

Perils of Precedent in Nuclear Software Qualification

Fred Pollard, Martin Rendell, and Daniel Finnigan

Rolls-Royce SMR Limited

Abstract

This paper examines the continuing use of the United Kingdom nuclear industry's current precedent in the qualification of software and programmable logic used in nuclear control and instrumentation systems. In this context, software qualification has conventionally relied on a two-legged argument comprising "production excellence" and "independent confidence-building measures" (the latter including independent static analysis, independent dynamic analysis, and statistical testing). Drawing on precedent from Sizewell B and subsequent UK nuclear practice, this paper argues that these measures have too often become in-effect "prescribed" techniques rather than measures selected to answer a clearly stated safety question. This paper reviews the claimed contribution of these techniques to safety and confidence-building, considering their practical and technical limitations, their dependence on developer interpretation, their cost and schedule implications, and their effect on through-life modification. It concludes that post hoc re-verification of completed software can provide little additional safety benefit where modern standards-based development, verification, validation, and independent functional safety assessment have already been performed, and may inhibit future safety improvements to a system by making modification disproportionately difficult. The paper proposes a possible alternative approach to software qualification, but notes that until the goal of qualification is determined, it is not clear how to assess the validity of any qualification performed above and beyond that which is prescribed by international standards.

1 Introduction

1.1 Background

The UK nuclear industry has developed a relatively unique approach to qualification of software in control & instrumentation (C&I) systems since the 1990s, with the original catalyst for the establishment of the approach being the Primary Protection System (PPS) at Sizewell B Nuclear Power Station (SZB) in Suffolk, UK. This has established a UK-nuclear-specific precedent that is frequently used as a basis for software justification within this industry.

This software-qualification approach was inspired by the "special-case procedure" for plant structural components that have very small random-failure rates — systematic defects dominate random failures (May et al. 1995). This procedure established a "multi-legged" safety case for such components. It was considered a helpful analogy for software, which

cannot fail randomly — only systematically¹; thus, a special-case procedure was established for software-based systems as well.

This software-qualification approach, comprising a “two-legged” qualification framework, focuses on demonstrating software quality through:

- a confirmation that the software-based system’s development lifecycle and design meet expectations based on relevant international standards (the “production excellence” leg); and
- independent confidence-building activities, such as analysis and test techniques and measures, usually performed on the final version of the software (or integrated system) after it has completed its development lifecycle (the “independent confidence-building measures” (ICBMs) leg) and preferably in a diverse manner to the development lifecycle’s equivalent activities².

It is the ICBM leg in particular that is often identified as contributing significantly to the cost and schedule of software-based C&I systems in the UK, especially for systems performing Category A nuclear safety functions³. The ICBM leg, and the activities contained within it, is the primary focus of this paper’s examination of the value of UK-nuclear software-qualification precedent. This is because unlike the production excellence leg, which is informed by useful precedent and international consensus in an evolutionary way through the development of standards and bodies of recognised good practice across safety industries, the ICBM leg is a construct unique to the UK nuclear industry, the basis for which has not significantly changed in nearly four decades (Hunns and Wainwright 1992) (ONR 2020).

The primary references for the two-legged software-qualification approach is guidance found in the Office for Nuclear Regulation’s (ONR) Safety Assessment Principles (SAPs) (ONR 2020) and the ONR’s technical assessment guide for computer-based systems (ONR 2023).

1.2 Purpose

This paper takes a step back from the software-qualification approach conventionalised within the UK nuclear industry since the 1990s to examine what the authors believe that the current approach achieves in reality, looking beyond the theory laid out in existing literature.

We focus in particular on five techniques which appear in regulatory documentation as additional activities to be performed over and above the activities already performed in a modern, safety-oriented development process:

1. Independent tool review

¹ Given the same inputs, same external environment (including the hardware with which it is interacting) and same internal state, software will always generate the same output. Non-determinism in a software system’s output, i.e. different output for same input, is due to incomplete definition of the system’s behaviour, leading to incomplete software requirements.

² ICBM activities are expected by UK regulatory guidance (ONR 2020) (ONR 2023) to be performed specifically on the final post-validation product, rather than on pre-final product versions or on the development process itself. Activities on the development process are the remit of “production excellence” activities, and the expectation is that any pre-validation testing is within scope of the product developer’s own processes.

³ Nuclear safety function categories and system classes are defined in BS EN IEC 61226:2021. Category A functions play a principal role in preventing anticipated operational occurrences or design basis accidents from leading to high-severity consequences, or whose failure could directly lead to an unmitigated accident with high-severity consequences. The class of a system is informed by the categories of function(s) allocated to it and the system’s contribution to them.

2. Certification by an independent body
3. Independent static analysis
4. Independent dynamic analysis
5. Statistical testing

We review these techniques, in terms of their method and theoretical value, and consider how their application as ICBMs contributes to the plant safety case. We consider their interaction with the development lifecycle of the systems themselves, and the plant, and consider how human factors may affect the value to the safety case of these techniques.

1.3 Structure of this Paper

This paper comprises two core sections:

- Section 2 examines the use of the ICBM techniques listed above that appear in UK nuclear precedent, and the benefits that they offer. It also considers the challenges to safety that may arise by focusing effort on post-development verification.
- Section 3 explores the pre-requisites for constructing a useful qualification if it is not to be driven solely by the precedent from previous UK nuclear projects.

Note that in this paper, “software” is used as a catch-all term that includes hardware description language (HDL)-programmed devices such as field-programmable gate arrays (FPGAs), as such devices are considered to have similar relevant properties to conventional software as far as qualification is concerned. This terminology aligns with the UK nuclear consensus, as well as some other safety-critical industries worldwide.

2 Examination of Techniques Used in UK-Nuclear Precedent

2.1 Overview

The background in Section 1 has described the structure of prior demonstrations for nuclear-classified software, but this section will consider the activities that have previously been undertaken as part of these demonstrations. The “production excellence” part of prior demonstrations is not objectionable in principle, in that it addresses the need for the application of internationally-recognised standards and showing that these have been complied with. This form of demonstration appears in all nuclear jurisdictions of which the authors are aware, and so we do not discuss this aspect of the safety demonstration further.

The independent confidence-building leg has previously involved the execution of a series of activities, typically graded by classification. Table 2 in NS-TAST-GD-046, the ONR Technical Assessment Guide for computer-based safety systems (ONR 2023), which is entitled “*Example graded approach to the selection of measures to support justification of commercial off the shelf devices*”, only applies to smart devices, and it is not formally a requirement that licensees follow this in any case. In practice, however, this table appears in one form or another with the same list of techniques in several contexts for so-called Computer-Based Systems Important to Safety (CBSIS) in existing facilities. In civil nuclear power plants, the representation of that table typically includes additional gradation with

two separate columns for Class 1 differing only by reliability claim so that more onerous techniques are reserved for a 10^{-4} pfd/pfa claim⁴.

The ICBM techniques for a Class 3 claim according to this table are generally achieved by a good development process and the application of regular commercial practice. The differentiator in gradation is the extent to which the following techniques are applied:

1. Independent tool review.
2. Certification by an independent body (supported by evidence).
3. Static analysis of the source code.
4. Dynamic analysis of the source code.
5. Statistical testing.

The application of these techniques as independent activities (i.e. as part of the ICBM leg, separate from and subsequent to the development process) is not mandated by international nuclear standards such as BS EN 61513:2013 (or standards within its family such as BS EN 60880, or BS EN IEC 62138), nor indeed by any other functional safety standard that the authors have examined.

The activities represent precedent, and so it is helpful to consider whether these techniques should be included in a given nuclear software qualification programme. However, the basis for inclusion cannot simply be that it is what has been done in the past, since precedent reflects the technological context in which it was developed. Instead, precedent should be evaluated for any benefit it offers relative to the time, effort and cost involved in undertaking the additional activities in the present context.

For a system delivering nuclear safety functions, benefit can come in two forms: establishing a *safety benefit* and *building confidence*. These are related concepts, but they are not the same.

Safety benefit implies eliminating or reducing hazards, strengthening controls to eliminate risk or reducing the likelihood of error. In essence, safety is improved by an activity when it truly reduces the level of risk.

Building confidence is an exercise in increasing the trust that a system is safe or otherwise fit for purpose. Here, the activity is about changing the understanding of stakeholders, be they a customer, regulator, or even an assurance function within a development organisation. The aim in a safety context is to assure these stakeholders that safety has been achieved by demonstrating that the system will operate correctly.

By developing a system well, applying the safety lifecycle rigorously and introducing safety improvements throughout, strong technical evidence is naturally generated. These activities directly contribute to the safety of the system and any improvements in them should correspondingly produce benefits for safety.

