HSC
Health & Safety
Commission

The use of computers in safety-critical applications

Final report of the study group on the safety of operational computer systems

HSC
Health & Safety
Commission

The use of computers in safety-critical applications

Final report of the study group on the safety of operational computer systems

First published 1998

Formerly ISBN 0 7176 1620 7
And now available in electronic format only

Contents

Preface

I should like to thank the Chairman and the members of the NuSAC Study Group on the Safety of Operational Computer Systems for their efforts in producing this report on a topic of considerable current interest. The Committee endorses the report and its recommendations, and commends it to all involved in the field.

Sir David Harrison CBE,FEng
Chairman of NuSAC

Summary

Computer systems are becoming ubiquitous in most industries, and the nuclear industry is not alone in finding computers entrusted with important safety-critical roles. The Study Group believes that computer systems can bring many advantages in safety and cost reduction to the nuclear industry. There is empirical evidence, after the fact, that their use in safety-critical operations has been successful (although complacency is inappropriate since the amount of operational evidence so obtained is far from sufficient to justify a conclusion that any of these systems is free from design faults that could cause them to fail, perhaps catastrophically). On the other hand, it has often proved difficult and expensive to demonstrate at the time of licensing that software-based systems would be sufficiently dependable in operation, so many of the conclusions and recommendations in this report concern improvements in ways of establishing safety claims.

Whilst much of this report addresses the peculiar difficulties that are posed by the use of computers, readers should not see this emphasis as implying that the Study Group takes a generally sceptical view of the efficacy of computers in safety-critical roles. On the contrary, we see our report partly as a contribution to the debate in the technical community aimed at solving these problems, so that the huge potential of computer systems - particularly the potential to increase the safety of other systems - can be released for the benefit of society.

Computer systems differ from conventionally engineered electrical and mechanical systems mainly because of the essentially discontinuous nature of discrete logic (in particular, of software). This limits the claims that can be made about future behaviour from successful testing, and makes the effect of any changes much less predictable.

Computer systems are vulnerable because they almost invariably contain design faults in their software (and perhaps in their hardware) that are triggered when the computer system receives appropriate inputs. Many of these faults will have been present from inception, and others will have been introduced during any changes that have taken place throughout the system lifetime. The reality is that even programs of surprisingly modest size and complexity must be assumed to contain design faults. It is the responsibility of the designer, having done whatever is possible to minimise the number of residual faults, to try to ensure that any remaining ones do not have an unacceptable effect upon other systems with which the computer system interacts: in particular, that they do not compromise the safety of the wider system.

The major advantage of computers over simple hard-wired hardware lies in their *general purpose* nature, derived from their programmability. This allows much of the specific functionality in a particular application to be implemented in software and thus avoid some of the unreliability arising from component failures that would be incurred in a purely hardware design. Indeed, in many industries computers allow functionality that would have been inconceivable in hardware alone. Often, as in the nuclear industry for example, the use of computers allows safety enhancing features, such as more sophisticated monitoring.

The very advantages of software-based systems, however, bring some special problems of *difficulty*, *novelty* and *complexity*. Designers tend to take on tasks that are intrinsically *difficult* as a result of the very flexibility of software: when a computer system is introduced into a context where a conventional hardware solution already exists, it is commonplace to take advantage of this flexibility to offer extra (sometimes excessive) functionality. In more and more applications, however, there is no hardware system that the computer is replacing: instead, the great flexibility of software is being used to do something completely *novel*. These trends result in great *complexity* in the end product, with the attendant problems of human comprehension.

These problems cause difficulties both in building software-based systems that are sufficiently reliable, and in assuring that reliability so that justifiable confidence can be placed in the

system's fitness for purpose. In arriving at the following conclusions, much of the Study Group's discussions centred upon the second of these difficulties concerning the contribution of software to safety cases.

Regulatory practice

1 There is evidence from our studies of several different industries that the safety cases in one industrial sector, even just within the UK, might not be acceptable in another. The reasons for this need to be investigated and understood. One area we have already identified as problematic - perhaps because it is so difficult - is the application of ALARP/SFAIRP to software-based systems.

2 There are considerable benefits to be gained from international regulatory consensus concerning the demonstration of fitness for purpose of future safety-critical computer systems. Current activities to this end are welcomed and should continue to be supported.

3 There should be a review of the current organisational and technical approach to independent assessment of computer-based systems in the UK nuclear industry, taking account of the experience of different practices in other industries, e.g. the DER (Designated Engineering Representative) system in operation in the US aircraft industry.

Safety cases

4 Current guidance, based upon appropriate empirical and theoretical support, should be further developed for the structure and content of safety cases for software-based systems. This guidance should address the somewhat different problems of new applications software, previously-developed software (PDS), and commercial off-the-shelf (COTS) software.

5 The design of the system and its safety case should be robust to such changes that are proven to be necessary throughout the anticipated life of the system.

6 The reliability levels that can be demonstrated for software are quite modest, and safety cases should not require levels of software reliability beyond these.

7 Although the use of software design diversity may have positive benefits, claims for statistical independence from diversity should be treated with great caution. In particular, claims for extremely high reliability levels based upon assertions of complete independence of failures in two or more diverse software systems are not believable (with the attendant implications for assessing systems in which one software system backs up another). More work is needed to understand the extent and nature of diversity and to address the difficult problem of evaluating the degree of dependence present in particular cases.

8 Confidence in assessments of software-based systems is usually less than for more conventionally engineered systems. We believe that attention should be given to incorporating formally in licensees' and regulatory guidance a recognition of the importance of the level of confidence that can be placed in assessments of risk within the concept of an 'adequate' safety case. What is needed is to clarify and define the notion of 'adequacy', such that it can be used to guide and justify decisions as to the required extent of activities that will establish the level of confidence that can be placed in a risk assessment.

9 An appropriate method for the formal analysis of competencies should be developed and used as a basic demonstration element within any safety case which aims to justify a high integrity claim for a computer-based safety.

10 In the face of the potential new types of ‘cyber threat’ it is important to ensure that appropriate design features, production practices, and operational controls are in place which will be effective in countering such (and similar) threats to the dependable operation of a computer system important to safety on a nuclear power plant.

Computer system design and software engineering

11 Attention should be given to better practices for the elicitation and specification of computer system requirements, since it is errors made at this early stage in system life that generally have the most serious impact.

12 Consideration should be given to the problems of safety demonstration from the very earliest stages of system design. Further work is needed on the development of designs for software-based systems (in particular those incorporating COTS software) that make them amenable to safety demonstration. Practical methods for documenting requirements, design and implementation, which allow safety requirements to be traced and verified through these stages, need to receive more attention.

13 Careful consideration should be given to the allocation of responsibilities between computers, humans and (conventionally engineered) hardware, and to interactions between these (particularly the impact of the introduction of computers upon operators). This allocation should be justified in the safety case.

14 One important guiding principle in the design of computer systems should be the avoidance of unnecessary complexity, whether this comes from the provision of unnecessary functionality or from the use of inappropriate structuring in the design. The safety case should contain rigorous justification of the functionality of the system and of its structure. Much more emphasis should be placed on structuring systems so as to achieve a high degree of isolation between the implementation of the most critical requirements, and all others.

15 Some techniques show particular promise for achieving and/or assessing safety and reliability in critical software-based systems, but would benefit from further investigation. Examples include:

- fault tolerance via design diversity
- probabilistic and statistical techniques for quantitative evaluation of safety and reliability
- formal methods for specification and verification of system properties.

16 Better tools, together with methods for qualifying them, should be developed to support the development and assurance of high integrity real-time software, including the difficult problems associated with concurrency and distributed systems.

Standards

17 Present standards are not of a form and content that would allow a positive assessment of conformance on its own to confirm fitness for purpose. Whilst standards are important, current ones do not reflect the UK safety case approach and ALARP, nor do they take adequate account of existing technologies and industry trends such as increased use of COTS software. The emerging generic standard, IEC 1508[1], and the standards for nuclear computer-based systems being developed by TC45A, should be reviewed and efforts made to negotiate amendments which would take account of these deficiencies. The UK nuclear industry and its regulator are urged to maintain active participation generally in the ongoing developments of standards relating to computer-based safety-critical systems.

[1] This has become IEC 61508 since the report was written.

Research

18 The recent experience of the different organisations involved in those aspects of the Sizewell B licensing that concerned the computerised primary protection system has valuable lessons for similar exercises in the future - both in the nuclear industry and in other industries. This experience should be recorded in as much detail as feasible whilst the collective memory remains fresh, perhaps as a project within the Nuclear Safety Research programme.

19 In our report we have identified several areas where technical progress is needed to maintain and improve our ability to design, build and evaluate safety-critical software-based systems. We are encouraged to see that many of the issues of most relevance to the UK nuclear industry are already being addressed as part of the UK nuclear research programme, and via links to programmes elsewhere. We recommend that NuSAC's Sub-Committee on Research use our report as a template for its on-going review of this part of the research programme.

Foreword

The Advisory Committee on the Safety of Nuclear Installations (ACSNI)[2] set up the Study Group on the Safety of Operational Computer Systems on 17 March 1995 with the following terms of reference:

> *to review the current and potential uses of computer systems in safety-critical applications;*
>
> *to consider the implications for the nuclear industry;*
>
> *in this context, to consider developments in the design and safety assessment of such computer-based systems, including other aspects of control systems;*
>
> *and to advise ACSNI where further research is necessary.*

This report addresses the broad principles upon which the Study Group believes the evidence and reasoning of an acceptable safety case for a computer-based, safety-critical system should be based. It also discusses, but does not attempt to cover in detail, the extent to which the UK nuclear industry already accepts these principles in theory, and the extent to which they act on them in practice.

An earlier version of the report was endorsed by ACSNI at its meeting in November 1996. The present and final version has been prepared following the comments we received at that meeting, as well as later written comments on behalf of several companies and individuals.

The membership of the Study Group was:

Professor B Littlewood	Professor of Software Engineering and Director, Centre for Software Reliability, City University, London *Chairman, ACSNI member*
Dr L Bainbridge	Formerly Reader in Psychology, University College, London *Former ACSNI member*
Professor R E Bloomfield	Safety-related Software Specialist/Consultant, Adelard, London
Professor P-J Courtois	Assessor, AV Nucléaire, Brussels *and* Professor of Computer Science, Département d'Ingénierie Informatique, Université de Louvain-la-Neuve, Belgium
Dr A H Cribbens, MBE	Technical Specialist, British Rail Research, Derby *To May 1996*
Dr R S Hall	Independent Consultant in Nuclear Safety, *ACSNI member, nominated by the CBI*
Mrs V L P Hamilton	Group Leader, Safety-Critical Systems Group, GEC-Marconi Research Centre, Chelmsford *From March 1996*

[2] Now the Nuclear Safety Advisory Committee (NuSAC).

Professor A D McGettrick	Professor of Computer Science, Department of Computer Science, Strathclyde University, Glasgow
Dr G Oddy	Technical Director, EASAMS Ltd, Borehamwood *To November 1995*
Professor B Randell	Professor of Computing Science, Department of Computing Science, University of Newcastle, Newcastle upon Tyne
Mr P Tooley	Group Head, Technical Investigations and Safety, Nuclear Electric Engineering Division *Nominated by the TUC*

The Nuclear Safety Directorate's observer was

Mr D M Hunns	HM Superintending Inspector, Nuclear Safety Directorate, Health and Safety Executive

The Study Group was supported in its activities by:

Mr N Wainwright	HM Principal Inspector, Nuclear Safety Directorate, Health and Safety Executive *Technical Secretary*
Mr C Simon	Nuclear Safety Directorate, Health and Safety Directorate *Secretary*

The Study Group, in preparing this report, met first on 10 May 1995 and a further 15 times, including three residential meetings.

The Study Group was extremely fortunate to be able to consult experts in the French nuclear industry, the UK railway industry and the US and UK aircraft industries. Visits were made by small sub-groups to the following organisations:

Westinghouse Signals Limited and GEC-Marconi Avionics in the UK;

Boeing Commercial Airplane Group and the US Federal Aviation Authority in Seattle, USA;

Direction de la Sûreté des Installations Nucléaires and Institut de Protection et de Sûreté Nucléaire in Paris, France.

We would like to express our thanks to the organisations involved, and to the individuals with whom we had extensive discussions:

Mr J Mills, Mr S D J Pilkington, Mr H Ryland (Westinghouse Signals)

M F Feron (DSIN, France)

Mme N Guimbail, M J-Y Henry, Mme B Soubies (IPSN, France)

Mr I Newton, Mr J Taylor, Mr N Wright (GEC-Marconi Avionics)

Mr D Hines, Mr J McWha, Ms L Schad (Boeing Commercial Airplane Group, USA)

Mr M deWalt (Federal Aviation Administration, USA)

In preparing this final version of the report, the Study Group was fortunate to be able to take account of extensive comments that we received on the earlier version. In particular we would like to thank the following:

Mr J Baker, Scottish Nuclear

Dr L Beltracchi, USNRC

Mr A W Clarke, OBE, Magnox Electric

Mr R Endsor, Nuclear Electric

Mr J Gallagher, USNRC

Mr K Geary, MoD

Mr D J Hawkes, CAA

Lt Commander P D Jordan RN, MoD

Mr D J Western, Nuclear Electric

Mr B Wichmann, NPL

Mr R J Williams, BNFL

The Study Group expresses its thanks to all those others who have been helpful in their many ways, not least to the ACSNI Secretariat and Nuclear Safety Directorate staff involved.

London
October, 1997

Safety of operational computer systems

1 Introduction

The increasing ubiquity of computer systems is having a dramatic effect upon almost all industries and upon society at large. It is now common-place to see computers entrusted with functions upon which human life can depend: examples include embedded computers in heart pace-makers, anti-lock braking systems in cars, fly-by-wire flight control systems for civil airliners. In the nuclear industry, software-based systems have become widespread, and in recent years have also started to play roles that are safety-critical, in many cases because they are specifically intended to protect a nuclear plant from becoming unsafe. A notable example of such a software-based safety system is the primary protection system (PPS) of the Sizewell B reactor, a system which aroused widespread interest in the UK scientific and engineering communities, as well as in the world at large.

Several times over the past six years, ACSNI has held meetings in which issues concerning the impact of computers upon nuclear safety have been discussed. Whilst these discussions often arose in the context of the Sizewell B PPS, it became clear that the issues were of wider significance for the nuclear industry and for regulators. For example, there was a concern about how future reactors might be licensed in the likely event that they contained safety-critical computer systems: the industry trend is such that manufacturers are likely to be offering only plants which depend upon safety-critical computers. Indeed, there is a trend for ever more safety-critical responsibilities to be taken over by computer systems. Furthermore, the nature of computer systems (hardware and software) is evolving faster than other technologies, and these new developments, and the pace of change itself, give rise to new problems for licensees and regulators.

In March 1995 ACSNI set up this Study Group to investigate these issues and report back to the full Committee. The terms of reference for the Study Group were:

> *'to review the current and potential uses of computer systems in safety-critical applications;*
>
> *to consider the implications for the nuclear industry;*
>
> *in this context, to consider developments in the design and safety assessment of such computer-based systems, including other aspects of control systems;*
>
> *and to advise ACSNI where further research is necessary.'*

The aim of the report that follows is primarily to inform NuSAC (the renamed ACSNI) of the present state of the art in this area and of likely future trends, and to give our views on some desirable developments. Whilst our primary audience is, of course, NuSAC, the report is also addressed to nuclear licensees and regulators. Furthermore, since we believe that issues surrounding the safety of critical computing systems in the nuclear industry, and indeed elsewhere, are of legitimate concern to society at large, we have attempted to make the report accessible to an intelligent lay audience. For a technical reader familiar with the area, this may mean that we occasionally state the obvious: we make no apologies for this. We should also make clear that we have no pretensions for the report to be either a guidebook or a standard.

We recognise that primary responsibility for nuclear safety is vested in the nuclear industry and that in the first place it must decide how to proceed with this important topic, although we would hope that, in the process, due account will be taken of the opinions, conclusions and recommendations expressed within our report.

In the report we use the term ‘safety-critical’ to encompass both:

(i) *safety systems:* computer systems that are part of a nuclear safety system, i.e. systems which respond to a potentially hazardous plant fault by implementing the safety action necessary to prevent the associated radiological consequences, and

(ii) *safety-related systems:* any other computer systems that could through their actions, or lack thereof, have an adverse affect on the safety of a nuclear system (e.g. a control system which maintains operation within pre-defined limits by responding continuously to normal plant perturbations).

The report concentrates in the main on reactor safety systems, but much of it will be of relevance to all types of safety-critical system in the nuclear industry. In general, particularly where significant risk is involved, the NII requires any safety-related computer system to operate under the overriding protection of a separate safety system, albeit not necessarily a computer-based safety system. It therefore regards safety systems as being of higher criticality than safety-related systems, though this is not implied by the above definitions.

The report places a major emphasis upon *safety cases* for software-based systems[3]. Rather informally, a safety case can be thought of as the totality of arguments and evidence that is used to justify a claim that the system is sufficiently safe. Current experience with computer technology in the nuclear industry in Canada, France and the UK suggests that the major difficulties arise in this judgement of adequacy, i.e. in the assessment of the computer systems. Experiences in operation have tended to show that the systems performed satisfactorily, but great difficulties were experienced in demonstrating at an early stage that this would indeed be the case.

Lastly, of course, the computer system is always part of a wider system which includes the physical plant, the human operator and operator supports, and the organisational culture. Proper design of their interactions is an essential component of a complete plant safety case, but we decided that such issues were beyond our scope and we have therefore concentrated mainly upon the computer system itself, and particularly upon *software*.

2 What makes computer-based systems special?

One of the most notable differences between most conventional mechanical and electrical systems, and computer systems, lies in the essentially discontinuous behaviour of discrete logic, and in particular of *software*[4].

---

[3] It does not strictly make sense to talk of a ‘software safety case’, since a safety case can only apply to a system that can directly act, potentially adversely, upon its environment. Nevertheless, much of the discussion in this report will concern the contribution made by software to the safety case of such a wider system.

[4] Much of our discussion makes a distinction between ‘software’ and ‘hardware’, but this can be somewhat misleading: modern hardware based on the use of ‘Very Large Scale Integration’ (VLSI) has much more in common with software than with conventional electrical and mechanical hardware when viewed from a reliability perspective. Perhaps the most important distinction is between those systems that fail largely because of random physical causes, and those that fail because inherent design faults are triggered when appropriate conditions occur. Our main concern is with the latter. Because of the complexity of VLSI hardware, its failures tend to be because of inherent design faults - just like software. Thus in what follows much of what is said about software should be taken as also relating to VLSI.

This shows itself in two main ways. In the first place, when a computer system is tested on a particular input and found to work correctly, one cannot always be certain that it will work on any other input, even one that is ‘close’ to the one tested. In other branches of engineering, on the other hand, one can usually assume continuity. So, for example, if a structure such as a bridge survives a load of 10 tonnes, it is often reasonable to claim that it would survive loads less than this. Because of the discrete, logical nature of software there are usually no simple equivalents of ‘load’ and ‘less than’. This weakens the claims that can be made from software testing, since it does not allow us to extrapolate with certainty from a small set of successful tests to infer that the software will perform correctly elsewhere. Of course, one usually cannot test *all* possible inputs, since the number of these is almost invariably astronomically large.

The second problem arises when software is changed. In conventional engineering it is often possible to justify a claim that a change to a system will be certain to be beneficial because there is usually well established theoretical and empirical evidence that can be used to analyse the impacts on other parts of the system. For example, if a component in the bridge is replaced by one that is stronger but otherwise identical, it is often reasonable to claim that the overall structure will have been strengthened (or at least not made weaker) - i.e. its tendency to failure will have decreased. (Caution is required, of course, since impact analysis may show that such a change may transfer stresses elsewhere and cause another component to fail earlier, thus decreasing the overall reliability of the bridge.) One generally cannot so easily have confidence in the benevolence of a software change, and there are many well-attested examples of software changes whose impacts have not been properly analysed which have had catastrophic effects upon system behaviour. For example, a change made to the software in telephone switches in the US several years ago was regarded as sufficiently ‘small’ as not to require testing: in fact, it contained a fault that brought down the long-distance telephone system of the Eastern seaboard for several hours.

Similar remarks apply even to those software changes that are merely intended to remove particular faults that have been identified in a program. From a reliability point of view, the concept of replacing a failed component by another one that is known to be working (but otherwise identical) is generally familiar, and in these circumstances it is reasonable to assume that the structure will be restored to the state it was in before the component failed. In particular, the repaired structure will be as reliable as it was before it failed. The nearest analogy to this in software is that of reloading a piece of software which has been corrupted - we can assume that the reloaded software will be as reliable (or otherwise) as the original load. Software ‘maintenance’, on the other hand, refers to changes in the software *design* that are either intended to correct a fault, or are in response to a change in the specification because the software does not perform as the user wishes. This is a completely different set of circumstances from the replacement of an item of failed hardware, which merely recovers the original system functionality. The removal of a software fault constitutes a change to the *design* of the system. It is therefore much harder, and sometimes impossible, to be certain that such a change has not introduced a new fault which has made the program less reliable than it was before: there are many recorded examples of ‘small’ changes, supposedly just fixing faults, causing serious reductions in reliability.

These difficulties associated with software changes are not only of concern to the designers of these systems, whose main concern is with the *achievement* of reliability, but also to those with a responsibility for *assessing* software reliability and its impact upon plant safety. Thus every change must be analysed to establish it is correct and that its impact on all aspects of the system is understood. In principle, if not always in practice, the disciplines and tools that software engineering provides for change control (see Section 7) can ensure that the required analysis is performed and the necessary understanding obtained. However, in many cases the only safe course is to treat a program that has been changed as if it were a new program, for which evaluation of reliability or safety must begin afresh.

It is the nature of software that failures of a program can only occur as a result of *design faults*[5]- what are commonly called bugs. Non-software-based systems can also suffer from design faults, of course, and a recent study of failures and accidents in pipework and vessels under the Seveso directive found that 23% of accidents were due to design faults (a further 3% were due to inadequate human factors review).

A particularly interesting class of software design fault is the so-called ‘millennium bug’. Many programs will not be able to handle the transition between the years 1999 and 2000 because they represent the year by using only two digits - they might, for example, treat 2000 as if it were 1900 with potentially catastrophic consequences. What is striking about the problem is its widespread nature: since representation of time is needed in most programs, a large proportion of all programs is likely to be affected. The costs of checking to determine whether software is susceptible to this problem, and making appropriate changes, is reported to be billions of dollars world-wide.

Design faults pose particular difficulties to those responsible for building and assessing safety-critical computer systems. Although good design practices might be expected to minimise the number of faults that find their way into a program, there are no general procedures that allow us to avoid them completely. Indeed, since software systems often are designed to provide much more complex functionality than the conventionally engineered systems they replace, they are more prone to design faults. *We must therefore assume that any program that is of a reasonable size will contain bugs*[6].

Given that any program will contain bugs, a major concern is the unpredictability of the outcome when one of these is triggered. Traditional hardware systems embody much of their functionality within the components which comprise the systems. The failure modes of these components are relatively few and can be dependably predicted - hence the system impacts can be analysed *a priori*. It is also relatively easy to devise tests which can prove the satisfactory functioning of these components within a system prior to permitting its operational usage. When equivalent functionality is invested in software, the possible failure modes become extremely difficult to bound, and hence to test. Similarly, complete analysis of the possible failure modes in terms of their expected effects upon the overall system is not practicable.

Finding and removing design faults in software is difficult for reasons that are given above. Error detection and recovery provisions at some higher level of the system can help to mitigate the effects of residual design faults, but achieving a level of fault tolerance that will entirely mask the effects of such faults is much more difficult. Clearly, since any design fault will be reproduced in identical copies of a program, simple notions of redundancy based on component replication, such as those used to protect against random failures of hardware systems, cannot be applied. Software *diversity* - where two or more versions of a program are developed ‘independently’ and their output adjudicated at execution time - has been used with some success in several industries to achieve a useful degree of fault masking. However, there is considerable evidence from several experiments and from recent theoretical work that the benefits fall far short of what could be expected had statistical independence of failures of the versions been achieved. Since independence cannot be claimed, it is necessary for the assessor to measure the degree of

---

[5] We shall be eclectic in our use of the term ‘design fault’. It will be taken to include ‘trivial’ coding errors as well as more subtle misunderstandings and omissions. As we have already indicated, software has the unfortunate property that faults that are ‘trivial’ in their logical nature can be far from trivial in their consequences and in their ease of detection.

[6] A conservative view would be that this is equally true of hardware devices that are of sufficient complexity. Modern microprocessors, for example, must be assumed to contain (and all too often are proved to contain) design faults.

dependence that is present in order to evaluate the reliability of the fault-tolerant diverse system[7]. There are no agreed mechanisms for assessing the degree of dependence of two pieces of software, other than testing them back-to-back and looking for coincident failures: if the aim is to demonstrate the achievement of a specific reliability, this will require as many tests as would be needed for a single version.

Finally, we must mention that software-based systems pose some of their most challenging intellectual problems to their designers in coping with tight timing constraints and concurrent activities of a number of inter-connected computers. Not only is it very hard to solve these problems, but convincing an independent assessor that they have been solved, and thus will not be a source of failure in operation, can itself be an immense technical challenge.

3 What advantages do computers bring?

The major advantage of computers over simple hard-wired hardware lies in their *general purpose* nature, derived from their programmability. This allows much of the specific functionality in a particular application to be implemented in software and thus avoid some of the unreliability due to physical component failures that would be incurred if it were implemented purely in electro-mechanical hardware.

In the most extreme cases, software-based systems have been implemented successfully that would have been inconceivable using older hardware technology alone: examples range from fly-by-wire aircraft flight control systems, through sophisticated automobile engine controllers, to the now ubiquitous office systems such as word-processors and spread-sheets.

Similarly, computers allow systems to perform in ways that would not be possible if they were under only human control. So-called ‘smart’ systems (i.e. systems under computer control) can have reaction times which are far shorter than human response times, enabling control of dynamically unstable physical systems. Thus we now have aerodynamically unstable military aircraft whose continuing safe flight is completely dependent upon computers, and chemical process control that operates at extremes of pressure and temperature that would not be possible if solely using human operators.

The nuclear industry has perhaps been more conservative than some others in taking up the new technology in safety-critical contexts, but experience reports are becoming available, and these are generally encouraging. CANDU was among the first reactors, in the early 1980s, to use programmable digital computers for the trip decision logic. In 1991, the operating experience over a total of 288 computer-years of operations was reported [Ichiyen & Joannou 1991]. Every computer failure had been a safe failure, in contrast to the conventional parts of the system where about one in four failures were potentially unsafe (failures were called *safe failures* if they increased the probability of a spurious trip, while failures that decreased the chance of a correct trip were considered *unsafe*). This was largely attributed to a design employing features such as systematic processor and memory self-checks, continuous testing, watchdogs, etc., which contribute to the early detection of potentially unsafe failures and to their conversion into safe ones. Interestingly, and remarkably, although safe failures tended to increase the chance of a spurious trip, there had not actually been any spurious reactor trips attributed to computer-related failures during those years.

[7] In the light of the new understanding of diversity that has come from these *software* studies, it is interesting to speculate on how justified are the claims for independence that are used in assessments of the safety of *non-software* systems.

Success has also been reported from other industries, notably telecommunications [Kuhn 1997]. The US Public Telephone Switched Network (PTSN) is the US portion of what is probably the world's largest distributed computer system in operation. The PTSN contains thousands of switches, and software for a single switch may contain several million lines of code. Since 1992, US telephone companies have been required to report to the US Federal Communications Commission all outages that affect more than 30,000 customers. Kuhn reports on these outages over two years to determine the principal causes of PTSN failures. The measures of failure effect used are the number of outages by category of failure, and the downtime by category measured in 'customer minutes' - the number of customers affected by an outage multiplied by the outage duration in minutes.

The PTSN averaged an availability better than 99.999% in the time period studied. Major sources of failure were human errors (28%) - half by telephone company personnel (14%) and half by others (14%) - acts of nature (18%), and overloads (44%). An unexpected finding, given the complexity of the network and its intensive use of software, is that software errors caused less downtime (2%) than any other cause except vandalism (1%), while hardware failures were responsible for 7%. If we exclude from the outage and downtime percentages those caused by failures like overloads, acts of nature, etc., which may be rather specific to a telephone network, we find that software accounts for 24% of the remaining outages and 9% of customer minutes downtime, while hardware accounts for 32% of these outages, and 29% of downtime.

Kuhn thinks that several factors may explain why software makes such a small contribution to unreliability. An important reason is the use of extensive software mechanisms for error detection and recovery. Second, by its very nature, the telephone network is highly distributed: failures are likely to be localised, and re-routing algorithms can divert traffic to avoid failed nodes[8]. Finally, the major telephone switch manufacturers are among the world leaders in computing and software technology. Nevertheless, there are concerns, caused by recent major outages, about the ability of the PTSN to maintain recent reliability levels as even more complex systems are designed and built.

Operational data from the US PTSN, and other experiences, provide empirical confirmation that making extensive use of the powerful *fault detection mechanisms* of computers can substantially increase both safety and availability. In addition, periodic routine verification tasks, e.g. to test the equipment or to calibrate sensing devices, can be performed by software at much higher periodicity, with more discrimination and with less risk of human error or negligence than is possible with manual procedures. These new technologies also contribute to safety and reliability (and bring economic benefits) by reducing system down times.

For robustness and reliability, and to allow for graceful degradation, advanced nuclear safety system designs tend to take advantage of distributed architectures not only at equipment level but also at control level (Sizewell B, Temelin [Temelin 1993], System 80+ [NRC 1994]). The co-ordination between autonomous subsystems required by these architectures needs extremely reliable data communications; this can only be provided by digital communications and programmable systems, through sophisticated protocols, and communication error detection and correction mechanisms. The same recent advances in coding theory and communication technology which are driving all types of audio and video communications towards digital technology should thus bring benefit to these nuclear systems.

Software implementations allow more sophisticated on-line monitoring in the form of more elaborate calculations or procedures to make inferences from the observed plant parameter

---

8 This re-routing, however, does not come free. It requires the maintenance - by software - of a complex consistent data-base across the whole network.

values. Not so long ago (before the Three Mile Island accident), in many control rooms the core saturation temperature, the sub-cooling factor and other operational parameters were computed by hand from tables. More and more now, computer systems may continuously monitor all the plant safety-critical functions. In normal operations, they can allow the system under control to produce more power and to use more efficient operational regimes than hitherto, without reduction in safety.

Computers are probably also the only solution to the ‘Christmas tree’ effect of nuclear plant control rooms. Sets of alarms that are capable of signalling every single detected abnormal indication - especially those caused by malfunctioning of the control and instrumentation systems themselves - are too large in number for human operators to cope with directly. They must be aggregated into a limited number of high-level alarm types that can be made visible and/or audible on the control room panels. To detect and locate the original cause of the incident, operators must then often look for LEDs in various racks. In contrast, computer-based displays can give all the necessary information to help identify and locate the cause of every single plant abnormal indication. They can also discriminate, prioritise, filter and mask irrelevant alarms for the different modes of operation and testing of the plant (e.g. Chooz-B1 [EDF 1994]). In addition they can provide sophisticated diagnostic aids and help the operators in troubleshooting, fixing the problems, and restoring safe operation. Many support systems of this kind, using relational or deductive database technology, are currently under development in research institutes (e.g. the OECD Nuclear Energy Agency’s Halden Reactor Project [Dahll & Applequist 1993]) or industry. However, it must be mentioned that effective operator support systems are not just a matter of developing the technology. If an operator support system is not compatible with the operators’ thinking, or is wrong or incomplete, it can cause problems for the operator rather than being an aid.

The use of computer technology, *correctly implemented*, offers the means of introducing specific safety-enhancing features which formerly it was not reasonably practicable, or even possible, to achieve by the traditional engineering means. For example, in addition to those mentioned earlier, continuous on-line calculations can be performed of key characteristics of plant performance (the so-called ‘calculated parameters’) which provide both faster and additional diverse means of safety monitoring for use by automatic protection systems.

Computer systems allow the number of cables to be reduced (and the commensurate fire risk), and hence the numbers of potentially unreliable connections/terminations, by the use of multiplexing, whilst increasing the integrity of the equivalent highways by use of sophisticated noise filtration and on-line checking calculations.

There are, in addition, a number of other well-known advantages offered by digital programmable technology which are of more general interest. When they are well designed, software-based systems offer more flexibility to accommodate system modifications and parameter tuning, both during the design, and later during operations, although this advantage can be constrained by the rigour of the change process.

