SAFETY JUSTIFICATION FRAMEWORKS: INTEGRATING RULE-BASED, GOAL-BASED AND RISK-INFORMED APPROACHES

Peter Bishop, Robin Bloomfield and Sofia Guerra
Adelard LLP
3-11 Pine Street, London EC1R 0JH, UK
pgb@adelard.com; reb@adelard.com; aslg@adelard.com

Nguyen Thuy
EDF R&D
6, Quai Watier, 78401 Chatou, FRANCE

ABSTRACT

The reliability and safety of the digital I&C systems that implement safety functions are critical issues. In particular, software defects could result in common cause failures that defeat redundancy and defence-in-depth mechanisms. Unfortunately, the differences in current safety justification principles and methods for digital I&C restrict international co-operation and hinder the emergence of widely accepted best practices. These differences also prevent cost sharing and reduction, and unnecessarily increase licensing uncertainties, thus creating a very difficult operating environment for utilities, vendors and regulatory bodies.

The European project HARMONICS (Harmonised Assessment of Reliability of MOdern Nuclear I&C Software) is seeking to develop a more harmonised approach to the justification of software-based I&C systems important to safety.

This paper outlines the justification framework we intend to develop in HARMONICS. It will integrate three strategies commonly used in safety justifications of I&C system and its software: *rule-based*—evidence of compliance to accepted standards; *goal-based*—evidence that the intended behaviour and other claimed properties has been achieved; and *risk-informed*—evidence that unintended behaviour is unlikely. The paper will present general forms of safety case that can be adapted to a variety of specific topics.

Key Words: Safety justification, I&C systems, Software

1 INTRODUCTION

The reliability and safety of the digital I&C systems that implement safety functions are critical issues. This is in particular due to the fact that software can usually not be proven to be completely defect-free, and that postulated residual defects could be suspected of leading to common cause failure that could defeat redundancy and defence-in-depth. Unfortunately, the differences in current safety justification principles and methods restrict international co-operation and hinder the emergence of widely accepted best practices. They also prevent cost sharing and reduction, and unnecessarily increase licensing uncertainties, thus creating a very difficult operating environment for utilities, vendors and regulatory bodies.

The paper will present research work currently being undertaken within the European project HARMONICS (Harmonised Assessment of Reliability of MOdern Nuclear I&C Software) on justification frameworks for software-based I&C systems important to safety.

Section 2 describes the technical basis for structuring a safety justification. Section 3 illustrates the approach applied to a much simplified I&C system. Our preliminary conclusions are given in Section 4.

2 TECHNICAL APPROACH TO SAFETY JUSTIFICATION

2.1 Justification strategies

One very widely used approach to justifying an I&C system and its software is to provide evidence that they have been designed and verified following a well-structured development process, and applying the requirements and recommendations of rigorous standards. This type of justification approach is often called the *rule-based approach*, as it relies on the application of pre-defined rules.

The rule-based approach works well in stable environments where best practice was deemed to imply adequate safety. However, it does not always provide direct evidence that the I&C system and its software achieve the behaviour or the properties required, to the desired level of reliability. In addition, there might be cases where one finds no suitable set of applicable rules that can confer the desired level of confidence in a given property or that can be demonstrated to have been met. To overcome these, a more goal-oriented justification approach may be applied, where the justification explicitly demonstrates that the desired behaviour, property or reliability has been achieved. This type of approach can be called the *goal-based approach*.

Lastly, particularly for safety systems, one also needs to justify that the I&C system and its software will not have unacceptable behaviour, that the postulated hazards have been addressed, and that the risks incurred by the remaining vulnerabilities have been reduced to an acceptable level. This type of approach can be called the *risk-informed approach*.

One important aim of HARMONICS is to improve safety justifications by integrating these three approaches (rule-based, goal-based, and risk informed) to get a coherent process for justifying software-based I&C systems.