Confidence, however, is established when the system is assessed as a whole, bringing that evidence together into a coherent, system level analysis that explores whether the safety functionality of the plant has been correctly implemented. This holistic review demonstrates that the lifecycle has been applied consistently, that the development activities have genuinely reduced risk, and that the integrated system behaves safely. In this way, good engineering creates safety benefits, and structured assurance creates confidence.

⁴ Where pfd = probability of failure on demand, and pfa = probability of failure per annum.

In the context of international standards, the safety benefits are achieved by rigorous specification, design, implementation, verification and validation, coupled with hazard and failure analysis at each stage that establishes potential issues that can be resolved as part of the development lifecycle. These same standards will typically include an independent functional safety assessment (or equivalent) that fulfils the need for confidence-building by performing (BS EN 61508-4:2010):

[an] investigation, based on evidence, to judge the functional safety achieved by one or more ... safety-related systems and/or other risk reduction measures.

The independence here is crucial to provide the confidence, with greater independence typically required as the safety significance of the functionality increases. However, note that this independent assessment from international standards does not re-execute development activities, including verification⁵. Instead, it probes the evidence already generated to form a judgement on the overall safety achieved.

Within international functional safety standards then, we already have the required elements of building safety into the plant and confidence-building for stakeholders. These activities are already effectively required in the UK nuclear regime as part of the “production excellence leg”. Since the ICBMs are an additional layer of verification over and above the development activities and an independent assessment of those activities as described above, the remaining question is whether they provide meaningful additional benefits in the form of safety improvement or confidence-building.

2.2 Independent Tool Review and Certification

We can briefly state that tool review is already addressed by international standards, which require various forms of tool qualification. This is already incorporated into both the developer’s activities (on the assumption that we have determined that the developer is following modern safety system development practices), and the independent assessment thereof. We therefore assert that performing another independent assessment of tool qualification is redundant, and so adds no safety benefit nor meaningful improvement in confidence.

Table 2 in NS-TAST-GD-046 also mentions certification by an independent body. If applied correctly, certification by an independent body may result in benefit in the form of safety improvements because certification authorities increasingly engage with developers during the process to provide independent advice, thus enhancing the development lifecycle quality. However, in the UK nuclear regime certification alone is not accepted and licensees are expected to review all evidence to confirm themselves that the development of components is adequate and in line with expectations. Therefore, in the UK nuclear context, certification at a component level might at most provide some initial guarantee that the additional independent audits and assessments are likely to be satisfactory, but the certification does not in of itself offer any usable evidence of quality. Consequently, it is judged that, in practice, certification by an independent body would be equivalent to an independent functional safety assessment for matters of significance⁶. This is already part of

⁵ Note that in the UK nuclear industry, it is increasingly the case that software-based systems are wholly or partly pre-developed (e.g. smart devices and pre-developed platforms/programmable logic controllers, respectively). Consequently, the UK nuclear qualification approach, with its focus on *post hoc* independent re-verification and re-validation, can have little to no influence on the quality of the underlying software.

⁶ Noting that different assessors may place different emphasis on different aspects of the system development.

demonstrating “production excellence”, so an exercise in certification would simply demand a second independent assessment, which would be redundant, and so add no safety benefit nor meaningful improvement in confidence.

This leaves static analysis of the source code, dynamic analysis of the source code and statistical testing to consider in the coming subsections. However, first it is instructive to consider the structure of a typical ICBM activity and its impact on safety.

2.3 Structure of an ICBM Assessment Activity

For source-code ICBMs, there are three principal parties involved:

1. The *analyst* is the person or team responsible for performing the independent verification.
2. The *licensee* is the operating organisation responsible for the safety of the plant.
3. The *developer* can be any of the people, team(s) or organisation(s) who have specified, designed, implemented, verified and validated the safety system that contains the software.

The typical relationships and reporting arrangements between these parties are illustrated in Figure 1.

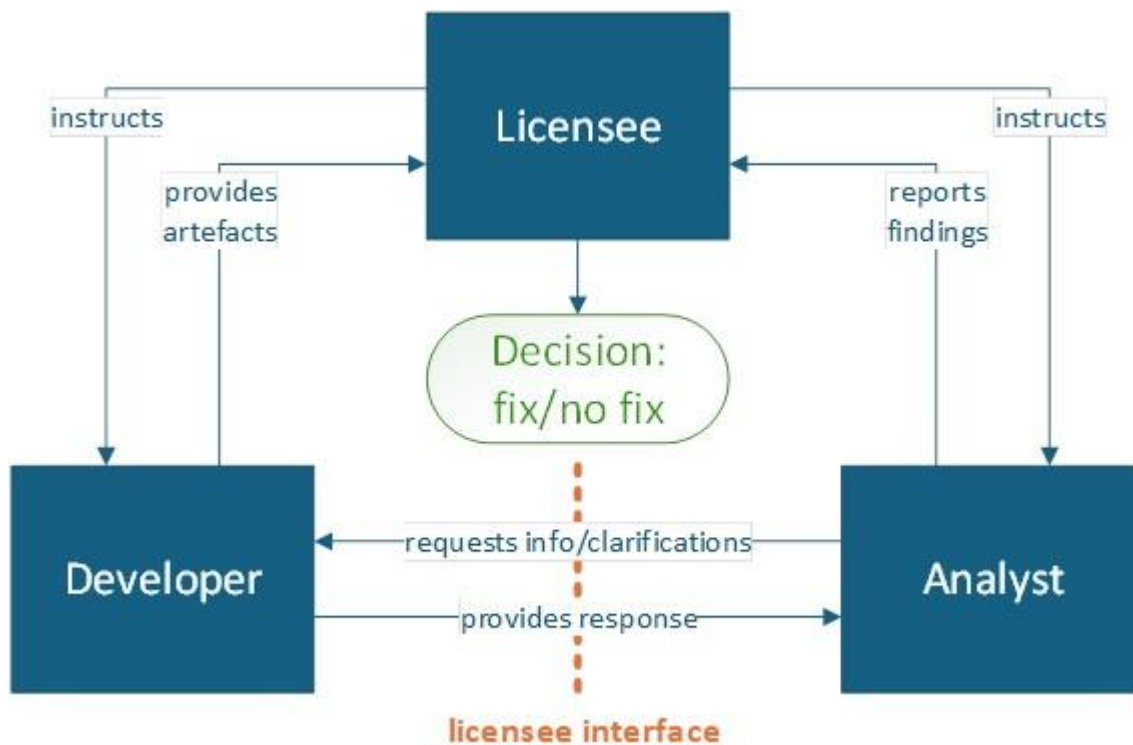


Figure 1 ~ Typical Relationships During ICBM Activities

The analyst typically reports to the licensee who is planning to operate the plant. The developer reports to the licensee as a supplier. Interactions between the analyst and the developer are in practice variable in their formality but good practice is that they should be documented and mediated by the licensee. In the course of their work, the analyst will make findings, which are technical outcomes that the analyst believes are anomalies or defects in the system. These findings are then sentenced in an engagement between the analyst and

developer, chaired by the licensee who makes the final determination on the impact of the finding to safety⁷.

This arrangement is unavoidable, because the analyst must be paid, and the licensee is ultimately responsible for safety and so must oversee and ultimately accept all such findings. Efforts have previously been made to have dual roles within a single licensee organisation, but this is less apparent in recent practice, which instead emphasises the independence of the analysts themselves from the developers on the assumed premise that this ensures the result is not affected by bias or mindset.

It is worth highlighting that this arrangement is nonetheless extremely challenging to a good safety culture. The timing of ICBMs, which must by their nature be delayed until the source code has undergone sufficient verification and validation, and which cannot be completed until after system validation, tends to place them on the critical path of a project. Since any defects requiring source code changes would result in a costly and time-consuming modification cycle that would also risk injecting new faults, the licensee is disincentivised to agree to any categorisation of a finding that will result in changes because of cost and schedule pressures. This is not to suggest in any way that any licensee would make a conscious decision to ignore a safety-related finding, but in a case where the effect (perhaps in aggregation) of many findings is unclear, e.g. non-trivial/unidentified interactions, or subject to debate, there is always going to be a pressure that favours a less impactful categorisation.

Having established the reporting structures that are required for such a third-party independent re-verification, the next sub-sections consider the different techniques that have been applied in this way in prior precedent within the UK nuclear industry and examines their benefits to both safety and confidence-building.

2.4 Static Analysis

The term “static analysis” is extremely broad and can refer to any method of analysing software (including HDL) that does not involve executing it. In the past at Sizewell B, and more recently at Hinkley Point C, this has involved formal verification of software using the MALPAS Compliance Analyser for Class 1 systems with a claim of 10^{-4} pfd/pfa, and informal verification via symbolic execution using the MALPAS Semantic Analyser for Class 1 systems with a claim of 10^{-3} pfd/pfa. The static analysis family of techniques also incorporates techniques such as source-code comparison, which has previously been applied to systems with claims of 10^{-4} pfd/pfa (for example at Sizewell B), and attempts to provide confidence that the compiled program is a faithful representation of the underlying source code.