Software, by itself, does not age or wear out, and is not affected by the environment (heat, electromagnetic interference, earthquakes, etc.). In principle, once correct, it can remain correct forever without maintenance. Thus, the more functionality one can put into the software, the more insensitivity to those external perturbations one can achieve in the integrated system.

Under current pricing, software implementations are usually cheaper than hardware implementations of the same functionality. However, if the software is badly structured, the cost of testing, debugging, inspection, justification and maintenance may remove this advantage. UK experience shows that the licensing process associated with the justification of the design must be expected to be a very significant cost factor.

Finally, for mathematical and technological reasons, more accuracy and (numerical) stability can often be obtained from the digital technology than from analogue computing and control devices.

4 Safety, reliability and system issues

When discussing the use of computers and software for safety-critical functions, it is useful to clarify the distinction between reliability and safety. We shall typically be interested in the safety of an overall system, i.e. a nuclear plant, which is comprised of interacting lower level systems, such as instrumentation systems. These lower level systems are themselves comprised of lower level systems, and so on. This report addresses safety-critical computer systems, such as might form a part of the overall safety system of a nuclear plant. Much of our concern, in turn, will centre upon the role of software in such a computer system.

Safety is the attribute of a system - e.g. a nuclear power plant - to be free from the occurrence of accidents, i.e. from the undesired events that lead to catastrophic consequences such as health and environmental effects of radiation and radioactive contamination. Safety is achieved through the use of *reliable* structures, components, systems and procedures. *Reliability* is the probability that a system or component will perform its intended function for a prescribed time and under stipulated environmental conditions.

Reliability may thus be determined by the probability of failure per demand, whilst safety is also determined by the possible consequences of these failures. In a reactor safety system, for instance, the primary functionality concerns the requirement to shut the reactor down safely when needed, and keep it in a safe state for a specified period of time. If the software in a safety system is unreliable, i.e. if there is a too-high probability of its not carrying out this shut-down function correctly when demanded, then there will be an unacceptable effect upon the safety of the wider system. If a computer-controlled control panel prioritises or filters alarm signals incorrectly, there can also be an adverse effect on safety.

The achievement of the required reliability by the hardware and software alone, however, is not enough to guarantee overall plant safety. If the specification of the safety-critical system is inadequate then the overall nuclear system may be unsafe even though the hardware and software implementation of the safety-system is completely reliable (with respect to its specification). Moreover, in a more general context, the events leading to an accident are almost never limited to a computer failure, but are a complex combination of equipment failures, faulty maintenance, human actions and design errors. Some accidents even result from a sequence of events, none of which may involve a component failure. Each component may reliably work as specified, but together they may create a hazardous system state.

An example that reliability alone is not enough to guarantee safety is provided by domestic fuses [Leveson 1995]. Their failure frequency is estimated at $10-^{6}$ or $10-^{7}$ per year. Fuses, however, can be wrongly calibrated or unthinkingly replaced by copper wire. The frequency of these errors has been estimated at 10^{-3} per year. Further improvements of the reliability of individual fuses would therefore not overall make their use safer.

In nuclear power plants as elsewhere, there are also sources of unreliability which would not be regarded as contributing to the overall plant risk - for example, components that, depending on their failure modes, can affect the plant safety in different ways. For instance, the protection relays (nowadays often microprocessor-based), which protect the power supplies of safety injection mechanisms against transient peaks of voltage or current, may not necessarily be considered as impacting safety if they fail to trip on demand. Depending on their role at the plant level, their failure to trip may leave the plant in a safe state. Likewise, the reliability contribution required from a diverse system may be much lower than that of the primary safety system or line of defence. An example is given by the ‘off the shelf’ computer-based radiation detectors that are sometimes used as an ultimate detection mechanism for containment isolation in the event of the protection system failing to detect a loss of coolant accident (LOCA).

Thus, it is important when discussing reliability and safety to have in mind both the system (or sub-system) of interest *and* its environment (often a wider system). In fact in the nuclear engineering community, it is normal to reserve the word *safety* for use as a property of an overall nuclear system, and to refer only to the *reliability* of any computer system involved as well as to the adequacy of its set of requirements that have been identified and specified. With this usage - which we adopt in this report - unreliability is associated with undesired departures from the specified behaviour. In contrast, breaches of safety are associated not just with inadequate reliability, but also with unexpected and undesired behaviours that had not been specified or had been inadequately specified.

A natural consequence of these considerations is that solutions to safety issues must start with nuclear system, rather than software, engineering. Clearly, the identification of the possible events that are to be regarded as important to safety is a key part of the determination of the safety requirements of, e.g. a nuclear protection system. This is a complex task which belongs to the world of nuclear safety engineers. They have to anticipate *all* possible failure modes of the protection system, define its functionality, and non-functional behaviour (e.g. internal monitoring), and do this *whatever* technology will be used: hardware, software, hydraulics, human operator, etc. The safety requirements are thus translated into functional and non-functional requirements of a protection system. It is then decided what technology to use, and hardware, software and/or human performance specifications are written. What is required from the implementation is demonstrable satisfaction of its specified reliability.

In other words, the impact of safety on design should ideally be confined to the functional and non functional *specifications* of the system. Software and hardware should then simply be required to be a reliable implementation of these specifications - demonstrably correct and tolerant of hardware random faults. In these circumstances, the somewhat ambiguous notion of ‘software hazard’ used by certain authors could be dispensed with.

This is not to say that safety can be ignored within the software implementation team (or any other implementation team). The initial version of the specification will not be perfect - it may have omissions, inconsistencies and requirements which are at the very least non-optimal. One of the important tasks of the implementation team is to refine the specification and in so doing they may have to feed back changes to the system team to be included in new updated versions of the specifications. In addition, where there are design options for the implementation team, the relative effect on safety should form part of the decision making process. So, concern for safety is indispensable within the software team, but the matters of concern and the techniques applicable are different from those of the system engineers who specify the safety requirements.

Thus concern for safety is a crucial issue and is related to the role of the individual engineers in large projects where software is a component only. At stake is the production of system specifications that are both complete and understandable by all parties involved in the design. The challenge is to develop methods for specifying in the system requirements *everything* concerning safety in a way which is understandable to computer hardware and software designers.

This necessity is confirmed by reported experience of incidents involving software. An example is the Therac-25, a computer-controlled radiation therapy machine, which massively overdosed six people who died or were severely injured. The machine relied on software for safety operations. These accidents could probably have been avoided if software designers had been given specifications that included a correct and complete description of the possible failure modes of the machine, and of the possible hazardous misuses of the operator interface [Leveson 1995].

Whilst there is a certain element of judgement involved in the identification of possible events that are important to safety, it is usual in those industries in which safety-critical systems are common to have a systematic safety analysis process which includes procedures to identify these events. In some nuclear safety systems, for example, the existence of a safe state makes the problem simpler than for other systems: reliable delivery of appropriate responses to the demands placed upon the safety system is the major safety issue, and the number and nature of these demands can be assumed to be well-understood. In other applications, however, particularly those involving systems that do not have a safe state (e.g. an operator assistance system, or an aircraft flight control system) and where safety issues are mainly associated with ensuring that unexpected and undesirable events do *not* happen, their prior identification and analysis can be much harder, so that the completeness and correctness of the safety analysis procedure is itself a major concern and thus should itself be taken into account in the safety evaluation.

The nuclear industry has extensive experience of safety cases, i.e. the formal arguments and evidence in support of claims that systems are sufficiently safe to be allowed to operate, which are required as part of the regulatory process (see Appendix A). However, it has to be said that in the case of software-based systems, there is less experience of safety cases than for older technologies. In fact, such safety cases are not yet common in all industries, nor are they well-supported by standards. As we shall see in Section 10, this suggests that safety cases involving software-based systems should involve some elements of *diversity* - both in the kinds of evidence used (what we shall call the 'legs' of the argument), and in the personnel involved (independent assessment).

It should also be emphasised that safety, like reliability, is not an absolute. We can say that a system is sufficiently safe, but not that it is completely safe. Much of our discussion will centre on the evidence and arguments we need to deploy in the safety case to make claims for this *sufficiency*. Ideally, such claims will be expressed numerically; thus we might require that the probability of failure upon demand of a safety system - depending on the consequences of the failure - be smaller than some number emanating from the wider plant safety case, which provides the context for the requirements specification of the safety system.

Finally, this distinction between levels at which safety and reliability requirements apply helps to clarify some implications of the ALARP principle which is discussed in section 9, especially if ALARP requirements must apply to the design of systems of limited demonstrable reliability that have to be used within high risk environments.

5 Hardware, humans and computers

There is a school of thought that says 'keep it simple and hard-wired and we do not need to worry about design faults'. This view was held until recently in parts of the UK nuclear industry, and is certainly not one to be dismissed out of hand. Similarly, it is often argued that, *in extremis*, the human operator should always be allowed to have the final say. These views are, of course, simply examples of design choices: they are particular ways of allocating responsibilities among humans, electro-mechanical hardware and computers. However, in order to make such choices rationally, it is important to know what are the particular strengths and weaknesses of these different system 'components'.

The decisions that are taken at the very highest level concerning the allocation of responsibilities between people and machines are some of the most crucial in determining the final reliability and safety of a system. In recent years the capability of automatic systems has increased greatly in many industries as a consequence of the introduction of software-based computer systems, resulting in a tendency for greater responsibility being given to these systems and correspondingly less to humans and to simple 'hard-wired' (non-programmable) hardware. Examples include highly automated civil aircraft, where computer systems can overrule pilot commands, and railway signalling and control systems in which safety interlocks are based on software rather than mechanical hardware (see Appendices D and E). For Sizewell B, the automatic safety system - which in this case comprises a software-based primary and a hard-wired secondary - has prime responsibility for the first 30 minutes after the detection of the onset of a potentially hazardous situation, and its actions to shut down the reactor safely cannot be overridden by the human operators during this time.

There are three main direct[9] sources of failures in systems such as these. They can fail as a result of random physical component failures. They can fail because of logical faults (design and implementation faults) being triggered in either hardware or software. They can fail because of human errors during operation. The safety case for a plant, or top level system, will need to allocate achievable reliability levels to all system components whatever their sources of potential failures, and an important part of system validation will be a demonstration that each such target has been achieved. Our main concern here is the target for the computer system and in particular its software - how this target can be achieved, and how it can be assured.

In general, there are human contributions to three key themes of this report :

- in the development and assessment of safety cases: in areas such as development and application of techniques; expert subjective estimates of risk, quality of evidence, cost-effectiveness, confidence, compliance, etc.; negotiation of agreement;

- in the system specification and design of software: these may be strongly affected by limitations to human comprehension and communication in the handling of complexity, and minimising human errors resulting from these limitations is important, through, e.g., training, techniques of software engineering and quality assurance;

- human functions in the on-line operation of the system, fault management, and plant maintenance.

Other sections of this report make many points about the first two of these areas, although there is still considerable scope for more explicit understanding of the human cognitive and social processes involved. What follows in this section is primarily concerned with the issue of allocation of operational functions.

5.1 Allocation of function to hardware and computers

Random hardware failures now mainly affect mechanical and electrical components. For micro-electronic devices on the other hand, such as computer chips, this kind of random failure is quite rare, and thus contributes comparatively little to their unreliability. Moreover, there are well-understood techniques, often involving redundancy, for mitigating the effects of random hardware failures upon the wider system; for example, in aircraft auto-land systems there are typically triple redundant computer channels, in nuclear plants there are now four such channels.

The recent growth in complexity of micro-electronic components, however, means that the potential for failures due to design faults is becoming greater. Software failures, of course, are *always* the result of design faults. The increasing dependence upon, and the increasing complexity of, both software and micro-electronics point to a general increase in the importance of unreliability arising from design faults.

---

9 Of course, the direct causes of failure may hide some important indirect causes. Poor management can be a source of failures, for example, by inadequate training of operators who then do not know how to react to warnings, or by not having in place an adequate means of reporting system maintenance changes to operators. In such cases it could be misleading to lay responsibility for system failure directly at the door of the operator. Equally, it has to be said that analyses of accidents - after the event - suggest that causes are usually multiple rather than single.

Most conventional reliability theory and methods concern only random physical failures of hardware. Whilst it could not be said that physical hardware reliability is completely understood - there are still unresolved issues concerning common cause failures, for example - it is true to say that this understanding is greater than it is for design faults in computer hardware and software and for human sources of failure. This greater understanding brings with it a possible hidden advantage to which we alluded at the beginning of this section: the impact of excessive complexity upon hardware reliability is so well understood by designers of these systems that there is a tendency to keep them as simple as possible, and this in turn lessens the likelihood of their containing *design* faults. Making it hard-wired can mean making it *simpler* - and thus safer and more reliable.

On the other hand, there are some functions that are irreducibly difficult, and it is here that designers increasingly only have a choice between human and computer control.

5.2 Allocation of function to computers and human operators

It is worth pointing out at this stage that design faults themselves - including those in computer software - also arise from human failures. If an aircraft crashes because the software in its flight control computers fails, this is a result of a human failure during the control system design process. Designers have to *imagine* all the circumstances in which the aircraft might find itself, and ensure that the computer system can respond to each and every one of these appropriately. They have the advantage of doing this in conditions which are less stressful, and less constrained by the imperatives of time, than the pilot. Pilots, on the other hand, have the advantage of *knowing* at some level of detail the precise circumstances aircraft are in. On the other hand they are likely to be suffering from the stress of needing to make immediate decisions under time constraints, and their reaction times are likely to be slower than those of an automatic system. A good design will allocate tasks to machines and human operators according to their respective strengths and weaknesses.

In the nuclear industry, the role of computers has advanced from simple tasks such as monitoring and scheduling in the early 1960s, to sophisticated information processing and direct control of the reactor in present-day plant. Although automatic safety systems response has always been a feature of nuclear plant protection, nevertheless in the early days the human operator was responsible for general plant monitoring, anticipating problems, and for many aspects of plant fault management. However, with the evolution of modern plants, the human operator has become progressively less in a position to perform these primary roles dependably without automated assistance. This arises for several reasons:

- the greater complexities of modern plants, and their more sophisticated control regimes;
- automatic control disguises the first symptoms of plant faults, so it is more difficult for the human operator to notice that problems are developing;
- if an operator is not directly involved in plant operation, and therefore keeping up to date with its state, it takes time to get the information required and to work out what is happening and what to do about it when asked to take over.

As mentioned earlier, typically as at Sizewell B the safety system (which is computer-based) has prime responsibility for the first 30 minutes, during which the human operator can gain sufficient understanding of what is happening so that he/she can be effective. There is a requirement to demonstrate that the computer is sufficiently dependable to meet the safety requirement of the overall plant without the intervention of the human operator during those 30 minutes. (The operator can intervene during this time in ways that are known to be ‘benevolent’ - for example to switch in alternative diverse cooling systems.) Our main concern in this report is with computer systems that carry this sort of ultimate responsibility, though our findings relate to any kind of safety-critical system.

Of course, hardware has long been used to automate human action. What is new is that software is now used to automate human *cognition* - it is entrusted with the analysis and decision-making that was previously carried out by the human operator. When we apportion responsibility between the computer and the human operator in the high level design, we must make a trade-off between the different potentialities for failure in the design process and in the operator: decreasing the amount of human responsibility in the operation of the plant increases the amount of human responsibility in the design of the plant.

The aim here is to allocate the different decisions and actions between humans and computers ‘optimally’: each should, as far as possible, be assigned to whichever is the least likely to fail in its execution. Simply put, computer systems are good at performing well-defined tasks, accurately, quickly and continuously. They should thus be given the following sorts of tasks:

- tasks which a human would find boring, such as monitoring equipment;
- repetitive tasks;
- tasks which consist of lots of simple elements, such as long calculations, where a human might be careless;
- tasks which require superhuman response times.

Some tasks which used to be beyond the capabilities of computers can now, in principle, be tackled by systems using novel technologies such as expert systems or neural networks. The claimed capabilities of these systems include: limited ability for learning from experience and optimising behaviour, pattern recognition, and forming conclusions from vague data or data from disparate sources. However, our considered view is that these novel technologies could not at present be considered sufficiently dependable to be used for the most safety-critical applications. For example, the very ability of these systems to learn poses particular difficulties in assessing and predicting their failure behaviour: their reliability must be assumed to change as they evolve, and past good behaviour could not be regarded as a guarantee of future good behaviour.

Human beings are still required to deal with unanticipated or vague situations because it is not known how to program for situations which cannot be anticipated: computers are unable to be creative and hence generally cannot invent novel solutions when confronted with something unexpected.

There are four categories of allocation:

- the task is always done by the computer;
- the task is usually done by the computer, but the human operator is expected to take over in unusual circumstances;
- the task is always done by the human operator;
- the task is usually done by the human operator, but the computer is expected to take over in unusual circumstances (this is very difficult to design effectively).

As things now stand, any increase in dependence upon computers may make the tasks remaining for the human operator more difficult, and so support for the operator needs to be considered more carefully. Human beings can only maintain the skills and commitment needed to carry out tasks by actually doing those tasks, and increased computer operation may give the operator less opportunity for this. So any increase in computer responsibilities means that more care is needed in supporting operators by interface and job design, and by training, to ensure that they can carry out their remaining responsibilities effectively.

The main focus of this report is, of course, upon the role of the computer, and particularly its software, in such a safety-critical system. A detailed discussion on attributing the causes and predicting the likely rates of human error is beyond our scope: there is a large literature on these topics, and interested readers should refer to the reports of the earlier ACSNI Study Group on Human Factors [ACSNI 1990, ACSNI 1991, ACSNI 1993]. It is worth noting here, however, that most of the evidence available about human error is relevant only to manufacturing and operating tasks: in particular, it does not address the errors that humans make during design processes, such as the design of complex computer software.

6 What problems do computers bring?

For various reasons - including the effectiveness of redundancy as a protection against random failures - the physical hardware of computers is now extremely reliable, so much so that the major worry of systems designers using computers is not that they will fail randomly like mechanical devices, but that they might contain design flaws which will be triggered in all copies when appropriate external conditions are encountered. Most system functionality and hence complexity is normally, and quite appropriately, ‘allocated’ to the software, so such residual design faults are more often found in software than in digital electronics.

It is in fact the very advantages of software, stemming from its general purpose nature, that often bring disadvantages from the point of view of achieving sufficient reliability, and of demonstrating its achievement so that the contribution of the computer system to the safety case can be assured. Rather informally, these problems stem from the *difficulty* and *novelty* of the problems that are tackled, the *complexity* of the resulting solutions, as well as the (previously discussed) inherently *discrete* behaviour of digital systems.

There is a tendency for system designers to take on tasks that are intrinsically *difficult* when building software-based systems. The fact of a system being based on software frees the designer from some of the constraints of a purely hardware system, and allows the implementation of sometimes excessive extra functionality. The more difficult the task, the more likely that mistakes will be made, resulting in the introduction of faults which cause system failure when triggered by appropriate input conditions.

The difficulty of the tasks that a software-based system has to perform is often accompanied by a degree of *novelty* in the tasks which is greater than in other branches of engineering. Whereas in the past computer-based systems were often used to automate the solution of problems for which satisfactory manual solutions already existed, it is becoming increasingly common to seek computerised solutions for previously unresolved problems - often ones that would have been regarded as impracticable using other technology. This poses particular difficulties for systems with high reliability requirements, since it means that we can learn little from experience of previous systems. Other branches of engineering, by contrast, tend to have a more continuous evolution in successive designs. Whilst changing from a non-digital electronic control system to a software-based system might be regarded as a natural step, it is perhaps better to regard it as a step-change in technology. Equivalent step changes in other branches of engineering are known to be risky, for example the attempt to introduce new materials for turbine blades that led to insolvency and nationalisation for Rolls Royce in 1971. In the nuclear industry, the present generation of computer-based safety systems, such as the Sizewell B PPS, undoubtedly contain some functionality that is novel when compared with the older hard-wired systems.

Software system technology is not always concerned with step changes, however, and adding functionality to an existing software system can often be accomplished successfully as a natural, progressive evolution.

Most importantly, these trends to new and increased functionality in computer-based systems are almost unavoidably accompanied by increased *complexity* in the internal structure and external interfaces - most particularly in the software. For our purposes here, sheer size (e.g. measured in terms of the number of lines of code) will often give a good indication of the internal complexity, and the size of the user manual the complexity of its interfaces[10].

Of course, some specific design features that add to the line count and number of functional interfaces may be deliberately introduced in the expectation that they will improve the safety of the system; an example is the use of functional diversity, with separate processing of the different physical parameters or diagnostics. However, such features also add to the concurrency and synchronisation problems, i.e. to those aspects of the design which are most difficult to validate.

Finally, as we have seen, the inherent discreteness of behaviour of digital systems makes it particularly difficult to gain assurance of their reliability: in contrast to conventional mechanical and electrical systems, it is almost invariably impossible to extrapolate from evidence of failure-free operation in one context in order to claim that a system will perform acceptably in another context.

Before discussing the problems of novelty and complexity in a little more detail, it is worth observing that software differs from physical hardware and from humans in that it admits the *possibility* of a complete guarantee that there will be no failures during operation. Since software only fails when design faults are triggered, it follows that their absence means it cannot fail. The same cannot be said about random failures of physical hardware (although we may be able to limit the effect of these) or of human failures. This observation has led some to argue that our goal in designing and building safety-critical software (and VLSI) should be complete perfection. Why seek to measure the impact of design faults upon reliability and safety when we can demonstrate their absence?

6.1 The difficulty of achieving logical perfection: the role of formal methods

Engineers from older disciplines sometimes ask why software engineers cannot simply ensure that their programs are completely correct. In fact there do exist formal mathematical techniques that allow exact correspondence between a specification for a program, and the program itself, to be asserted with certainty (assuming the proof techniques are themselves valid and applied correctly). Following such a proof, we might be tempted to claim that the software was completely fault-free: we would certainly have a guarantee that the software would do what the specification says it should do - no more, no less.

However, the specification involved in the proof here has to be a formal object itself, which we *know* to be a complete and correct embodiment of the high-level ‘engineering requirements’ - what we *really* want the software to do. In fact, experience suggests that a high proportion of serious design faults arise as a result of misunderstandings of these more informal system requirements. Such faults become embedded in the formal specification, which is then imperfect, and thus so will be any program written from that specification.

For these reasons, leave alone any doubts as to the complete validity of the proof process, it is very rarely possible to assert credibly that software is completely reliable and that failures are impossible; indeed, we believe that such claims for complete perfection cannot be sustained even for programs of only relatively modest complexity.

10 Formal mathematical measures of complexity have been proposed by the software engineering research community, but with mixed success: this is still an active research area. These measures do not capture what we want here, namely the informal notion that complexity impedes understanding and comprehension.

This is not to say that formal proof of this kind has no value. On the contrary, it would clearly be useful to know that a program is completely free of implementation faults (those that arise from the activities involved in turning the specification into the executable program); but it does mean that we are still left with the task of evaluating the impact upon safety of possible faults in the specification. Formal methods can even help here, for example by checking for consistency of a specification.

Assuming for the moment that a formal specification can be written which accurately and completely captures the engineering requirements, the feasibility of proving the functionality of the source code depends on the style and structure adopted during design. Formal verification of the functionality of software designed from the outset with formal verification in mind is becoming feasible, even for quite large programs. The cost is very high, but not excessive compared with, for example, the cost of the sort of testing normally expected for safety-critical systems. The problem is that most real-time software has structures that are inherently difficult to verify, and functionality that depends on non-functional properties such as scheduling and memory usage. Whilst these can be modelled formally, they tend to be difficult to analyse. Thus although it is possible to perform full formal verification in some circumstances, in many projects it is impractical to do so.

Of course, it could be argued that complete correctness, although desirable, is not necessary, even for safety-critical software. Some of the failure modes of even safety-critical systems will be ones with relatively benign consequences, and others may be masked by built-in logical redundancy, so that it would be sufficient to know that there were no faults present that could cause (suitably defined) *catastrophic* failures. Unfortunately, complexity can make it extremely hard for even these claims to be substantiated. For example, if the software performs extensive non-critical functions, as well as safety-critical ones, the presence of the former can compromise the claims made about the latter. Essentially the only practicable way to make claims for perfection with respect to safety-critical faults is to isolate these as much as possible from other functions, so that they can be reasoned about independently of most other functionality, and to ensure that the degree of independence is sufficient for the results of this reasoning to be trustworthy.

6.2 Novelty

In all branches of engineering, the design process is a mixture of *novelty* and *legacy*. A new system will contain aspects of design novelty when compared with earlier systems, but there will also be aspects of the design that are carried over from earlier systems. It is from the novel aspects of a design that the 'added value' arises, but there is an accompanying risk that new design faults will also be introduced. Extensive re-use of tried-and-tested system components, on the other hand, may reduce the risk of introducing new faults, but at the price of placing constraints upon the designer in his provision of required system functionality.

Maintaining a balance between the conflicting demands of system functionality and reliability will always be an important role for designers, but the introduction of computer-based systems has made the problem more acute. The relative ease with which sophisticated tasks can be implemented using software has trapped many projects into undertaking designs which are far more novel and complex than is perhaps wise. Indeed, this has all too often resulted in designs which are not merely unreliable in operation, but so complex and imperfectly understood that their development becomes unmanageable and they are abandoned before becoming operational.

Of course, designers usually do not introduce novelty for its own sake. Even in those circumstances where the new system performs roughly the same function as its predecessors, the novelty is often there because of genuine additional requirements, over and above those required from these previous implementations. And in those cases where there is a near complete novelty in the functions that the system performs, in the sense that there are no ancestor systems, even ones of less extensive functionality, it is often claimed that the intention is precisely to use the extensive functionality to *increase* safety. Examples of this are widespread, ranging from recent computer-based aircraft control systems which provide protection against inadvertent stall and airframe over-stress (see Appendix D), and computerised automobile braking and steering systems, to more general information systems in fields ranging from banking to medicine.

Just as novelty is not always a vice, so legacy is not always a virtue. The use of existing design can sometimes make the job of a designer more difficult. For example, the software in the Sizewell PPS was made more complex partly because of the decision to have a highly configurable design that could be re-used in future plants. This configurability made the static analysis (and also understandability) of the software more difficult. Similarly, when a digital system has replaced an electronic or mechanical device, there may be a carry-over of spurious requirements which were originally intended to cope with the foibles of these earlier (non-computer) implementations: examples include offsets to avoid transitions through zero and the use of analogue rather than true digital control algorithms.

Even in those cases where reuse of a tried-and-tested component seems eminently sensible, problems can arise because of subtle differences between the new application and the ones upon which the confidence in the component were based. For example, the failure of the first launch of the Ariane-5 rocket in 1996 was initiated by the software of an inertial guidance system carried over from Ariane-4. The software failed when the inertial platform was subjected to higher accelerations than it had experienced previously. Here the specification for the software changed, and a program that might have been correct for the earlier Ariane-4 specification was not correct for the later Ariane-5 one. (It is reported that a recommended programme of testing under conditions similar to those of Ariane-5 was vetoed on grounds of cost.)

There are some general rules of thumb for the designers of safety-critical systems emerging from this. Firstly, novelty that only provides non-safety functionality should be viewed with suspicion since it is likely to detract from the achievement or assessment of safety functionality. Secondly, re-use of tried and trusted designs is desirable if it allows greater confidence to be placed in the design of which they will be a part, but a relentless re-use of old designs - e.g. for purely economic reasons - can have a deleterious impact upon safety and reliability. It is a courageous project manager who regards old designs as merely prototypes and authorises a complete rewrite to remove the undesirable element of legacy, but sometimes this proves to be the more reliable and cheaper solution.

6.3 Complexity

The principal problems of designing and validating safety-critical computer systems stem from their discrete nature (as we have seen earlier) and their internal (and perhaps external) *complexity*. Such ‘discrete complexity’ can also increasingly be found in the logical design embodied in VLSI chips, but is in the main to be found in the software: this is not a criticism of software or of software developers, rather it is just the consequence of a valid response to the differing costs and time scales involved in creating and replicating VLSI and software designs. Here we will for simplicity refer just to the problems of software.

Great complexity brings with it many dangers. One of the greatest is difficulty of understanding: it is often the case that no single person can claim to understand a program completely, and this brings with it an associated uncertainty about the properties of the program - particularly its reliability and safety. The sheer size of a program will often give a rough idea of its complexity: the French SACEM train control system has about 20,000 lines of code, the Sizewell PPS has 100,000 lines of code (plus about the same amount of configuration data), and a typical telephone switch can have in excess of 10m lines of code. Some feel for these numbers comes from observing that the average modern novel is about 10,000 lines long, and is written in a naturally understandable language - even in cases of the most unrelenting modernism. Moreover, it should be noted that complexity, at least as it affects understanding, does not usually increase merely linearly with size.

Furthermore, only for the most trivial of software components is exhaustive enumeration of all possible behaviours feasible: we cannot simply test the software for all conditions it could ever meet in operation and thus guarantee that it will be failure-free in operation[11].

For these reasons, the most important issues involved - at design time and/or with respect to the resulting system - in achieving and demonstrating the achievement of software safety are the minimisation and control of complexity.

These issues are as important for formal software specifications as they are for software implementations. Indeed, there is conceptually rather little difference between a software component (i.e. the software implementation) and its specification - whether this is a complete specification, or just relates to particular system requirements. Such implementations and specifications are both digital systems, written in a formal language, and both may contain faults. Ideally the specification is much shorter and simpler than the component design/implementation, but particularly where efforts are made to have a 'complete' specification, the issues of complexity, and indeed discreteness and novelty, are as relevant to software specifications as they are to software designs. In other words a formal specification is just as much a 'system', which can contain faults, and cause failures (of the design process), as is the software that is designed to meet this specification.

One has to distinguish, albeit informally, between two kinds of complexity:

- unavoidable (i.e. required) complexity - whether in the specification, or in the design that is supposed to match this specification - arising from difficulties that are inherent to the task at hand;

- unwarranted complexity, arising from inadequate skills and experience, techniques and tools, particularly those related to achieving an appropriate system structure.

The first of these cannot be addressed simply within the software context - rather the questions which have to be answered are whether the apparently required complexity is actually needed, and whether the requirements placed on the software cannot be reduced without excessive detriment to the system of which the software is but a component. For example, it is generally undesirable to incur extra complexity to gain benefits (e.g. economic) that are not related to an increase in safety if this further complexity compromises either the achievement or the assessment of the safety. This points up the importance of the decisions taken early in the design process about how much functionality can be expected of the system, and the estimates of the impact of this upon safety.

As regards the second point, it is evident that the question of the adequacy of the skills and experience of the people involved is an issue which has to be addressed in any safety case, for any technology. However, there has been considerable study of system structuring as a 'divide and conquer' approach to controlling computer system (and specification) complexity.

---

[11] Consider the hypothetical case of a system where each input consists of a vector of readings arising from 50 sensors, each of which has passed through an analogue-to-digital converter with output on a 100-point scale: there are 100^{50} possible different inputs to the software (although not all will be feasible). Even if each of these cases could be tested in one second, a complete test would take some 10^{92} years. Such numbers are actually modest compared with those encountered in some real systems (and they do not allow for the fact that the *timing* of inputs may be as critical as their *value*). This observation suggests that it would be inappropriate to be complacent about systems that have shown themselves to be failure-free even in very extensive operational use: such evidence is far from sufficient to conclude that they will *never* fail.

Finally, it is perhaps worth mentioning here a different kind of threat that complexity can pose, namely to the successful completion of a software development project. There are numerous examples of systems whose developments have been abandoned, at great cost, because they grew too complex. Many of these were in the military field, but the French Cegelec Controbloc P20, which was to have been used for Sizewell B, is a cautionary example from the nuclear industry (see Appendix C).

7 Computer system design and software engineering

The development of software can involve many years of human effort, with the consequent difficulties of co-ordination and management. Each new piece of software represents a new solution to a new problem. It is equivalent in other engineering disciplines to the manufacturing prototype used to create the production line. The goal of systematising the software production process, and making it more dependable and cost-effective, motivated the invention of the term 'software engineering' nearly thirty years ago, and much subsequent effort aimed at providing a body of techniques and tools that justified such a term. In safety-critical systems, the tools and techniques of most importance are those concerned with reducing as far as humanly possible the number of design faults that are introduced (post-specification) and remain in the operational system, and in protecting against their effects.

Because of the difficulty, complexity, size and novelty of the typical software task, mistakes are bound to be made and unless these are subsequently detected and removed, they will result in defects in the product. In order to reduce the number of design faults introduced, the discipline of software engineering includes management practices, design practices and quality assurance techniques. Planning and using a well-defined development process are essential. However, the software development process is an intellectual one: it is far less predictable and controllable than a factory production line. The technical and administrative capabilities of the software developer organisation and its staff are of crucial importance. Various attempts to characterise the capability of organisations are currently being promoted. These are still rather immature, however, and lack a sufficiently firm scientific underpinning to be of much relevance to assessing the suitability of an organisation's products for use in safety-critical applications.

