (1986) The Social Production of Technical Work: The Case of British Engineers MacMillan: Basingstoke
White, C (1990) 'Possible Areas of Work in Safety Critical Software' MoD, High Holborn, August 1990

Youll, DP and Simms, PG (1988) Study of the Training and Education Needed in Support of Def Stan 00-55 Cranfield IT Institute Ltd, September

Edinburgh PICT Working Paper Series

1. *Edge, David,* The social shaping of technology.
2. *MacKenzie, Donald,* 'Micro' versus 'macro' sociologies of science and technology.
3. *Williams, Robin and Russell, Stewart,* Opening the black box and closing it behind you : on microsociology in the social analysis of technology.
4. *Fleck, James,* Innofusion or diffusation? The nature of technological development in robotics.
5. *Peláez, Eloína,* Parallelism and the crisis of von Neumann computing.
6. *MacKenzie, Donald,* Parallel computing : some issues for social research.
7. *Williams, Robin,* The development of models of technology and work organisation with information and communication technologies.
8. *Webster, Juliet,* New technology, old jobs : secretarial labour in automated offices.
9. *Fleck, James,* The development of information-integration : beyond CIM?
10. *Peláez, Eloína,* What shapes software development?
11. *Molina, Alfonso,* Information technology : anatomy of a successful European collaboration.
12. *Webster, Juliet,* Gender, paid work and information technology.
13. *MacKenzie, Donald,* The influence of the Los Alamos and Livermore National Laboratories on the development of supercomputing.
14. *Fleck, James, Webster, Juliet and Williams, Robin,* The dynamics of implementation : a reassessment of paradigms and trajectories of development.
15. *Rooney, Brendan,* The implementation of CIM in rubber manufacturing : a flexible response : case study of Avon Rubber.
16. *Fransman, Martin,* Cooperation, competition and international competitiveness : the case of Japanese central office switches.
17. *Webster, Juliet,* The shaping of software systems in manufacturing: issues in the generation and implementation of network technologies in British industries.
18. *Oakley, Brian,* Look back in Alvey: why support for R&D is not enough.
19. *Molina, Alfonso,* The development of public switching systems in the UK and Sweden: the weight of history.
20. *Fransman, Martin,* Controlled competition in the Japanese telecommunications equipment industry: the case of central office switches.
21. *Webster, Juliet,* Automation in the social office: women's skills and new technology.

22. *Tierney, Margaret & Williams, Robin,* Issues in the blackboxing of information technologies.

23. *Peláez, Eloína,* From symbolic to numerical computing: the story of thinking machines

24. *Molina, Alfonso,* Building up a neural network sociotechnical constituency: a contribution to the formulation of a UK strategy.

25. *Tierney, Margaret,* The formation and fragmentation of computing as an occupation: a review.

26. *Howells, John & Hine, James,* Competitive strategy and the implementation of a new network technology: the case of EFTPoS.

27. *MacKenzie, Donald,* Economic and sociological explanations of technical change.

28. *Edge, David,* Mosaic array cameras in infrared astronomy.

29. *Tierney, Margaret & Fleck, James,* The management of expertise: knowledge, power and the economics of expert labour.

30. *Tierney, Margaret,* The evolution of Def Stan 00-55 and 00-56: an intensification of the 'formal methods debate' in the UK.

31. *Howells, John, Alexander ,Nicholas & Hine ,James,* New technology and the changing bank - retail industry relationship.

32. *Howells, John,* A socio-cognitive model of innovation.

33. *Tierney, Margaret,* Negotiating a software career: informality and 'the lads' in an Irish software installation.

34. *Fleck, James,* Selectionism dominant: an essay review of books by Basalla, Giere, and Hull.

35. *Faulkner, Wendy,* Understanding industry - academic research linkages: towards an appropriate conceptualisation and methodology.

36. *Molina, Alfonso,* Pressures for change in the global distribution of the microprocessor industry: is US domination about to come to an end?

37. *Fleck, James,* Innovation during implementation: configuration and CAPM.

Please supply ..... copy(ies) of **Edinburgh PICT Working Paper(s)** Nos.....

at a cost of £5 per copy inclusive of postage.

I enclose a cheque for .....

Please make cheques payable to '**The University of Edinburgh**' and send to Mrs Barbara Silander, RCSS, The University of Edinburgh, 56 George Square, Edinburgh, EH8 9JU

Edinburgh PICT Research Report Series

1. *Molina, Alfonso,* The transputer constituency : building up UK/European capabilities in information technology.

Edinburgh PICT Student Paper Series

1. *Hedges, David,* A review of value added and data services and their implications for the City of Edinburgh.

2. *Hosein, D.L. Abbass,* A critical evaluation of the industrial robot : use of a strategic perspective to enhance management ability to derive corporate benefit from its implementation.

3. *Harremoes, Steffen,* Consequences of the Single European Market on main exchange manufacturers : prospects for UK producers.

4. *Hardstone, Gillian,* A simple planning board and a jolly good memory: time, work organisation and production management technology.

Please supply.....copy(ies) of **Edinburgh PICT Research Report(s)** Nos.....

at a cost of £20 per copy inclusive of postage (reduced rate of £5 for academics).

Please supply.....copy(ies) of **Edinburgh Student Paper(s)** Nos.....

at a cost of £5 per copy inclusive of postage.

I enclose a cheque for .....

Please make cheques payable to '**The University of Edinburgh**' and send to Mrs Barbara Silander, RCSS, The University of Edinburgh, 56 George Square, Edinburgh, EH8 9JU.

PICT at Edinburgh

The Programme on Information and Communication Technologies (PICT) is a major initative of the Economic and Social Research Council, which aims to explore social science perspectives on the rapidly evolving Information and Communication Technologies (ICTs) and inform policy debate in the field. The research is conducted by a network of six centres - Brunel University (CRICT); Polytechnic of Central London (CCIS); The University of Edinburgh (RCSS); UMIST (CROMTEC); University of Newcastle (CURDS); and University of Sussex (SPRU) - and coordinated from the University of Oxford.

Edinburgh PICT research is based at the Research Centre for Social Sciences and draws on expertise in the Departments of Business Studies, Economics and Sociology, as well as the Science Studies Unit. The group starts from the assumption that the development and implementation of new technologies cannot be wholly explained by technical considerations, but that complex social, political and economic factors are involved. The research effort therefore focuses on the 'social shaping' of ICTs , at the level of detailed technical design. It aims to elucidate the considerable scope which exists - for both producers and users of technology - to influence the direction and consequence of technological change. Much of the research involves building strong links with the policy community, in industry and in government.

Edinburgh PICT is part of a strong and growing base of socio-economic research on technology at the University, and runs a Doctoral Programme of Social and Economic Research on Technology within the Faculty of Social Sciences. Both teaching and research activities benefit from close links with departments in the School of Information Technology. In addition, members of the Edinburgh group collaborate with researchers in neighbouring Higher Education Institutions, and with other centres in the PICT national network.