Note that this sub-section will describe these techniques in the context of the MALPAS tool, this is because it is the tool that has been used in precedent. However, the evaluations of each technique do not imply anything is amiss with the tool and would be the same for any tool used in a similar way.

At Sizewell B, formal verification (compliance analysis) was used to demonstrate that the software source code complied with both its software requirements specification and its software design specification (Ward, 1993). The analyst manually converted the details in

⁷ This is typically a four-category system denoting an impact to nuclear safety (Category 1), documentation error (Category 2), non-functional or non-critical defect in the code such as comments (Category 3), and no anomaly (Category 4); for example see (Ward 1993).

these specification documents, which are written in a natural language specification, into a formal mathematical notation. The source code for each function was then modelled using an intermediate language that could, albeit with considerable additional human input, automatically demonstrate that the modelled source code matched its modelled specification⁸.

This appears to be a rigorous process, though it suffers from only performing an analysis on individual components, rather than on the system as a whole. Minimal confidence can be developed on the safety of the overall system based on an in-depth analysis of one part of it. For example, such an approach cannot consider interface errors between system components, such as those caused by timing or communications faults⁹. Recognition that components of a system can individually function as intended, but nonetheless in combination still result in hazardous behaviours at the system level, is a key insight in modern safety engineering, and has motivated more recent approaches such as System-Theoretic Process Analysis (STPA).

Given that it offers little-to-no insight into system-level behaviours, the apparent objective of individual-component analysis is to establish confidence that the development process achieved its intended aims for that component, additional to that derived from the “production excellence” process itself. It can therefore be concluded that there is no direct safety benefit from the application of this technique, since there is only an intent to retrospectively demonstrate the achievement of a safety-related property, not to influence the development of the system itself.

Furthermore, behavioural formal verification, i.e. formal verification of the software functionality, used in this way can only generate confidence insofar as it concerns implementation against the software requirement and software specifications; no validation is performed of the correctness of the requirements and specifications, nor the correctness of earlier development lifecycle stages. However, it is not clear that confidence-building in even this narrow aspect is meaningfully achieved by behavioural formal verification of the source code either for reasons we shall now examine.

As noted, to perform behavioural formal verification on software not developed with formal methods, the analyst must convert a natural language specification into a mathematical formalism. However, natural language specifications are ambiguous, and do not explicitly outline developer assumptions that underpin them. Translation is dependent on analyst skill and judgement, rather than the ideal of unambiguous formal equivalence (Ward, 1993). There are therefore always going to be multiple, perhaps many potential translations from the natural language to formal definition.

It is therefore likely that either the analyst will require clarifications, or more likely that the formal verification itself will produce a finding based on a mismatch of the specification to the source code. In both cases, the only source of information is the developer of the system.

Clarifications will allow developer assumptions to be codified in the independent-verification mathematics, and their response to the findings in sentencing will be to highlight errors in the assumptions that the analyst has made in modelling the intent of the original specifications. The analyst will then take the reasonable course of action to adjust the

⁸ The Sizewell B MALPAS analysis took 4.5 years and had a nominal team size of 15 people, rising to more than 80 in the later stages (Ward, 1993).

⁹ Bruns and Anderson (1995) cite Littlewood and Strigini (1993), noting that “*faults which resist discovery by testing are typically the result of unforeseen interaction between components*”. A similar argument can be applied to component-wise formal verification.

mathematical formalism based on this new information, perhaps highlighting that this suggests that the documentation should be made less ambiguous or to contain additional information.

At this point, consider what has occurred: the analyst's re-interpretation exercise is no longer independent of the developer in substance. Any mistakes that the developer has made in interpreting their specifications, or any erroneous assumptions made during the development now have a high likelihood of being encoded in the formal model. Indeed, the developer is always in a privileged position in these circumstances because they built the system and so know it better than anyone else. This gives an unavoidable authority bias during interactions, which the licensee is in no better a position to counteract. Since the only basis for the analysis is the natural language specification, the interpretation of the developer with respect to how the source code has been implemented (i.e. the aggregation of the programmer, reviewer, tester, etc., all of whom have converged on the same interpretation of the specification by the end of system validation) will therefore generally dominate.

The result is that this type of behavioural formal verification is likely to be based on a mathematical model that has converged on the developer's intent. The verification activity will therefore tend to confirm that the software behaves in the way the developers believe it is meant to behave¹⁰, rather than what the software specification necessarily originally intended. This would tend to lead the analysis to focus on corrective measures to the documentation to aid analyst clarity, rather than to the software itself.

This is borne out empirically from the analysis at Sizewell B, which identified (as of 55% completion) approximately 2,000 findings (Ward 1993), of which:

1. No Category 1 errors — that is, no errors that impacted on safety.
2. 40% of the errors were Category 2, relating to the need to clarify the documented specifications.
3. 8% of the errors were Category 3 errors which proposed improvements to code defensiveness but were not critical defects.
4. 52% were classified as Category 4 (not requiring any action).

At the completion of the analysis campaign, there were still zero Category 1 findings, and only 6% of the total findings were Category 3, with the rest either Category 2 or 4 (Wichmann et al. 1995). The findings were therefore overwhelmingly dominated by categorisations requiring no change to the software, with the majority of findings at worst requiring clarification of the specification so that it reflected the implementation. The authors' experience performing and overseeing the similar work elsewhere is that the proportions of finding types are similar and for similar reasons.

Based on these figures, one of the following must be true: that the software was free from critical defects (in which case this suggests that the "production excellence" and its associated non-formal test and analysis activities were actually sufficient to assure safety and the formal verification offered limited additional insight into how the system would behave); that the *post hoc* analysis was incapable of finding significant defects in the software; or the *post hoc* analysis process is dominated by the authority a competent developer has over their code and their understanding of it.

¹⁰ This may be particularly error-prone if the original software developers are no longer available, and so the personnel supporting the analysis in their stead are themselves having to infer the intended functionality. There is a risk here of the substitutes using what is written in the source code as the source of truth, or a deferral to the "authority" of the original developers.

It is effectively impossible to know which scenario above is most valid. In any case, there does not appear to be a meaningful safety benefit arising from the activity, with the risk that there is a false confidence generated from the presumption that the analysis is somehow meaningfully independent of the development.

This result appears unavoidable regardless of what stages of the lifecycle this type of after-the-fact formal verification is applied to¹¹. If the analyst attempts to start modelling at the system level, they will still require extensive developer input to make the transition to the lower levels because for any system there is more than one way of designing and implementing a given set of requirements, and the analyst would need to codify all the assumptions the developer has made in doing so. Behavioural formal verification is therefore not a beneficial activity for safety, nor for a meaningful improvement in confidence when performed after the fact¹².

Semantic analysis suffers from the same issues as behavioural formal verification, but to a greater degree because the specifications are never formalised. Instead, the method converts sequential logic in software source code to a mathematical representation that the analyst then manually compares to the natural language specification. This again places reliance on the interpretation of the specification, where the developer will have a greater authority over the intent of the natural language specification.

There can be limited confidence that this is a genuinely independent approach offering significant improvement in confidence. It is worth noting that the proportions of findings presented earlier in this section incorporated the outputs from both compliance analysis and semantic analysis, so the conclusions are similar. We therefore conclude that it is not a beneficial activity for safety nor for a meaningful improvement in confidence.

A counterpoint to these conclusions could be that any additional verification activity that has the potential to identify defects is definitionally going to improve confidence, *particularly* if it finds no errors. To this, there are two replies: the first is that a modern, safety-standards compliant approach already performs a lot of verification, and this verification will almost certainly find errors in the initial implementation that are then resolved and verification repeated until the final exercise finds no defects of significance. Performing additional verification on already-verified code will suffer from diminishing returns¹³. This leads to the second point, which is that there has to be a point at which the time, effort, and cost of additional verification tasks has to be weighed against the safety benefit achieved, in line with the principle of reducing risk so far as is reasonably practicable (SFAIRP).

The time and costs associated with formal verification (compliance analysis) and semantic analysis can be estimated using the productivity ranges provided in the *MALPAS Advanced Software Analysis and Verification — Introductory Guide* (Atkins 2008) and some estimates of analyst cost and the number of lines of code (LOC) in a modern reactor protection system. For our estimates, we assume:

¹¹ i.e. whether *post hoc* analysis is performed on the artefacts of the completed system requirements, or on the artefacts of the completed coding phase, for example. The actual timing of the *post hoc* analysis is largely irrelevant.