One characteristic of the software development process is that, in general, the earlier a fault is introduced, the more serious its impact is and the more expensive (in time and cost) it will be to fix it. Thus mistakes in the planning, requirements definition and design concept phases are of most concern in managing the development of software. This observation is one factor that motivates the use of tools that assist in requirements elicitation and in such matters as the determination of a system specification, its design, and prototyping. Bridging the gap between systems requirements and implementation - and so achieving the necessary visible traceability - is another factor in motivating the use of (suitably validated) tools.

7.1 Requirements specification

The system or plant requirements specification is arrived at as a result of a process of requirements elicitation which involves experts from a range of disciplines. The specification is typically based on an analysis of the safety of the plant and involves any other relevant information, e.g. details of earlier accidents. The requirements specification of the computer system is derived from the system requirements specification and describes the functionality of that computer system including any protective actions and performance (in particular safety) criteria.

The computer system requirements specification is the basis for all future development. As such it should possess certain characteristics that support and encourage an efficient approach to design and implementation. These include ensuring that it is:

- simple and easy to understand for all parties including licensers, suppliers, designers and users; this includes being unambiguous, well-structured and readable; it also implies that extraneous information (e.g. unnecessary implementation detail) is absent;
- complete in the sense that:
  - all important functional properties are described, including the human computer interface and other operator supports, the human behaviour in situations for which the operators are responsible, e.g. sensor failure;
  - non-functional characteristics such as safety, reliability, dependability (and this may involve security issues) and timing properties are all precisely described, and shown to have been derived from the plant safety analysis;
  - a precise definition of all system boundaries exists (sensors and actuators, other external systems, human operators);
  - physical constraints are identified for the values of all environmental variables and parameters;
- easily checkable and analysable; it must also have properties that make it easy to use for the purposes of verifying the adequacy of the final computer system and this is related to the ease with which it can be used as a reference document and support traceability;
- able to support the process of design with the necessary traceability being simply accomplished.

Descriptions are also needed for each of the different states in which the system can reside together with the events that cause the change of state. These ideas are best supported with the use of a model-based specification, i.e. a specification based on a model of the system that uses well-understood mathematical entities such as sets and functions. In particular the requirements must specify actions of the software in the event of failures of sensors or other such devices.

Model-based languages include text-based specification languages such as Z, VDM or B and graphical languages such as those used in the railway industry, or LOTOS which is popular in telecommunications applications. Such formal languages have well understood formal notations with a clearly defined syntax and semantics. However, in the nuclear industry model-based approaches to specification are not well established. Rather, dataflow-like languages tend to be employed in which behaviour is described in the form of an algorithm. Examples are the use of logic or control flow diagrams where boxes represent algorithms and the connections are the data flows. In model-based languages, invariant properties can be used to capture the notion of safety and automatic checks can then be made to ensure that the system remains in a safe state. Dataflow languages do not make explicit such concepts as the invariant property of a system. Essentially, the problem with dataflow-like languages is that there is insufficient logical separation between the specification and the implementation. Hence using these languages the validation process is more difficult.

The degree of formality of such systems is variable. In France Merlin-Gerin (a subcontractor to Framatome, a main contractor and designer of protection systems) for the purposes of specification utilises natural language augmented with diagrams which include such devices as and/or gates. This has the apparent benefit of simplicity; manufacturers are required to express the requirements in a manner that conforms to the diagrams. The Merlin Gerin language is called

SAGA [Bergerand & Pilaud 1989]. In Germany Siemens also use a graphical approach using Teleperm XS and XP [Bock *et al.* 1988].

In the case of the Sizewell B Primary Protection System, the traditional approach was adopted [Boettcher & Tooley 1994]. This approach followed a progressive and systematic breakdown of requirements into details, and the definition of these requirements in a structured set of English language documents of predefined scope and contents. These documents were supplemented by a set of functional logic diagrams which were used to specify the system functional requirements in an unambiguous way.

Since the requirements specification is developed at an initial phase of the life cycle its accuracy and completeness have the potential to have an enormous influence on the effectiveness and the efficiency of the overall development process. Accordingly there is merit in ensuring that any specification language is machine processable.

Appropriate software tools can be employed for a range of purposes, e.g. to test at the earliest possible stage that the specification has properties such as self-consistency and certain kinds of completeness, or to allow the specification to be animated so that there is confidence in the accuracy of the specification. In addition tools can be used to support the process of design by transforming or otherwise manipulating the specification, an approach that tends to ensure traceability automatically. The validation and the selection of appropriate software tools of proven quality is a matter of some importance, for their selection provides a potential source of common mode failure for the system. Where these are successful, their use has the benefit of removing tasks from the validation and verification activity and allowing the software developers to devote their energies to other activities.

7.2 Architecture

In the design phase of development the implementor takes the requirements specification and from this derives an approach to implementation. A major challenge for the designer is the need to produce a (demonstrably) reliable system from (known to be) less reliable components. For hardware components the designer would expect to have reliability estimates (at least concerning operational faults). However, for both hardware and software components reliability estimates concerning residual design faults are problematic.

The means by which the computer system architecture is demonstrated to be adequately safe and to meet the computer-based system requirements (functional and non-functional) must also be addressed at this stage. The methodical and systematic derivation of the design requirement specifications for the architecture is important for the safety demonstration. It is not a trivial task, and it must be addressed carefully and properly. To a large extent, the design process is an empirical and heuristic process based on past experience gained in previous designs. Besides, new designs are often driven by technological considerations (e.g. use of a new, more efficient or more reliable technology) or by commercial objectives (e.g. better configurability, easier maintenance). As a result, many architectures which are proposed, even the most recent ones, appear, from a safety point of view, as being somewhat arbitrary instead of derived from the system requirements.

As part of the architectural design there will be a need to identify and to select technologies capable of realising the system in a way that ensures the system can acquire the required functional and non-functional requirements. The definition of the hardware and software structures of the computer system architecture must address the non-functional requirements (such as performance, reliability, maintainability, security, replaceability) even more directly than the functional requirements.

The architectural design will involve decisions about overall system design as well as the details of software/hardware partitioning. Safety integrity levels[12] (SILs) will typically have a significant role to play in these decisions. Decisions need to be made to ascertain:

- what to implement in hardware and then which particular choice of hardware - the latter needs to be of proven quality; whether a centralised or distributed system is to be used, and then whether a dedicated or general purpose machine is to be employed;
- what to implement in software and how; thus how should the software be partitioned so that undesirable interference does not occur between sub-components of the software;
- what structure the safety-critical software should have (e.g. whether a safety kernel is appropriate);
- what functions are to be allocated to pre-written software (perhaps using libraries such as window systems, run-time code, or code generators), what languages are to be used for new software and what compilers and other tools will be used to support development;
- whether a single computer system is sufficient or whether a particular kind of distributed system will be required, with the latter having implications for communications software and possibly for co-ordination;
- what role the human operator will have, and how the interface, training, organisational climate and other operator supports should be constructed so that the necessary safety levels are always achieved.

These various decisions will include attention to such matters as: what part of the system has responsibility for detecting and handling device errors, hardware errors, programming errors, co-ordination errors, timing errors, system errors, operator errors and so on.

There are important guidelines that tend to underpin good design: they arise from the desire to control complexity and they give rise to issues such as structuring, simplicity, modularity, testability, information hiding, etc. Correctness with respect to the specification is another important property. But a further set of established principles that underlie the design of safety-critical computer systems then include the following:

- separate the safety and non-safety functions;
- use redundancy of hardware, with separate channels being separated physically and electrically;
- employ diversity of hardware;
- if possible design systems and sub-systems so that they detect their own faults, defaulting to a safe state; in the nuclear industry this is an obligation, with the licensee being asked to demonstrate that mechanisms have been devised to ensure that all devices do detect their own faults; indeed if error correcting codes are used and if some correction occurs there is an obligation in the nuclear industry to have this recorded;

---

[12] IEC1508 defines four Safety Integrity Levels (SIL) with SIL4 being the most onerous. A level N SIL defines the range of the average probability of failure to perform its design function on demand as $\geq 10^{-(N+1)}$ to $< 10^{-N}$.

- complement software fault avoidance with software fault tolerance mechanisms (e.g. exception handling, n-version programming, recovery blocks, error correcting codes, self-checking programs) when deficiencies in design are well understood;
- design to ensure easy testability, analysability (for hazard analysis, for example, but also for performance), verifiability, traceability (back to requirements), for maintainability and for assurance; the latter implies that justification should be provided for all design decisions;
- if possible identify safe states and see these as havens in the event of failures.

Of the above, the concept of software diversity cannot yet be said to be an established principle within the nuclear industry. Rather, its use is still a matter of discussion and debate. For example, exception handling is used, n-version programming arouses controversy, and recovery blocks are never used.

Ultimately designs should exhibit a number of desirable properties. At a functional level these include such properties as: completeness, consistency, testability, accuracy, traceability to the specification. At the non-functional level there is typically a need to address such matters as timing, performance, safety, reliability. The design of interfaces should be included in these considerations.

One particularly interesting approach to some of these issues has been made by the Boeing Company in the design and implementation of the Boeing 777. The company took the view that if it did not have sight and control of the manufacturing process of a component (e.g. computer hardware or a compiler) then it would use replication and hardware diversity (with subsequent voting) as a means of guarding against error. To reduce further the possibility of common mode error it would:

- select components that were manufactured using different technologies;
- employ people from different backgrounds and trained in different ways;
- continually monitor operation to check for performance and common errors.

The best approach to architectural design remains a topic which is not fully understood. Some general guidelines and principles exist and have been outlined above but there is room for considerably more research in the area.

In the nuclear industry, the need for designs to be validated and licensed imposes further requirements that are not always addressed. For example, attention needs to be paid to the documentation associated with the design, including traceability and validation results, since this forms the basis of the information for the licensor.

7.3 Software engineering

Managing the task of developing safety-critical systems which are explicitly software-based is widely recognised to be more challenging than similar developments which are based on more conventional technologies. Simplicity is a key principle that underpins all development. Within the industry the main problems stem from

- the very great difficulty of proving and thereby guaranteeing safety properties, even relatively simple ones;
- designing software paying attention to the documentation necessary for licensing and independent assessment.

The essence of the software engineering problem in the industry is about finding disciplined and rigorous ways of managing this. If complexity is present this provokes adverse reaction. Convincing evidence of safety properties is sought, and design that supports this is viewed favourably.

Generally there is no widely accepted view of the most appropriate techniques to use in the development of safety-critical systems or any clear view of the inadequacies of current methods. The relationship between current approaches and the ease with which reliability levels can be evaluated is in need of further clarification.

7.3.1 Languages

For safety-critical systems the choice of language should support the development of programs that are easy to read, easy to understand, have predictable behaviour and are capable of handling errors. In the process of language selection a number of other factors need to be taken into account, e.g. the programming language needs to fit comfortably with the facilities used at higher levels of the design process, it needs to be familiar to those producing code, and appropriate (validated) tools need to be available.

In the nuclear industry in general, the emphasis in language selection is on preferring features that permit static checking. Thus the normal review process and static analysers are favoured. Conversely the emphasis is on strenuously avoiding features or properties that reduce the verifiability of programs and that can be verified only by appealing to mentally executing the program or by dynamic testing.

Taking these observations into account then, one can identify several desirable characteristics of languages, for example:

- a well-defined syntax and semantics;
- strong typing and range checking;
- block structured;
- well conceived concepts and structuring mechanism as well as a rich selection of data types to encourage modularity, simplicity, re-use, abstraction, decomposition, information hiding and separate compilation;
- facilities to support self test and accessing hardware;
- support for the development of programs that are easy to read and understand;
- mechanisms to support fault tolerance (e.g. exception handling).

One can also identify features of languages that are considered undesirable since they tend to reduce clarity. These can be addressed through language selection or through coding standards which need to be policed by appropriate tools. Thus attention needs to be given to the use of pre-written libraries of code (unless such code is guaranteed to have been thoroughly validated), undefined variables, changing priorities of operators, assumptions about default values, and various other unsafe constructs (e.g. unchecked conversion in Ada). Use of pointers and dynamic storage including recursion, real arithmetic, excessive overloading, and concurrency all need to be used with special care.

7.3.2 Programming

Certain approaches to programming support the development of safety-critical software. These include defensive programming whereby the assumptions made when code is written are formalised and checks are made to ensure that these assumptions hold at run time. Another similar concept involves identifying assertions or annotations (usually about ranges of variables or about relationships between values of variables) which should be true at run time and including checks to ensure that these do hold.

Coding standards (ideally automatically checked) often provide a framework within which coding occurs. These are intended to ensure such desirable qualities as simplicity and readability, and the absence of undesirable characteristics. Thus there may be guidance on depth of nesting of constructs (e.g. of subroutines within subroutines), lengths of routines, use of certain constructs, numbers of parameters, and the readability of identifiers.

In the nuclear field there are two main approaches to development of software:

- the traditional approach using a specification, and from this producing code in a language such as C or PL/M;
- capturing requirements in a dataflow like specification language, validating this using appropriate tools, and from this automatically generating code, e.g. in C; the program generators are modelled on the use of various operators and other boxes for which there is pre-generated code.

In the case of the Sizewell B PPS, the traditional approach was used, with the majority of code produced in the Intel PL/M 86 language. The implementation was restricted by a set of coding standards which aimed to ensure the production of good quality code (e.g. easy to understand; avoiding practices known to be error-prone).

Another practice which is commonly employed in these safety-critical systems is defensive programming. Typically this is included in the specification of the system. Basically checks are made at run-time to ensure that: inputs lie within the expected range of values; devices including memory remain active through the use of watch dog checks; other relationships between variables remain true. These ongoing checks made at run time ensure that the system remains in a safe state, that devices continue to respond, that memory is still operational and so on.

7.3.3 Verification and validation

The twin topics of verification and validation are applicable throughout the life-cycle: colloquially verification can be described as ‘building the system right’ whereas validation is ‘building the right system’. An important principle is that the sooner in the life cycle an error or deficiency is identified the better. Verification and validation tend to be more effective when a more formal approach is taken to the process. In particular steps should be taken to carry out checks on the initial requirement specification that are as near exhaustive as possible.

A wide range of possibilities exists ranging from reviews, inspections and walkthroughs through to a range of approaches to testing (that include both static and dynamic approaches) and to the extensive use of mathematically-based (‘formal’) methods. Verification and validation methods available include functional testing, structural testing, statistical testing, equivalence partitioning, cause effect analysis, boundary value testing, stress testing, top-down methods and bottom up methods, performance testing, timing.

The validation and verification process needs to address systematically not only the functional capability of the system but also the non-functional requirements. In particular where software is being reused and has been developed in one environment, it should be verified and validated - with careful management this may benefit from automation - in each new environment.

In these activities having some model of the plant supports the development of requirements; then the validation and testing of the specification is driven by this. The testing should reflect the full range of possible uses. It should address situations where the system has to operate in a range of possible degraded states.

When a system is operational the testing process needs to continue, because of the likelihood of (operational) faults in the hardware, for which there is typically a regime whereby testing occurs every two or three months. In addition the specification will indicate the presence of other checks (captured in the concept of defensive programming mentioned in the previous section) which ensure the ongoing well-being of the plant.

The compilation process can give rise to the inclusion of code to support some run-time activities. Such code needs to go through the verification and validation process. This can be done through special arrangements with the compiler vendor or reverse engineering techniques can be employed. Some manufacturers take the step of producing their own version of the run-time system so that they can be confident that the code is well understood.

Of course, the compiler needs to be maintained throughout the life-time of the plant; a developer therefore needs to strike a balance between utilising a new version of a compiler or remaining with a version of the compiler whose defects are known but manageable. In this context it is often important to avoid optimisations in the code. If the new object code is not compatible with the old code this can lead to difficulty and point to preferring an older version of a compiler; a sense of balance is required.

The verification and validation process is usually extensive, time consuming and often regarded as involving a high level of drudgery. Yet it is important. Further work is needed to understand how to carry out this activity in a manner that reduces the effort and yet ensures the development of systems which attain a required and proven level of reliability. Safety-critical systems often have a long life, and in the nuclear industry there is an obligation to record in the safety case file tests and the results of tests as well as, for example, the details of proofs. The implication is that if maintenance occurs then considerable saving may be possible by replaying these automatically. All relevant information needs to be carefully structured to ensure an efficient approach to these activities.

7.3.4 Maintenance

The information that is associated with the development of reliable programs is typically extensive and leads to the need for careful handling of the related software systems configuration data. This information typically includes:

- specifications and designs;
- configurations and versions;
- traceability information including the reasons for design decisions;
- source code and object code;
- test data and the results of tests;
- proofs;
- errors and incident information with subsequent reliability growth information to ensure healthy evolution.

Maintenance of software (the system software or indeed tools used for development) occurs for various reasons, e.g. to remove bugs from programs, to add new functionality. This needs to be done in a carefully controlled manner which involves impact analysis (may include hazard analysis, etc.) with modification occurring in a manner that ensures a disciplined approach to its control and the ongoing evolution of the software. This will involve subsequent validation and verification with such activities as regression testing.

The problems of changing software are now well recognised by the nuclear industry, and appropriate measures are taken during the design process. These measures can include modular design, strict procedures for control of changes, impact analysis, and regression testing and re-analysis to provide a demonstration of the maintenance of the safety case.

7.4 Tools

A wide range of tools can find use in the developing of safety-critical systems. Apart from the normal range of system software such as a wide variety of language processors (including systems to check adherence to a carefully chosen language subset), there are typically tools to support each stage of the systems life-cycle. Thus tools can be used to:

- test properties of specifications and animate them;
- support the design process;
- analyse and measure aspects of code;
- support the system configuration management and version control;
- support the verification and validation phases, e.g. with theorem provers, and software to support maintenance using regression testing;
- support a range of techniques such as hazard analysis, event tree analysis, and software fault tree analysis.

Often tools themselves have limitations. The major static analysis tools, for example, are SPARK/SPADE and MALPAS plus translators. They carry out static analysis of source code and design documentation in the form of special annotations of the source code. These tools are mainly used in the UK. The analysis tools can only handle a limited subset of computer language constructs, so either the programming must be done in a limited subset (such as the SPARK subset of Ada) or tool assisted translation has to be done. Translation is time consuming and is a potential source of errors in the static analysis. These tools are designed to handle purely sequential problems and their models can only be adapted with a great deal of manual effort and difficulty to handle very simple concurrent processes.

The tool SAGA, based on the language LUSTRE, is an industrial tool used in France by Merlin-Gerin. It carries out the familiar type checking activity but also carries out the automatic code generation. This implies that code produced in other ways has to be fully integrated into these fragments, but that seems not to present any serious difficulty.

The very use of tools in the development process means that validating or otherwise demonstrating the quality of a tool is necessary for high integrity systems. Such activity needs to address specifically the introduction of code that could lead to possible threats to security that would compromise safety.

The use of tools such as high level language compilers introduces the possibility of common mode failure. It is often argued that with significant and widespread use, and removal of identified deficiencies, software errors are reduced to a minimum. Although approaches do exist for the validation of certain kinds of tools, the precise details and standards applied are often confusing and uncertain.

Various approaches exist to resolving any difficulty, for example:

- mistrust the compiler and embark on such procedures as reverse engineering the object code to produce the original source code, or verifying the authenticity of the object code;

- use multiple tools (with diversity of design) and use diversity to check the agreed results, as done with the Ada compilers in the Boeing 777 development - the complexity of the voting algorithms often presents enormous difficulty;

- have compilers validated (or ideally proved correct though for common languages this is beyond the state of the art); so at least the aim should be to ensure that compilers do not introduce errors into code or introduce dangerous code, e.g. unreliable run-time code.

In all cases there should be continuous monitoring to identify and document (hopefully with the intention of ultimately removing) any compiler errors. When diverse compilers are used, this same activity allows the gathering of evidence to support or otherwise the arguments about independent development of the compilers. Changing tools repeatedly, e.g. to adopt new versions, can be enormously expensive in terms of time and other resources. There is a balance to be struck between the benefit these bring and the cost of becoming familiar with a new product and noting the differences from the old.

7.5 Final observation

In safety engineering generally a lack of maturity in a technology can be seen as a threat to safety, and as such something to avoid; if users do not fully understand the implications of use then there may be effects that compromise safety. Within software engineering new languages, new tools, new methods or approaches all fall into this category. Likewise approaches that rely on some deep understanding or complex software system need to be treated with caution.

The use of certain technologies such as some branches of Artificial Intelligence, e.g. those employing probabilistic rule-based systems or neural networks, merits considerable caution. These systems can have unpredictable behaviour and make recommendations that cannot be readily validated. This makes it very difficult for those using or maintaining systems employing these technologies to fully understand their implications, and so their use for systems of high integrity is not recommended.

8 The software failure process

8.1 The inherent uncertainty in the failure process

People who are new to the problems of software reliability often ask why probabilities need to be used. After all, there is a sense in which the execution of a program is completely deterministic. It is either fault-free, in which case it will never fail; or it does contain faults, in which case any circumstances that cause it to fail once will *always* cause it to fail. This contrasts with hardware components which will *inevitably* fail if we wait long enough, and which can fail randomly in circumstances in which they have previously worked perfectly.

Reliability engineers often call failures due to software (and other failures arising from design defects) *systematic*, to distinguish them from *random* hardware failures. This terminology is somewhat misleading, inasmuch as it seems to suggest that in the one case an approach involving probabilities is inevitable, but that in the other we might be able to get away with completely deterministic arguments. In fact this is not so, and probability-based reasoning seems inevitable in both cases. When we use the word *systematic* here it refers to the fault mechanism, i.e. the mechanism whereby a fault reveals itself as a failure, and not to the failure *process*. Thus it is correct to say that if a program failed once on a particular input (i.e. a particular set of input values *and* timings) it would always fail on that input until the offending fault had been successfully removed. It is from this rather limited determinism that the terminology arises.

However, our interest really centres upon the failure *process*: what we see when the system under study - and in particular the software - is used in its operational environment. The software failure process arises from the *random* uncovering of faults during the execution of successive inputs. We cannot predict with certainty what all future input cases will be: in the case of a reactor safety system, for example, there would be inherent uncertainty about the physical state of the reactor during a future demand (although the set of all possible demands is well understood, as are the frequencies of their occurrence). As well as this uncertainty about which inputs will be received by a computer program, there is also uncertainty about its faults. We would not know which inputs, of the ones we had not yet executed, would produce a failure if executed (if we did know this, we would use the information to fix the fault).

There is another little-understood source of uncertainty: that of our understanding of the boundaries of the input space. Typically, in building software systems, judgements are made based on engineering knowledge that certain combinations are not possible and that therefore the software need not be designed to handle them. Practical experience, however, shows that many failures categorised as software failures only arose when the software was in a state that it was never designed to handle. The Ariane 5 failure, mentioned earlier, provides a topical example of this. The failure occurred when Ariane 4 software, reused in Ariane 5, encountered during launch different conditions from the ones for which it was designed. Whilst nuclear systems are generally designed to be robust - i.e. to cope with unexpected states - there is still the potential for unexpected input conditions to create a failure.

There is inevitable uncertainty in the software failure process, then, for several reasons. This uncertainty can only be captured by probabilistic representations of the failure process: *the use of probability-based measures to express our confidence in the reliability of the program is therefore inevitable*. The important point is that the language and mathematics of reliability theory are as appropriate (or inappropriate) for dealing with software reliability as they are for hardware and human reliabilities. This means that it is possible, during the construction of the safety case for a plant, to assign a probabilistic reliability target to a subsystem even when, in the most general case, this is subject to random hardware failures, human failures, and failures as a result of software or hardware design faults.

8.2 Measurement of safety and reliability

Once it is accepted that there is inherent uncertainty in the failure process, we must express our reliability and safety requirements probabilistically. The way in which this is done will vary according to the nature of the system. Thus it is natural to demand that a safety system, which needs to respond only rarely to demands, should have a requirement expressed as a probability of failure *on demand*; a control system on the other hand, which needs to keep a physical system continuously within acceptable operating bounds, would more appropriately have its requirement expressed in terms of a failure rate per unit *time*. Other measures can also be of interest. For example, the *availability* of a system - measured, say, in expected down-time per annum - may affect safety: an availability requirement would need to be satisfied, for example, by an air traffic control system (together with a reliability requirement).

There is often more than one such requirement imposed upon a system, each being associated with a different kind of failure. In the case of Sizewell B, for example, the reliability needed corresponded to 10^{-3} probability of failure on demand *(pfd)* for the computer-based PPS (and 10^{-4} *pfd* for the hard-wired secondary system). This figure of 10^{-3} refers only to those safety-related failures of the PPS to trip the reactor and maintain it in a safe post-trip state. Clearly, there is an additional requirement for undemanded trips to occur sufficiently infrequently: this is a safety issue, too, albeit a less important one than failure to trip on demand. In general, it is possible for there to be several such reliability and safety requirements associated with different types of undesirable events.

There are severe limitations on the levels of reliability that can be measured statistically from direct observation of the failure behaviour of a program. In the case of a protection system, for example, where the reliability requirement of the software would be expressed as a probability of failure on demand, some 4600 failure-free demands would need to be executed to claim that this *pfd* was better than 10^{-3} with 99% confidence. For 99% confidence in a 10^{-4} *pfd*, 46000 failure-free demands would be needed, and so on. If each demand requires minutes[13] of computer time (and time on a simulator generating the demands), it is clear that there are severe practical limitations to the levels that can be measured. We may be able to justify a claim of about 10^{-4} *pfd* for a protection system in this way - albeit at considerable expense - but levels that are orders of magnitude higher than this will be completely infeasible. It should be said, however, that the present reliability requirements for computers in safety systems in the UK nuclear industry appear to be more modest than those in other industries: see, for example, Appendix D on safety-critical control systems in civil aircraft.

Notice also that this unforgiving statistical limitation on what we can measure is accompanied by some stringent practical difficulties: the simulator that generates the demands needs to do so in a way that accurately reflects the frequencies with which the different classes of demand will be seen in real-life operation, and the results of each and every test have to be evaluated absolutely correctly. Achieving this is by no means easy.

These remarks do not imply that it is impossible to *achieve* (rather than measure beforehand or predict) very high levels of reliability in software. On the contrary, there is evidence in several industries that very high levels have been achieved as evidenced by failure-free working for extremely long times in operational use. For example, in the aircraft industry, software has operated without failure over many years in hundreds of aircraft (see Appendix D). However, having this confidence after the event is very different from having it at the beginning of the life of the system, when a decision has to be made as to whether it is acceptably safe to be operated.

It should also be emphasised that seeing previous systems exhibit failure-free working for long times in operation is only relatively weak evidence for the reliability of a new system. It may constitute indirect evidence, since it might make us more confident in the process by which the systems were built, but there will typically be aspects of the new system that are unique, and we can expect the process itself to have changed considerably from the time when the evidence of failure-free working was collected on the earlier systems.

---

[13] The dynamic test harness for the Sizewell B primary protection system requires about five minutes on average for each test.

9 Background to nuclear plant safety

9.1 ALARP

The Health and Safety at Work Act (HSWA) places a duty on employers (licensees, in the nuclear context) to ensure that risks are reduced 'so far as is reasonably practicable' (SFAIRP). The popular terminology which expresses the application of this duty is captured by the acronym ALARP - i.e. the risk should be As Low As Reasonably Practicable. The interpretation of ALARP has been set into the context of 'tolerability of risk' as a clarification arising from the Sizewell B Public Inquiry. This found its expression in the well known 'carrot diagram' (see figure below), which has become the standard means for the exposition of the principle.

Central to this thinking is the notion of an upper level of risk above which the operation of a plant would not be accepted; below this 'intolerable level', however, further risk reduction continues to be sought on an ALARP basis - i.e. until the cost becomes disproportionate to the improvement gained. In other words, it is not sufficient merely to ensure that risks are tolerable, when an additional safety margin can be provided at reasonably practicable cost. The achievement of ALARP clearly requires some evaluation (qualitative or quantitative) of the reduction in risk associated with adopting some particular measure and a clear view of the costs.

The ALARP principle is based on the assumption that it is possible to compare *marginal improvements in safety* (marginal risk decreases) with the *marginal costs of the increases in reliability*. Nuclear risks may offer this possibility when they are quantified (i.e. in terms of event probability and of radiation releases), and when the failure rate improvements of the systems controlling the relevant events can be evaluated. This comparison of marginal variations does not in principle require a common measure, but simply that both risk and the marginal cost or efforts to improve reliability can be realistically assessed. This assessment can however be problematic, especially when design faults have to be taken into account.

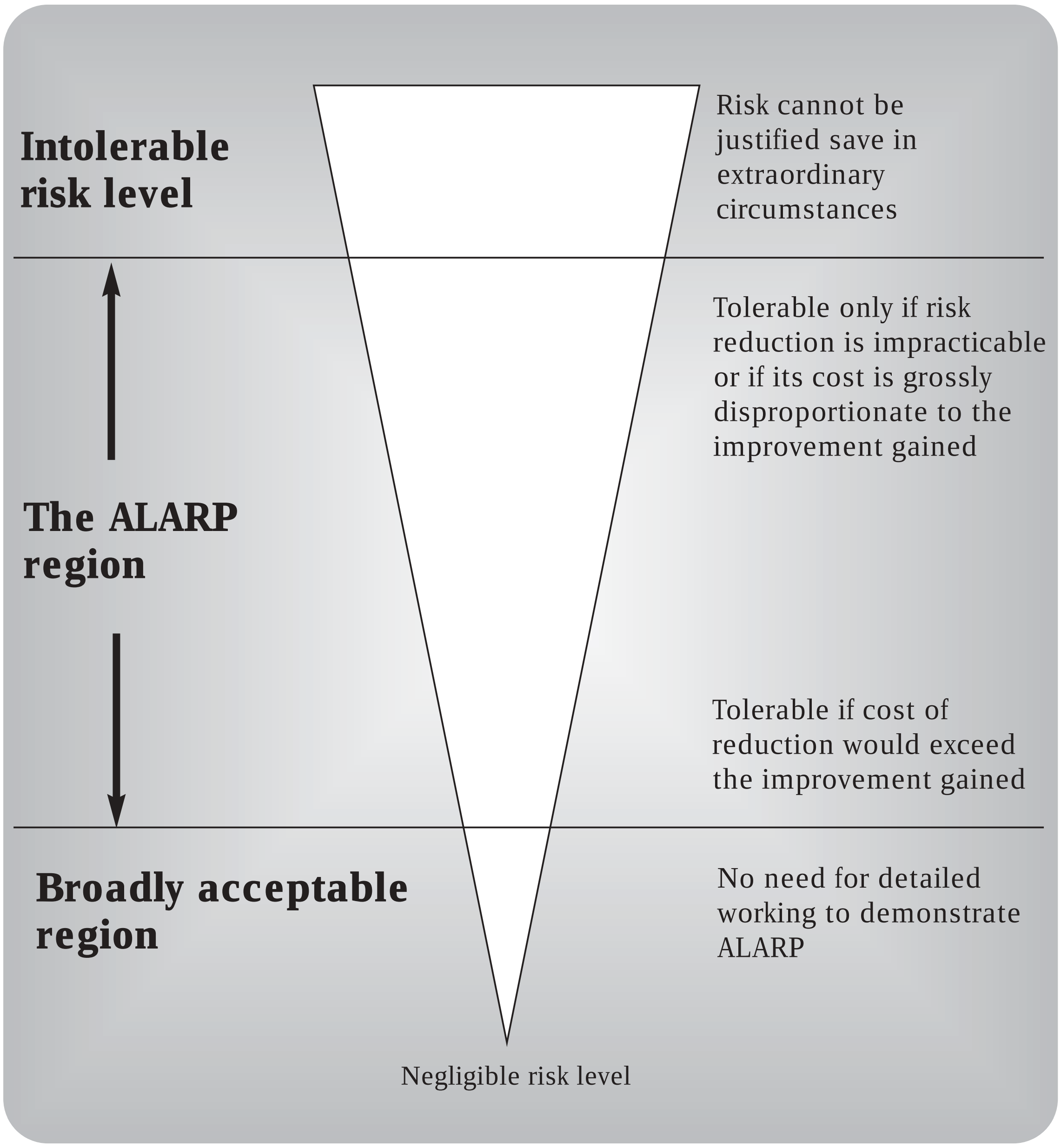
Another issue that can be raised in practice by the application of the ALARP principle is that one may have to be able confidently to evaluate these potential marginal variations before the detailed design and the implementation of the modifications are actually completed, or even started.