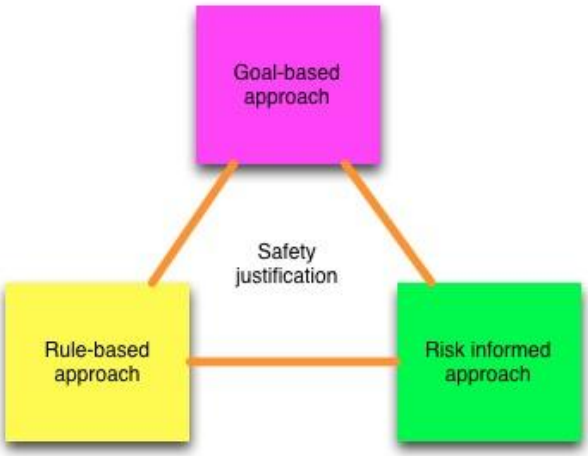

Fig. 1. Integrated justification strategy [9]

The exact form of the safety case based on these approaches depends on the application and on negotiation by the parties involved, but the paper presents general forms of safety justifications that can be applied to a variety of specific subjects.

2.2 Structuring a safety justification

Over the past 10 years there has been a trend towards an explicit goal-based approach to safety justification and considerable work has been done on the structuring of safety arguments [1, 2, 3, 4, 5, 7]. The approach to be used in the HARMONICS project is the CAE (Claims Argument Evidence) approach [1, 2, 3]. It supports the description of how sophisticated engineering arguments are actually made. The key elements of the approach are:

- Claims, as the name suggests, are assertions put forward for general acceptance. They are typically statements that we are adequately confident that a system or some subsystem has a certain property.

- Evidence that is used as the basis of argument. This can either be facts (e.g. based on established scientific principles and prior research), assumptions, or sub-claims, derived from a lower-level sub-argument. Sources of *evidence* may include the design, the development process, prior field experience or source code analysis.

Arguments link the evidence to the claim. They are the "statements indicating the general ways of arguing being applied in a particular case and implicitly relied on and whose trustworthiness is well established" [8], together with the validation for the scientific and engineering laws used.

The basis elements of a safety justification are shown in Fig. 2 below.

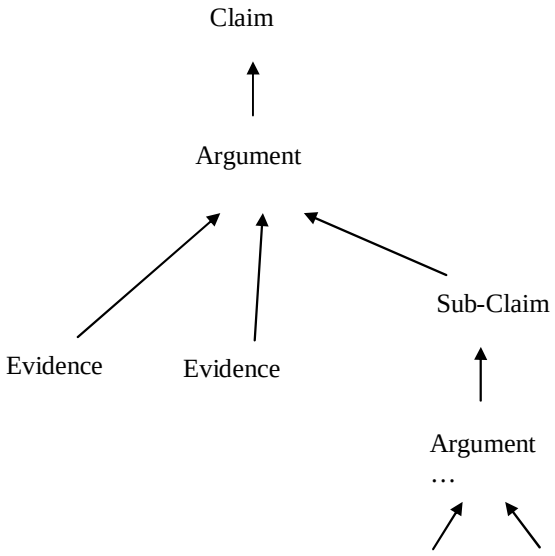

Fig. 2. CAE safety justification structure

Rather than linking evidence directly to the claims via an interposing argument, the structure can be extended by incorporating sub-claims that can be expanded into further arguments and sub-claims. The final nodes of the justification should always be supporting evidence.

There are a number of well-founded methods for decomposing claims into sub-claims including:

- architectural decomposition, where a claim about the system is decomposed into sub-claims about the components and their interconnections,
- functional decomposition, where a system-level function is partitioned into sub-functions,
- enumeration (often used in rule-based and risked based approaches), where the relevant items are

identified and then addressed by supplying evidence,

- attribute decomposition, where a claim about the behaviour of the system is decomposed into sub-claims about different aspects of the behaviour (like timeliness, accuracy, robustness).

The argument has to justify that the decomposition into sub-claim is valid, and in a rigorous justification, the decomposition would be related to some model of the system design.