¹² There is acceptance that formal methods used within the development process offer many benefits, but this is not widespread practice, nor mandated by standards at present.

¹³ Software reliability growth modelling usually becomes less convincing at high reliability claims for this reason — it becomes increasingly difficult to demonstrate that the observance of no test failures shows high reliability due to the low expectation of failures at that reliability level.

1. A system with 1 million LOC. The Sizewell B PPS had approximately 100,000 LOC, and our experience is that an order-of-magnitude increase in the intervening 30 or so years is a reasonable estimate¹⁴.
2. For compliance analysis, a productivity of 30 LOC/day. This is the midpoint of the range given by (Atkins 2008) of 10 to 50 LOC/day, which reasonably accounts for initially low but gradually improving productivity over the course of an analysis.
3. For semantic analysis, a productivity of 60 LOC/day. This is the midpoint of the range given by (Atkins 2008) of 20 to 100 LOC/day.
4. An analyst cost averaging £800 per day, based on our experience of market rates between £600 and £1000 per day.
5. 260 8-hour days per person-year, which is typical in planning to account for holidays, weekends, etc.
6. A 25% cost uplift to account for management of the work, verification activities, assurance activities, etc.

For compliance analysis, this produces a per-system estimate of ~33,300 person-days, or 128 person-years for a cost of ~£33.3m. For semantic analysis, this produces a per-system estimate of ~16,600 person-days, or 64 person-years for a cost of ~£16.7m. Note that these figures are still optimistic, because it also does not include the cost of developer time to support technical queries and sentencing meetings, which is typically charged to the licensee in addition to the development and supply costs for the C&I system. It also does not include the costs associated with the feasibility studies, tool developments or training associated with these activities.

In this example of a nuclear reactor protection system, these costs generally exceed the cost of developing and manufacturing the system itself¹⁵. Given the lack of safety benefit and the lack of meaningful improvement in confidence, we conclude that these activities are disproportionate to the associated time, efforts, and costs.

The presence of *post hoc* formal and semantic analyses in UK nuclear precedent is based on the technical landscape at the time of Sizewell B. Standards such as IEC 880:1986 (later to become BS EN 60880) did exist and were used as the basis for development of the Sizewell B PPS, but the functional safety standards relied upon in modern developments did not emerge until later¹⁶.

However, the technical community was (and to some extent still is) convinced of the need for formal methods to assure high-integrity systems, to the extent that the original publication around that time of INT.DEF.STAN.00-55 (the Ministry of Defence software standard) *mandated* the use of formal methods for defence software projects (Watts 1989). In this environment, the *post hoc* behavioural formal verification of the Sizewell B PPS makes some sense since it was regarded as good practice to have a formally verified system.

¹⁴ This change in scale itself causes us to question whether earlier precedent is applicable or appropriate.

¹⁵ In the case of smart devices, which typically cost on the order of hundreds or thousands of pounds, ICBM programmes alone typically cost several multiples more than this.

¹⁶ There is evidence of national standards preceding the first edition of IEC 61508, which was published in 1998. For example, IEC 61508 was heavily influenced by the 1989 German standard DIN V 19250. However, this was a national standard and largely used only in Germany and Central Europe and there is no evidence of its direct influence on the development and licensing activities for Sizewell B. In any event, it is widely accepted that IEC 61508 was the first complete functional safety standard, and this came several years after Sizewell B entered service.

Even this, however, would be to misrepresent the technical opinion at the time which was that the PPS should have been *developed* using formal methods. It is unclear that the approach of applying them after-the-fact was ever satisfactory to practitioners of the time. The modern technical landscape is also different, and formal methods find use for integrity-checking, but are still not commonly applied for formal behavioural verification when developing software, particularly for nuclear systems. Having mentioned it in the previous paragraph, it is notable that after initial publication, subsequent issues of DEF.STAN.00-055 successively weakened and then removed the mandate for formal methods. This reflects the fact that the expectation, prevalent in the late 1980s and early 1990s, that behavioural formal verification would become a standard engineering practice has not yet materialised.

Another type of static analysis is integrity-checking, which looks for vulnerabilities in the code that may reduce the systematic reliability of the system. The arguments against behavioural formal verification or semantic analysis do not apply to integrity-checking, where software is examined for compliance to a set of general properties or rules (e.g. checking for accesses to arrays out of bounds). This is because there is no specification to interpret and the analysis is therefore more objective and independent of the developer. For example, a result that mathematically demonstrates the possibility of array access out of bounds cannot be defended based on a misinterpreted specification.

There is also widespread tooling available for this activity at this level of formalism, suggesting inherent feasibility. Such tooling, such as Polyspace Code Prover, does not require an analyst to specify formal mathematical properties, which are instead built directly into the tools. These tools are also generally able, to some extent, to consider the entire code base, rather than individual functions, when checking integrity properties, which can reduce the number of false positives requiring sentencing. As a result, the skills required for the work are likely to be more widely available and the attendant time, costs and effort are likely to be proportionate to the benefit obtained.

However, this also means that they are in widespread use *within* safety system development lifecycles, because modern standards require or recommend them and because they afford a relatively cheap means of reducing risk of remedial work later in the development process, or after the system has entered service. If the work has already been done by the developer, and already independently assessed for adequacy (including whether the tools were configured correctly for the task) then it seems questionable whether benefits would arise from a further analysis of a similar nature. We will return to this point later in the paper.

The other form of static analysis that appears in UK nuclear precedent for systems with claims of 10^{-4} pfd are activities to confirm that the source-code compilation tools have not negatively impacted on the design intent expressed in the software source code. A technique called source-code comparison (Pavey and Winsborrow 1993) was devised and executed by Nuclear Electric (later EDF Energy) for the Sizewell B PPS. EDF Energy has adapted this method and is attempting to employ it at Hinkley Point C (AREVA NP and EDF 2012). In a sense, this is another form of integrity-checking, albeit a specialised one.

This method involved decompiling the object code, processing both it and the original source code and constructing MALPAS models that would allow a demonstration of semantic equivalence. This technique is proprietary and owned by EDF Energy and, crucially, must be reconstructed for each compiler and each processor type (which may have different instruction sets); compiler/processor-specific modelling is required in the equivalence-checking tool (Pavey and Winsborrow 1993). The technique at Sizewell B was achievable because of the limited optimisations performed by the compiler, and the project coding

standard constraints that avoided language constructs most likely to cause errors in the compiled code.

This allowed a clear source-object correspondence to be established at Sizewell B, which is key to making the technique tractable. Modern systems typically make use of optimisations, including whole-program optimisations, to meet performance requirements and other constraints (such as software size) which are likely to make automation of a complete formal demonstration of equivalence impossible, with consequent impacts on tractability. These complexities are being explored in the academic field of translation validation, but there is limited evidence of any industrial use and is largely an area of active research.

Of other industries examined, none has adopted such a formal approach to demonstrate equivalence between source and compiled image. In aviation, for DAL A software assurance under RTCA/DO-178C, there is a requirement to identify whether any additional code has been inserted by the compiler that is not directly traceable to the source code and to subject such code to additional verification (see Clause 6.4.4.2b of RTCA/DO-178C). However, this is linked to dynamic analysis activities, so the correspondence is established through a combination of test to establish correspondence from source code to executable object code and further test and/or analysis to examine any additional code that has been inserted and to determine its impact on the output. Extensive industrial tooling exists for this type of activity, but its formalism does not extend to that of source-code comparison.

The challenge of relying on precedent here, unlike for other static analysis methods explored in this subsection, is not so much the value of the technique if it can be performed¹⁷. The challenge is that it cannot be achieved for many modern systems to the same level of rigour as in preceding systems. A more extreme, but increasingly pertinent, example of this would be a system developed with FPGAs. FPGAs are developed using HDL, which is usually converted by a series of tools into a representation that reflects the configuration of the physical elements within an FPGA called the place-and-route netlist. This is then converted into a proprietary format called a bitstream using the FPGA vendor's tool, and this bitstream is what is used to configure the circuit.

However, the proprietary nature of the bitstream means that a technique like source-code comparison cannot be applied unless one has privileged access to the FPGA chip vendor's underlying intellectual property, which is not generally forthcoming¹⁸. This leaves an apparent gap in any justification because we are unable to “do what we have done before”, and it has caused significant difficulties in deploying this useful technology within the UK nuclear industry, likely resulting in a detriment to safety improvements. The error here is not that the conversion process should necessarily be ignored, but that by focusing on prescribed techniques as the principal means of demonstration, we can lose sight of what questions we are trying to answer to satisfy ourselves that the system is safe. The focus on following UK nuclear precedent also diverts qualification away from newer approaches such as those described above for avionics software, or the use of formally-verified compilers such as CompCert (2026).