Nevertheless this does not rule out the application of the concept where risk reduction can only be judged qualitatively, rather than quantitatively. For example, the simple addition of a further safety feature, which costs relatively little, may be obviously worthwhile - qualitative judgements of this nature can often be readily made. Thus the application of ALARP should not be seen as restricted only to those circumstances which are amenable to quantitative reliability analysis.

9.2 Safety case and PSA

The requirement for a nuclear plant safety case arises from several Licence Conditions. Importantly, a safety case must demonstrate, by one or other means, the achievement of ALARP. Other Licence Conditions, e.g. the establishment of Operating Rules, in the satisfaction of their requirements draw upon the contents of the safety case(s). Unlike other industries, such as off-shore and railways, there is no single piece of legislation that defines the term safety case. In the Health and Safety Commission's submission to the Government's 'Nuclear Review'[14] a Safety Case is defined as 'a suite of documents providing a written demonstration that risks have been reduced as low as reasonably practicable . . . It is intended to be a living dossier which underpins every safety-related decision made by the licensee.'

[14] The review of the future of nuclear power in the UK's electricity supply industry.


Intolerable risk level
Risk cannot be justified save in extraordinary circumstances
Tolerable only if risk reduction is impracticable or if its cost is grossly disproportionate to the improvement gained
The ALARP region
Tolerable if cost of reduction would exceed the improvement gained
Broadly acceptable region
No need for detailed working to demonstrate ALARP
Negligible risk level

The ALARP principle: levels of risk are divided into three bands. Width of wedge represents level of risk.

The core of any safety case is (i) a deterministic analysis of the hazards and faults which could arise and cause injury, disability or loss of life from the plant either on or off the site, and (ii) a demonstration of the sufficiencies and adequacies of the provisions (engineering and procedural) for ensuring that the combined frequencies of such events will be acceptably low. Safety systems will feature amongst the risk reducing provisions comprised in this demonstration, which will thus include qualitative substantiations of compliance with appropriate safety engineering standards supplemented (where practicable) by probabilistic analyses of their reliabilities. Other techniques which may be used for structuring the safety case include fault and event tree analysis, failure mode and effects analysis (FMEA) and hazard and operability studies (HAZOPS).

The safety case traditionally contains diverse arguments that support its claims. These arguments are sometimes called the ‘legs’ of the safety case and are based on different evidence. Just as there is defence in depth in employing diversity at a system architecture level, so we see this as an analogous approach within the safety case itself. Another important feature of the safety case process is independent assessment. The objective of independent assessment is to ensure that more than one person or team sees the evidence so as to overcome possible conflicts of interest and blinkered views that may arise from a single assessment. The existence of an independent assessor can also motivate the assessed organisation. The actual evidence in these different assessments will, however, be largely identical and in that way should not be confused with the different legs of the safety case.

Probabilistic Safety Analysis (PSA) is now an accepted aspect of the demonstration of safety. The PSA is based on the overall plant design and operation and covers all initiating events that are in the design bases. It is performed using best-estimate methods and data to demonstrate the acceptability of the plant risk. The PSA provides a comprehensive logical analysis of the plant and the roles played by the design for safety, the engineered safety systems and the operating procedures. The PSA also demonstrates that a balanced design has been achieved. This means that no particular class of accident of the plant makes a disproportionate contribution to the overall risk. The PSA provides information on the reliability, maintenance and testing requirements for the safety and safety-related systems. The techniques used for safety analysis are various with fault trees, event trees and FMEA being the dominant methods, and HAZOPS being used in fuel reprocessing applications.

The PSA provides a check on the level of safety achieved in terms of plant risk and provides a means of claiming that the risk is lower than the intolerable region established by the NII's Safety Assessment Principles. The PSA can then be used to demonstrate that risks are as low as reasonably practicable by investigating the effect on plant risk of modifying the plant safety provisions. It follows that levels of unreliability will have been ascribed to the systems that determine the overall plant risk, and the achievement of these levels will need to be demonstrated in a robust manner in the safety case. However, there are some aspects of safety that the Principles recognise as not readily amenable to simple quantification of failure. The role of human factors (at an individual, group and organisational level) in achieving safety and initiating accidents is hard to quantify meaningfully, especially when knowledge based activities are concerned. Similarly the contribution of good management practices is hard to assess, although research and some progress has been made in this area. Other areas identified that are difficult to quantify, are common mode failures and other types of failure due to design faults or specification omissions. The latter are particularly important from the point of view of this report, since they include failures due to software faults.

9.3 Safety categorisation

Software is a pervasive technology increasingly used in many different nuclear applications, but not all this software has the same criticality level with respect to safety. Thereby not all the software needs to be developed and assessed to the same degree of rigour. On the contrary, since development and V&V resources are always limited, attention in design and assessment should preferably be weighted to those parts of the system and to those technical issues that have the highest importance to safety.

We have already discussed in Section 4 why the importance to safety of a computer-based system and of its software is essentially determined by the functions it is required to perform in the plant. This importance is therefore evaluated by a plant safety analysis with respect to the safety objectives and the design safety principles applicable to the plant. It is also determined by the consequences of the potential modes of failures of the computer system and of its software. The latter evaluation however is usually difficult because software failure occurrences are hard to predict.

The distinction between the safety and safety-related classes defined in the introduction is essential in the nuclear industry [IAEA 1980, IEC 1993, IAEA 1994]. Safety, design and V&V requirements are in principle different for the two classes. In this respect, it is again interesting to distinguish the roles played by safety and reliability. The class to which a system belongs is solely determined by the importance to safety of its functions, no matter how reliable the system is.

These categories can be, and indeed are, advantageously used to relax the *reliability* constraints that are imposed on software-based systems. Additional lines of defence - external to the software-based system - can and should be used to reduce the importance to safety of these systems so that requirements in terms of reliability, availability and security can be lessened.

There are however serious problems when these classes are applied to determine the expected or achieved reliability of a software-based system or its fitness for use in a safety system. The difficulty comes from the impossibility of quantifying the 'quality' of a piece of software, of guaranteeing a given level of quality, and of tailoring and controlling this quality by enforcing design and development procedures. If categorisation is used to relax the requirements on the quality of the development and V&V processes for lower safety categories, the consequences of such relaxations on the eventual reliability of the software is in general unpredictable.

There are standards that allow certain relaxations of this kind, for instance by requiring the use of more stringent test coverage metrics for more critical categories [RTCA 1992, MoD 1997]. There is an implication that each additional requirement upon the development and verification process tends to demonstrate a reduction in the number of faults of a particular class. The difficulty lies in determining what effect such a reduction would have had on the reliability of the software or the safety of the system. The exact philosophy for such an approach is not made explicit in the standards, but there is a logical case for requiring greater effort to detect possible faults for the higher criticality systems. With this interpretation the requirements of the standards are consistent with the principles of SFAIRP and with limiting at least some of the causes of uncertainty in the risk assessment.

One is also faced with the problems raised by pieces of software which support functions of different criticality and which also must somehow interact, or communicate, or merely coexist on the same hardware. If one cannot prove that the less critical parts - whatever their behaviour, correct or not - cannot adversely affect the more critical ones, it is a common conservative practice to allocate the same highest criticality level to all of them.

This problem, however, requires more attention and research. There is clearly a balance to be achieved here between the amount of design and V&V efforts that results from this conservative approach, and the amount that would be needed to obtain evidence that - despite possible interactions - the safety cannot be compromised by the behaviour of the less critical parts.

Many computer system designs seem to ignore the possibility that the effort of the second kind may be reduced if the design is adequate. Separate processors coupled with one way simple proven protocols could be more advantageously used. It is also often forgotten that components can be isolated from one another not only physically but also logically. Logical firewalling can be spatial, e.g. through separate virtual memory spaces and protected memory segments, and/or temporal by enforcing appropriate time schedules protecting the more critical executions from overruns of the less critical ones.

Other logical mechanisms are worthy of further investigation. An example is the 'object model' approach to software system structuring, which provides a means of defining software system components ('objects') and controlling the flows of information between them. Such controls could in principle make it feasible to build software systems from components of differing levels of criticality, and to restrict the propagation of errors between levels. Advanced architecture designs of this kind are under development (see [Hoyne & Driscoll 1992, Bernstein & Kim 1994] for avionics examples).

10 Safety cases for computer systems

10.1 Introduction

Although computer technology changes rapidly in developing increased performance and penetrating new applications, it is hardly a new technology anymore. It would seem a reasonable aim that the production of software safety cases should become much more part of the overall engineering approach and not something that gives rise to undue project or technical risk. However, this is not the case at the moment, despite the considerable experience in both the nuclear and non-nuclear sectors in developing what are in effect software safety cases (see Appendices C, D, E). The problems of software engineering in general are discussed elsewhere in this report. In terms of the safety case, past problems have been due to:

- lack of initial explicit definition of the technical approach to be taken in justifying safety cases;
- confusion over what is necessary to achieve dependable systems and the evidence to justify a system: for example, many standards contain an undifferentiated mixture of project development and assurance activities;
- lack of a documented, agreed, accessible engineering process for computer safety cases (rather than for their development);
- difficulty in justifying 'reasonably practicable' and demonstrating ALARP.

In this report we discuss a number of areas where greater understanding or consensus is required. We now examine in more detail the impediments and how we might address them by improving present practice or encouraging specific research.

10.2 Integration with systems engineering and engineering principles

The safety case for a computer system cannot be divorced from the safety justification of the wider system of which it forms a component. The UK nuclear industry has in the past used conservative engineering and design principles (such as the separation of control and protection) which enhance safety. It is prudent to maintain these existing principles and if necessary reinterpret them as the technologies change. Below we examine areas where some reinterpretation is needed: claim limits; PRA; safety case process; diversity; independent assessment.

In considering the reinterpretation of these principles we have studied in some depth the systems engineering approaches in different sectors (see Appendices C, D, E).

10.2.1 Conservative requirements

One area of conservative design is the use of claim limits. This design principle limits the reliability that can be claimed for a redundant system and is important as it can lead to a recognition of the need for additional diversity and defence in depth in design. As discussed in Section 8.2, there are quite stringent limits to the claims that can be made for the reliability and safety of software-based systems. In particular, since there are no accepted means for assuring that a software-based system of other than trivial complexity has ultra-high reliability, reliability goals for such systems must be sufficiently modest to be *achievable* and *assessable*.

The practice in the UK nuclear industry of limiting claims for non-diverse redundant systems to 10^{-5} *pfd* is particularly relevant where identical copies of software are running on redundant channels. In fact the claim may need to be more modest, depending upon the complexity and novelty of the system. The application of a more modest claim was particularly evidenced in the safety case of the Sizewell B nuclear reactor, where identical copies of the software were running in a 4-way redundant primary protection system. NII had judged in 1991 [Hunns & Wainwright 1991] that the reliability of the software of a safety system could be considered comparable to less than 10^{-4} probability of failure on demand *(pfd)* provided that the software comprehensively met the elements of its 'special case procedure' (see Appendix A). An unreliability figure set at the $<10^{-5}$ level was considered to require a level of demonstration not judged to be practicable using the available techniques - for example, the contribution which might be offered by the application of formal methods was restricted by the lack of industrial strength in such techniques at the time. This then set a natural claim limit of $<10^{-4}$ *pfd* for a software-based system.

However, it should be recognised that the 'special case procedure' represents a stringent set of requirements and that any falling away from these requirements challenges that $<10^{-4}$ claim. Moreover, where the application is safety-critical, the demonstration must be such as to provide a high confidence that the required integrity has been achieved - i.e. there should be a distinct margin of additional demonstration to allow for inherent uncertainty. With this in mind a demonstration of the minimum reliability requirements for the computer-based safety and safety-related systems is sought through an investigation of the sensitivity of the plant risk to variation in their reliability. This minimum requirement should, of course, be less than that which is targeted by the demonstration.

The use of claim limits results in the need, in some applications, for diverse systems to meet the overall system requirements. However, justification for the use of one software-based system to back up another cannot be based upon a reliability calculation which assumes that diversity is sufficient to claim *independence* of failures. The extent of the benefit that will be gained from diversity - and thus the level of reliability that can be claimed for the diverse system - depends upon the closeness to independence that has been achieved, but this is extremely difficult to measure [Eckhardt & Lee 1985, Littlewood & Strigini 1993]. Indeed, a defensible rule of thumb, based on some experimental evidence, is to assume that the use of software diversity will achieve little more than a single order of magnitude increase in reliability over a non-diverse system [Knight & Leveson 1986].

In some circumstances, a very simply designed back-up system (possibly hard-wired) may be required, since it is only in such cases of extreme simplicity that one could have any confidence that the system will be free of some classes of design faults (e.g. algorithmic ones). The assessment of the benefit from diversity here, and thus of the overall system reliability, would depend upon this claim, rather than on an argument for statistical independence.

10.2.2 Engineering principles and PRA

The UK nuclear industry has a very mature systems engineering infrastructure. There is emphasis on sound safety engineering principles leading to good design that has the safety margins and defence in depth that is required. Probabilistic risk assessment is used to confirm that the design is adequate and it is used to balance or optimise the design.

The extent to which PRA is coupled to the design process, rather than to a certification or validation activity, varies between industries and countries. The formal position in the French nuclear industry (see Appendix C) is that PRA is not a requirement nor part of the design loop, whereas in the UK PRA has a much stronger presence in the design loop. This is in part due to the use of ALARP in the UK[15]. The UK systems engineering approach has led to a greater emphasis in

[15] However, the formal positions may reflect different policy presentations of what, underneath, is a more similar approach than the policies indicate.

the UK on demonstrating reliability figures for software, either implicitly or explicitly. One can also see the difference with the US aerospace industry (see Appendix D). There the justification that the system has reached the 10^{-9} probability of failure per hour target for design errors is based, in the first instance, on the application of engineering principles and good process: it is only quantified retrospectively when data is available on the safety of a large fleet of aircraft over its lifetime.

10.2.3 Safety case process

The safety case should be developed in the context of a well-managed quality and safety management system and be executed by competent persons (see Section 10.3.4). The delivery of the safety case should be phased along with other project deliverables and integrated into the design process.

Evidence to support the safety case is produced throughout the system life cycle, evolving in nature and extent as the project progresses. In the earlier phases there will be *management* evidence of plans and high level technical design decisions. Later, *design evidence* will accumulate concerning the features that contribute to the safety of the system (e.g. fault tolerance, fail safety). Evidence will also be produced during the *development* of the software. This will include evidence of the *process* that has been followed, such as the results of reviews and the nature, source and method of detection of any faults found. It also includes the results of the verification and validation activities including those from simulation, testing, proof and static analysis. Once the system becomes commissioned there will accumulate *operational* evidence of actual experience with the system and its maintainability.

The safety case should be a clear embodiment of the evidence and arguments used, so that agreement can be reached on the validity of the conclusions. It is a document with a diverse readership, and should be amenable to review by well-informed generalists as well as by specialists. This implies that the safety case should be structured to allow access at different levels of detail and from different viewpoints, that the documentation should be hierarchical, and that the safety arguments should be presented at a number of levels of abstraction.

The safety case is a living, cradle-to-grave document. Since changes to software are changes in design, and have potentially unlimited effect, it is particularly important that there are mechanisms for demonstrating the maintenance of safety through all possible changes in the computing system and/or the plant it is controlling and/or protecting.

10.2.4 Independent assessment

Independent assessment is an accepted part of the system safety case process that has been and, we believe, should continue to be applied to computer systems in the UK nuclear industry. Indeed, since the assessment of the dependability of software-based systems does not have as strong a scientific underpinning as some more conventional technologies, there will probably continue to be a large element of expert judgement involved. This element of subjectivity may make it more difficult for an assessor to resist any pressure that he may receive, thus increasing the relevance of independent assessment.

Practices vary widely in the extent and organisation of independent assessment. One extreme approach seems to be that of the Sizewell B independent design assessment (IDA), where the purchaser of the system conducted extensive evaluations comparable to the effort put in by the developer. In other sectors, such as defence, one sees a model that seeks to place the responsibility on the vendor to provide the evidence [MoD 1996, MoD 1997] and to limit the extent of IDA work to perhaps a few percent of the development budget. Similarly we see different models for the interface between the independent assessors and the regulator. For example, the practice in the US aviation industry is for licensee staff to report directly to the regulator (see Appendix D).

It is important not to confuse independent assessment with the use of multiple legs (nor with QA) in a safety case. The classical independent assessment will examine, but not produce, the different legs of the safety case. In practice, though, these notions can be mixed. The strength of the classical view of assessment is that someone is casting a critical and independent eye over what evidence is produced. The weakness is that for the assessment to be meaningful the assessor needs to do some confirmatory independent analyses to check the results. This analysis could be based on different evidence and so the situation could develop where the IDA evidence becomes one leg in the safety case. This in fact happened in the case of Sizewell PPS: independent assessment was carried out using different techniques and generating different evidence from that of the vendor of the system, so that one part of the IDA generated one of the independent legs.

The increased trends to COTS software or PDS (See Section 10.4.1) will require a method for judging the worth of independent assessments already carried out. These assessments may range from internal independent review within an organisation to third party certification. Some of the assessments may be done in the context of formal regulatory approval, others may be generic ones commissioned by the vendor. As noted in Section 11, the present state of standards makes this particularly difficult to tackle.

It is not only these industry trends that raise questions about independent assessment. The technology of computer systems also brings particular issues to the assessment work. For example, there is the issue of what mix of techniques should be used in assessment. There is some data, from Sizewell and from software experiments, that scrutiny by experts can be very effective at finding faults in code. It is also apparent that there are classes of faults that would be very hard to find that way, and that different review techniques seem to catch different types of fault [Fenney 1994]. One of the motivations for moving towards mathematical methods is that they provide a more rigorous notation for review [Joannou & Harauz 1990]. But again, what level of rigour is required? There is some evidence that whenever machine assisted proofs are conducted discrepancies are found between the code or hardware design and its specification [Rushby 1993]. To what extent are the results of these techniques dependent on an independent assessor and how much could be left to the vendor? To what extent is the use of an independent tool a substitute for an independent person?

To sum up, changes to the nuclear industry and the continuing changes in the C&I sector, coupled with the particular technical problems of reviewing software meaningfully, raise questions on the appropriate extent and organisation of independent assessment. There is a requirement for a scientific underpinning that would allow industry and regulators to assess and reach consensus on appropriate approaches.

10.3 Building a convincing and valid argument

We have discussed above how there is a need to integrate the computer safety case with the system engineering approach used in the nuclear industry. It is also necessary to have understanding and valid technical arguments at the heart of the safety case. Building a convincing and valid argument for the safety case depends on a number of difficult aspects:

- the different types of convincing evidence that support the claim being made ;
- the combination of evidence to provide a robust safety case;

- the demonstration that sufficient evidence is available;
- judging the competency of those involved.

While these are not specifically software related problems, as we have noted above the particular uncertainties in our knowledge of software, together with changes to the industry and to the market, increase their importance. In our consultation we sought to establish the extent to which argumentation in support of the safety case had been adequately studied already. We came to the view that it would be helpful to have a more explicit approach to argumentation and a greater awareness of argumentation as a subject for study (the role of separate argument legs, etc.). These aspects are now examined in turn.

10.3.1 Types of evidence

In the safety-critical industries the need for justification for software-based systems has led to the development of a variety of different approaches and there is evidence from our studies of different industries that the safety cases in one industrial sector might not be acceptable in another. We have also found that the relative weight given to evidence of the quality of the software production process (compared with a direct evaluation of the reliability of the operational version of the software), and the stress on design for assurance, vary considerably (see Appendices C, D, E). However, although the different industries put different emphasis on the different components of evidence, it is clear from our surveys of practice and other work that the evidence comes from a combination of:

- *Design features* - as in the architectural requirements for diversity, the railways' attention to failure detection [CENELEC 1995] and the nuclear fail-safe systems [Keats 1983].
- *Process* - this can be characterised as evidence of the quality of the practices and procedures and of the rigour and competence with which these have been applied. It is exemplified by the avionics standard [RTCA 1992] - see Appendix D - and generally provides indirect evidence of the behaviour of the system.
- *Experience* - real field experience of systems that are stable and operating continuously can provide valuable evidence of their fitness for purpose. For demand-driven systems the evidence is far less compelling unless they have received a significant number of demands in relation to their reliability requirement, which is not the case for nuclear reactor protection. In this case simulated experience is more valuable [Keats 1983, Fenney 1994].
- *Analysis* - such as animation of safety properties, formal verification that an implementation satisfies a specification, and worst case timing analysis, can be used to show absence of certain types of faults and support claims about behaviour.

It is also clear that different types of evidence convey a different level of conviction. While we have found general rules hard to arrive at - there are always counter examples - we believe it is helpful to rank evidence as follows:

- *deterministic* evidence is usually to be preferred to statistical (e.g. a proof that a certain class of failure cannot occur may carry more weight than a statistical claim that these failures occur at a low rate);
- *quantitative* evidence is usually to be preferred to qualitative (e.g. numerical data on numbers of faults found during analysis and testing may carry more weight than claims for the quality of the development processes used);

- *direct* evidence is usually to be preferred to indirect (e.g. in a claim for a software reliability level, a statistical measure based upon operational testing may carry more weight than evidence of comprehensive test coverage).

Given the important role of PRA in the UK plant safety case and licensing process, one of the sources of evidence for the safety case for reactor protection systems should, we believe, be a measurement of software reliability. One way to do this is via observation of failure-free working (or at least working free of safety-critical failures) of the software during extensive simulation of operational use. In other cases, there may be actual operational experience of the software in similar circumstances to those pertaining to the safety case, in which case this should be used to evaluate reliability.

10.3.2 Combining evidence

One of the key concerns for the safety case is how the evidence is combined together to form a valid and convincing argument. A safety case should be deliberately constructed to call upon many different types of evidence: i.e. the 'legs' that support the argument. While this is intuitively appealing it is surprisingly difficult to put on a more formal footing. Work is needed to understand and formalise the current intuitive notions of 'independence' of evidence and to understand the different contribution of elements of the evidence to the acceptability of the case. While the multiplicity of different arguments can inject diversity, provide protection against possible failures of reasoning or of assumptions, if not well-presented it can also lead to confusion and lack of focus in the safety case.

Different system safety case strategies can be deployed in arguing the acceptance of a component of that system. Often our ignorance or uncertainty in the behaviour of a component will lead us to adopt a particular approach. For example, uncertainties in the ability of an operator may lead us to examine the worst case consequence. Difficulties in estimating the reliability of software may lead us to increase the dependence on a physical component such as a valve. Currently, little is documented about these different approaches to designing safety cases. In our view, a greater understanding is needed of the safety cases for heterogeneous systems that include people, software and other equipment.

Bearing in mind that judgements of the suitability for purpose of software-based systems will inevitably continue to be based upon disparate evidence - e.g. logical evidence from formal analyses, evidence about aspects of the design such as fault tolerance, direct measurement of reliability and/or safety from statistical testing, and so on - we need general, formal ways of combining such evidence into an overall evaluation, preferably quantitative, of safety and reliability. In the shorter term, more rigorous means of *qualitative* reasoning are needed.

Some techniques for increasing the accuracy of human subjective judgements, and for combining evidence from different judges, are discussed in the report of the ACSNI Human Factors Study Group [ACSNI 1991]. The potential for a more systematic use of these techniques in software safety cases should be addressed.

10.3.3 Determining that sufficient evidence is available

Having established the main sources of evidence for assuming the fitness for purpose of a computer-based system, we need to address how much evidence is required. For this purpose it is important to distinguish precisely between what is needed to achieve a specified dependability and what is needed to demonstrate that achievement. The former determines the dependability which is built into a system, and therefore is subject to ALARP in terms of 'enough', whereas the latter is associated with 'demonstration adequacy', which is a matter for judgement by those who assess the safety case. As such (and quite separate from ALARP) there can be scope with the latter for negotiation between licensee and regulator. The extent of evidence required will be driven by a number of factors, including the approach taken to judging confidence, and the cost of evidence. We now discuss these separate issues in more detail.

ALARP/SFAIRP

One area we have already identified as problematic - perhaps because it is so difficult - is the consistent application of ALARP/SFAIRP to software-based systems. As we have seen in Section 9.1 ALARP is a principle applied at a system level to negotiate the extent of further risk reduction measures. Through a system level discussion of ALARP the reliability of the protection functions can be adjusted. Often, as in the case of the Sizewell B PPS, the overall risk may have a range of insensitivity to changes in the reliability of individual C&I components because of the redundancy that is built into the overall plant design.

To apply ALARP one needs to know the costs and benefits of changes to the engineering of the computer system. This is an area of long standing problems within software engineering. However, some dependability enhancing activities (for example, randomised dynamic testing as part of the independent validation phase) can be costed since the extent of work required is well known in theory and there is now experience with turning this into practical engineering [May *et al.* 1995]. Other similar activities may have a small incremental cost (e.g. using a specific software engineering technique if the team involved is already expert in it) and the benefit be demonstrable or at least plausible.

Many of the software engineering options are fault avoiding measures and we will need estimates of the impact that these have on reliability (and other qualities). For software reliability the problems of judging the impact are well rehearsed, but in terms of a safety case it is often *orders of magnitude* changes in *mtbf* or *pfd* that are significant. If we have a set of techniques that we feel, in general, can provide us with systems of modest safety criticality (e.g. SIL1 or 2 on the IEC1508 scale) - which is not unreasonable since there is evidence that typical industrial C&I systems achieve SIL 1 (see figure in Section 10.4.1) - we need techniques that are two orders of magnitude better to gain the criticality required for some nuclear safety systems (e.g. SIL3 or 4).

These orders of magnitude improvement offer severe challenges in both achievement and assessment. To move from industrial good practice, as described for example in IEC1508 SIL1 to two higher SIL levels will, if it is possible at all, probably require a number of special measures. So, for example, although the use of COTS software might be the way of the future it is not something that is obviously practicable: the amount of additional evidence needed to support the required claims may be prohibitively large.

In practice SFAIRP is used where the risk reduction benefits of a technique are not known or hard to quantify. For example, many of the softer non-functional requirements, such as maintainability or usability, involve a large degree of subjective judgement, and it is difficult to relate them to risk reduction even if the qualitative benefits may be clear.

However, SFAIRP is also an important principle in its own right in its appeal to generic good practice as well as risk reduction. It therefore embodies the traditional engineering approach that quality components are needed for safety-critical systems. This is particularly important since in well designed redundant systems the overall risk is usually insensitive to variations in a single component, so that the decision to adopt a particular measure often revolves around a consideration and judgement of reasonableness. Standards, particularly international standards, and industry good practice are key to demonstrating what is ‘practicable’. As with ALARP, considerations of cost are important once minimum standards have been met.

Confidence

Application of the ALARP/SFAIRP principle to software-based systems seems to raise some special difficulty because the *confidence* that can be placed in evaluations of their reliability or safety is usually less than it would be for more conventionally engineered products.

There are in fact two ways in which this uncertainty applies. Firstly, there is the question of deciding the set (and extent) of production process elements needed to imbue a system with a given level of dependability; which, according to costs versus benefits, might be negotiable under ALARP. Secondly, there is the extent of evidence of those processes, supplemented by other demonstrations, considered necessary to constitute a convincing case; which (as earlier mentioned) is outside the ALARP arena, as currently understood. Both aspects, would benefit from further elucidation.

A recent example of this problem arose in the assessment of the Sizewell B software-based Primary Protection System. An assessment of the reliability of the PPS software, based on evidence such as the quality of the development process, analysis of delivered product and dynamic testing, resulted in a consensus among the safety case assessors, including the regulator, that it was sufficiently reliable to satisfy its part of the safety case (10^{-3} *pfd*). Others, however, believed that greater confidence could have been gained by conducting operational testing based upon statistically representative simulated demands, allowing a direct statistical evaluation of the probability of failure on demand. The point here was not that such further evidence would allow a claim for higher reliability, but that a claim for the same level of reliability could be made with greater confidence. In fact, there seemed to be no objective basis (or principle) for settling whether or not this extra confidence, via statistical testing, was necessary.

The above example concerned confidence in quantitative evidence, but as with the case of combining evidence discussed above, we also need to be able to deal in valid and defensible ways with confidence in *qualitative* evidence. An example is the situation in which confidence might be eroded by a series of problems each of which is, in itself, minor.

We believe that attention should be given to formally incorporating into licensees' and regulatory guidance a recognition of the importance of the level of confidence that can be placed in assessments of risk within the concept of an 'adequate' safety case. What is needed is to clarify and define the notion of 'adequacy' in this context, such that it can be used to guide and justify decisions as to the required extent of activities that will establish the level of confidence that can be placed in a risk assessment.

Costs

We have not addressed the costs of building and assessing safety-critical computer systems in any detail. Clearly, however, cost will be an important factor for any system, both direct costs of purchasing the system, developing the safety case, and setting the system to work, as well as indirect costs arising from 'time to market' considerations and impact on plant availability.

Although a regulator could demand, within the constraints of ALARP, the provision of a given safety feature without regard to its cost, the reality is that a cheaper feature is likely to be volunteered in those circumstances where there is a choice. In fact, the more expensive provisions tempt the mounting of ALARP arguments for their avoidance and, where the arguments appear finely balanced, this can lead to extensive discussion. The relatively inexpensive provisions, on the other hand, whether finely balanced or not in terms of their benefits, tend to be adopted with little or no debate. Thus, in reality, the extent of safety features built into a system, and its safety demonstration, are affected by the costs of these features. We are talking here, of course, not about 'absolute' acceptability, which must always be comprehensively demonstrated: it is the margin by which this limit is exceeded which is the potentially vulnerable area.

There is a need to be alert to the interplay, which is often subtle, between the costs of alternative types of design solution and production method for a system, and those of establishing and maintaining its safety case. In the example of the Canadian Darlington protection systems the particular design solutions adopted led unfortunately to the position where changes to the software, after its commissioning, could not be sanctioned because establishing their benign impacts upon reliability was judged infeasible. This had a serious life-time cost implication. In general, features which contribute to the achievement of high integrity in the original production of a software design will also assist its modifiability. Nevertheless, some features which assist design may hinder verification and, in certain cases (e.g. the subsequent unavailability of a tool used to generate the code), may make more difficult the later implementation of changes.

Although the actual costs of different forms of evidence are very project specific, the general form of cost functions can be estimated. For example, it is known how the degree of testing required for a reliability demonstration varies with integrity level, it is reasonably clear on what scale of system formal methods are applicable, and what the costs are of varying degrees of rigour. Some Government organisations systematically collect data from projects on such costs. There is thus the basis for a more systematic approach to assessing the costs of different forms of evidence and different safety case strategies.

Recent experience in the UK on Sizewell B has been very expensive in terms of the cost per safety function given the rather modest integrity that was accepted for the system. Developments in industry are aiming for much lower (e.g. orders of magnitude lower) costs in the next five years and this may open up new applications in the nuclear industry providing the safety case issues have been addressed.

10.3.4 Training and competency

The well known difficulty of evaluating the dependability of any computer system, simply by examination of the product alone, means that the adjudication of fitness for purpose must take into account also the additional evidence of how the system was produced, and hence of the inherent quality injected by the production process itself. This fundamentally depends, of course, upon the competencies of the contributing individuals, who may well comprise a considerable team, especially where large systems are involved and high integrities need to be demonstrated - as in the case of a reactor safety system.

Competency is an individual or team attribute - it might be described as the ability ‘to do’, with reference to one or more recognised standards of performance. Thus, we can be generally confident that an individual possessing competence in a particular activity will perform in such a way as to vest that activity with an expected level of quality. As an entity, competency can involve, of course, a complex of many factors, although the possessions of relevant training, knowledge and experience are clearly core aspects.

The evidence of an individual’s formal training is normally available and, to the extent desired, can be factually evaluated. The content of the training can be checked for relevance and comprehensiveness; and, where there is a record of examination, this indicates the trainee’s associated up-take of knowledge and skill. Similarly, an individual’s relevant working experience (and demonstrated abilities) can be reviewed, especially where a systematic record of work achievements (as well as work areas) has been maintained. Furthermore, the periods and dates of the relevant experiences and training will normally be known, and hence can be judged with respect to the currency of the associated competencies.

In principle, this type of objective information (at individual, team or company level) could be incorporated as part of a safety case. Naturally, some individuals may question the appropriateness, and indeed the value, of utilising their more personal information in this way; nevertheless it seems an unassailable principle that professional people should be able at any time to describe the competencies which entitle them to function in their given roles.