This approach was developed in earlier EU-sponsored research projects (SHIP and CEMSIS). It has subsequently been refined by application of the approach to systems in the defence, nuclear and medical sectors. It is now widely accepted by the nuclear industry in the UK and it is the basis for the Generic Design Assessment for new build in the UK.

2.3 Safety justification strategies

The strategy for the safety justification approach in HARMONICS will be primarily goal-based but aspects of the rule-based and risk-informed approaches will also be incorporated. So, in addition to system properties, claims may be related to standards compliance (rule-based) or about control of potential hazards (risk-informed).

From a rule-base perspective, compliance with standards or with sound design principles may not contribute directly to show appropriate behaviour of the system, but they contribute to the overall case by increasing our confidence in the correct behaviour of the system and on its justification. For example, standards often include long-established design principles such as segregation of systems with different criticality, the single failure criterion, and design diversity. The safety justification would have to show that theses constraints apply. In practice such claims of compliance would assist a goal-based argument by limiting the impact of failures, and hence limiting the scope of goal-based arguments to a subset of the I&C equipment. The design principles often ensure simple systems designs are used. This makes it easier to generate evidence for goal-based claims.

Compliance with development process aspects of the standards may also be claimed. The development process contains activities that provide for the planned generation of evidence. In parallel, configuration management and other quality assurance activities support the production of traceable and consistent evidence. A key aspect of a good safety justification is that it should be supported by trustworthy evidence. Evidence that a sound development process and quality assurance principles have been followed increases our confidence in all the evidence generated to support the safety justification and, consequently, in the overall justification. For example, we have confidence that tests have been properly performed, recorded and applied to the correct subsystem.

Clearly, for a purely behaviour based case, we would not need any claims about the process. Process evidence would be incorporated, if appropriate, through its impact on the claims about the product behaviour.

Our confidence in a case, and its constituent parts, comes from a scientific and social process of elaboration, challenge, update and acceptance. As anyone who has been involved with technical research or developing cases knows, this cycle of review and challenge is fundamental to achieving confidence. We would no more believe a case that had not been reviewed than we would believe a scientific result without peer review. As part of this challenge it is important to seek negative evidence that could potentially disrupt the claims (a risk-informed approach). Any credible hazards should be explicitly identified and addressed in the safety justification. This activity can increase confidence in the primary goal-based justification if no serious problems are identified. This cycle of challenge can result in the claim being modified, the confidence in the claim being adjusted or both, and the process of review and challenge can involve a range of different stakeholders.

2.4 Construction of a safety justification

The construction of a safety justification for an I&C system will be addressed at a number of different levels. At the top-level, the claims will be related to the functionality, performance and dependability attributes required to maintain plant safety. The validity of these requirements have to be related to the nature of the plant and would need to be justified in the overall plant safety case. Often, the plant safety functions to be performed for Category A (as of IEC 61226) safety function are conceptually quite simple. However, there are additional non-functional requirements, including reliability, availability, defence against common-cause failure and fail-safety, that also need to be claimed at the top level.

At the next level down, we need to relate the system functional and non-functional claims to claims about the system architecture. To satisfy the non-functional requirements, additional complexity is introduced, such as:

- redundant and diverse components,
- system functions partitioned across components,
- additional functional requirements for components (like self-testing and voting) required to meet non-functional requirements at the system level,
- requirements to satisfy design principles (e.g. for diversity, segregations, etc.),
- operation and maintenance requirements.

At this level, we need to demonstrate that claims made at the top-level level will be satisfied if the claims at the architecture level are true, e.g. a failure per demand target claim at the system will be satisfied, subject to claims about:

- component hardware reliability,
- component software reliability,
- common mode failure,
- component interconnection and fault segregation,
- component fault diagnosis and repair probability,
- repair time.

Similarly claims about functionality at the architecture level should support a claim that the top-level function is implemented provided some minimum set of components are still operational.