¹⁷ Note however that exercising this technique still costs on the order of millions of pounds once analysis tool development is taken into account.

¹⁸ One tool, PV-Bit by Graf Research, claims to be based on such privileged access but the nature of the commercial arrangements involved makes it challenging for a licensee to qualify the tool for this use since the outputs cannot be systematically verified against the inputs.

Indeed, for all the static analysis techniques, this is an issue that continues to arise: why are these techniques being executed? What questions are they trying to answer? This topic will be addressed in Section 3.

2.5 Independent Dynamic Analysis

It is unclear when independent dynamic analysis became established as an ICBM (for example in Table 2 of NS-TAST-GD-046 (ONR 2023)), and there appears to be limited precedent for its actual execution in this form. Other techniques have their origins in the licensing of Sizewell B, but independently performed dynamic analysis was not among the five activities that comprised the independent assessment (Ward 1993). For the Protection System at Hinkley Point C, for example, the relevant part of the ICBM programme consists of an independent review of the dynamic analysis undertaken by the developer, rather than an independent re-execution of the work (AREVA NP and EDF 2012). The appearance of independent dynamic analysis among the techniques associated with UK nuclear precedent therefore does not seem to arise from its repeated application to systems on previous projects. This makes it particularly important to ask what safety concern the activity is intended to address before treating it as an expected part of a qualification programme. In the authors' experience, however, independent dynamic analysis is applied to off-the-shelf COTS products, such as smart devices as part of their ICBM programmes.

Dynamic analysis is the technique usually employed by the developers of safety-critical or high-integrity systems, because it is required in relevant standards. It consists of executing requirements-based tests across units of code, or aggregations of such units, with the aim of demonstrating that the software complies with its design requirements. Separately, standards typically also require that some structural coverage metric is assessed to ensure that there is no code that is not traced to requirements and that other features such as conditions within the software are completely exercised.

As with static analysis, there is then an open question of what a completely independent execution of dynamic analysis is trying to demonstrate. Such an activity would involve one of:

1. Re-executing the analysis on a different tool;
2. Re-executing the analysis using different people;
3. Targeting different requirements; or
4. Some combination of the above.

If it is true that using a diverse set of tools and people could identify more issues, then this suggests that there is something wrong with the tools used by the development team (e.g., an inappropriate tool or incorrect configuration of it), or that there are process issues in its execution. We have already seen that modern safety standards require tool qualification (which should identify capabilities and limitations of the tool); and independent assessment should confirm the competence of personnel and the correctness of the tool qualification. If configuration and use of the tool is a specific concern, then it is more efficient to check that the configuration/usage was correct than to completely re-execute a verification programme from scratch. Independence is also of limited benefit, since a compliant development process would already require appropriate independence of the development and verification teams. Independent execution also relies heavily on developer interpretation of the requirements when findings are made, so this suffers similarly to the authority bias effects described for post-development behavioural formal verification.

In addition, modern test environments (and test harnesses) can be large, requiring significant implementation effort by the analyst, and cost. This can be tackled by independent re-implementation by the analyst or through re-use of developer artefacts (harnesses, etc.), the latter of which may diminish independence claims.

If the intent is to target different requirements, then it is unclear in general what these should be. Precedent for smart devices in the UK nuclear industry suggests that dynamic analysis may be targeted at robustness properties, or other areas of particular interest. However, even if these are not already addressed by standards that promote fault injection, stress testing and related techniques, it may be that simply asking the developer to perform more tests is a more straightforward route to achieving this. The key here is the independence and diversity of thought involved in devising the tests, not the way they are executed.

There is therefore no obvious benefit to an independent repetition of dynamic analysis in terms of improvements in safety or in increasing confidence, particularly if in addition to an adequate independent assessment of the original analysis. Since the costs will be of a similar order-of-magnitude to the developer's work themselves, as well as the need to develop a test environment, generate independent understanding of the code structure and licensing tools, it is apparent that this would be a substantial undertaking for the negligible value in outcome.

Independent dynamic analysis, like static analysis, focuses on software implementation. This focus gives rise to a number of other objections and detriments to safety, which are discussed in the next sub-section.

2.6 Perils of *Post Hoc* Source Code Analysis

Post-development source code analyses of developed software do not address the key risks associated with software, which lie in the specification and design of the overall system and how this is translated into software requirements, design and integration, rather than the implementation of the software itself.

Figure 2 shows the target of the source code-level ICBMs from precedent in context with the overall lifecycle of a C&I system. As stated previously, this highlights that only the implemented software is examined, but none of the preceding steps that dictate what that implementation is intended to achieve.

Any assurance derived from these activities, by focusing only on a demonstration that an implementation conforms to its specified behaviour that has already been made by the developer through other means, therefore provides limited benefit in understanding risk or demonstrating that it has been managed. This is because these activities provide limited insight into the behaviour of the system when the underpinning assumptions (such as system specification correctness and completeness, or interface guarantees) may be violated.

Incorporating post-development source code-level verification activities can also inhibit the longer-term achievement of safety. In the life of a C&I system, modifications to the software may be desirable, for example, to implement safety improvements to the system based on operational experience; to allow for wider plant modifications that have a safety benefit; or to patch the software to address identified security vulnerabilities, or changes in the threat landscape. However, requiring specific additional activities can introduce a bar to modification so high that such safety improvements will not be made because of the associated costs of re-executing analysis and the difficulty and costs associated with maintaining a pool of independent experts able to undertake them.

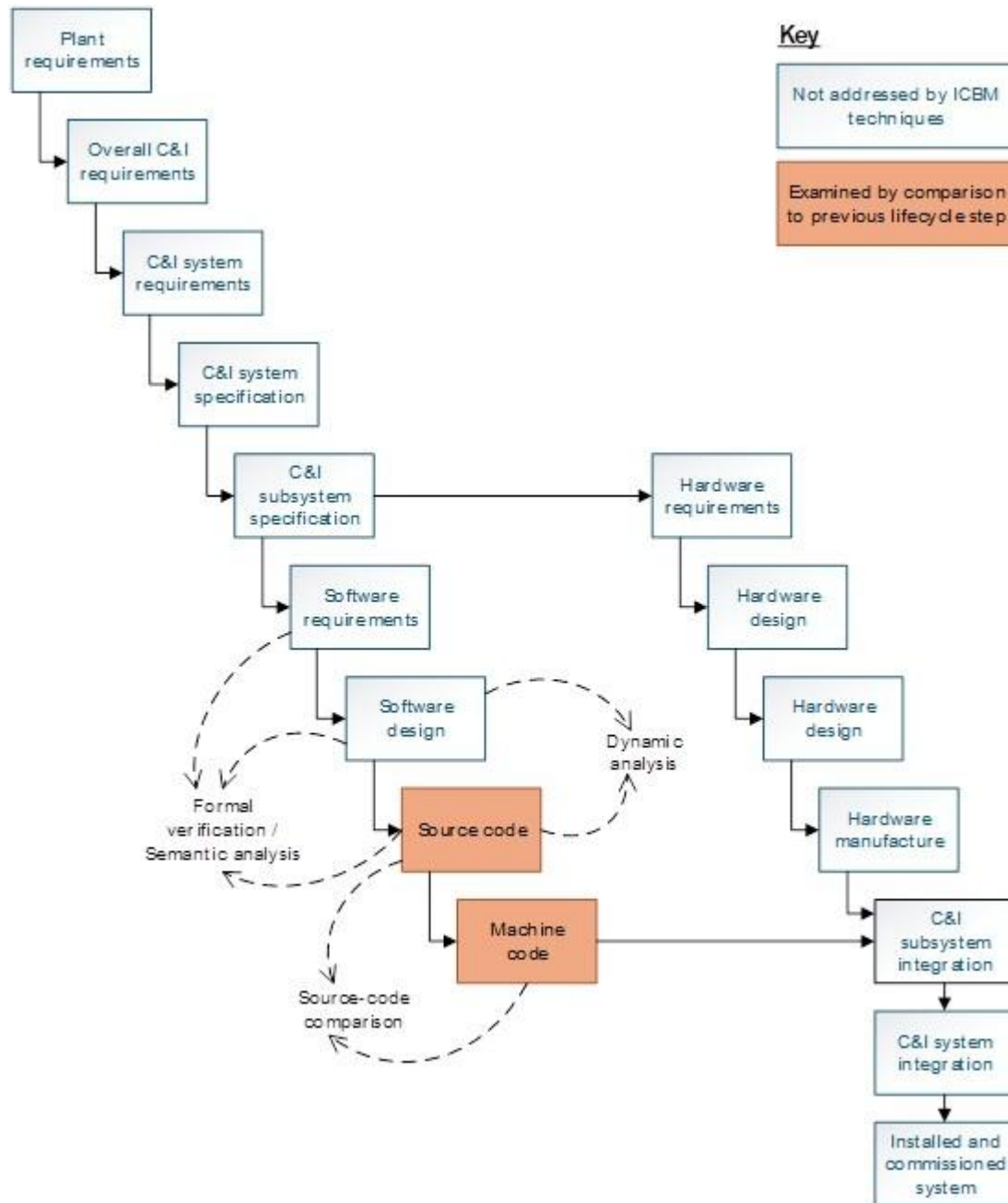


Figure 2 ~ C&I System Lifecycle Showing Focus of ICBM Source-Code Techniques

The effect of this is to add substantial costs into the consideration of whether an in-service improvement is regarded as reasonably practicable. This challenges whether an improvement contributes to reducing risk SFAIRP. Where improvements would be beneficial, but are essentially optional, the additional *post hoc* analysis costs will tend to encourage an operator not to implement them on the grounds of excessive additional cost and effort. This decision will be driven as much by perception of complexity as by a genuine evaluation of costs, but the effect will be that any such incremental improvements to plant safety are unlikely to be made.