It is clear, however, that there are other competency-related factors, less readily documentable, which can substantially influence the quality of performance achieved by an individual. Important among these are the various personality characteristics, such as self-discipline, judgement, communication skills, honesty, ability to work with others, responsibility, maturity, flexibility, creativity, etc. Similarly, organisational factors (employer expectations, work loadings, rewards and sanctions, etc.) exert their influence. There can be no doubt that all of these significantly affect how an individual performs, and the quality of the work produced. In some instances it may be possible to make subjective evaluations of such factors, and if it could be provided the inclusion of a well constructed analysis which appropriately addresses these aspects would weigh positively as part of an overall case supporting a high integrity claim.

Nevertheless, the most objective demonstration of an individual's competency must be the evidence of the relevant training, experience and associated achievements, albeit that even this information cannot be used mechanistically to predict the error-freeness (say) of that individual's work. Despite the latter shortcoming, this in no way denies the important, even essential, nature of the support to a safety case which this evidence affords. Even though the gearing to 'integrity of product' may be more directional than specific, it is worth bearing in mind that a strong demonstration of competency strength is an equally strong demonstration of lack of weakness. Since, particularly for the high integrity applications, the latter might be regarded in many quarters as essential, it is this avoidance of the negative rather than the more tenuous evaluation of the positive which arguably constitutes the clearer criterion of acceptability.

Notwithstanding how such information is judged, the evidence of relevant competencies appears to be a legitimate and desirable element within any safety case purporting to demonstrate, as part of its overall argument, a high quality of production process. The extent of the confidence engendered, however, will be maximised if the evidence is developed and presented in a thorough and systematic manner.

It is accordingly recommended that an appropriate method for the formal analysis of competencies should be developed and used as a basic demonstration element within any safety case which aims to justify a high integrity claim for a computer-based safety[16].

10.4 Future focus and emerging concerns

There are a number of issues that we judge will have increased importance in the future due to a combination of market conditions and technology developments. In this section we focus on three of these issues: the use of previously-developed software; the increased perceived and actual threats to computer security and the threats this might pose to safety-related systems; and the increasing reliance on computer-based tools.

There are also other changes that we can perceive in the market. The nuclear industry will not be a significant market sector for safety-critical software and most developments will be likely in sectors such as railways and aerospace. Railtrack alone, for example, is committed to investing £2500M in signalling in the next ten years and a significant proportion of this will be on what we would classify as safety-critical C&I systems. The nuclear industry is likely, in the next 5-10 years, to be solely driven by replacement and refurbishment of existing systems with perhaps some new applications as costs and licensing risks are reduced. Other sectors will drive the price of safety-critical systems down. As in industry generally, the structure of the supply market is likely to continue to change as is the structure of the nuclear industry itself. It may therefore be appropriate for the nuclear industry to liaise more closely than has traditionally been the case with other sectors and to seek partnerships so that appropriate technology is available to it.

[16] A starting point, but no more than that, would be the Software Engineering Institute's Capability Maturity Model [Paulk *et al.* 1993].

10.4.1 COTS software

The term pre-existing software, or commercial off the shelf (COTS) software, is taken to mean software which is used in a computer-based system, but which was not produced by the development process under the control of those responsible for the implementation of the system. It may also be described as 'pre-developed' software or even 'software with an uncertain pedigree'. The importance of COTS software is recognised internationally and it is being considered by the IEC Committee addressing the modernisation of IEC880.

Typical uses of pre-existing software are:

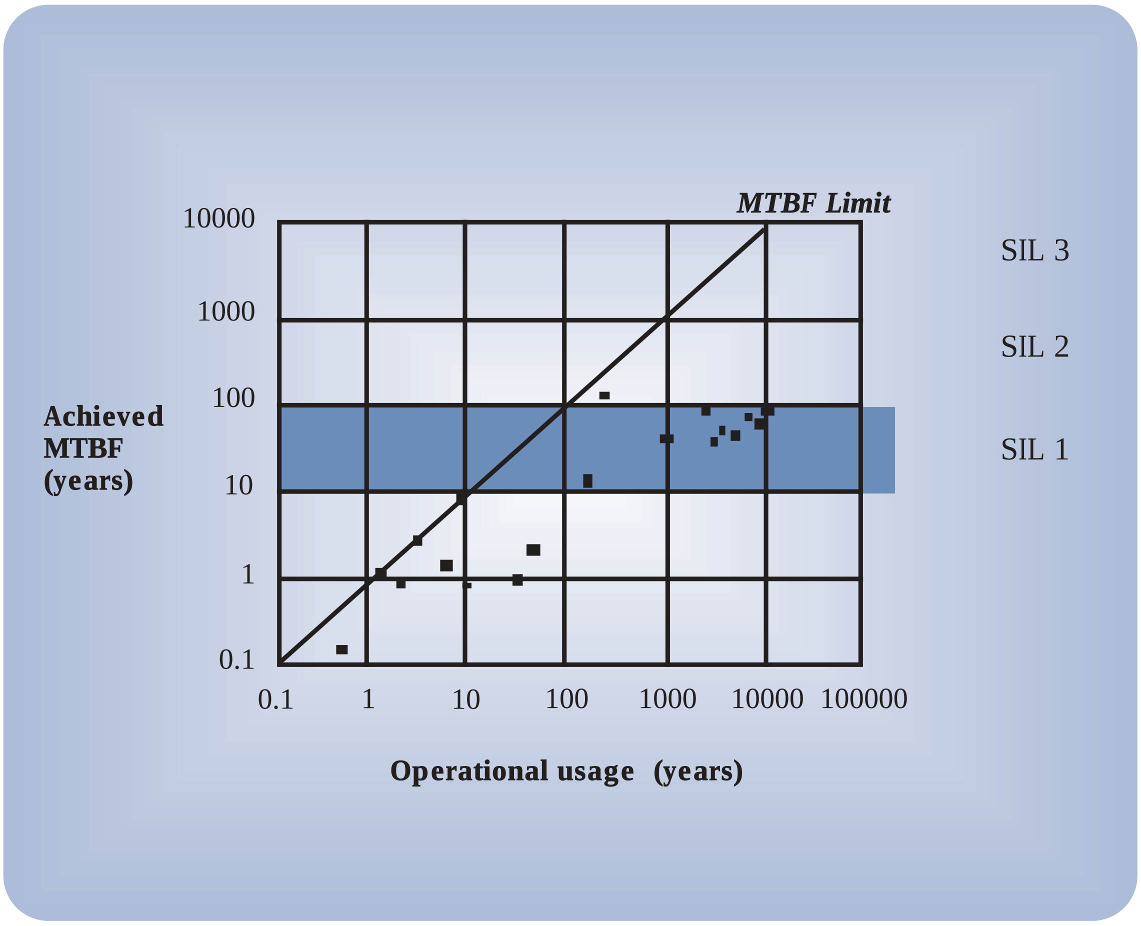
- software modules serving the same function, but previously produced in another system development;
- the run-time software of compiled code;
- components of an operating system, of a data base, graphical user interface, of a communications system, an input-output driver;
- the basic software of a general purpose process control system that can be configured for different applications.

The use of pre-existing software can be attractive because it may reduce development costs. Besides, if it has been properly validated, and if it has been extensively exercised in other systems under operating conditions that can be shown equivalent to the current one, the re-use of pre-existing software can in principle offer considerable reliability assurance. In some organisations COTS software is seen as the answer to the risks and costs of software intensive projects.

However, the key issue associated with the use of pre-existing software in safety-critical applications is whether and how this software can be assessed and demonstrated to meet the dependability requirements of the particular application. Its ability to perform the required functions correctly must be ensured with the same level of confidence as for the custom-developed software, if it is not to be a weak link in the system and in the safety demonstration. The challenges to be met in making this demonstration is illustrated below from data captured in the CEC-sponsored SHIP project [Bishop & Bloomfield 1995]. This shows the mean time between failures (*mtbf*) of industrial and telecommunications software in comparison with IEC 1508 SIL1 (the middle horizontal band).

As can be seen from this figure most systems are below SIL2 and many systems fail to meet SIL1. However, not all failures will be safety related so there is an element of pessimism in these figures as well as an element of under-reporting (only 'large' failures get reported or lead to changes). What is shown clearly is that some industrial systems, even with many years of experience, do not show the reliability growth that might be expected. This is in part due to functionality being inevitably added as the system matures: this new functionality diminishes or cancels out the reliability growth from increased use.

If hazard assessment of the potential use of pre-existing software indicates that it is not safety-critical, and if there is sufficient evidence that it has performed correctly in similar conditions, approval of its use may not require access to its code or to its development process. In certain cases, one might consider approving the use of a restricted, well-defined set of pre-existing simple software functions (e.g. operating system functions) under the sole conditions that they are well identified, are always called upon through the same well-defined and tested interface, have been extensively exercised under similar or simulated operating conditions, and are well isolated in the sense that they cannot in any way be affected by the execution of any other function or part of the system.



COTS software data from nuclear, chemical and aerospace (control and protection) industries. Claimed MTBF is limited to the total usage time. Some systems exhibit growth rates 100 times less than the best case.

More generally, however, if pre-existing software is used in a safety-critical application, and if it was not developed in accordance with the relevant standards, it may be extremely difficult or impossible to establish its adequacy retrospectively. If the system developer does not hold access rights to the pre-existing software, it may be difficult to obtain from the supplier the necessary documentation. Even if the documentation is available, it may be inadequate for *a posteriori* verification and validation. In such cases, the software developer has to create all the necessary specification and the design documentation, and to apply it to all the analysis procedures required by the standard for the new software. A related concern is the quality of the documentation of the interface between the pre-existing software modules and the custom-developed modules. Again the issue arises of what contribution different evidence, or lack of evidence, makes to the overall safety argument. In our view, therefore, it should be investigated whether it is possible give more guidance on the nature and the amount of combined evidence that should be required from pre-existing software, and that could be related to its past usage, its development and its code.

Another key issue is the possibility of maintaining pre-existing software over the required period of time. Unless a contractual guarantee of long-term support is established, adequate vendor support may not be obtained for a particular version. It may also be impossible to influence the form and the quality of further versions that may be needed. Financial and other risks may be involved.

To sum up, the current reliability of industrial software and the extensive measures that would be needed to increase this by several orders of magnitude appear as significant obstacles for the use of COTS software in safety-critical applications, especially safety systems. Issues that should be addressed include:

- what additional assurance activities are there for COTS software and how can their need be assessed (e.g. when to reverse engineer, conduct black box tests);
- how the data reporting and collecting regime can be assessed to ensure the representativeness of the data provided (e.g. defence against under reporting of failures);
- how the evolution of the product and different configurations of the product can be taken account of in exploiting field experience data;
- how to ensure that changes to the software or the application, and differences between the application environment and the proposed one, do not undermine the benefits of COTS software and can be justified;
- whether the reliance of field experience as a form of evidence is too restrictive in the event of future changes to the software;
- whether type approval by a third part organisation is credible and desirable.

10.4.2 Security

Over the past decade much attention has been given to computer and communications security mechanisms and procedures, with numerous initiatives at the national, European and international levels to develop standards and technical solutions. There are however broader problems than just security of information. In the USA a Presidential Commission on Critical Infrastructure has been established following concern that certain national infrastructures (including electrical power systems) are so vital that their incapacity or destruction would have a debilitating impact on the defence or economic security of the United States [DoD 1996][17]. Threats of electronic, radio-frequency, or computer-based attacks on the information or communications components that control critical infrastructures (‘cyber threats’) were identified as an important concern.

Such attacks may be regarded as, in effect, another cause of faulty behaviour by a system through (i) exploitation of a previously unrecognised design fault (i.e. deliberate activation of some pre-existing part of the system in some way that was unanticipated by the system designers), and/or (ii) introduction of a new fault into a system (e.g. surreptitious insertion of so-called ‘Trojan Horse’ code, despite whatever precautions were taken to prevent such insertions). Evidently, if such attacks cannot be prevented from occurring and affecting the system, or their consequences are not protected against by adequate defensive measures in the system design, system failures might be provoked.

The set of engineered provisions which together determine the dependability of a computer system (and the related risk from the parent plant) thus may include a subset of provisions which specifically covers security - and which features accordingly in the plant’s ‘security case’. It will be apparent that the form and content of a security case have much in common with (and indeed may overlap) those of a safety case.

[17] Extensive information about the Commission and its activities are available at its Web site Home Page: http://www.pccip.gov/

Although any specific consideration of how computer systems security is assured has been outside the scope of this report, it is our broad understanding that the security of computer-based safety systems in the UK nuclear industry, and of the computer systems supporting their design, relies (amongst other measures) upon traditional defensive practices, which have been adapted as appropriate to keep pace with, and to capitalise upon, ongoing technological developments. Especially bearing in mind the growing concerns about cyber-threats it is obviously important to be assured that current practices remain fully in tempo with evolving threats and the most up-to-date security advances. Furthermore, given the trend of increasing dependence upon computers (plant modifications and upgrades, etc.), it is equally important to be assured that such security practices are systematically and effectively applied to identify and cover all instances of where computer systems are employed in safety-critical roles within the UK's nuclear plants.

In the face of the potential new types of 'cyber threat' it is important to ensure that appropriate design features, production practices, and operational controls are in place which will be effective in countering such (and similar) threats to the dependable operation of a computer system important to safety on a nuclear power plant. We believe that the HSE should be requested to confirm to NuSAC its satisfaction with the current arrangements for ensuring adequate computer systems security on the UK's nuclear installations.

10.4.3 Tools

The use of appropriate automated tools has the potential to increase software product reliability and the feasibility of establishing the product's fitness for purpose. Tools can assist in the elicitation of requirements, generate code from specifications, reverse engineer the code, automatically check for adherence to rules of construction and standards, generate proper and consistent documentation, support change control, etc.

However, the production of safety-critical software may be adversely affected by the use of tools in several ways. For example, transformation tools may introduce faults by producing corrupted outputs; and verification tools may fail to reveal faults that are already present. In addition, faults introduced by tools may lead to common cause failures.

The level of qualification required from a tool is not easy to determine. Essentially, it will depend on the safety category for which the tool is being used, and on the consequences of an error in the tool. Thus, there is a need for better tools which can cross-check and validate each other, for methods to determine the level of qualification required from a tool, and for a better definition of the evidence required to demonstrate that a tool provides the adequate level of dependability.

There is evidence of the need for more and better tools to support the development of safety-critical software. In particular, there is a shortage of tools which support the real-time aspects of software. Tools are also needed to enable audits to be performed more efficiently so that the frequency of audits can be increased.

There is also the issue of tool support for the safety case itself. The increased trend to more explicit argumentation, and the large amount of documentation associated with a safety case, lend themselves to computer-based support. The ability to link, search, select and annotate documentation could add to the reviewability of the safety case and support the corporate memory of those organisations involved.

10.5 Conclusions

Although computer technology is continually evolving it is hardly a new technology anymore. It would seem a reasonable medium term goal that the production of software safety cases should become much more part of the overall engineering approach and not something that gives rise to undue project or technical risk. The industry should be able to move to a regime where:

- the safety case process is more repeatable and predictable;
- there is a cross-industry and international consensus on the technical bases for judging validity of the safety case;
- the costs and benefits of the different safety case options (e.g. regarding type of evidence) are understood;
- the project risks can be identified and mitigated.

The goal of achieving a more normative approach to computer safety cases must be seen in the context of the continuing changes to the nuclear industry and supply-side structure and the insignificance of the nuclear C&I market to the overall safety-critical market.

The main impediments to achieving these goals have been discussed and are:

- the need to reinterpret the conservative design and engineering principles for the their application to computer systems - in particular the use of claim limits; PRA; diversity and independent assessment;
- the need for a convincing and valid argument to demonstrate that the software is adequately safe.

We make a number of specific recommendations to address these impediments; these are detailed in section 12.

11 Standards and regulatory consensus

In this section we consider two different aspects of international collaboration: regulatory consensus and standards.

11.1 Benefits of regulatory consensus

The very significant cost to the licensee (and also the complementary regulatory effort required) of achieving an acceptable demonstration of fitness for purpose of Sizewell B's computerised primary protection illustrates the level of resource which may be needed in order to demonstrate the acceptability of safety-critical software even against a relatively modest reliability target.

Every aspect of the production process, from the initial requirements specification through to the final on-site commissioning, has to be subjected to an appropriate level of examination and analysis so that the required confidence in the system can be secured.

As discussed earlier, the supplier market for engineering systems of this kind is now very much an international one. Moreover, with the substantial fall-off in the growth of the nuclear industry across the world, the ability of the nuclear sector to command bespoke designs is greatly diminished. Hence, for future new plant and for upgrades to existing plant, the available platforms for safety-critical applications are likely to be those which have been produced for the general market and which, accordingly reflect non-sector-specific production standards - such as the emerging IEC 1508.

It is thus more than ever desirable that the national nuclear regulators attempt to establish, and promulgate widely, a consensus on what they would all expect to see as constituting an acceptable demonstration of fitness for purpose for a computer system in a nuclear power plant safety role (see Appendix D for a description of the extensive regulatory consensus that has been achieved in the civil aircraft industry). By so doing, this establishes an authoritative world-wide reference, which should communicate to suppliers, buyers and safety case writers alike. By so setting the base-line expectation, this should assist the industry to plan and organise ahead, confident in the knowledge of what is required when embarking upon the provision of a system of this kind.

The process of achieving consensus is, itself, a formative and educative exercise, not the least because it provides the opportunity for each national regulator to test in debate the strength of its own particular approach. Experience of this activity has shown that although individual regulators have national constraints, outside their control, on the flexibilities which they can offer, nevertheless it has proved possible to reach commonly acceptable positions even in the areas of greatest potential difference.

There are several good reasons for striving to achieve the widest possible international regulatory consensus on this topic. Most important are the likely progressive infiltration on existing nuclear power plants of software-based technology for up-grades and improvements, and the loss (due to the diminished construction programme) of the nuclear sector's commercial base for influencing the C&I systems' suppliers. Consensus would enable the testing and refining of national ideas within an international debate. It would bring the clarity and authority to the international nuclear industry (licensees and their chosen suppliers) of a multi-national, regulatory statement of the elements needed to achieve an acceptable safety case for a software-based system in a safety-critical nuclear application.

Having once achieved a round of consensus, however, this cannot be regarded as a one-off exercise. Inevitably, the technology as well as the associated nuclear-sector applications will evolve, and commensurably the consensus achieved must be routinely reviewed as an ongoing commitment into the future.

11.2 Standards

In the course of the industrial visits undertaken as part of this work we have investigated the use of standards. In addition we commissioned a review of some twenty-four national and international standards with a view to determining how they might enhance the approaches and practices of the UK nuclear industry when producing and assessing safety-critical, software-based systems requiring high integrity. Particular attention has been given to standards from the aviation industry, the railways industry and the UK defence industry. The review has also included the current draft IEC standard IEC 1508 plus a number of in-house Canadian nuclear industry standards that are in the public domain.

The UK nuclear industry has traditionally supported and followed the International Atomic Energy Agency's (IAEA) Nuclear Safety Standards (NUSS) guides and the International Electrotechnical Commission's (IEC) range of standards. Standards, particularly those from the IEC, are important as they provide an authoritative basis for negotiating an agreed approach among all interested parties for a particular system.

The particular IEC standards specific to the nuclear industry are IEC 880 for the software of safety-systems (i.e. those with the highest integrity requirements), IEC 987 for the hardware and IEC 1226 for the classification of the systems. However the current IEC standard for nuclear reactor software does not reflect the safety case-based approach used in the UK and the application of ALARP, and this has reduced its value in recent applications in the UK and also in France. Also

IEC 880 does not take into account current technologies for building systems nor, most importantly, the fact that most systems to be licensed in the future will be a heterogeneous mix of new software, configured software and previously-developed software. While there are attempts within the standards process to address these shortcomings for the nuclear industry, the international standards process is unlikely to produce timely results of the required technical quality.

The emerging generic standard (IEC 1508) is to be welcomed as it will have a significant impact in raising awareness in the market place but, as with IEC 880, it does not reflect the UK safety case-based approach. However, it is necessary to recognise certain difficulties. Because of the comparatively rapid changes in computer technologies and the ponderous nature of international standards making, standards tend to lag further behind good practice than in other branches of engineering. It is not feasible to expect the international standards process to produce the required results in a timely manner: these criticisms of the standards process and the quality of its products are not new. Means of closing this gap, whilst retaining the powerful normative pressure of standards, should be encouraged. Examples include the use of international ‘agreement groups’ which aim to interpret and augment current standards for use in a specific application.

With all standards there is generally insufficient information or guidance on the criteria for demonstrating compliance: such a demonstration is, of course, an important element of a safety case demonstration. In addition, there is a lack of direction on the production of an acceptable safety demonstration for a system in terms of evidence and reasoning required. Of particular importance in the presentation of this evidence and reasoning is its reviewability: again specific requirements are not available.

Many current systems are based on generic software that is customised for a particular application. Current nuclear standards provide no requirements or guidance for such ‘data driven’ designs. From our review of standards we feel that such requirements and guidance should be produced for the nuclear field: and to this end we recommend that standards such as those of the railway and aviation industries, which treat this topic quite well, should be used as baseline material.

The nuclear industry should be encouraged to use generic standards where possible and the current emerging generic standard (IEC 1508) could be very significant. The development of IEC 1508 and the amendments to IEC 880 should be tracked and representations made to the standards bodies on possible amendments that would provide ‘hooks’ into the UK approach and ensure that the international standards reflect as far as possible UK needs. Guidance should be developed and promulgated based on best safety case practice in UK industry. This could be of benefit to other industrial applications not just the nuclear industry.

As with many markets, the structure of the industrial safety and control equipment market is changing with alliances and amalgamations of vendors. The effect of the single market is encouraging reuse of applications in different sectors, and vendors are seeking certification of products in one application, or by one authority, to be recognised elsewhere. This mutual recognition is based on claims of conformance to standards. This provides both an opportunity for reduced cost (e.g. by spreading the cost of assessment via type approval) and a threat in that present standards are too weak for an assessment of conformance to be taken to mean fitness for purpose.

Allied to the issue of conformance to standards is the issue of how the safety criticality of systems and their components is classified. Currently there are a variety of schemes and it difficult to understand whether a level B avionics software product is equivalent to a safety-related nuclear one or SIL 2 in the generic standard. The assessment of the safety criticality of tools used in the software engineering process has become important as developments become more dependent on tools.

12 Conclusions

In this section we summarise, as brief bullet points, our conclusions from this study. Whilst much of this material applies principally to computer systems, readers will also encounter much that applies more generally to other systems. We include this material not only for completeness, but because we believe that many of these issues apply *a fortiori* to computer systems: for example, the dangers of unwarranted complexity. Similarly, the material here addresses the problems of safety-critical computing quite widely, and we do not restrict ourselves to those issues that are of importance to the nuclear industry: our recommendations concerning these can be found in Section 13.

12.1 Recent practice and how it can be improved

12.1.1 Computer system design and software engineering

- Digital computer systems can bring many advantages through their (hardware) reliability and their potentiality to provide extensive - often novel - functionality (via software). Whilst these advantages are usually obvious in terms of commercial benefits, for example in control and information processing systems, they can also bring benefits of increased safety, not least in diagnostic testing and more extensive safety functions, for example in protection systems.

- Problems can arise in the use of digital computer systems when their discrete nature is accompanied by great complexity. Complexity is a source of error and unreliability. Simplicity of requirements, design and implementation will facilitate successful assessment: the safest design is generally the simplest overall solution for a particular requirement.

- Some of the most important decisions in system design concern the early allocation of responsibilities for different tasks to computers, humans and mechanical and electrical hardware.

- There is a need for design for assurance (i.e. designs amenable to dependability demonstration), particularly at the architecture level of systems.

- The design should take account of the potential for any one design error, no matter how trivial in cause, to result in a catastrophic defect in the product (e.g. by using design fault tolerance).

- Software fault tolerance via design diversity can be an effective means of achieving increased reliability. However, evaluation of the achieved reliability in a particular instance is hard, since independence of version failures cannot be assumed.

- More emphasis should be placed on structuring safety-critical systems so as to achieve a high degree of isolation between the implementation of the most critical requirements (e.g. trip function), and all others.

- Real-time systems, and highly concurrent/distributed systems, are particularly difficult to validate and assess: many current techniques and tools are adequate only for software components that do not have these characteristics. More generally, immaturity in technologies, methods, techniques, tools and human expertise should be regarded as a threat to safety.

- In order to reduce the number of errors introduced, planning and using a well-defined software development process are essential. Errors in the planning, requirements definition and design concept phases are of most concern in managing the software development.

- Quality Assurance (QA) will not compensate for poor design.

- Current attempts to characterise the capability of software development organisations are too immature, and lack a sufficiently firm scientific underpinning, for their use as a predictor of high integrity in safety-critical applications.

12.1.2 Safety cases

- There is evidence from our studies of different industries that the safety cases in one industrial sector in the UK might not be acceptable in another. The reasons for this need to be investigated and understood. One area we have already identified as problematic - perhaps because it is so difficult - is the consistent application of ALARP/SFAIRP to software-based systems. Guidance should be developed on how the ALARP/SFAIRP principles can be applied to computer based systems.

- Steps should be taken to develop and maintain international consensus to ameliorate the problems of licensing systems approved by other national agencies. This issue would become crucial if there were requirements in future to license foreign reactors.

- The existing nuclear design and engineering principles should be interpreted for computer based systems and guidance developed for computer aspects of a safety case. This should capture current best practice and lessons learnt from recent UK and international experiences.

- The current practice of requiring safety cases to have several separate, ideally independent, chains of argument ('legs') is to be welcomed. However, the current regulatory practice allows for many different approaches and interpretation. It would be beneficial to explore these further with the aim of identifying those strategies which are optimal, firstly in terms of safety benefit and then with regard to cost effectiveness.

- The safety case should recognise that, all other things being equal, different types of evidence will have varying value for a given claim:

  - deterministic evidence is usually to be preferred to statistical;

  - quantitative evidence is usually to be preferred to qualitative;

  - direct evidence is usually to be preferred to indirect.

- There is inevitable uncertainty in the failure process of faulty software, so probability-based measures must be used to express its reliability.

- One of the sources of evidence for the safety case should be a measurement of software reliability. This could be via observation of failure-free working during testing, or from actual operational experience.

- There are quite stringent limits to the claims that can be made for the reliability and safety of software-based systems. Reliability goals for such systems must be sufficiently modest to be *achievable* and *assessable.*

- The practice in the UK nuclear industry of limiting claims for non-diverse redundant systems to 10^{-5} *pfd* should continue where identical copies of software are running on redundant channels. In fact the claim may need to be more modest, depending upon the complexity and novelty of the system.

- In some circumstances, a very simply designed back-up system (possibly hard-wired) may be required, since it is only in such cases of extreme simplicity that it may be plausible to claim that the system will be free of some logical design faults.

- The issue of confidence in the accuracy of the risk assessment (i.e. measures may have to be taken that increase confidence but do not necessarily increase the safety of the system) should also be addressed.

- The safety case is a living, cradle-to-grave document. The lifetime costs of different forms of evidence should be evaluated and taken into account (e.g. what is most cost-effective for the initial assessment of a system may not be most cost-effective at later stages of its life).

- Since changes to software are changes in design, and have potentially unlimited effect, it is particularly important that there are mechanisms for demonstrating the maintenance of plant safety through all possible changes in the computing system and/or its environment (the plant it is controlling and/or protecting). This applies in equal measure to so-called recalibration 'data', since these are just another form of software.

- The safety case is a document with a diverse readership, and should be amenable to review by well-informed generalists as well as by specialists. This implies that the safety case should be structured to allow access at different levels of detail and from different viewpoints.

- The safety case should be developed in the context of a well managed quality and safety management system and be executed by competent persons. In order to ensure the effective delivery of the safety case, integrated into the design process and phased along with other project deliverables, it is necessary for early agreement to be secured between licensee and regulator on the composite elements and associated acceptance criteria.

- Recent experience in the UK has been very expensive in terms of the cost per safety function given the rather modest integrity that was accepted for the system. Developments in industry are aiming for much lower (e.g. orders of magnitude lower) costs in the next 5 years and their potential applicability to the nuclear industry should be investigated.

- In building computer systems for safety-critical applications, there are typically components and tools in whose construction the system designers are not involved. The steps taken to compensate, so as to guarantee the requisite safety levels, should be explicitly stated in the appropriate safety case. Guidance should be developed on the use of COTS software both for use in the nuclear industry and to express the industry's concerns and requirements to the wider industry.

- In the face of the potential new types of 'cyber threat' it is important to ensure that appropriate design features, production practices, and operational controls are in place which will be effective in countering such (and similar) threats to the dependable operation of a computer system important to safety on a nuclear power plant. We believe that HSE should be requested to confirm to NuSAC its satisfaction with the current arrangements for ensuring adequate computer systems security on the UK's nuclear installations.

12.1.3 Safety standards

- Current activities aimed at achieving international regulatory consensus for safety-critical computing systems are welcomed and further work in this direction should be encouraged.

- Standards, particularly those from the IEC, are important as they provide a basis for negotiating an agreed approach among all interested parties for a particular system.

- The current emerging generic standard (IEC 1508) and the amendments to IEC 880 should be monitored and representations made to the standards bodies to ensure that the international standards reflect as far as practicable UK needs.
- Given the difficulty of influencing international standards, guidance should be developed that would address the needs of the UK nuclear industry and promulgated as an interpretation of IEC 1508 based on best safety case practice in UK industry.
- An alternative strategy should be developed in case the development of IEC 1508 and additions to IEC 880 are subject to further delays and uncertainties.
- Current standards are not of sufficient technical quality that they can be used as the basis for certification.
- The third party assessment against standards provides the opportunity of reducing both licensing costs and the threat of inappropriate approvals. The claims made of 'approval to IEC 1508' by third party agencies should be carefully monitored and assessed for adequacy.
- Current standards do not reflect the UK safety case-based approach, nor the application of ALARP.
- Current standards do not adequately address the use of previously-developed software, nor technologies such as formal methods and statistical testing.
- There is a need to understand and document the different industrial approaches to safety classification.

12.2 Open topics for research or study

12.2.1 Computer system design and software engineering

- A better understanding is needed of the different strengths and weaknesses of humans, computer systems and mechanical and electrical hardware.
- Further investigation is needed of system architectures which have all the advantages of extensive functionality (including some extra safety advantages) of software, without incurring insurmountable difficulties in ensuring overall system safety.
- Investigation is needed of the adequacy of present techniques for concurrent and distributed systems, to identify areas for further research and provision of tool support.
- More empirical evidence is needed of the efficacy of software engineering techniques, to support 'good practice' in the design and building of critical software-based systems, and to provide measures of the confidence we might place in a system that has been developed using such good practice.
- Better mechanisms are needed for collecting and disseminating field experience of software-based systems.
- Some techniques show particular promise for achieving and/or assessing safety and reliability in critical software-based systems, but would benefit from further investigation. Examples include:
  - fault tolerance via design diversity;
  - probabilistic and statistical techniques for quantitative evaluation of safety and reliability;
  - formal methods for specification and verification of system properties.

- There would be benefit in identifying and classifying in a systematic manner the competencies that should be exhibited by practitioners and organisations working in the area of safety-critical computer systems.
- Greater clarity is needed of the relative contributions of testing and formal proof to the validation and verification process.
- Better software tools are needed, as well as better means of qualifying the tools for use on safety-critical system development.
- Better methods are needed for determining the amount of test coverage needed to meet the required integrity level for a system, and the degree of re-testing (and other measures) needed to restore confidence following the detection and correction of a fault.
- Better methods are needed to identify the modes and assess the consequences of software failures.

12.2.2 Safety cases

- We need a greater understanding of the safety cases for heterogeneous systems that include people, software and other equipment.
- A better understanding is needed of what makes a safety case well-structured, of the properties it should exhibit and of the checks that should be made to ensure that these desirable properties are present. Having identified these, there then arises the question of what methods and techniques can best contribute to presenting and reasoning about the safety case.
- A better understanding is needed of what can be claimed from multiple diverse arguments for the reliability of software-based systems: this needs to address the number of arguments required, the strength of the individual arguments, issues of dependence between arguments, and issues of how to combine disparate evidence.
- There should be an investigation of means of making demonstrably *conservative* claims for software, similar to those that can be made in certain other branches of engineering.
- More work is needed to understand what measures would be needed to make software acceptable beyond the present claim limit of 10^{-4} *pfd*.
- More work is needed to understand the extent and nature of diversity and to address the difficult problem of the degree of independence achieved.
- A better understanding is required of the contribution of commercial off-the-shelf software to a safety case.