Typically the evidence to demonstrate that architectural claims are consistent with system level claims will be demonstrated by some form of system modelling and analysis, but the evidence used may vary depending on the type of claim that needs to be demonstrated

We also have to take account of the fact that the architectural design could introduce potential hazards, so a further claim needs to be made that potential hazards present in the system are identified, their impact on the top-level claims are assessed, and the risks mitigated.

Finally, there are claims about the implementation of components (i.e. that they meet the requirements and constraints placed upon them by the architectural decomposition). In practice these components could also have subcomponents, so further levels of claim decomposition may be needed. For example, the claims at component level might relate to sub-components like:

- the computer hardware
- the operating system platform
- the application software

Claims made at this level need to be supported by evidence about the components or subcomponents and their interactions. For component reliability, the evidence could be statistical testing for the overall system, but there will be additional evidence about subcomponents, such as evidence of the platform hardware reliability from prior operational experience. Here again the implementation choices for the component could introduce hazards, so a further claim needs to be made that potential hazards present in the system are identified, their impact on the top-level claims are assessed, and the risks mitigated

3 ILLUSTRATION OF THE APPROACH

The approach presented in the previous section will be illustrated here on a simple example that we are currently developing: an I&C system S consisting of only one computing unit and implementing a set of I&C functions F in order to guarantee a condition C in the plant. Even though it is a much simplified example, it illustrates some of the principles that can be applied to more realistic, full-scale I&C systems.

Our top claim is that system S is adequate to guarantee condition C. However, for further simplification, the example will address only the functional and software aspects of the I&C system (e.g., it does not cover the hardware aspects, nor the firmware of input / output boards).

Our top claim can be decomposed into two main sub-claims:

a) The functional requirements specification for S (Spec(S)) is adequate to guarantee C.

b) The software of S implements Spec(S) correctly.

3.1 Spec(S) is adequate to guarantee C

As the requirements specification of an I&C system is the starting point of the safety lifecycle, any er ror or inadequacy will be faithfully expanded by the rest of the development process. In fact, the operatio nal experience of highly reliable computer-based systems (in all industrial sectors) shows that defects in f unctional requirements specifications are a significant cause of failure that should not be ignored.

The argument for this sub-claim may combine three types of evidence: evidence of a sound, rigorous elicitation and specification process for Spec(S) (the rule-based part), more direct evidence that Spec(S) d oes indeed guarantee condition C or has essential, necessary properties (the goal-based part), and evidenc e that Spec(S) does not imply unsafe behaviour (the risk-informed part).

The rule-based part may cite compliance to suitable standards. Such standards would typically requir e / recommend a list of actions, such as the identification of all relevant input documents, the use of appro priate specification methods and languages, verification and functional verification of the completed speci fication (possibly by an independent team). They would also typically require / recommend addressing a l ist of subjects such as functions, response times, interfaces, safe failure modes, etc. The rule-based part w ould typically not make any direct mention of condition C.

The goal-based part would directly justify that Spec(S) will guarantee condition C, in the context of the plant for which system S is intended. Therefore, it could for example mention model-based verification (with models of the plant and of Spec(S)). It would also directly justify that Spec(S) has important properties, such as clarity, precision (so that two different readers will not have diverging interp retations) and consistency (i.e., absence of contradiction, and intrinsic completeness).

The risk-informed part would justify that in the context of the plant for which system S is intended, t he implementation of S does not imply unsafe behaviour, neither by commission nor by omission, and tha t residual risks are acceptable.

Though these three parts have been presented in separate paragraphs, in practice, they are not comple tely independent. For example, information collected to support the rule-base part may also be used for th e goal-based or the risk-informed parts.

3.2 The software of S implements Spec(S) correctly

As for the previous one, the argument for this sub-claim may combine the three types of evidence. However, only the goal-based argument will be detailed here.

The goal-based argument needs first to identify the different sub-goals that need to be justified in order to claim that "the software of S implements Spec(S) correctly". Here, we will focus on one specific sub-goal, which is that "the software of S correctly implements the required functions F".