The ICBM approach also tightly couples the demonstration of safety to specific tools and techniques. Over the life of a plant or system, advances in technology or methods are hard to adopt because it requires fundamental change to the entire basis of substantiation. The

licensee must maintain the skilled workforce and tooling required to be able to maintain the system (tool obsolescence being an inevitable problem over the lifetime of a nuclear facility), including executing what are often specialist techniques, thus incurring additional costs and barriers to modification through the multi-decade life of the facility. This is a challenge for the development tools and substantiation tools, which, particularly in the case of ICBMs for high-integrity systems, are more likely to be bespoke, further increasing the burden.

It would instead be preferable to adopt a goals-based approach, so that new techniques and methods could be adopted and used to drive later substantiation. Contemporaneously with Sizewell B's licensing, it was noted that the nature of the MALPAS analysis meant that it was process-based (Bruns and Anderson 1995). For example, the analysis is performed uniformly across the code rather than placing additional scrutiny on critical areas, or driving the analysis based on system requirements.

However, to adopt a goals-based approach, one must have some understanding of what the goals are, which is a topic we will revisit in Section 3.

2.7 Statistical Testing

Statistical testing is the only system-level activity that appears in ONR guidance for ICBMs (ONR 2020) (ONR 2023), which purports it to be a means of demonstrating the numerical reliability of the system after it has been validated¹⁹. This is an unusual inclusion in the first instance, because, despite its presentation as good practice within the regulatory guidance, there is no precedent for its use in licensing protection systems and yet there is strong emphasis on it being needed for Class 1 systems. The PPS at Sizewell B was not licensed based on statistical testing (May et al. 1995) (Hunns and Wainwright 1992)²⁰. The proposed statistical testing at Hinkley Point C has not yet been performed and has reported encountering challenges with practicability, leading to divergence from pure statistical testing (Walker 2024). The problems with statistical testing are outlined in greater detail in (Rendell and Pollard 2026) but are summarised here.

Rendell and Pollard (2026) argue that the approach proposed by NS-TAST-GD-046 (ONR 2023) is insufficient in test volume to demonstrate the safety claims placed on multi-function protection systems. NS-TAST-GD-046 proposes ~4,600 tests on the system in order to demonstrate evidence of 10^{-3} pfd for the system, yet in reality the claims for reliability are on the lower-level safety functions in the plant engineering schedule that are allocated to the system. These functions' individual reliability claims cannot be aggregated into a single reliability figure for the system. Instead, Rendell and Pollard (2026) argue that the correct calculation should be ~4,600 tests of *each* of these functions. To suggest otherwise is to offer developers a perverse incentive to collect as much functionality as possible into a single but complex system to minimise the volume of testing required. For example, if a developer

¹⁹ The discussion in this subsection is primarily focussed on the use of statistical testing as described in, for example, BS EN 61508-7:2010, Annex D, performed on the integrated, validated system, for the purposes of providing evidence of the system's reliability in performing its safety functions. This is the form of statistical testing utilised by ICBM programmes. This discussion does not discuss statistical testing as done during the development lifecycle for the purposes of subjecting software or units to high-volume tests, potentially in simulation. The "core conditions" of statistical testing for producing a reliability claim, as utilised in UK nuclear precedent, preclude the use of simulation due to the need to run tests on the same hardware and software as the operational system — research into use of simulation for reliability claims of this type is incomplete, e.g. May (2017). This subsection does not comment on large-scale (potentially simulated) random testing that may not require an operational profile.

²⁰ A statistically informed testing campaign was performed after licensing, but available papers imply that it was effectively only performed to ~40% statistical confidence, with this testing only comprising suites of 5,000 tests for the 10^{-4} pfd reliability target.

has two systems with claims of 10^{-3} pfd each, then based on NS-TAST-GD-046, each requires ~4,600 tests for a total of 9,200 tests to demonstrate the reliability of its functions. If the developer combined the systems into one, they could limit themselves to 4,600 tests, and so save considerable time, cost and effort to apparently claim the same level of reliability for all the combined functions of both systems with half the volume of testing²¹. This is intuitively wrong and yet it is the logical conclusion of targeting a “system reliability” that has no meaning, compared to the actual safety claims, which are the reliabilities of each function. Given that, for example, reactor protection systems have multiple safety functions (tens to hundreds), these necessary test-volumes are likely to be impracticable, given that each test is likely to be on the order of minutes to tens of minutes, and use of simulation is unproven for this specific purpose (Rendell and Pollard 2026).

As described in Rendell and Pollard (2026), even if the test regime were executed, the inability to rigidly meet the assumptions required by statistical testing to achieve a reliability measure mean it is likely unknowable what level of confidence in the reliability claim is achieved. Further, deliberately diverging from pure statistical testing methods, e.g. to aid practicability, has further unknowable effects on the numerical outcome. A technique with an unclear outcome cannot meaningfully contribute to a genuine improvement in confidence, and so it is unclear what is achieved by a statistical test of this type.

Similar to the discussion on post hoc analysis of source code earlier, the ICBM-type approach of requiring statistical testing as part of software qualification (ONR 2011) potentially adds an additional disproportionate hurdle to future software modifications. If a modification requires a partial or full run of a statistical testing campaign, then expertise is needed to, for example:

1. Determine what level of testing is needed — a full statistical testing campaign, a reduced-size campaign, i.e. lower statistical confidence, a reduced-scope campaign, e.g. only testing certain fault scenarios? This will require specialist statistical testing expertise and impact analysis of the system modification.
2. Assess and potentially update the operational profile to account for any plant changes in the interim period.
3. Update the oracle to reflect any changes to the system under test.
4. Modify the test harness to accommodate signal changes.

Available research indicates that an answer to point #1 is not straightforward (Korhonen et al. 1997), but in some models may require a larger test suite than the original size target to be run after discovery and fix of a defect, and be dependent on when in the suite the failure occurred (Littlewood and Wright 1997). That a single line of code being modified may impact the whole system is challenging, especially when statistical testing is intended to be a black-box technique.

The test equipment and software may become obsolete faster than the system being tested, meaning that some of the above may even need to be rebuilt from scratch (this might be particularly pertinent given the earlier discussion about the disincentive for making modifications unless exceptionally necessary). In any case, the amount of effort needed is likely to be significant, thus adding further disincentive to make a modification, noting that

²¹ If the two hypothetical functions are coupled, such that they would both be demanded by every test case in either of these hypothetical scenarios, then it may be justifiable to cover them both in the same system with only 4,600 tests. However, this is a limited special case and, for example, it is highly unlikely that any realistic test cases exist that would demand every function in a reactor protection system in the same test.

to achieve full statistical rigour, even a very minor modification may warrant a full test campaign.

Precisely estimating the time, effort and cost associated with a statistical test campaign is difficult without a specific reference system because there is a dependency on the duration of testing, which is unknown at this stage of the development and depends on the reliability being claimed. However, there are significant elements of effort and cost that can be estimated to a degree because they arise regardless of the claimed reliability. For example, for a reactor protection system:

1. At least one division of the system must be purchased for testing, in addition to the ones that will be deployed on the plant. The costs for this can be estimated to be of the order of £10m.
2. A test harness must be procured and/or developed. For a multi-functional system, this is estimated to be effort on the order of 10 person-years, with equipment costs ranging from the hundreds of thousands to millions of pounds.
3. Software for the test case generation, oracle and output comparator must be developed, which is estimated to involve development effort on the order of 5 person-years.
4. The event design itself for the operational profiles is estimated to be on the order of tens of person-years.