13 Recommendations

The Study Group believes that computer systems can bring many advantages in safety and cost reduction to the nuclear industry. There is empirical evidence, after the fact, that their use in safety-critical operations has been successful (although complacency is inappropriate since the amount of operational evidence so obtained is far from sufficient to justify a conclusion that any of these systems will *never* fail). On the other hand, it has often proved difficult and expensive to demonstrate at the time of licensing that software-based systems would be sufficiently dependable in operation, so many of the recommendations that follow concern improvements in ways of establishing safety claims.

Regulatory practice

1 There is evidence from our studies of several different industries that the safety cases in one industrial sector, even just within the UK, might not be acceptable in another. The reasons for this need to be investigated and understood. One area we have already identified as problematic - perhaps because it is so difficult - is the application of ALARP/SFAIRP to software-based systems.

2 There are considerable benefits to be gained from international regulatory consensus concerning the demonstration of fitness for purpose of future safety-critical computer systems. Current activities to this end are welcomed and should continue to be supported.

3 There should be a review of the current organisational and technical approach to independent assessment of computer-based systems in the UK nuclear industry, taking account of the experience of different practices in other industries, e.g. the DER (Designated Engineering Representative) system in operation in the US aircraft industry.

Safety cases

4 Current guidance, based upon appropriate empirical and theoretical support, should be further developed for the structure and content of safety cases for software-based systems. This guidance should address the somewhat different problems of new applications software, previously-developed software (PDS), and commercial off-the-shelf (COTS) software.

5 The design of the system and its safety case should be robust to such changes that are proven to be necessary throughout the anticipated life of the system.

6 The reliability levels that can be demonstrated for software are quite modest, and safety cases should not require levels of software reliability beyond these.

7 Although the use of software design diversity may have positive benefits, claims for statistical independence from diversity should be treated with great caution. In particular, claims for extremely high reliability levels based upon assertions of complete independence of failures in two or more diverse software systems are not believable (with the attendant implications for assessing systems in which one software system backs up another). More work is needed to understand the extent and nature of diversity and to address the difficult problem of evaluating the degree of dependence present in particular cases.

8 Confidence in assessments of software-based systems is usually less than for more conventionally engineered systems. We believe that attention should be given to incorporating formally in licensees' and regulatory guidance a recognition of the importance of the level of confidence that can be placed in assessments of risk within the concept of an 'adequate' safety case. What is needed is to clarify and define the notion of 'adequacy', such that it can be used to guide and justify decisions as to the required extent of activities that will establish the level of confidence that can be placed in a risk assessment.

9 An appropriate method for the formal analysis of competencies should be developed and used as a basic demonstration element within any safety case which aims to justify a high integrity claim for a computer-based safety.

10 In the face of the potential new types of 'cyber threat' it is important to ensure that appropriate design features, production practices, and operational controls are in place which will be effective in countering such (and similar) threats to the dependable operation of a computer system important to safety on a nuclear power plant.

Computer system design and software engineering

11 Attention should be given to better practices for the elicitation and specification of computer system requirements, since it is errors made at this early stage in system life that generally have the most serious impact.

12 Consideration should be given to the problems of safety demonstration from the very earliest stages of system design. Further work is needed on the development of designs for software-based systems (in particular those incorporating COTS software) that make them amenable to safety demonstration. Practical methods for documenting requirements, design and implementation, which allow safety requirements to be traced and verified through these stages, need to receive more attention.

13 Careful consideration should be given to the allocation of responsibilities between computers, humans and (conventionally engineered) hardware, and to interactions between these (particularly the impact of the introduction of computers upon operators). This allocation should be justified in the safety case.

14 One important guiding principle in the design of computer systems should be the avoidance of unnecessary complexity, whether this comes from the provision of unnecessary functionality or from the use of inappropriate structuring in the design. The safety case should contain rigorous justification of the functionality of the system and of its structure. Much more emphasis should be placed on structuring systems so as to achieve a high degree of isolation between the implementation of the most critical requirements, and all others.

15 Some techniques show particular promise for achieving and/or assessing safety and reliability in critical software-based systems, but would benefit from further investigation. Examples include:

- fault tolerance via design diversity;
- probabilistic and statistical techniques for quantitative evaluation of safety and reliability;
- formal methods for specification and verification of system properties.

16 Better tools, together with methods for qualifying them, should be developed to support the development and assurance of high integrity real-time software, including the difficult problems associated with concurrency and distributed systems.

Standards

17 Present standards are not of a form and content that would allow a positive assessment of conformance on its own to confirm fitness for purpose. Whilst standards are important, current ones do not reflect the UK safety case approach and ALARP, nor do they take adequate account of existing technologies and industry trends such as increased use of COTS software. The emerging generic standard, IEC 1508, and the standards for nuclear computer-based systems being developed by TC45A, should be reviewed and efforts made to negotiate amendments which would take account of these deficiencies. The UK nuclear industry and its regulator are urged to maintain active participation generally in the ongoing developments of standards relating to computer-based safety-critical systems.

Research

18 The recent experience of the different organisations involved in those aspects of the Sizewell B licensing that concerned the computerised primary protection system has valuable lessons for similar exercises in the future - both in the nuclear industry and in other industries. This experience should be recorded in as much detail as feasible whilst the collective memory remains fresh, perhaps as a project within the Nuclear Safety Research programme.

19 In our report we have identified several areas where technical progress is needed to maintain and improve our ability to design, build and evaluate safety-critical software-based systems. We are encouraged to see that many of the issues of most relevance to the UK nuclear industry are already being addressed as part of the UK nuclear research programme, and via links to programmes elsewhere. We recommend that NuSAC's Sub-Committee on Research use our report as a template for its on-going review of this part of the research programme.

Appendix A: The UK regulatory approach

A1 Introduction

In general, health and safety at work in the United Kingdom is governed by the Health and Safety At Work etc. Act 1974 [HSWA 1974]. The particular type of regulatory regime which is applied to premises varies with the natures of the business and the risks which are involved. Many premises where work with radioactive substances or generators of ionising radiation is undertaken are regulated through the HSW Act and the Ionising Radiation Regulations [IRR 1985]. However where inventories of nuclear matter are large, the associated operations are subject to a licensing regime. This licensing regime is enacted in the UK through the Nuclear Installations Act 1965 (as amended) [NIA 1965] - a statutory provision of the HSW Act and subsidiary to it. The NI Act stipulates that, with certain exceptions, a licence, granted by the Health and Safety Executive (HSE), is required before a site may be used for the purpose of installing or operating any nuclear facility such as a nuclear power station, a nuclear reprocessing facility or a nuclear fuel manufacturing and isotope production facility. An important section in the NI Act empowers the Executive to attach at any time to the nuclear site licence conditions which they deem to be in the interests of safety. These conditions apply to the design, siting, operation, maintenance and decommissioning of any plant or other installation on, or to be installed on the site.

The Nuclear Installations Inspectorate (NII), part of HSE's Nuclear Safety Directorate, has the responsibility for granting nuclear site licences and attaching the conditions. Licence Conditions provide the NII with the powers to require the licensee to produce documents and have them approved by NII; to require consent from NII before performing certain operations, or starting nominated phases of new plant construction; and to enable NII to direct the licensee to carry out specified actions.

This legislation, however, places the primary responsibility for safety with the licensee. The licensee must ensure that any plant or activity is safe *so far as is reasonably practicable* (SFAIRP) - i.e. where the risk from the plant has been reduced to the point where the cost of further reduction becomes grossly disproportional to the safety benefit achieved. This is done by producing a safety case which demonstrates firstly that the risks involved are tolerable, i.e. below specified limits. It must then demonstrate that all that is reasonably practicable has been done to reduce the risks. The regulator's role, in respect of these licensed sites, is to ensure compliance with the site licence and any other relevant statutes, and to review the safety case and assess its acceptability.

A2 The assessment of a safety case

The design of a nuclear plant is based on design guides, codes of practice, sound safety engineering principles and, where necessary, supporting research and development activities. NII's assessment of the associated safety case is undertaken with reference to a set of Safety Assessment Principles (SAPs) [HSE 1992a]. These Principles do not have a statutory status in the legal sense but NII nevertheless uses them as its yardstick for judging the acceptability of the safety case for a nuclear plant, and for consenting to that plant's operation (submissions which differ from these principles will be considered, however, if supported by a suitable justification). In certain cases these principles require further interpretation, therefore Assessment Guides may support the regulator in assessing the degree of compliance with the SAPs.

The safety case will need to contain an analysis of the safety of the nuclear plant based on the above design. This usually starts with a list of all the known possible initiating faults due to internal and external hazards, and personnel errors, and is followed by details of the safety systems incorporated to protect against those initiating faults whose expected frequency is greater than an agreed figure: these are known as design basis initiating faults. In the case of the

NII's SAPs, this figure is set at 10^{-5} per annum. Fault sequences beyond the design basis also need to be analysed to determine their effects on the plant. This latter analysis should then be used to formulate accident management strategies.

A3 Probabilistic Safety Analysis

Additionally, NII seeks a Probabilistic Safety Analysis (PSA), based on the overall plant design and system of operation, and covering all initiating faults. The PSA is a methodical accident analysis which produces a numerical estimate of the risk from the plant. It provides a comprehensive, systematic and logical analysis of the potential for things to go wrong on the plant, which in turn determines (or confirms) the roles to be played by the safety provisions. It also provides the means of showing that a balanced design has been achieved such that no particular class of accident or feature of the plant makes a disproportional contribution to the overall risk. As well, the PSA provides information on the maintenance and testing requirements for the safety and safety-related systems. Finally, in addition to demonstrating that the risk is lower than the intolerable region (see paragraph 4 of the SAPs), the PSA should also be used, where applicable, to justify that risks are *as low as reasonably practicable* (ALARP), e.g. by investigating the effect on plant risk of modifying the plant safety provisions.

A4 Sensitivity analysis

Despite the difficulty of quantifying the reliability achieved by the computer-based, safety provisions, it is nevertheless possible to make helpful use of the PSA via the facility of sensitivity analysis - i.e. the demonstration of the minimum reliability requirements (in terms of tolerable plant risk goals) for the computer-based, safety and safety-related systems through an investigation of the sensitivity of the plant risk to variations in their reliability.

A5 Categorisation

Whilst the above probabilistic analysis plays a significant role in the safety case, the SAPs require a demonstration that the plant is based on sound engineering principles. These deterministic principles, which lay down proven good practice developed over time, are delineated in the SAPs and, in fact, form the major part of them. One particular principle, P69 (see paragraphs 116 and 131), has relevance in the context of computer-based systems. This principle requires that all structures, systems and components should be allocated a safety category which takes account of the consequence of their potential failure and of the failure frequency requirements placed on them in the safety analysis. Hence all structures, systems and components should be designed, constructed, inspected and tested to the highest standards commensurate with their safety categorisation. This safety categorisation is determined on the following basis:

> Category 1 - any structure, system or component which forms a principal means of ensuring nuclear safety;
>
> Category 2 - any structure, system or component which makes a significant contribution to nuclear safety;
>
> Category 3 - any other structure, system or component.

For a safety category 1 system, conservative design and construction standards are sought together with a strict interpretation of the SAPs in line with the ALARP requirement. This means that the desirable high level of confidence in its claimed dependability can be assured. Most safety systems fall into this category although there can be circumstances (e.g. a secondary safety system, say, with a very modest reliability requirement) where a lesser safety category allocation, and the corresponding less demanding standards of engineering, will be justified. Equally where a safety-related component - for example a reactor pressure vessel - is in an application for which the provision of a safety system is not practicable, this single system, as the principal means of ensuring nuclear safety, may then need to adopt the safety category 1 mantle.

A6 The required safety demonstration for computer-based systems

As stated above, the SAPs require that all computer-based systems in a nuclear installation be allocated a safety categorisation in accordance with the principles set down in the SAPs. The design, construction and testing of these systems should then be in accordance with the assigned category.

However, in order to demonstrate that risks have been reduced so far as is reasonably practicable (SFAIRP), as a rule the safety cases of all computer-based systems associated with a nuclear installation should contain evidence that generally accepted good software engineering practice have been followed. Hence, commensurate with the allocated safety categorisation, all reasonably practicable measures and techniques should have been employed such as will (a) militate against the introduction of errors, (b) aid in their detection and correction, and (c) ensure that any remaining errors are tolerated. Since the effect of such measures on the overall plant risk cannot be evaluated at present in a quantitative way, the criteria for determining whether a measure is reasonably practicable (the ALARP principle) has to be based on accepted good practice and engineering judgement. More generally, the rigor sought in the safety case for any system in a nuclear installation (computer-based or otherwise) will depend on the influence of the reliability of that system on the overall plant risk; the greater the influence, the more robust must be the safety case demonstration. This, of course, will be reflected in the allocated safety categorisation. For the purposes of the next discussion it will be assumed that category 1 equates to safety systems and category 2 equates to safety-related systems since, as was pointed out above, this is generally the case.

A7 Computer-based safety systems

Safety systems rate the highest level of importance since they are designed to detect potentially dangerous fault sequences, and implement appropriate safety actions, that is, they serve to terminate a fault sequence or mitigate its consequence. It follows that the safety case of any computer-based, safety system designated as safety category 1 will need to withstand a most critical examination (comparable for example with that of the Sizewell B computer-based protection system). Such an examination should ensure that the safety case provides an adequate demonstration, through the provision of sufficient[18] evidence and associated reasoning, that:

> the safety system has achieved the level of reliability required to ensure that plant risk is in the tolerable region;
>
> and the probability of the safety system's failure has been reduced so far as is reasonably practicable in relation to overall plant risk.

---

[18] Principle P83 in NII's SAPs explicitly calls for the use of 'conservative design and construction standards' for safety category 1 systems. The general theme of demonstrable safety margin recurs throughout the engineering SAPs. Hence 'sufficient' in this context may be interpreted to mean the amount and quality of evidence and associated reasoning as will provide a high degree of confidence in the achieved reliability. This may be regarded as applicable to safety systems generally.

A robust safety case capable of withstanding this examination thus requires the application of basic safety principles, such as simplicity, redundancy, segregation, diversity and fail-safe design (see Principles 77 to 81 of the NII's SAPs). However, the particular demonstration, by predictive analysis, that the system meets its specified reliability target is not achievable by today's techniques in the case of a software-based design: nor, so far, is any dynamic demonstration of numerical reliability sufficiently proven to be definitive for such quantification. This, then, throws the weight of the safety demonstration onto the qualitative deterministic approach, making the case more difficult to establish for systems requiring particularly high integrity.

NII's approach in such circumstances is to involve the SAPs' Special Case Procedure as the basis of assessment (see paragraph 240). This Procedure is used where a particular plant item, important to safety, has characteristics inherent to its technology/manufacture which render it unamenable to the traditional means of integrity justification. In such cases it becomes necessary for the licensee to establish an alternative, and usually a multi-legged, basis of demonstration. The Special Case Procedure, developed in NII's SAPs for software-based safety systems (see Principle P179), itemises the general aspects which a multi-legged demonstration would be expected to cover. It is clearly for the licensees, aware of the thrust of NII's assessment, to propose and acceptably justify their chosen basis of demonstration.

For its part, within the elements of this Procedure, NII would typically seek acceptable demonstrations with respect to:

- the application of accepted international standards - e.g. IEC 880 [IEC 1986] for software;
- the implementation of an appropriate QA programme and plan;
- the correctness of the requirements specification, and of its subsequent refinement into the detailed design requirements - here the traceability of design features to requirements is considered important and any functionality added during the refinement phase should be reflected in the requirements specification;
- the use of independent verification and validation with a clear demonstration of the traceability of the analyses, tests and inspections to the requirements specification;
- a comprehensive commissioning test programme for the system as an integrated part of the overall plant;
- the establishment of a sound arrangement for the implementation of modifications to the software and hardware;
- the carrying out of a comprehensive independent assessment activity, applied to the final product and devised to mount a level of examination sufficient to show confidently whether or not the required integrity has been delivered (this activity would be expected to involve manual and tool-assisted analysis techniques, review of test coverage and change process adequacy, confirmation of the adequacy of tools - especially those 'in-line', examination of the supporting documentation and a programme of dynamic testing of the integrated hardware/software system using signal trajectories selected from the operational input space).

This set of requirements, quite obviously, represents a heavy burden in terms of safety demonstration.

A8 Computer-based safety-related systems

Where the computer systems are not the principle means of ensuring nuclear safety, e.g. safety category 2 control, surveillance and general data processing systems, the safety demonstration requirements may be less stringent. In each case the contribution to the overall risk needs to be assessed and a safety demonstration commensurate with that risk developed. Such systems should be designed, constructed, inspected and tested to the appropriate national or international standards, commensurate with their particular roles and functions. The use of good quality process control equipment for the majority of duties in this category, where the reliability requirements are very modest, has long been accepted by the UK regulators. However, it has to be recognised that safety-related systems require a higher reliability, albeit not as high as that of the safety system, for certain duties associated with accident management, e.g. the monitoring of a reactor protection system's status or the supporting of the equipment required to achieve and monitor safe shutdown. Such reliability cannot be assumed for general commercial computer-based equipment, hence it may even be necessary to use hardwired controls and indications to achieve the overall plant risk goals. The adequacy of the total accident management provisions has also to be demonstrated in terms of the amount of monitoring provided, and the ability of the computer-based, safety-related system to handle the data rates that might be generated under accident conditions.

An important deterministic principle in any safety case is the demonstration of the separation, both functionally and physically, between protection and control since this reduces the potential for common mode failures. This division is sought since it also enables the safety categorisation of the control aspect to be demoted thus enabling greater resources to be concentrated on the safety demonstration of the safety-category-1, safety systems.

As part of the safety demonstration, the SAPs call for an analysis of the foreseeable ways in which, under fault conditions (including multiple faults), the control systems could place demands on the safety systems. Such failure modes might be assessed, for example, by notionally failing the outputs of the plant in such ways as to represent the worst case scenarios (including unstable behaviour), together with the use of a conservative frequency of occurrence. The safety case would then need to show that the associated safety system is designed to protect against these faults and that the frequency of occurrence of the control system failures, causing protection demands, is not excessive. Additionally, the safety demonstration should contain an analysis of the amount of failed safety-related control and instrumentation equipment that can be tolerated before the plant must be shut down.

Human operator advisory systems and safety support systems again need determinations of their contributions to plant risk, and appropriate levels of safety demonstration. Their roles in the management of accidents require systematic analysis, and demonstration of adequacy since, through incorrect advice, they could increase the severity of the consequences. Obviously, properly designed and validated advisory systems should reduce the frequency of demands on the safety systems. However, it is considered important that the means be identified and provided by which such advice may be checked before the operator takes action on it since erroneous operator action might result in the defeating of correct protective action, or the placing of unusual or more frequent demands on the safety system.

A9 Concluding comments

The foregoing represents a general outline of NII's approach with respect to judging the acceptabilities of computer-based systems proposed for main-line or indirect safety roles on nuclear power plants. It should be emphasised that the nuclear regulatory regime in the UK is essentially non-prescriptive - i.e. the safety performance requirements are clear but there is flexibility for a licensee to propose the means, in each particular instance, of meeting them. Such means, however, must be acceptably justified, with due reference to the legally established principle of 'reasonable practicability'.

Appendix B: Standards

A review of relevant standards was conducted by the HSE for the Study Group, and informed its subsequent discussions. The following is a list of the standards that were considered.

B1 National and international standards

Aerospace

RTCA/DO-178B	*Software considerations for Airborne Systems and Equipment Certification*, Requirements and Technical Concepts for Aeronautics, Washington, DC, 1992.
PSS-05	*ESA Software Engineering Standard*, European Space Agency, 1994, ISBN 0-13-106568-82

Automotive

MISRA	*Development Guidelines for Vehicle Based Software*, the Motor Industry Software Reliability Association, November 1994, ISBN 0 9524156 07.
	Diagnostics and Integrated Vehicle Systems, MISRA Report 1, February 1995.
	Integrity, MISRA Report 2, February 1995.
	Noise, EMC and Real-Time, MISRA Report 3, February 1995.
	Software in Control Systems, MISRA Report 4, February 1995.
	Software metrics, MISRA Report 5, February 1995.
	Verification and Validation, MISRA Report 6, February 1995.
	Subcontracting of Automotive Software, MISRA Report 7, February 1995.
	Human Factors in Software Development, MISRA Report 8, February 1995

Defence[19]

Def Stan 00-55	*The Procurement of Safety-critical Software in Defence Equipment*, Parts 1 & 2, Ministry of Defence, London, 1993.
Def Stan 00-56	*Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment*, Ministry of Defence, London, 1993.

[19] Later versions of the following standards are now available: see [MoD 1996, MoD 1997].

Medical

IEC 601-1-4	*Medical Electronic Equipment, Part 1: General requirements for Safety. 4 Collateral standard: Safety requirements for Programmable Electronic Medical Equipment*, International Electrotechnical Commission, SC62(sec)73, July 1994 (Secretariat draft).

Nuclear

IEC 880	*Software for Computers in the Safety Systems of Nuclear Power Stations*, International Electrotechnical Commission, 1986 (Basic nuclear sector standard).
IEC 880 Supp 1	*Software for Computers Important to Safety for Nuclear Power Plants as a First Supplement to IEC 880*, International Electrotechnical Commission, IEC TC45A(Secretariat)189, July 1994 (Secretariat draft).
IEC 987	*Programmed Digital Computers Important to Safety for Nuclear Power Stations*, International Electrotechnical Commission, 1989 (Basic nuclear sector standard).
IEC 1226	*Nuclear Power Plants - Instrumentation and control systems important for safety - Classification*, International Electrotechnical Commission, 1993 (Basic nuclear sector standard).
ANSI-7-4.3.2	*IEEE Standard criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations*, American National Standards Institute/Institute of Electrical and Electronic Engineers, 1993.
NUREG/CR-4640	*Handbook of Software Quality Assurance Techniques Applicable to the Nuclear Industry*, United States Nuclear Regulatory Commission, August 1987.

Railways

prEN-50126	*Railway Applications - The specification and demonstration of dependability, reliability, availability, maintainability and safety (RAMS)*, European Committee for Electrotechnical Standardisation (CENELEC), TC9X draft, November 1995.
prEN-50128	*Railway Applications - Software for Railway Control and Protection Systems*, European Committee for Electrotechnical Standardisation (CENELEC), Draft No. CLC/SC9XA/WG1(sec)78, February 1994.
prEN-50129	*Railway Applications - Safety-related Electronic Railway Control and Protection Systems*, European Committee for Electrotechnical Standardisation (CENELEC), Draft No. CLC/SC9XA/WG2(sec)79, December 1993.

Other

ISO 9000-1	*Quality Management and Quality Assurance Standards - Part 1 - Guidelines for Selection and Use*, International Standards Organisation, 1994.

ISO 9000-2	*Quality Management and Quality Assurance Standards - Part 2 - Generic Guidelines for the application of ISO 9001, ISO 9002 and ISO 9003*, International Standards Organisation, 1993.
ISO 9000-3	*Quality Management and Quality Assurance Standards - Part 3 - Guidelines for the Application of ISO 9001 to the development, supply and maintenance of Software*, International Standards Organisation, 1995.
BS EN ISO 9001	*Quality systems: Model for quality assurance in design, development, production, installation and servicing*, British Standards Institution, 1994.
UL 1998	*Standard for Safety Related Software*, Underwriters Laboratory, January 1994, ISBN 1-55989-550-0.
Draft IEC 1508	*Functional Safety: Safety-related Systems;*
	Part 1: General requirements;
	Part 2: Requirements for electrical/ electronic/ programmable electronic systems;
	Part 3: Software requirements
	Part 4: Definitions and Abbreviations of Terms:
	Part 5: Guidelines on the application of Part 1;
	Part 6: Guidelines on the application of Parts 2 and 3;
	Part 7: Bibliography of techniques and measures;
	International Electrotechnical Commission, Technical Committee No. 65: Industrial - Process Measurement and Control, Sub-Committee 65A: System Aspects, 1995.

B2 Company Standards

The following company standards in the review were from Atomic Energy of Canada Limited (AECL) and the Canadian utility Ontario Hydro, since these high level standards are effectively in the public domain. In the UK the nuclear industry standards are not in the public domain. The Study Group commissioned from HSE a review in which the reviewer had access to the company standards used by BNFL, Magnox Electric and NE. The Study Group itself had access to the NE approach through [Hughes & Hall 1992] and received a copy of the Magnox Electric guidelines at a late stage.

AECL	*Standard for Software Engineering of Safety-critical Software*, Atomic Energy of Canada Limited/Ontario Hydro standard, Rev 0, December 1990
	Guidelines for Categorisation of Software in Nuclear Power Plant Safety, Control, Monitoring and Testing Systems, COG-95-264 Rev 1 1995.
Ontario Hydro	*Guideline for Computer System Engineering*, Rev 0, No. 907-C-H-69002-0203, April 1993.

Software Engineering of category II Software, Rev 0, No. 907-C-H-69002-0100, May 1993.

Software Engineering of category III Software, Rev 0, No. 907-C-H-69002-0200, May 1993.

Guidelines for the Application of Software Engineering Standards to Configurable Software, Rev 0, No. 907-C-H-69002-0204, April 1993.

Guidelines for the Modification of Existing Software, Rev 0, No. 907-C-H-69002-0205, April 1993.

B3 Additional Material

S. Bhatt, L. Chanel. *Comparison of International Standards for Digital Safety System Verification and Validation*, IAEA specialist meeting on 'Software Engineering in Nuclear Power Plants', Chalk River, Canada, 1992.

D. S. Herrmann. "A methodology for evaluating, comparing and selecting software safety and reliability standards", *Proceedings COMPASS 1995, Washington,* IEEE 1995.

Safety Assessment Principles for Nuclear Plants, HMSO 1992, ISBN 0 11 882043 5.

Tolerability of Risk from Nuclear Power Stations, HMSO 1992, ISBN 0 11 886368 1.

Railway Safety Cases, Railways (Safety Case) Regulations 1994, Guidance on Regulations, HMSO 1994, ISBN 0 7176 0699 0.

IEEE Standards collection, Software Engineering, IEEE 1994, ISBN 1 55937 442 X.

Title 10, Code of Federal Regulations, Part 50, Domestic Licensing of Production and Utilization Facilities, USA.

International Atomic Energy Agency, *Safety Standards: Code on the Safety of Nuclear Power Plants: Design*, Safety Series No. 50-C-D (Rev1), 1988.

Appendix C: The French nuclear industry

C1 Introduction

What follows is a comparison between the French and UK approaches to the use of computer-based systems in nuclear power plants. Four members of the Working Group - P.-J. Courtois, R. Bloomfield, B. Littlewood and N. Wainwright - visited the French 'Institut de Protection et de Sûreté Nucléaire' (IPSN) on June 30th 1996, where discussions were held with Mme Soubies and Mr J.-Y. Henry from the Institute, and with Mr. Féron from the 'Direction de la Sûreté des Installations Nucléaires' (DSIN). We are grateful for the openness of these discussions, which have greatly informed the contents of this Appendix. Readers should note, however, that the views expressed here are solely the responsibility of the authors.

C2 Industrial and regulatory framework

Electricité de France (EDF) is the owner and operator of French nuclear power plants (NPPs) and applies for the licenses. Framatome is the main contractor and designer of the nuclear safety systems. The Schneider company, Merlin-Gerin, is a Framatome subcontractor, and is the designer of the Digital Integrated Protection System SPIN (Système de Protection Intégré Numérique).

EDF has three main departments: R&D, Design, Operations & Distribution. Transfer of responsibilities from the design to the operations & distribution department takes place when the reactor reaches criticality for the first time.

DSIN, the regulator, deals only with the EDF operations & distribution and design departments. DSIN works for the Industry and Environment ministries, and co-ordinates all the procedures to start a new plant. Furthermore, DSIN is entrusted with definition of the regulations and their application to the main permanent nuclear installations. A standing advisory group, the 'Groupe Permanent Réacteurs' (GPR), is consulted by the DSIN on the most important matters (examination of preliminary and final safety analysis reports, major modifications to plants). GPR consists of representatives from EDF, ministries, IPSN and others, named for their personal competencies.

IPSN (Institut de Protection et de Sûreté Nucléaire) is the technical support organisation. It is consulted and responds to requests from DSIN on technical matters and submits reports and analyses to GPR. IPSN may also raise issues in its own right which are then for DSIN, at its discretion, to take forward. There is no legal obligation for DSIN to consult IPSN.

The IPSN was constituted by law in 1976, and is part of, and (for convenience) administered by, the Commissariat à l'Energie Atomique (CEA), but is functionally independent. The IPSN budget is voted by the national assemblies and is separated from the CEA budget. However, there is discussion in France regarding the true extent of the independence of IPSN, so much so that some people think that the Institute should be taken out of CEA, and made an independent public agency.

IPSN is entitled to ask EDF for all documents deemed necessary for its missions. For the SPIN software, IPSN analysed and assessed the documentation, the methods used for specification, design, programming and testing, carried out some structural analysis of selected representative parts of the software to assess its conformance with programming rules, and carried out a software FMEA for some selected critical functions.

The position of IPSN, with its direct funding from parliament which protects it from interference, contrasts with the situation in the UK where the technical support comes from private sector organisations which are acceptable to the nuclear regulator for carrying out specific independent assessment work. This UK *modus vivendi* offers flexibility for securing the most appropriate experts and tools for every single V&V task. The complexity and the specificity of these tasks may indeed require more resources than can be expected from a single nuclear safety technical organisation, whose primary specialism is nuclear safety analysis, not computer technology. IPSN or DSIN may also, if necessary, subcontract to third parties.

On the other hand, when validation work is allocated to private independent organisations, the regulator is obliged to conduct repeated evaluations of suitability and competency. Other considerations include, access to confidential information, understanding and familiarity with the system being validated.

The use of independent V&V is essential for high integrity system development. However, it is difficult to establish the appropriate level of independence between the design and the V&V teams who must themselves have suitable levels of understanding of the system requirements, design, documentation, test tools and oracles.

Independent V&V carried out by an organisation completely separate from the designer is an object of debate in the nuclear sector. The cost-benefit ratio of this practice is perceived differently by nuclear manufacturers, utilities and regulators ([CNRA 1997]). When the independent assessor is the technical support of the nuclear safety regulator, there arises the issue of who qualifies the tools used by this assessor for the V&V tasks; where is the reference basis?

The ‘Designated Engineering Representative’ approach followed by the US aerospace industry (discussed in Appendix D) might be a viable compromise, although there is no *a priori* guarantee that it combines all the advantages, and none of the disadvantages, discussed above.

IPSN assesses EDF quality assurance, verification and validation *plans* at a very early stage of a project, allowing the licensing to proceed more smoothly once those plans are approved. This is also to be contrasted with the UK, where the safety case is presented by the licensee and seems to be much more ‘evolving’ in time.

This does not mean, however, that everything can - or even should - be frozen before the project starts. IPSN is still today, for instance, issuing requests to EDF for complementary documentation and actions to be taken regarding the SPIN development and validation.

C3 French digital protection systems

The first French 1300 MW unit (Paluel) started up in 1984 and was from the beginning equipped with a digital integrated protection system (first generation SPIN), digital logic controls (Controbloc N20) and plant computers used for information processing and alarm filtering. On the new series of plants (1400 MW N4 units), the reactor protection equipment supplied by Framatome/Merlin Gerin comprises mainly three systems:

- The nuclear instrumentation system which provides other equipment with signals from excore nuclear detectors. Nuclear flux measurement equipments are classified 1E. Equipment delivering signals to the control systems and to the core monitoring units are not classified.

- The control rod system which comprises the 1E rod position indication system (protection algorithms make use of rod positions) and the non-classified rod drive controls.

- The reactor protection system (new generation SPIN) which consists of four data acquisition and processing units and two logic output units which generate reactor trip and safeguard actuation signals.

In comparison with the first generation SPIN, the architecture remains similarly based on 2/4 logic: 4 protection channels, 2 output logic protection units, 8 reactor trip breakers. Motorola 32-bit microprocessors are used and programmed in C. Communications between units use local area networks.

C4 Licensing approach

As in the UK, where the licensee presents his safety case, in France EDF produces a provisional safety analysis report describing the planned design together with a demonstration of its safety. The demonstration follows a rather standard pattern, structured into chapters and sections covering well-defined topics and similar to the safety analysis report in the US and other countries. This report is assessed by IPSN.

The demonstration aims at establishing compliance of the system with safety rules (the 'Règles Fondamentales de Sûreté') and is essentially based on two types of arguments:

- compliance with design criteria such as the single failure criterion, and arguments of diversity;
- the construction and design basis which explains which standards, and how applied.