Additional to this are the costs and effort associated with the execution of the testing itself, with both factors rising with overall test duration because testing cannot be wholly unattended. There will be additional hardware costs associated with the testing, including replacement parts for the test harness, and effort associated with maintaining calibration. There are also effort and costs associated with the supplier of the system, with whom substantial engagement is required to integrate the system into the test harness²². In conjunction with the effort and costs established above it is sufficient to observe that the overall effort associated with statistical testing of a single system will be of the order of tens of person-years, and costs in the order of tens of millions of pounds.

The key aspect is that this expense would go to answer a question about the safety of the system that we are not asking. We want to know how reliable each of our safety functions are, but this is not what this expensive technique is able to practicably tell us.

2.8 Conclusions

This section has examined the precedent for applying verification techniques to software after the completion of development, and the use of statistical testing. In the manner performed in an ICBM programme, these are generally found to at best provide a false confidence based on what they purport to demonstrate and do not provide any meaningful benefit to safety, nor to establishing confidence in the achievement of safety at a plant level. The time, effort and cost of these activities as applied to a modern C&I system has also been examined and found to be disproportionate to any negligible improvement in safety or confidence in safety that is achieved. As a result, these techniques are judged to not be useful precedents to adopt in the qualification of modern C&I systems and may actually be detrimental to safety because of their impact of the preservation of qualification through the

²² This interaction with the supplier should also be handled carefully, to avoid “polluting” the independence in a manner similar to that which was described for static analysis.

life of the system, and because the rigid desire for precedent prevents modernisation of existing facilities and new designs.

In each case, the examination has turned to the question of what it is we are seeking to demonstrate by applying each technique. The answer we have landed on in general is that we are not answering anything beyond “*does this piece of software behave in accordance with the evidence we already have?*”. However, this is a common problem with the ICBM approach based on precedent. It has been, uniquely within UK nuclear regulation, an in-effect prescriptive regime, which contradicts the goal-based approach upon which substantiation is meant to be based. As a result, every licensee or applicant for reactor design assessment has formulated a different question to which the ‘required’ ICBM techniques are the answer. It is more appropriate to start with the goal and determine the evidence on that basis. This is explored further in the next section.

3 What Question Must Be Answered?

In Section 2, we examined the techniques that have generally been applied to systems based on the precedents set at Sizewell B in the early 1990s, and concluded that they offered at best negligible benefit to safety or justifiable improvement in confidence in the safety of the system. The general approach to this form of independent re-verification after-the-fact was also critiqued because it acts as an inhibitor to safety improvements through the life of the plant.

A reasonable challenge might be that we have not considered alternative techniques and that if some further examination of techniques was undertaken, replacement techniques could be identified that would be more appropriate. However, the more fundamental question is why additional techniques are required at all — what is the principled basis upon which to decide to do more work than the original development and suitably-independent verification required by modern safety standards?

A view is taken in other jurisdictions and in other high-hazard industries that a development process according to a modern, safety-oriented standard, graded according to the safety significance of the system, is sufficient to ensure that the system is adequately dependable within its application. There is an argument that can be made here, where the system and its software have been developed in a lifecycle that begins with a plant-level hazard analysis (the plant design basis in nuclear jargon) and proceeds through a structured lifecycle of safety function definition, system allocation, development, validation, installation, and commissioning. In this case, all decisions and all of the implementation can be traced directly to reducing plant risk and all the evidence generated by the development lifecycle will ultimately trace to this purpose.

This argument may fall down in practice, especially in modern systems where much of the implementation (hardware, software, and HDL) is pre-developed. This could be pre-developed in the form of a platform (like a programmable logic controller or some other computerised platform), or in the form of a commercial off-the-shelf product providing some functionality for the system, e.g. a sensor transmitter. In these cases, the development lifecycle is focused on implementing capabilities that anticipate the needs of future users by following lifecycles that are intended to control or avoid latent systematic faults. Although these capabilities may be essential for the correct operation of the system and may have been developed well, they are somewhat disconnected from the plant-level hazards that the

system is trying to prevent, or the plant-level hazards that spurious operation of the system may itself induce. We have in this case therefore arguably lost traceability to plant-level hazards, in that requirements for various system components are no longer directly driven by the hazard analysis process. Furthermore, there should be concern about the interaction between the various pre-existing software elements as well as between these elements and any application-specific development.

We propose that the goal with CBSIS in general is to demonstrate that:

The risks associated with software and/or programmable logic used in implementing a safety function have been reduced so far as is reasonably practicable and to a sufficient level to support safe system operation.

Satisfying this goal requires an understanding of how the intended and unintended behaviours of the CBSIS could affect plant safety, how consequential behaviours are prevented or controlled, and whether the available evidence is adequate for the safety significance of those behaviours.

This is not a goal addressed by the ICBM approaches in UK nuclear precedent. These look only at software implementation correctness with respect to the software specification, or require some degree of system-level testing that does not focus on safety functions and their implementation directly. They do not examine risk properly, and they do not examine behaviours that may emerge in systems combining software and programmable logic of different origins. It is in this respect that they are extremely limited.

This goal could potentially be addressed by the independent assessment of the development process, as that forms a judgement on the level of safety achieved. However, it is not obvious that the structure of development evidence, particularly where derived from a supplier audit in the case of COTS products necessarily wholly underpins that demonstration on its own.

As part of ongoing study and regulatory engagements in this area, the authors have instead proposed that the plant-level hazard analysis is linked to a software architecture failure analysis (performed using, for example a software interface failure modes and effects analysis for simple software architectures) via successive levels of failure analysis on the system, subsystems and components. This would permit the identification of controls within the overall system that prevent the propagation of postulated software failures, and those software failure modes that can propagate to the plant level with a consequential effect.

Armed with this analysis, each claimed propagation control can be closely examined to see if its specification and verification confirm that the failure that we are trying to control will be prevented. Any unmitigated postulated software failure modes can draw the assessor's attention to the existing verification evidence to explore whether these failure modes have been adequately addressed by the existing development work, using techniques and measures that are appropriate given the safety significance.

In this way, the evidence from the development lifecycle is re-purposed to a demonstration that the risks from the software to the plant have been adequately controlled. If there are gaps in this demonstration, it may highlight the need for extensions to the existing evidence (such as increasing the number of unit tests) or completely new analysis. This work remains in progress, and we hope to report further on it once we have trialled it on a design. However, it is not entirely novel, being an adaptation of the software architectural safety analysis described in Annex E of BS ISO 26262-6, augmented to allow it to be applied after-the-fact.

Nonetheless, given its lack of maturity, the approach outlined above is not presented as a universal replacement qualification method, which in any case would be counter-intuitive in a paper where we seek to identify problems with a prescriptive approach to qualification. Its purpose here is to demonstrate that alternatives to routine independent re-verification exist, and that these alternatives can begin with a consideration of system behaviour rather than with a predetermined set of techniques.

Care is needed not to replace one form of prescription with another. If our approach were adopted simply because it had become the new precedent, without identifying the question it was intended to answer, the same underlying problem would remain to some extent. The appropriate analysis or verification activity will depend on the system, its application, the available development evidence and the uncertainties material to the plant safety case.

Earlier in this section, we proposed a possible goal, but we do not claim that this is necessarily the best or most complete expression of it. It may require refinement to distinguish between demonstrating that a system is adequately safe and demonstrating that further risk reduction would not be reasonably practicable. Nevertheless, expressing a goal of this kind changes the basis on which assurance activities are selected. It allows a licensee to begin with plant risk, identify the claims and uncertainties associated with the CBSIS, examine the available evidence and undertake further work where that evidence does not adequately support the safety case.

By contrast, the UK approach to date has used precedent for very specific confidence-building measures without clearly articulating the residual safety concern that those measures are expected to address after standards-based development and independent safety assessment have been completed²³. Licensees are consequently encouraged to begin with historical techniques and then explain how they integrate into the overall safety argument (ONR 2011), rather than beginning with a safety claim or material uncertainty and identifying a suitable means of addressing it.

Regulatory guidance would therefore be more useful if it expressed the objective of any additional software qualification activity. Such an objective should identify what remains to be demonstrated, if anything, beyond production excellence and independent functional safety assessment as defined in standards, while allowing the licensee to propose evidence appropriate to the system, its application and its safety significance. This would preserve independent regulatory challenge without turning the methods used on one project into assumed requirements for the next.

The tentative approach described in this section may not be the best means of satisfying such an objective. More fundamentally, the goal proposed by the authors may not itself be the right one. This uncertainty does not justify reverting to precedent. Rather, it demonstrates the need to articulate and debate the regulatory goal before selecting the techniques by which it is to be satisfied.

Before asking what should replace ICBMs from precedent, it is therefore necessary to establish the unresolved safety concern that additional assurance is intended to address, and the goal against which any proposed activity will be judged. If these cannot be expressed sufficiently clearly to distinguish relevant evidence from irrelevant activity, there is no principled basis for requiring either the historical techniques or replacements for them.