French law requires preliminary, provisional and final versions of the safety report. DSIN asks the GPR advice on the safety of the installation, and IPSN acts as rapporteur to present the case, its conclusions and recommendations to the GPR. To resolve misunderstandings, EDF and IPSN hold meetings before presentation to GPR. In case of disagreement the GPR makes a decision, and the DSIN may decide to follow or not.

C5 Reliability quantification

The French approach is basically deterministic and supplemented by PSAs. PSAs are not requested before licensing: they are used only to gain a global overview of the plant safety and to determine possible weaknesses. Besides, the PSAs do not cover software.

EDF gave IPSN convincing arguments that due to the high quality of the software design and of its V&V process the software installed in NPPs is of a much higher quality than commercial software.

EDF was asked to inform DSIN on the ability and possibilities of quantifying software reliability. They convinced IPSN that, at the present time, no reliability figures for safety software can be meaningfully produced. Research is nevertheless continuing.

A sensitivity analysis was also carried out by EDF. As a reliability figure of 10^{-5} was claimed for hardware, a figure of $2x10^{-5}$ including the software was assumed for computer based systems in this study. EDF was furthermore asked to analyse the sensitivity of the plant risk to this assumption by considering the most important potential failure sequences with an initiating event frequency higher than 10^{-3} which place demands upon the software-based protection system. Results have not been issued yet. This analysis is typical of the use of PSA in countries that strictly follow deterministic safety design and assessment approaches.

C6 Requirements specifications

The issue of how requirements for safety-critical C&I systems are specified and approved was discussed in our meeting. Specifications in natural language accompanied by diagrams - using e.g. AND/OR gates - are usually produced by Framatome. They are used as ‘top level’ software specifications. The bidding manufacturer has to conform to those diagrams. He must re-express what is his understanding of the requirements, and this reformulation is checked against what Framatome originally asked. The bidding manufacturer must also define how he is going to design and to apportion the hardware and software.

As a result of a bid, different solutions may appear. In the case of the N4, Framatome chose the Merlin-Gerin solution, and ‘everything with it’. Framatome re-wrote what they understood from Merlin-Gerin and asked for confirmation in order to close the ‘loop’ in the requirements specification approval process. The documents including the Merlin Gerin solution re-written by Framatome were forwarded to the DSIN.

The manufacturer has to show that his solution complies with safety, and in particular with non-functional requirements (Note: these do not include software reliability figures). The DSIN does not give early approval with regard to requirements and design options, except if explicitly asked to do so.

Non-compliance with the single failure criterion would be a strong concern. This criterion is applied in France to *systems*, including systems containing software; it has not been applied to software alone.

C7 Verification and validation

The 1300MWe SPIN system is considered by DSIN and IPSN as being ‘good’. This judgement is based on three things: operational feedback; the assessment of the design and of the development process; the tests that have been made.

As far as operational feedback is concerned, the system has been working well for many years. It has gone through several versions, but the same architecture and the same design have been maintained (the SPIN system is now in its 9th version since first operation, essentially because changes were necessary in safety margins). The number of actual trips is about one per year per unit, and four times per year with periodic tests which do not provoke actual trips. In addition, spurious trips are also of the order of 0.1 per year per unit. Spurious activations of ESFAS are of the order of 0.2 per year per unit.

It was stated, however, that field experience is only considered as a ‘plus’ in the safety case, not as a substitute for other arguments.

For the N4 series, Hartmann and Braun produced experience from use of Contronic E in Germany (not in nuclear plants) to show that they were able to handle software modifications. This is considered by IPSN as a sign of vendor ability to handle modifications and evolution, but not as a means of supporting software reliability claims.

The non-acceptance of evidence from previous use in non-nuclear systems is often in contrast with the willingness to take into account the general efficacy of the design process. It would be interesting to analyse the reasons which justify this somewhat counter-intuitive position: it seems at least as important to know that a system has worked well in the past, albeit in slightly different circumstances from the ones of interest, as to know that the development process is good. The latter is a judgement which could be valid even if it is based on the experience of the application of this process on completely different systems. One possible explanation is that the Merlin Gerin development process has already been applied on different variants of nuclear protection systems.

On the question of testing, for the 1300 and 1450MWe series IPSN made tests additional to those made by EDF. Complementary tests were made to verify some software self-testing mechanisms because testing of these facilities had not been properly covered. IPSN also carried out for the N4 series some software FMEA analysis to analyse the propagation of certain failure modes, establish the criticality of certain failures and study their impact on the most important protection functions.

However, neither statistically representative dynamic testing as in Sizewell B, nor quantitative estimation of the test coverage, were made. Nonetheless, for the N4 series an analysis of the manufacturer's test coverage of the set of software functional paths was carried out on part of the SPIN software. This was not intended as a basis for further statistical analysis but as part of the determinisitc demonstration.

C8 Tools used for the N4 series software assessment

The SAGA tool [Bergerand & Pilaud 1989], based on the LUSTRE language [Caspi *et al.* 1987], is an industrial tool which has been used for some time by Merlin-Gerin for software development.

Two facilities provided by SAGA are of concern to IPSN:

- *Validation facilities:* Certain checks, like type checking, are automatically done by the tool. This is used as an argument by Merlin-Gerin not to have these checks repeated by the V&V teams. Merlin-Gerin had to provide evidence that these facilities were indeed efficient and sufficient.

- *Automatic Code Generation*: SAGA produces some code (C) automatically. Originally problems in the tool were found by checking against the code (about 100 lines each) - 250 'views' were independently generated. These problems have been corrected and Merlin-Gerin now argue that their feedback of two years experience with non-nuclear systems have stabilised the tool. Integrating manually produced code with code produced by SAGA did not show any problem. IPSN is not entirely satisfied with this situation, but did not find errors in the samples of the code they reviewed. IPSN uses the CLARE tool as a cross-check.

Furthermore, the static analysis tool MALPAS was used by IPSN to analyse certain characteristics (control and data flow) of a part of the C code manually produced (in the same way as the LOGISCOPE tool could be used). MALPAS was also used to focus attention on some parts of the C code and to translate them into the MALPAS tool's Intermediate Language (IL). It was also used to verify the C code of one simple functional sequence (the pressuriser level), to produce back (reverse engineer) the original equations and to compare them with the specifications entered into SAGA.

MALPAS was certainly not used by IPSN as extensively as by NE on Sizewell B PPS. There may be several practical reasons for this, one of them being the considerable investment of effort which large-scale use entails. Certainly it does not automatically imply that NII may have asked for too much MALPAS analysis.

C9 Common mode failures and diversity

Two diverse (hardware and software) computerised systems are implemented in French NPP N4 units: The SPIN in charge of the reactor trip and ESFAS (Engineered Safety Features Actuation System) functions, and the Contronic E system in charge of most C&I unclassified or 2E classified functions.

In 1982, during the run-up to the 'first criticality' for Paluel, the PSA had confirmed that sensors and actuators were failing at a frequency higher than required by the reliability of the scram function (10^{-5} per demand). It was decided to have a back-up secondary system with specific sensors (related to steam generator water levels) and using specific tripping logic and actuators. In case of absence of trip, this system would be in charge of the more likely incidents - namely those of category II whose occurrence frequency is in the range of 1 to 10^{-2} per year.

EDF proposed to use the PWR 1300 Controbloc, which is of a different technology. While the SPIN is based on Motorola 6800 microprocessors, the Controbloc system used wired logic controllers.

Later, for the N4 1400 units, EDF proposed new instrumentation principles but also to repeat the same approach for the SPIN back-up. This was agreed by DSIN. EDF proposed the Controbloc P 20 system from Cegelec. The software of this system - because of its complexity - could not be successfully developed and was abandoned by EDF. The existing Hartmann and Braun Contronic E control system was chosen as a replacement. The H&B Contronic E had not been qualified for 1E functions and had not been previously used in nuclear plants; hence, it could not be claimed to sustain the same level of reliability as the SPIN.

The SPIN system software is designed 'to ensure a high degree of quality and a virtual absence of defects' (F. Féron, 'Overview of N4 series C&I system', [CNRA 1997]). However, a common mode failure of this software is postulated and a diverse system was felt necessary. The diverse system (ATWS: anticipated transient without scram) is provided by separate equipment, the SCAT system, using the Contronic E technology. This palliative system, designed by EDF, acts as a diverse system of the SPIN system for postulated initiating events of class II (frequency higher than 10^{-2}). While the US NRC does not require the ATWS system to be safety graded, France does.

For classes III and IV postulated initiating events (frequency of at most 10^{-2} and 10^{-4} respectively), the SPIN alone is in charge. This is summarised in the table below. One could conclude from this table that the 'implied *pfd*' of the safety subsystems in charge is the reliability claimed from the combination of these subsystems to achieve an unconditional *pfd* of 10^{-6}, considering the median value of the PIE frequency range.

Classes of Postulated Initiating Events (PIE)	**PIE Frequency (per year)**	**Safety Subsystems in Charge (implied *pfd*)**
Class I	Normal Conditions	_
Class II	10^{-2} to 1	SPIN or ATWS (10^{-5})
Class III	10^{-4} to 10^{-2}	SPIN alone (10^{-3})
Class IV	10^{-6} to 10^{-4}	SPIN alone

C10 Deterministic versus probabilistic safety evaluation

Although the PIE (Postulated Initiating Event) classification is widely accepted in the nuclear industry, its interpretation and use for design and safety demonstration purposes may differ between countries depending on whether probabilistic safety goals are being adopted or not. Besides, these differences are not always clearly understood.

The PIEs are classified according to their anticipated frequencies of occurrence. These frequencies are estimated from operational experience data from similar plant designs across the world. The PIEs correspond to single well-defined types of initiating faults (e.g. a pipe break). To each class are associated upper limits on the radiological consequences of any *single* PIE of the class. The basic principle is that the most frequent occurrences must yield little or no radiological risk.

In most countries (e.g. France, Germany, Belgium), the PIE categories are used deterministically. For *every* postulated initiating fault, including the failures directly caused by the PIE, a worst single failure is postulated in the safety systems, together with other conservative assumptions. It is then required to demonstrate that following this fault sequence no release can occur beyond the limits accepted for the class to which the PIE belongs. These analyses are done for *every* PIE. They do not require probabilistic safety arguments. They do not make assumptions on PIE occurrence frequencies.

This approach is different from an approach based upon an accepted probability requirement (as in [HSE 1992a] p44, or in the Netherlands) that may use the same PIE categories. The probability requirement may be that the *total* anticipated frequency of *accidents* that could give rise to a large uncontrolled release should have a frequency less than say 10^{-7} per year, or may impose upper limits on the *total* anticipated frequencies of *accidents* which could deliver predetermined dose ranges to persons outside the site ([HSE 1992a], p42). Then, the PIE classification could be used to impose upper limits on the rate of failure per demand of the equipments and systems in charge of controlling and mitigating the consequences of the initiating events. For example, in the situation discussed above, such probabilistic requirements *could* imply - as indicated in the table - that the SPIN should have a *pfd* less than approximately 10^{-3}. Such requirements should result from PSAs based on PIE anticipated frequencies and fault sequence frequencies that are combined into *total* accident frequencies.

C11 Concluding comments

French nuclear safety analysis is based on deterministic arguments, which are similarly applied to their digital systems. However, as explained above, the deterministic versus probabilistic dichotomy is probably not by itself the best ground to analyse the differences between the French and the UK safety approach, since both countries associate large dose releases with small frequencies.

More significant is probably the fact that many countries consider that, taking into account the uncertainties in the evaluation of the occurrence probabilities of rare events and in the evaluation of the reliability of systems designed to have a very low failure rate, it is extremely difficult - if not impossible - to set global risk criteria that have a practical and industrial meaning. Thus, distinct uses and interpretations are made of the notions of single initiating event and of maximum release associated to its class, on the one hand, and of maximum *total* predicted frequency of accidents giving rise to a maximum release on the other hand.

In particular, certain countries (e.g. France, Belgium) have no *legal* basis for *nuclear quantitative safety goals*, and in particular no *legal* basis for radiation dose limits (the figures used in France have been proposed by EDF and endorsed in the safety assessment by IPSN/DSIN). They hold the view that quantitative safety goals should only be used together with a detailed specification of the method to be followed to show compliance. As no PSA methodology has yet been standardised, and considering the intrinsic limitations of PSA approaches, they find it adverse to safety to reduce assessment just to PSA calculations.

It is also interesting to put this discussion in perspective with the so called ‘non-prescriptive’ regime of certain regulators - NII for instance - who state (cf. appendix A of this report) that: ‘... the safety performance requirements are clear but there is flexibility for a licensee to propose the means, in each particular instance of meeting them. Such means must be acceptably justified, with due reference to the legally established principle of “reasonable practicability”’. This position does not specify how to proceed when no consensus yet exists on the means or methods to follow, and would therefore be a potential source of difficulties for any licensing approach based on the quantitative safety goals discussed above.

Although the approach to demonstrating nuclear safety in the UK, unlike in France, makes formal use of PSA as a supplement to the safety case, nevertheless the deterministic basis of the case is comparable in both countries. Furthermore, both countries apply independent assessment (albeit in different ways) where a computer safety system is depended upon to protect against plant faults of the most frequent kind.

Appendix D: The US aircraft industry

This note is based upon visits by some members of the Study Group to Boeing Commercial Airplane Group and the FAA in the US, and to GEC-Marconi Avionics in the UK.

D1 Overview of the aircraft industry and certification

The airline and aircraft industry is necessarily international. Not only do airlines operate across international boundaries, they purchase aircraft and equipment from world-wide suppliers. Major new aircraft developments are also multi-national projects. Clearly, the industry could not operate without a consensus - international traffic requires that an aircraft which is considered operational in one country should also be allowed to operate in other countries. In practice this requires both high level and detailed discussion between regulators in different countries and consensus guidelines such as RTCA/DO-178B[20] [RTCA 1992] (see later).

Each country has an established legal framework for regulation of the industry, which includes an authority, such as the Civil Aviation Authority (CAA) in the UK and the Federal Aviation Administration (FAA) in the USA. The airframe supplier, such as Boeing, is responsible for ensuring that the aircraft as a whole is airworthy. This is achieved by means of a certification process, during which the supplier presents evidence for airworthiness to the authority. Note that certification of the aircraft by the authority does not imply that the authority is in any way responsible for the safety of the aircraft. That responsibility remains with the aircraft supplier and operator. The main role of the authority is to ensure that the companies involved have the right culture, procedures and processes so that they will be capable of producing and operating aircraft which satisfy the airworthiness requirements.

There are some differences between the various authorities and their approach to regulation and certification. The European aviation authorities together form the Joint Aviation Authorities (JAA), which issue requirements in the form of JARs. These requirements are adopted under national laws of the JAA member states to become their airworthiness codes. The FAA produces equivalent regulations (FARs) which differ slightly in that they are in themselves legal requirements.

The JAA approach to certification is via a series of reviews of the evidence. The investigation for certification of an aircraft type is performed by a multi-national team following JAA agreed procedures. The findings and recommendations will be put to the member states with a recommendation, where appropriate, for certification by each of the members. The FAA mainly works through the DER (Designated Engineering Representative) system, which is described later. Other nations follow either the JAA or the FAA approach, but it can be necessary for suppliers to consider cultural differences in attempting certification elsewhere. It is also possible for mutually exclusive certification conditions to be imposed i.e. for an authority in one country to require a particular piece of equipment to be fitted to the aircraft of its national airlines, while another authority will not permit the same piece of equipment to be fitted. In order to manage such differences, the other authorities will take an interest in the certification process of the first of a new model and they may, at the beginning of the certification process, impose the certification conditions which would be required within their jurisdiction. In the remainder of this discussion we will concentrate on certification under the FAA.

The FAA operates under a public charter and is answerable to Congress. It is responsible both for promoting safety and for promoting the aviation industry. The wording of the charter is that it is

[20] The European version of this is EUROCAE ED 12B, which is technically identical: the documents were produced by a joint EUROCAE/RTCA committee in a co-operative process.

responsible for promoting the 'highest public perception of confidence in safety'. This dual role contrasts with the ALARP principle applied in the UK in some other industries. However, if there were to be problems with safety in the US aircraft industry, this would affect public confidence and hence damage the industry. On the other hand, this dichotomy of interest will tend to increase pressure for rapid conclusions to any investigations into safety.

D2 The Boeing 777 Primary Flight Control System

Much of our discussion with representatives of Boeing centred upon the software for the PFC of the 777; accordingly, much of the discussion here will use this as a case study.

Before describing this system briefly, it is worth remarking that this avionics application differs markedly from safety systems, such as nuclear reactor protection systems. Most importantly, this is a critical *control* system, and there is no equivalent to the rigorous separation of protection and control that we see in the UK nuclear industry. For an aircraft, the only safe state is stationary on the ground, and a control system, which must operate continuously, is required at all other times. It is claimed that in current aircraft one could lose most of the functionality of the control systems in straight and level flight and not lose the aircraft, provided that the control systems can be restarted successfully. However, such arguments are not generally relied on as part of the certification process and in any case, there are critical parts of the flight, such as take-off and landing, where such margins are not available.

This 777 PFC was developed for Boeing by GEC-Marconi Avionics Limited in Rochester in the UK, but close relationships between the two companies were maintained as part of the Boeing philosophy of 'working together'. The formal programme for the 777 began in 1990, but there was a preceding prototype programme (called the 7-7) which began in 1981. The first flight of a 777 was on June 12, 1994, with the certification being completed in April 1995 and the first 777 entering service in May 1995.

The Primary Flight Control System is responsible for the control of the primary control surfaces of the aircraft - the flaperon, spoilers (7 per side) and outboard aileron on the wings and the stabilisers, elevators and rudder at the tail.

The system is triplicated for robustness reasons, with each channel using two sets of buses. One bus is used for critical data (the flight control bus); the other bus (the systems bus) serves the Airplane Information Management System (AIMS).

Flight crew commands are routed to the PFC via Actuator Control Electronics (ACE) units. The ACEs convert the analogue signals from transducers on the flight controls (column, wheel and rudder pedals) into digital signals on the databuses. The PFC uses these digital signals and in turn transmits control commands back to the ACEs. These control commands are used to drive the control surfaces, together with actuators on the flight controls which provide feedback to the pilot.

This digital control system permits additional functionality to be provided, compared with a conventional system, including:

- Bank angle protection;
- Turn compensation;
- Stall and overspeed protection;
- Pitch control and stability augmentation;

- Thrust asymmetry compensation (in the event of one engine failure);
- System health monitoring;
- Aircraft maintenance support;

The system has fall back modes of operation. Secondary mode is invoked if primary sensor data is lost. In this mode, the PFCs supply commands to the ACEs but envelope protection and autopilot are not available. Should there ever be a total failure of all three channels, loss of all bus data or the pilot invoke the PFC disconnect, a third direct mode is automatically invoked. In this mode, analogue control commands are routed directly from the cockpit controls via the ACEs to the control surfaces. This mode is not part of normal operating procedures, nor is a further degraded mode which uses an independent mechanical path to the stabiliser trim and one pair of spoilers.

The system is not configurable. Modifications would be required for anything that changes the control laws (for example increased gross weight). It is intended to adapt the system for the 747-X (the next generation of the 747), but software and hardware modifications will be required as the 747X has more control surfaces.

The development of the PFC required considerable effort (over a hundred people for the major part of the programme, the majority of whom were involved in software development and assessment). It would be hard to estimate the functionality from the control requirements as the engineers estimate that only about 20% of the software implements the control laws - the rest is concerned with redundancy management and ground maintenance. Overall the PFCs contain about 2500 modules of Ada containing about 250K lines of code. There are also about 20 lane-specific modules, some of which include assembler code. The software for each lane takes between 1 and 2 Megabytes.

D3 The certification approach: RTCA/DO-178B 'Software Considerations in Airborne Systems and Equipment'

We shall concentrate here upon the main avionics software standard, RTCA/DO-178B [RTCA 1992][21]. However, it should be noted that other systems and hardware standards form the context in which this is operated. These standards are not mandatory, but are recommended practice and represent an international consensus of the avionics industry. This means that all of the suppliers in even a long supply chain will have a commonality of approach, although they may well differ in detailed application. If the supplier is unable to comply with the standards in some respect then it will be necessary to propose some alternate means of compliance and to demonstrate to the certification authority that this alternate means is at least as good, for this particular case, as the method required by the standard.

DO-178B represents a consensus view of an approach that produces safe systems and is reasonably practical. The FAA took part in the development of the standard and have agreed that this process will produce software suitable for certification as part of avionics equipment and systems. There is, of course, no independent evidence that confirms whether the objectives set out in DO-178B are necessary and sufficient to produce airworthy software, other than the experience of the practitioners. However, there are plans to collect data and experience on using the standard.

During the certification process, the FAA will verify that the supplier has a process that results in software that complies with DO-178B. The FAA does not merely verify that the software complies

[21] The Boeing 777 PFC was subject to the earlier guidelines of DO-178A, because of the date of application, but many of the principles of DO-178B were applied.

with DO-178B. If the FAA finds a fault in the software they would regard it as indicative of a class of faults because of a weakness in the process and expect the loophole to be closed, not just the particular fault to be fixed.

DO-178B categorises software into levels A to D according to the severity of the consequence of a software failure on the airworthiness of the aircraft. Level A is the most onerous in certification requirements and relates to software whose failure could result in the loss of the aircraft. The amount of testing, configuration control (including change control and traceability) and reviewing required, as well as the degree of independence needed for the verification activities, varies across the levels.

DO-178B states *what* should be achieved, but leaves the choice of *how* to do this to the implementors. The requirements of the standard are summarised in a table of over 60 objectives. About 60% of the standard relates to *process* attributes, for example: quality assurance, configuration control, documentation, planning, design. The requirements for Level A software in this area, although they would appear very onerous to an organisation not used to developing critical software, are similar to the sorts of requirements in safety-critical standards in other industries. The remainder of the objectives relate more directly to the *product* of the development, for example: the extent of the testing including structural coverage and robustness testing.

DO-178B does not mandate or even recommend formal methods, although implementors could offer them under alternate methods of compliance. They may be included in a future version, although to be included, there would first need to be experience of using them, and their procedures and tools, within the avionics industry. DO-178B is intended to be a compromise between absolute proof of correctness (which no-one can provide) and every case being left to individual judgement. While no-one believes that exhaustive testing of such software systems is possible, the process required by DO-178B means that the functionality of the software has been extensively demonstrated.

In theory, DO-178B defines a process, which if followed will produce certifiable software. In practice, suppliers need to interpret it and to provide their own judgement in how they will apply it, and whether the letter of DO-178B is sufficient. Avionics suppliers therefore discuss and agree their proposed approach with the customer and FAA throughout the development process.

The key certification documents are the Certification Plan (this is roughly equivalent to a software safety plan) and the Accomplishment Summary (equivalent to a software safety case). These refer to other documents and configuration items which are produced as part of the development process. The accomplishment summary will include justification of the alternate means of compliance, which will include deviations from the planned approaches and any necessary work-arounds.

Traceability back to the specification is considered one of the major strengths of the RTCA/DO-178B approach. Detailed traceability is applied at all stages of the lifecycle, from specification, through design, to code. Where the customer's requirement is abstract (e.g. for redundancy management), the supplier traces the abstract requirement through to 'derived requirements' (i.e. the strategy for achieving the customer's requirement). All testing also has detailed traceability back to the specifications.

D4 The certification process: the Designated Engineering Representative (DER)

The FAA does not have the resources to perform the detailed compliance checking which it regards as necessary for certification. It therefore uses DERs (Designated Engineering Representatives). DERs are employed by the large companies such as Boeing which require certification, or they are self-employed consultants who are hired by companies (usually smaller) that do not have their own company DERs. However, each DER is individually approved and appointed by the FAA and has a direct reporting line to the FAA, via one of the six regional Aircraft Certification Offices (ACOs).

The criteria for appointment as a DER are defined in [FAA 1995]. The selection by the FAA is by CV and interview based on experience, ability and personal characteristics (communication skills and professional integrity are requirements for selection). Eight years of relevant engineering experience is required, but four of these may be as part of a University degree course. In the case of company DERs, the FAA also has to approve the company which employs the DER. DERs are monitored annually and must conduct some activity as a DER each year to maintain their appointment.

DERs are not allowed to interpret policy, for example to make decisions on new technology. Thus issues which are not explicitly covered by DO-178B, and precedents such as the use of software diversity, would not be left to DERs. This results in three types of project:

- FAA project, where the FAA does the review alone;
- DER project, where the DER approves the certification evidence with no further FAA review (In practice, in almost all projects, the FAA reviews the certification plan and the accomplishment summary and hence confirms that there are no policy decisions required);
- DER assisted project in which the FAA reserves the right to approve the certification evidence (In practice the FAA will approve the certification plan and the accomplishment summary and will mandate the evidence required for new policy areas; the DER will be entrusted to obtain the evidence and ensure compliance).

In the case of the Boeing 777 PFC there was a policy in place for everything except for the original proposal for dissimilar software, or dissimilar processors as it became in the final implementation. In fact, the FAA at present has no policy for assuring processors (there is a committee working on it). Therefore DERs cannot make decisions on this.

Note that FAA review usually involves one FAA officer who is assigned to the job. In general on novel or difficult areas of interpretation other FAA or external experts would be consulted. There are presently no criteria to state the competence or otherwise of an FAA reviewer to approve software documentation.

In the case of the PFC, there were four DERs involved. Their role was to ensure compliance with DO-178B, so as to be able to recommend certification to the FAA. Typical activities included carrying out walkthroughs, running random tests, following random paths through the supporting documentation, etc. However, it was reported that very little formal documentation was submitted by the DERs to the FAA.

The activities of the DER are varied. For example, the DER will take a view about the general planning of a project, e.g. that resources are inadequate in the widest sense. If these are deficient then the DER can take appropriate action. With software in particular there is a recognition that many issues need to be addressed at an early stage of the life-cycle.

The DER will not be prescriptive unless a compliance issue is involved. If the DER does not fully understand some technique this can lead to a need to attend courses or for other such action. When new methods are suggested by the implementation team, the DER tends to use criteria such as: are the methods feasible, are they well understood, are they well designed, what are the management procedures to deal with them, is the tool environment mature? If there are any deficiencies in these areas, this can impact safety.

This DER procedure has been in place for many years, and experience of it by both Boeing and the FAA has generally been a good one, and the system has passed three government reviews.

There seem to be several advantages in the procedure. The legal status of the DERs, and their direct reporting line to the regulator, the FAA, is very important - it contrasts strongly with an internal independent design assessment team reporting to internal company management. Being familiar with company practices and procedures, and usually being present from project inception, the DERs can be expected to have a greater knowledge of what is really going on than an outsider.

Possible disadvantages may lie in threats to independence. These are not likely to be overt - it is unlikely that companies would put direct pressure on DERs to respond to project needs - but may arise from possible divided loyalties on the parts of the individuals themselves. However, it should be noted that DERs are not supposed to have project management responsibilities in addition to their DER role, thus lessening the chance of a conflict of interest. Furthermore, DERs appear to have high status in their companies (in the case of the 777 PFC, the Chief Designer, Jim McWha, had himself been a DER), and are respected for their technical abilities. The FAA conducts continuous monitoring of DERs, and appointments come up for renewal each year; there have been cases (in other companies) where this has resulted in termination of DER status.

Of course, the very intimacy of the involvement that the DERs have with a project may preclude their having a fresh viewpoint, as an outsider might be expected to have. In our view, this risk seems to be outweighed by the advantages of their greater knowledge of the project - although there is no reason why a further layer of external independent review could not be added to the DER system to strengthen it (the FAA themselves already play this role to some degree).

D5 Reliability quantification

Reliability targets for whole aircraft are determined such that the reliability of a new craft will not be less than those already in service. As the overall accident rate for commercial aircraft is of the order of 10^{-6} per flying hour, this is set as the industry target. To place these figures in context, the overall human mortality rate in the West is about 1.6×10^{-6} per hour. Newer, large civil transport aircraft are in fact now achieving significantly better than this - typically 3.0×10^{-7} per hour. The Certification Authorities (FAA, JAA), after consultation with the industry, decreed that system failures should contribute no more than 10% of the accident rate and therefore the probability for catastrophic failures due to systems failure shall be less than 10^{-7} per hour. With something of the order of one hundred critical systems on an aircraft, this implies that the contribution to the total (catastrophic) failure rate of any single system shall be less than 10^{-9} per hour.

These targets are also subject to the single failure criterion, that “No single failure, however remote its probability of occurrence, can be allowed to have a catastrophic effect”. Such failure conditions must either be designed out of the system or be protected by redundancy so that multiple faults are necessary to cause a catastrophic effect.

There is an awareness within the aircraft industry that public confidence in flying could be damaged if the absolute number of accidents reported were to increase each year. Therefore, as the number of flying hours of commercial aircraft increases each year, the targets are gradually becoming even more onerous.

Clearly, achieving a failure rate of 10^{-9} per hour for any system is very difficult; demonstrating its achievement in a particular case is probably much harder. Even for conventionally engineered, highly redundant systems, it is hard to justify claims for complete statistical independence between failures, and it is for this reason that some industries, including the UK nuclear industry, place a limit upon what can be claimed for them. The problem is even more acute in the case of software-based systems, for the reasons that are discussed in the main body of this report: i.e. simple replicative redundancy will provide no protection from design faults, and there is strong evidence that design diversity does not succeed in providing complete independence in the failure behaviours of different versions of a program.

Nevertheless, there does seem to be a requirement that critical software-based systems, such as the 777 PFC, have a failure rate better than 10^{-9} per hour. Since the software forms a single point of failure in such a system, the reliability requirement for the software must be even more stringent.

There seems to be an ongoing debate in the aircraft industry on the issue of software reliability quantification, as there is in other industries. There is general agreement, expressed in DO-178B for example, that a level of 10^{-9} per hour cannot be measured in any scientifically convincing way. This leads some to argue that the requirement should be a zero failure rate for software, and that verification and validation procedures should aim to provide a deterministic (non-statistical) argument to support such a claim. We understood this to be the position of the Boeing representatives to whom we spoke.

The position in the aircraft industry with respect to reliability quantification for software-based systems contrasts with that in the UK nuclear industry. There the required reliability levels have been more modest - of the order 10^{-3} to 10^{-4} probability of failure on demand - and are amenable to direct measurement via statistically representative testing [May et al. 1995], as well as to indirect qualitative arguments.

D6 Design diversity and fault tolerance

The use of software diversity within the PFC was considered in the early stages of its development, but it was finally decided to build only a single version. There were several reasons for this decision.

In the first place, the academic studies of the efficacy of software diversity seem to have had a great influence on the whole industry. There is a perception that the increased complexity of a diverse architecture may not be counterbalanced by a sufficiently large increase in reliability over what could be attained from a single version. This problem of complexity may be more serious in aircraft control than it is in other applications because of the absence of a safe state: there is no equivalent of a simple trip command in the event of disagreement between version outputs.

On a more practical level, it was felt that the management problems posed by a three-fold development of diverse versions were likely to be extremely hard to control. Identifying three independent teams of developers, and three teams of verifiers, and then controlling the whole process so that they produced versions that were dissimilar yet had the same functionality, was regarded as a daunting task. There was a suggestion that in earlier applications of diverse software in the industry, the degree of diversity actually achieved may have been quite low: a figure of up to 80% commonality between ‘dissimilar’ versions was mentioned.

Finally, there was a view that the maintenance of dissimilar source code could be prohibitively expensive.

Whilst Boeing decided against the use of complete dissimilar software channels for the 777 PFC, dissimilarity was used in other areas. For example, the three lanes in each channel of the PFC use dissimilar processors and different compilers. Interestingly, Boeing found about 10 compiler problems in each case - i.e. instances where the compiler did the wrong thing and thus may have introduced a fault into the executable code - but there appeared to be no commonalities between these.

D7 Formal methods

Formal methods are not mandated in DO-178B, but there are static analysis requirements. Where formal methods have been used, they have been seen as complementary to testing.

GEC-Marconi Avionics are convinced of the benefits of both static and dynamic analysis. They applied formal methods to certain critical algorithms in the Boeing 777 PFCS. Formal specification was beneficial in some of the simpler algorithms, and problems were identified and resolved. However, formal proof was less beneficial in the same algorithms. The task was labour intensive of the highest grade of effort. Attempts to apply formal methods to more complex algorithms also required high grade effort and produced no useful result. The company have not been able to perform any confirmatory, parallel experiment but suspect that the same results could have been achieved by conventional methods given the same level of effort. Their general conclusion based on this experience was that formal methods had limited applicability.