²³ Or, as noted in this paper, some of the precedent is not even that – such as the “need” for statistical testing of reactor protection systems despite it never being done previously for licensing, and the “need” for independent execution of dynamic analysis, despite this technique historically being performed as a development-lifecycle activity.

4 Conclusions

This paper has examined independent static analysis (covering its various forms), independent dynamic analysis, and statistical testing, because these techniques have been repeatedly included in UK nuclear software qualification as activities performed after the development process has concluded; in many cases based on the precedent established during the licensing of Sizewell B. Although precedent can provide useful experience, repeated application has allowed these techniques to effectively become expectations without sufficient reconsideration of the technological context in which they arose or the safety purpose they are intended to serve. Where modern standards-based development, verification, validation, and independent functional safety assessment have already been performed, their routine use as additional independent confidence-building measures provides little additional safety benefit, or justified improvement in confidence relative to their considerable cost, effort, and through-life consequences.

This is not an argument against independent assessment, nor against undertaking additional analysis and testing where these are needed. It is an argument that additional work should address an identified safety claim, material uncertainty, or weakness in the available evidence. Independence of judgement should not be equated with wholesale repetition of development activities.

It is also not an argument that every project should deliberately start afresh and derive its approaches from first principles. Good practice should be recognised, and adopted where appropriate, and the experience of earlier projects should be used to improve the quality and safety of future ones. The argument is that “good” precedent for safety systems should have a material impact on the levels of safety achieved in the context of the system under consideration, primarily via the development process; blindly following precedents on the grounds that it is what has always been done is antithetical to the goal of improving safety overall. Our conclusion is that the precedent for the ICBMs established at Sizewell B is not appropriate in the modern technical landscape.

The possible goal- and failure-based approaches outlined in this paper are intended to illustrate that other ways of reasoning about software risk are available, not to prescribe a replacement set of techniques. The more important need is for the objective of any qualification beyond international standards to be stated clearly. Only then can licensees select proportionate evidence appropriate to the system, its application, and its safety significance; and regulators can then judge whether that evidence is adequate.

Precedent can provide useful experience, but it should not determine the answer before the safety question has been asked. Before requiring either established ICBMs or replacements for them, it is necessary to identify the unresolved safety concern and the goal that assurance additional to that performed as part of a modern safety development lifecycle is intended to satisfy. Without that foundation, there is no principled basis for determining what additional assurance activity might be appropriate, judging its adequacy or deciding whether it is needed at all.

Disclaimers

Opinions expressed herein are those of the authors, not necessarily those of their past and present employers or clients.

References

- AREVA NP, and EDF. (2012). *PCSR – Sub-chapter 7.7 — I&C tools, development process and substantiation*. Available from https://www.edfenergy.com/sites/default/files/sub-chapter_7.7_-_iandc_tools_development_process_and_substantiation.pdf. Accessed 14th July 2026.
- Atkins. (2008). *MALPAS - Advanced Software Analysis and Verification — Introductory Guide*. Available from <http://web.archive.org/web/20240301202919/http://malpas-global.com/wp-content/uploads/download/MALPAS%20introductory%20guide.pdf>. Accessed 13th July 2026.
- Bruns G, and Anderson S. (1995). *Gaining Assurance with Formal Methods*. In: *Applications of Formal Methods* (pp. 33-54). Upper Saddle River, USA: Prentice Hall.
- BS EN 60880. (2009). *Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions*. BS EN 60880:2009. British Standards Institution, London.
- BS EN 61508-4. (2010). *Functional safety of electrical/electronic/programmable electronic safety related systems. Part 4: Definitions and abbreviations*. BS EN 61508-4:2010. British Standards Institution, London.
- BS EN 61508-7. (2010). *Functional safety of electrical/electronic/programmable electronic safety-related systems Part 7: Overview of techniques and measures*. BS EN 61508-7:2010. British Standards Institution, London.
- BS EN 61513. (2013). *Nuclear power plants – Instrumentation and control important to safety – General requirements for systems*. BS EN 61513:2013. British Standards Institution, London.
- BS EN IEC 62138. (2019). *Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category B or C functions*. BS EN IEC 62138:2019. British Standards Institution, London.
- BS EN IEC 61226. (2021). *Nuclear power plants – Instrumentation, control and electrical power systems important to safety – Categorization of functions and classification of systems*. BS EN IEC 61226:2021. British Standards Institution, London.
- BS EN IEC 62138. (2019). *Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category B or C functions*. BS EN IEC 62138:2019. British Standards Institution, London.
- BS ISO 26262-6. (2018). *Road vehicles — Functional safety. Part 6: Product development at the software level*. BS ISO 26262-6:2018. British Standards Institution, London.
- CompCert. (2026). Website: <https://compcert.org/>. Accessed 27th August 2026.
- Hunns D. M, and Wainwright N. (1992). *Software-based protection for Sizewell B: The regulator's perspective*. In: International Conference on Electrical and Control Aspects of the Sizewell B PWR, (IEE Number no.361), Institution of Electrical Engineers, British Nuclear Energy Society, Institute of Energy, Institution of Mechanical Engineers, and Convention of National Societies of Electrical Engineers of Western Europe (EUREL), London.

- Korhonen J, Pulkkinen U, and Haapanen P. (1997). *Statistical reliability assessment of software-based systems*. Finnish Centre for Radiation and Nuclear Safety. Available from <https://inis.iaea.org/records/0sccc-2x936/files/28028382.pdf>. Accessed 26th September 2026.
- Littlewood B, and Strigini L. (1993). *Validation of ultrahigh dependability for software-based systems*. Communications of the ACM 36(11):69-80. Available from <https://openaccess.city.ac.uk/id/eprint/1251/1/CACMnov93.pdf>. Accessed 16th September 2026.
- Littlewood B, and Wright D. (1997). *Some conservative stopping rules for the operational testing of safety-critical software*. IEEE Transactions on Software Engineering 23(11):673-683.
- May J. H. R. (2017). *Theoretical feasibility of statistical assurance of programmable systems based on simulation tests*. In: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, Czech Republic.
- May J. H. R, Hughes G, and Lunn A. D. (1995). *Reliability estimation from appropriate testing of plant protection software*. Software Engineering Journal, November 1995:206-218.
- ONR. (2011). *Generic Design Assessment – New Civil Reactor Build: Step 4 Control and Instrumentation Assessment of the EDF and AREVA*. Office for Nuclear Regulation. Office for Nuclear Regulation, Assessment Report ONR-GDA-AR-11-022. Available from <https://www.onr.org.uk/media/tz0lphdm/ukepr-ci-onr-gda-ar-11-022-r-rev-0.pdf>. Accessed 26th September 2026.
- ONR. (2020). *Safety Assessment Principles*. Office for Nuclear Regulation. Available from <https://www.onr.org.uk/media/pobf24xm/saps2014.pdf>. Accessed 26th September 2026.
- ONR. (2023). *Computer Based Safety Systems*. Office for Nuclear Regulation. Technical Assessment Guide NS-TAST-GD-046. Available from <https://www.onr.org.uk/media/3def2aat/ns-tast-gd-046.docx>. Accessed 16th September 2026.
- Pavey D. J, and Winsborrow L. A. (1993). *Demonstrating Equivalence of Source Code and PROM Contents*. The Computer Journal, 36(7), 654-667.
- Rendell M, and Pollard F. (2026). *Statistical Testing: Qualitative Arguments for a Quantitative Outcome?* Safety Critical Systems eJournal, Vol. 5, Iss. 1. Available from <https://scsc.uk/journal/index.php/scsj/article/view/64/53>. Accessed 11th October 2026.
- RTCA/DO-178C. (2012). *Software Considerations in Airborne Systems and Equipment Certification*. RTCA Inc., Washington DC.
- Walker L. (2024). *Case studies in operational profile based statistical testing for the quantitative reliability estimation of software based commercial off the shelf (COTS) equipment*. Technical Meeting on Commercial Equipment in Nuclear Applications and Non-Nuclear Standards for Safety Instrumentation and Control Systems, 1st—3rd July 2024. International Atomic Energy Agency, Vienna.
- Ward N. J. (1993). *The Rigorous Retrospective Static Analysis of the Sizewell 'B' Primary Protection System Software*. SAFECOMP '93, (pp. 171-181).

Watts S. (1989). *Software row dogs nuclear power plans*. New Scientist, April 1st 1989. Available from <https://www.newscientist.com/article/mg12216580-200-software-row-dogs-nuclear-power-plans/>. Accessed 14th July 2026.

Wichmann B, Canning A, Clutterbuck D, Winsborrow L, Ward N, and Marsh D. (1995). *Industrial perspective on static analysis*. Software Engineering Journal, 10(2), 69-75.