The FAA's position on formal methods is currently similarly sceptical, and their policy seems to be to keep a watching brief. They are aware of the understanding and training problems associated with tools: there seem to be too few people who can use them effectively, and their economic feasibility is questionable. They also point out that the empirical evidence for the efficacy of formal methods is rather weak: in particular, they have not been shown to produce demonstrably safer systems than other methods. The FAA does not wish to be at the cutting edge of software technology, rather they need a well-founded confidence that methods actually work.

D8 Discussion

Flight critical aircraft control systems and nuclear safety systems clearly pose rather different problems to their designers and evaluators. The necessity for continuous control in the former case, compared with the need to respond only occasionally to reactor demands in the latter, mean that the reliability requirements differ by several orders of magnitude. Thus the reliability of a flight control system cannot be evaluated directly to confirm that the required goal has been achieved, whereas for the modest levels required in the nuclear case this kind of quantitative confirmation may be possible.

An impressive aspect of the 777 development was the role played by extremely extensive simulation. The aircraft industry seems to have a very detailed understanding of the flight conditions that its aircraft will meet, and can test against these in simulation without having to place a real aircraft at risk. The results of these simulations, together with considerable flight testing, must be a significant source of confidence in the safety of new aircraft, even though the intention is not to conduct a direct evaluation of the achieved reliability. There has been considerable progress in recent years in developing similar systems to test and evaluate nuclear protection systems, this work should be encouraged.

Notwithstanding the differences between the applications, some of the experiences in the aircraft industry may be relevant to the nuclear industry. An outstanding example is the high level of international regulatory consensus that is represented by DO-178B. This lies at the heart of a system that enables aircraft to be accepted as safe to fly in all countries around the world. If a similar regulatory consensus could be reached in the nuclear industry there would be considerable advantages in economy and possibly in enhanced safety.

The role of independent assessment in the US aircraft industry, through the DER system, is interesting and may be worth further study by the UK nuclear industry and regulators. The key components seem to be the formal legal basis of the reporting by the DERs to the FAA, coupled with the intimate knowledge of the development that comes from their day-to-day presence in the company building the aircraft. It is interesting to compare the DER system with the internal Independent Design Assessment (IDA) team set up by NE to evaluate the Sizewell B PPS. Whilst this team appeared to have the technical competence and access that were a feature of the DER process, their responsibilities lay solely with NE, and there was no legal obligation for them to report independently to the regulator (although the regulator has the legal right to inspect the licensees' documents at any time).

Appendix E: The UK railway industry

This note provides a partial review of practices in the railway industry focusing on concerns of interest to the Study Group. It has been developed from a visit to a vendor by members of the Study Group and from an informed interpretation of published literature.

E1 Licensing approach, safety management and standards

The railway signalling and control market is undergoing substantial structural change in response to the European open market and the privatisation in the UK. The renaissance in the railway industry is leading to ambitious plans for investment in new signalling equipment. In the UK Railtrack is committed to spending £2500M over the next ten years, a significant proportion of which will be on computer based equipment. This could have implications for the nuclear market as it may well be that procedures and innovations in the railway industry will drive the safety-critical computer market.

It was recognised by the industry and by HSE that the privatisation of the railways and the move away from a vertically integrated industry would lead to new arrangements for ensuring safety and these were formalised in the Railways (Safety Case) Regulations 1994. These govern the overall content of safety cases in the railway industry. There is substantial guidance available from the HSE [HSE 1994] on how these regulations should be followed in practice, in particular the handling of safety responsibilities between the infrastructure company (primarily Railtrack) and the operators. The guidance also advises on co-operation between different operators and their mutual responsibilities. Signalling systems, rolling stock and other equipment is subject to the Railways and Other Transport Systems (Approval of Works, Plant and Equipment) Regulations 1994. These require the approval of HM Railway Inspectorate, acting on behalf of the Secretary of State for Transport, of new or altered systems that may affect the safe operation of the railway.

The railway industry has a long tradition of attention to safety and learning from experience. This has led to the development of many (thousands of) industry specific standards, procedures and specifications that define requirements to be followed. Following privatisation there has been a large effort to review these documents and convert them to goal-based, performance standards that can be used effectively in the present contractual regime. These would provide requirements for the context in which computer systems might be used (e.g. physical and functional interface specifications, requirements for Electro-Magnetic Compatibility). However, because historically British Rail undertook much of the systems engineering and design work itself, there is a lack of standards on system aspects and, for example, the overall system design rationale and safety targets.

Partly in response to the market and regulatory developments there has been a considerable investment by both Railtrack and equipment suppliers in European standards being developed by committees in CENELEC TC9X [CENELEC 1993, CENELEC 1994, CENELEC 1995]. These standards broadly follow the IEC1508 model but differ in the approach to ‘baskets’ of techniques and in the emphasis on a data lifecycle as well as a software lifecycle. As many of the railway applications are configured by application-specific data (e.g. for a particular geographical layout) the specification, implementation, verification, validation and maintenance of the data is just as an important as that for the algorithmic part of the software. They also pay considerable attention to railway-specific approaches to computer system architecture and error detection. Unlike IEC 1508, they recognise explicitly the need for a safety case and provide useful guidance on how the development of a safety case fits into the system safety lifecycle. It was pointed out to us in the course of our consultations that, because of mutual recognition, the standards need to be sufficiently tough to exclude undesirable equipment from the marketplace but low enough to be practicable.

In order to address, and to be seen to be addressing, its safety responsibilities, Railtrack has been developing its own internal safety management systems. Recently, the directorate responsible for control and signalling systems published their Engineering Safety Management System [Railtrack 1997c, Railtrack 1997b, Railtrack 1997a, Railtrack 1997d]. This has a similar basis to those used in defence and nuclear applications (e.g. it contains very similar risk classification tables and text to those in Def Stan 00-56) but with additional supporting detail, procedures and work instructions. It does not provide much specific guidance on the content of safety cases for computer-based systems but does describe the generic assessment procedures.

There has not been a long tradition of QRA- or PRA-informed safety cases in the railway industry. The guidance from HSE does not address the overall safety targets for the industry - there is not a railway specific TOR document - but it does discuss the need to show that operations are as safe as they have been historically. However the recent Railtrack safety management principles adapts figures from the TOR report [HSE 1992b].

The translation of these overall figures into quantified performance targets for pieces of equipment remains problematic. The original safety management guide on risk assessment provided illustrative targets but acknowledged that the figures quoted for widely used equipment of 10^{-9}-10^{-12} probability of occurrence per hour are not practical to demonstrate. The draft European standard [CENELEC 1995] also provides illustrative figures of 10^{-11} failures/hr/system element for signalling systems. It would appear from our consultations that the approximations used in the derivation of the risk targets, especially the lack of detailed consideration of the mitigations between the signalling equipment failure and an accident, have led to this very high reliability figure (e .g 10^{-11} failures/hr/system element).

Although the comparison between the required reliability figures for railway signalling is difficult it would appear that the railway signalling requirements are at least as onerous as those for a single reactor protection system. It also appears that the railway industry will accumulate, because of the greater number of systems deployed, evidence to support or weaken the assumed conservative assumptions used in the nuclear industry on both the limits of reliability that can be claimed for computer systems and also for the general common mode failure rate cut-off rate.

The issues of assessment of fitness for purpose and the use of standards as the basis for certification are important as more systems become variants of already approved systems, and suppliers work with railway authorities in different countries. It is particularly important in the light of the significant planned investment and the known problems in developing safety cases for COTS software products (see Section 10 of this report).

E2 Present computer applications

Computers have been used in the railway industry since the mid 1980's for critical signalling and interlock applications. As in the nuclear industry there are also many other less critical but nevertheless safety-related systems. Extensive use is made of information and management systems whose failure can potentially initiate safety incidents or which can play an important role in mitigating or managing accidents (e.g. in provision of information to passengers). In this review we concentrate on the more critical applications.

The railway industry have a very successful record of computer application to critical systems. However we have been told privately of two safety-related faults found in equipment in another country with a highly advanced railway system. We also know of unexpected but apparently non-safety-related problems found by mathematical analysis of the software of a system currently extensively deployed.

Requirements specification

A notable feature of railway signalling systems is the use of traditional graphical specification languages that the railway signal engineer can use. There have been developments to enable configuration data to be automatically generated from these diagrams. Attention had been given to the assurance of these software data preparation tools that are judged SIL 2. There is also human factors based research looking at the loss of diversity that automated generation might bring. The resulting concept of cognitive diversity is novel and has potential application in the nuclear industry [Westerman *et al*. 1995]. The issue of tool integrity has also been a concern in the use of Atelier B and the theory base used by the tool has been the subject of independent review and analysis.

There have also been examples of work in which developments in computer science have been applied to the modelling and validation of signalling requirements. For example special purpose theorem provers have been used in Sweden to model specifications [Eriskon 1996], and some protocols have been analysed in detail with the HOL theorem prover as part of academic research [Morley 1993]. Network SouthEast also commissioned a large formal specification of a signalling control system. The ability to model the railway signalling philosophy and demonstrate with high confidence that changes to it are consistent and safe may be an important tool in facing the future challenges of new and hybrid control regimes.

Design for assurance

One of the noteworthy aspects of the railway signalling and control systems reviewed is that there is a concentration on designing for safety, on acceptable architectures that provide defence in depth and on a close coupling of the hardware and software safety cases. (This seems to us to be rather in contrast with the comments on the nuclear industry - see Section 7.2). Another important point is that designs of the signalling systems are often conservative and small (~64 Kbytes per module) with small teams of people involved.

One of the first computer-based signalling interlock systems in the UK was the SSI (Solid State Interlocking) system developed by British Rail and then licensed to equipment manufacturers to supply. SSI was developed by a small team, was written in assembler and received much scrutiny. This is not unlike the early approaches in the aerospace industry on engine controllers, apart from the lack of tool-based static analysis. In the defence industry these historical levels of scrutiny were not repeatable and this led the MOD to set down its requirements in a market shaping standard for safety-critical software that has now been issued and is being applied.

In other railway products one can see a continuation of the design for assurance concerns. Examples are the introduction of diversity in the design by using different normal forms for the logic, and different compiler options to give some protection against compiler faults. Other manufacturers use a form of encryption and special purpose hardware - the Vital Coded Processor [Chapront 1992] - as a defence against compiler and hardware faults. In all the systems reviewed the basic critical architectures use a small number of processors.

Software engineering

There are considerable differences between the software engineering approaches used in the railway industry by different contractors and the practices of the nuclear industry. For example in some current railway products the software was developed to good industry practice using JSD and Pascal and more recently, for less critical systems, some object-oriented techniques. There was some limited use of static analysis for control and data flow analysis. In contrast, there are examples from GEC-Alsthom in France of extensive use of formal methods and machine assisted mathematical verification. The use of formal methods originated in 1988 and has led to an increasing scope in their use, greater machine support achieved by the development of the B Method and the Atelier B toolset, and more recently the integration of the formal methods with the requirements engineering process. It has been applied to a number of train protection products (e.g. those used on the Cairo and Calcutta Metros) [Craigen *et al*. 1993]. The development of the B process and tools has been supported by the French Railways and Paris Metro. Interestingly, the regulators require the delivery of the proof file from Atelier B which they peruse.

Software reliability evaluation

None of the applications we reviewed had considered the use of statistical testing for software reliability evaluation. There is a general belief that the requirements derived from PRA would be too onerous for statistical testing and we did not find any systematic use of field data in software safety cases.

Licensing

It would appear from our brief review that some computer-based safety-critical systems would not be licensable in the nuclear industry for reactor protection. The reason for this and the policy implications for the HSE should be examined.

E3 Present and future challenges

Apart from the need to refurbish many of the existing non-computer-based signalling systems, there are some fundamental changes to the way in which train networks will be controlled. There is a general move world-wide to deploy transmission-based ‘moving block’ signalling systems that do away with the familiar signals at the side of tracks. These moving block systems, pioneered on the Paris RER system [Guiho & Hennebert 1990], allow for higher traffic densities, more flexible management and enhanced safety through the ready provision of automatic train control and protection. Transmission based systems also promise reduced costs on rural lines: a particularly attractive approach in the US where they will be combined with satellite based (GPS) positioning technology. There are also European studies examining driver-less freight wagons that move opportunistically around the network.

These developments, particularly the move to transmission based systems, have had a long gestation period and there has been much European research aimed at providing a standardised and progressive application of the technology [Brandi 1996]. In the UK the modernisation of the West Coast Mainline will be the first large scale application and major feasibility studies have just been completed. Even without innovation these new systems will pose significant challenges for the safety case and for the process of regulation. The systems will contain much safety-related software and a considerable amount of this is based on existing systems (e.g. digital telecomms). It will be interesting to see how the industry deals with the transition from modest systems, developed by small teams with an emphasis on design for assurance, to the new architectures.

The new systems will also introduce new failure modes. For example, security attacks are an obvious source of failures in transmission based systems. While confidentiality and integrity of data may be assured by encryption techniques, denial of service attacks will be much harder to counter. These are quite plausible, given experiences in other domains, and the traditional fail-safe response of stopping all trains may not be acceptably safe and certainly not secure. The role of the railways as a critical national infrastructure may lead to yet more onerous security requirements (See Section 10.4.2)

It may well be that the railway industry solves many of the problems facing the nuclear industry (i.e. in developing standards, applying COTS software, establishing the role of formal methods). It would be beneficial for the nuclear industry to establish more formal mechanisms for monitoring, evaluating and sharing experience with the railway industry.

E4 Conclusions

Current railway signalling systems are similar in the scale of their functional requirements to those for nuclear reactor protection. The railway industry also follows, and is required to follow by legislation, a safety case approach. So it is germane to compare experiences in the two industries. However, the comparison is complicated by the different applications and the difficulty in comparing the safety targets in the two industries.

There is considerable experience with safety-critical computer applications in the railway industry and there are some interesting aspects to the design and verification of signalling systems that should be noted by the nuclear industry:

- The emphasis on design for assurance is particularly important as is the small scale of the systems and the small teams involved.

- The adoption of formalised domain specific languages to express the requirements for signalling systems is seen as an important strength.

- The use of machine-checked formal methods is some of the most advanced in the world in terms of rigour and scale of application, and would appear applicable to some of the applications found in nuclear protection. There are also significant applications of theorem proving to model system specifications and requirements.

- There seems to be potential for making use of field data and/or statistical testing in justifying software reliability. While this may not form part of the current railway industry safety cases, the data could be of great interest in providing firm evidence in the safety-critical COTS software debate.

- The approach to tool integrity and the use of the concept of cognitive diversity may have application in the nuclear industry.

On the whole the industry appears to adopt processes and techniques which are conservative. Being cautious about change is one of the cornerstones of safety engineering yet this conservatism brings with it problems in judging SFAIRP, especially if one examines other industry sectors in defining good practice. However, it would appear from this limited review that some of the systems accepted for safety-critical applications in the railway industry would not be acceptable in nuclear ones. The reasons for this should be better understood.

To conclude then, the railway industry is facing many similar challenges to those anticipated in the nuclear sector and appears to have far greater resources to solve them. It would be beneficial for the nuclear industry and its regulators to establish more formal mechanisms for monitoring, evaluating and sharing experience with the railway industry.

Appendix F: The UK nuclear industry's research programme

F1 Introduction

Although the UK's major licensees (British Nuclear Fuels, Magnox Electric, Nuclear Electric and Scottish Nuclear) have responsibility for commissioning and managing the majority of safety research for their nuclear power stations, the Nuclear Safety Advisory Committee (NuSAC) is tasked by the Health and Safety Commission (HSC) with ensuring that the programme (known as the HSC Co-ordinated Programme) is adequate and balanced. This task is discharged through NuSAC's Sub-Committee on Research (SCR) which meets with the Industry Management Committee (IMC) and representatives of NII about three times each year. The research programme is directed via NII's Nuclear Research Index (NRI), which records the generic safety research issues considered to be significant by NII and the SCR. The industry's major licensees then use this index as the vehicle for commissioning nuclear safety research. The research programme is supplemented by the licensees' own research programmes, and research directly commissioned at the request of NII (funded through the HSE Levy Programme - a levy on the licensees for this purpose).

In July 1997, a small sub-group of the main Study Group met with an NII representative to discuss the nuclear safety research programme. This representative has responsibility for the control and instrumentation (C&I) aspects of the NRI and meets regularly with IMC's C&I Technical Working Group. He was thus considered best able to inform the Study Group's Research Sub-Committee about the current and past programme, and discuss any future proposals. The result of this discussion is captured below.

As with any research programme in leading edge technologies, there is always a difficulty in ensuring that research is sufficiently pragmatic to generate viable solutions without constraining potentially useful research avenues. Inevitably, some of the research goes no further than the technical report which documents the results, but the programme does have a facility for searching through previous results, before commissioning new work.

The research programme is also constrained by the lull in investment in new plant. There is currently little value in researching topics which are relevant only to new stations. Research topics of interest are those which are applicable to current instrumentation or which have potential in retrofit programmes.

In addition to its own programmes, the industry also takes part in European Research projects such as Halden and FASGEP.

The programme has recently adopted a policy of requiring the first task in each research project to be to produce a summary of the state of the art and the issues relating to the problem. In this way the research programme generates a set of reference knowledge which has value independent of its use in assessing the results of the research project.

F2 Research topics

There are two current projects which were seen as particularly high value in generating useful results which would be directly applicable. These are a project studying techniques for impact analysis in software change control and one studying concurrency.

Topics currently within the research programme include:

- A study of designs amenable to safety case demonstration. This topic is mainly aimed at distributed control systems such as might be used for safety integrity level 1 or 2 systems for control as opposed to protection systems. The topic is linked to the Advanced Control Programme of Strathclyde University.

- The topic of diversity is being revisited with a study of both diversity in the wider sense and in the specific sense of software diversity.
- The programme is a member of PRICES, which considers the role of humans in specifying and producing systems.
- Concurrent communicating processes are being studied, including both concurrency methods for software and the disconnect between the concurrent nature of control problems and the, often, sequential implementation of the solution.
- The problem of ASICs is being addressed with a prototype project where the regulator and supplier are exploring the regulatory context for ASICs by developing a real ASIC and exploring how it could be assessed.
- Replacement of legacy systems is being studied, using as an example a real system which will shortly become obsolete. The real example is being used to explore generic issues.
- Reliability and maintainability improvement is being addressed by a programme to update the industry PES guides, to take account of issues such as COTS software, IEC 1508 and smart sensors.
- There is a topic on statistical systems testing, which is working with the software diversity project and is building on the generic dynamic testing performed after the Sizewell PPS testing.
- There is a topic addressing smart sensors, and how software is being used to enhance sensor capability, but also considering the impact on the regulatory process.

F3 Relevance to the recommendations in this report

It is clear that the research programme is attempting, or has attempted, to address the topic areas which need further research. In some cases, the specific projects have not yielded promising results, but it may be that re-addressing the topic, in the light of subsequent experience or from a different viewpoint may now be needed. However, it should be noted that the topics are inherently difficult problems and there are no easy solutions. Furthermore, most of the topics relate to systems engineering and software engineering problems with far wider relevance than he nuclear industry alone. The nuclear industry can and does contribute to wider research programmes which seek to address such general problems.

The recommendations of the Study Group's report are now being used to influence proposals for research topics, which may include revisiting from a new direction previous research topics.

List of acronyms

ACE	Actuator Control Electronics
ACSNI	Advisory Committee on the Safety of Nuclear Installations
AI	Artificial Intelligence
AIMS	Airplane Information Management System
ALARP	As Low As Reasonably Practicable
ATWS	Anticipated Transient Without Scram
BSI	British Standards Institution
C&I	Control and Instrumentation
CAA	Civil Aviation Authority
CEA	Commissariat à l’Energie Atomique
COTS	Commercial Off-The-Shelf
DER	Designated Engineering Representative
DSIN	Direction de la Sûreté des Installations Nucléaires
EDF	Electricité de France
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulation
FMEA	Failure Modes and Effects Analysis
GPR	Groupe Permanent Réacteurs
HSC	Health and Safety Commission
HSE	Health and Safety Executive
HSWA	Health and Safety at Work Act
IAEA	International Atomic Energy Agency
IDA	Independent Design Assessment
IEC	International Electrotechnical Commission
IEE	Institution of Electrical Engineers
IMC	Industry Management Committee
IPSN	Institut de Protection et de Sûreté Nucléaire
IRRs	Ionising Radiation Regulations
JAA	Joint Aviation Authorities

JAR	Joint Aviation Requirement
LED	Light Emitting Diode
LOCA	Loss Of Coolant Accident
MTBF	Mean Time Between Failures
NII	Nuclear Installations Inspectorate
NRC	Nuclear Regulatory Commission
NRI	Nuclear Research Index
NSD	Nuclear Safety Directorate
NuSAC	Nuclear Safety Advisory Committee
NUSS	Nuclear Safety Standards
PFC	Primary Flight Control system
Pfd	Probability of failure on demand
PIE	Postulated Initiating Event
PPS	Primary Protection System
PRA	Probabilistic Risk Assessment
PSA	Probabilistic Safety Analysis
PTSN	Public Telephone Switched Network
QA	Quality Assurance
RTCA	Requirements and Technical Concepts for Aeronautics
SAPs	Safety Assessment Principles
SCR (NuSAC's)	Sub-Committee on Research
SEI	Software Engineering Institute
SFAIRP	So Far As Is Reasonably Practicable
SIL	Safety Integrity Level
SPIN	Système de Protection Intégré Numérique
TOR	Tolerability Of Risk
V&V	Verification and Validation
VDM	Vienna Development Method
VLSI	Very Large Scale Integration

References

[ACSNI 1990] ACSNI, *First report on training and related matters*, ACSNI Study Group on Human Factors, HMSO, London, 1990.

[ACSNI 1991] ACSNI, *Second report: Human Reliability Assessment - a critical overview*, ACSNI Study Group on Human Factors, HMSO, London, 1991.

[ACSNI 1993] ACSNI, *Third report: Organising for safety*, ACSNI Study Group on Human Factors, HMSO, London, 1993.

[Bergerand & Pilaud 1989] J. L. Bergerand and E. Pilaud, "Design and qualification of software for protection and control systems: the use of SAGA", in *SFEN International Conference on Operability of Nuclear Systems in Normal and Adverse Environments*, (Lyon), 1989.

[Bernstein & Kim 1994] M. M. Bernstein and C. Kim, "AOS: An avionic operating system for multi-level secure real-time applications", in *10th Annual Computer Security Applications Conference*, (Orlando), pp.236-45, IEEE Computer Society, 1994.

[Bishop & Bloomfield 1995] P. Bishop and R. E. Bloomfield, "The SHIP safety case approach", in *14th International Conference on Computer Safety, Reliability and Security (Safecomp95)*, (G. Rabe, Ed.), (Belgirate), Springer, 1995.

[Bock *et al*. 1988] H.-W. Bock, A. Graf and H. Hofmann, "Teleperm XS and XP: partners in I&C", in *International Conference on Computer Safety, Reliability and Security (Safecomp88)*, Pergamon Press, 1988.

[Boettcher & Tooley 1994] P. Boettcher and P. Tooley, "High integrity systems in defence in depth at Sizewell B", *High Integrity Systems Journal*, 1 (2) 1994.

[Brandi 1996] S. Brandi, "The European-wide cooperative research project for a standard railway traffic management system", in *World Congress on Railway Research (WCRR'96)*, (Colorado Springs), pp.583-7, 1996.

[Caspi *et a*l. 1987] P. Caspi, N. Halbwachs, E. Pilaud and J. Plaice, "LUSTRE: a declarative language for programming synchronous systems", *in ACM Symposium on Principles of Programming Languages*, (Munich), 1987.

[CENELEC 1993] CENELEC, *Railway Applications - Safety-related Electronic Railway Control and Protection Systems*, European Committee for Electrotechnical Standardisation, prEN-50129, December 1993.

[CENELEC 1994] CENELEC, *Railway Applications - Software for Railway Control and Protection Systems*, European Committee for Electrotechnical Standardisation, prEN-50128, February 1994 1994.

[CENELEC 1995] CENELEC, *Railway Applications - The specification and demonstration of dependability: reliability, availability, maintainability and safety (RAMS)*, European Committee for Electrotechnical Standardisation, prEN-50126, November 1995.

[Chapront 1992] P. Chapront, "Vital Coded Processor and safety-related software design", in *11th International Conference on Computer Safety, Reliability and Security (Safecomp92)*, Pergamon, 1992.

[CNRA 1997] CNRA, *Licensing of computer-based systems important to safety: Report of Workshop and CNRA Special Issues Meeting on Technical Support*, OECD Committee on Nuclear Regulatory Activities, OECD Nuclear Energy Agency, NEA/CNRA/R(97)2, 1997.

[Craigen *et al*. 1993] D. Craigen, S. Gerhart and A. Ralston, *An international survey of industrial applications of formal methods*, NIST, US Department of Commerce, GCR 93/626, 1993.

[Dahll & Applequist 1993] G. Dahll and T. Applequist, OECD *Halden Reactor Project - Licensing of safety critical software*, Halden, Report HWR 344, 1993.

[DoD 1996] DoD, *Report of the Defense Science Board Task Force on Information Warfare-Defense (IW-D)*, Defense Science Board, Office of the Under Secretary of Defense for Acquisition and Technology, Washington DC, November 1996.

[Eckhardt & Lee 1985] D. E. Eckhardt and L. D. Lee, "A Theoretical Basis of Multiversion Software Subject to Coincident Errors", *IEEE Trans. on Software Engineering*, 11, pp.1511-7, 1985.

[EDF 1994] EDF, "Palier N4: les centrales de Chooz B et Civaux", *Revue Générale Nucléaire* (Juillet-Aout), pp.5-10, 1994.

[Eriskon 1996] L.-H. Eriskon, "Specifying railway interlocking requirements for practical use", in *15th International Conference on Computer Safety, Reliability and Security (Safecomp96)*, Springer, 1996.

[FAA 1995] FAA, *Designated Engineering Representatives (DER) Handbook*, Federal Aviation Administration, Washington, DC, 1995.

[Fenney 1994] T. Fenney, "Sizewell B Primary Protection System: an assessment of the confirmatory review process", in *Technical Committee Meeting Organized by the IAEA*, (Helsinki/Espoo, Finland), IAEA, 1994.

[Guiho & Hennebert 1990] G. Guiho and C. Hennebert, "SACEM software validation", in *12th International Conference on Software Engineering*, IEEE Computer Society Press, 1990.

[Hoyne & Driscoll 1992] K. Hoyne and K. Driscoll, "SAFEbus", in *11th Digital Avionics Systems Conference*, (Seattle), AIAA/IEEE, 1992.

[HSE 1992a] HSE, *Safety Assessment Principles for Nuclear Plants*, Health and Safety Executive, ISBN 0 11 882043 5, 1992.

[HSE 1992b] HSE, *The Tolerability of Risk from Nuclear Power Stations*, HMSO, London, 1992.

[HSE 1994] HSE, *Railway safety cases: guidance on regulations*, HSE Books 1994.

[HSWA 1974] HSWA, *The Health and Safety at Work Act 1974*, HMSO, c. 37, 1974.

[Hughes & Hall 1992] G. Hughes and R. S. Hall, "Recent developments in protection and safety-related systems for Nuclear Electric's (UK) power plant", in *IAEA/OECD International Symposium on Nuclear Power Plant Instrumentation and Control*, (Tokyo), 1992.

[Hunns & Wainwright 1991] D. M. Hunns and N. Wainwright, "Software-based protection for Sizewell B: the regulator's perspective", *Nuclear Engineering International*, 36(446), pp.38-40, September, 1991.

[IAEA 1980] IAEA, *Protection systems and related features in nuclear power plants*, International Atomic Energy Agency, Vienna, Safety Series No 50-SG-D3, 1980.

[IAEA 1994] IAEA, *Software important to safety in nuclear power plants*, International Atomic Energy Agency, Vienna, Technical Report No 367, 1994.

[Ichiyen & Joannou 1991] I. Ichiyen and P. Joannou, "The CANDU approach to digital safety systems analysis", *Nuclear Engineering International*, 36(446), pp.35-7, September, 1991.

[IEC 1986] IEC, *Software for Computers in the Safety Systems of Nuclear Power Stations*, International Electrotechnical Commission, IEC 880, 1986.

[IEC 1993] IEC, *Nuclear power plants - instrumentation and control systems important to safety - classification*, International Electrotechnical Commission, Geneva, IEC 1226 (1993-05), 1993.

[IRR 1985] IRR, *The Ionising Radiation Regulations* 1985, HMSO, Statutory Instrument No 1333, 1985.

[Joannou & Harauz 1990] P. K. Joannou and J. Harauz, *Standards for Software Engineering of Safety Critical Software*, Ontario Hydro, Standards Procedures and Guides 98C-H69002-001, 1990.

[Keats 1983] A. B. Keats, "Fail-safe computer-based plant protection systems", in *International Conference on Computer Safety, Reliability and Security (Safecomp83)*, Pergamon, 1983.

[Knight & Leveson 1986] J. C. Knight and N. G. Leveson, "An Empirical Study of Failure Probabilities in Multi-version Software", in *Proc. 16th Int. Symp. on Fault-Tolerant Computing* (FTCS-16), (Vienna, Austria), pp.165-70, 1986.

[Kuhn 1997] D. R. Kuhn, "Sources of failure in the public switched telephone network", *IEEE Computer*, 30 (4), pp.31-6, 1997.

[Leveson 1995] N. G. Leveson, *Safeware: System Safety and Computers*, Addison Wesley, 1995.

[Littlewood & Strigini 1993] B. Littlewood and L. Strigini, "Assessment of ultra-high dependability for software-based systems", *CACM*, 36 (11), pp.69-80, 1993.

[May *et al*. 1995] J. May, G. Hughes and A. D. Lunn, "Reliability estimation from appropriate testing of plant protection software", *Software Engineering Journal*, 10 (6), pp.206-18, November 1995.

[MoD 1996] MoD, *Hazard Analysis and Safety Classification of the Computer and Programmable Electronic Systems Elements of Defence Equipment*, Ministry of Defence, Def-Stan 00-56, Issue 2, December, 1996 .

[MoD 1997] MoD, *The Procurement of Safety Critical Software in Defence Equipment*, Ministry of Defence, Def-Stan 00-55, Issue 2, August, 1997.

[Morley 1993] M. J. Morley, "Safety in Railway signalling data: a behavioural analysis", in *Higher Order Logic Theorem Proving and its Applications*, Springer, 1993.

[NIA 1965] NIA, *The Nuclear Installations Act 1965 (as amended),* HMSO, c. 57, 1965.

[NRC 1994] NRC, *Final safety evaluation report related to the certification of the System 80+ design*, US Nuclear Regulatory Commission, Docket No 52-002, NUREG-1462, August 1994.

[Paulk *et al*. 1993] M. Paulk, W. Curtis and M. B. Chrisis, *Capability Maturity Model for Software, Version 1.1,* Software Engineering Institute, Technical Report, CMU/SEI-93-TR, February 1993.

[Railtrack 1997a] Railtrack, *Engineering Safety Management System. Vol 1, issue 2: Principles of Train and Infrastructure Change Management*, Railtrack Electrical Engineering and Control Systems, 1997.

[Railtrack 1997b] Railtrack, *Engineering Safety Management. Vol 2, issue 2: Train and Infrastructure Change Management Processes,* Railtrack Electrical Engineering and Control Systems, 1997.

[Railtrack 1997c] Railtrack, *Engineering Safety Management. Vol 3, issue 2: Guidance*, Railtrack Electrical Engineering and Control Systems, 1997.

[Railtrack 1997d] Railtrack, *Engineering Safety Management. Vol 4, issue 2: Tools and techniques*, Railtrack Electrical Engineering and Control Systems, 1997.

[RTCA 1992] RTCA, *Software considerations in airborne systems and equipment certification*, Requirements and Technical Concepts for Aeronautics, DO-178B, July 1992.

[Rushby 1993] J. Rushby, *Formal methods and the certification of critical systems*, Computer Science Laboratory, SRI International, Menlo Park, Technical Report, SRI-CSL-93-07, 1993.

[Temelin 1993] Temelin, "Modernising the Temelin VVER power plant", *Modern Power Systems*, pp.34-7, July 1993.

[Westerman *et al*. 1995] S. J. Westerman, N. M. Shryane, C. M. Crawshaw, G. R. J. Hockey and C. W. Wyatt-Millington, "Cognitive diversity: a structured approach to trapping human error", in *14th International Conference on Computer Safety, Reliability and Security (Safecomp95)*, (G. Rabe, Ed.), (Belgirate), Springer, 1995.

Printed and published by the Health and Safety Executive C25 10/98