One possible approach is to follow the architectural decomposition of the software of S. The software architecture for S has the following main software components:

- An operating system (OS) that provides generic services.
- An application software component (AS) that implements the functions required by Spec(S).

Like the operating systems of most safety-grade I&C platforms, the OS has a number of properties of particular interest for our purpose:

- OS-SP1: It has a time-based behaviour (as opposed to an event-based behaviour, where events are signalled by interrupts). This means that it operates cyclically, in cycles of fixed duration. At each cycle, it performs the same sequence of actions: read inputs boards and place inputs at predefined memory locations, start the AS and wait for its completion, take AS results from predefined memory locations and write them to output boards, process operator requests if any, perform auto-tests.
- OS-SP2: The OS is not directly affected by what occurs in the plant. In particular, its behaviour is not altered by what it reads from input boards, nor by what it writes to output boards. Also, the only influence of the AS is the time it takes the AS to execute at each cycle. (Timing issues are not treated in this example).

These two properties are what can be called *structural properties*. With this background, the goal-based justification that "the software of S correctly implements the required functions F" (a *functional property*) will need to consider the following sub-claims:

a) The OS has structural properties OS-SP1 and OS-SP2.

b) At the beginning of each cycle, the OS transfers correctly values from the input boards to corresponding pre-defined memory locations (functional property OS-FP1).

c) Next, the OS starts the AS and waits for its completion (functional property OS-FP2).

d) Next, the OS transfers correctly AS results from pre-defined memory locations to the corresponding output boards (functional property OS-FP3).

e) The processing of operator requests by the OS has bounded side effects (structural property OS-SP3), and these side-effects are consistent with all the other properties claimed for the software of S (functional property OS-FP4).

f) The auto-tests performed by the OS have bounded side effects (structural property OS-SP4), and these side-effects do not affect any of the other properties claimed for the software of S (functional property OS-FP5).

g) The AS performs functions F, taking its inputs where the OS places what it has read from the input boards, and placing its results where the OS takes the values to be written to output boards (functional property AS-FP1).

h) The executable binary code of S is consistent with the software representations (e.g., the source code) on which the preceding claims have been ascertained (equivalence property).

As we develop the case we will demonstrate rigorously how these sub-claims imply the top level claim. Each of the sub-claims will naturally need to be supported by corresponding evidence. The HARMONICS project also investigates practical means for providing such evidence, in particular by static source code analysis and formal verification. There is in fact a close interaction between these two aspects of the project (safety justification frameworks and formal verification) as using brute force to formally verify that the software of S (OS + AS) correctly implements functions F is often not feasible. Therefore, it is necessary to take a more progressive, analytical approach that is described in a structured manner with the justification framework.

4 CONCLUSIONS AND FURTHER WORK

The justification framework is still under development, but it will provide a sound basis for identifying what evidence is needed to support the claims that the overall system has the required safety properties. Work is in progress to elaborate the approach, including ways of constructing and justifying the decomposition of the claims into sub-claims. We will also identify appropriate techniques for generation of evidence. This will be done by applying the justification framework to a number of realistic I&C case studies that will be performed within HARMONICS. The overall goal is to develop a Claims-Argument-Evidence template for reactor protection system.

5 ACKNOWLEDGMENTS

HARMONICS is co-funded by the European Commission, the UK C&I Nuclear Industry Forum (CINIF) and the consortium organisations: VTT Technical Research Centre of Finland, Électricité de France (EDF), Institute for Safety Technology (ISTeC) from Germany, Adelard LLP from UK and Strålsäkerhetsmyndigheten (SSM) from Sweden. The authors also acknowledge the contributions of the other HARMONICS project members.

6 REFERENCES

1. Adelard Safety Case Development Manual, © Adelard, ISBN 0 9533771 0 5, 1998.
2. P.G. Bishop and R.E. Bloomfield, The SHIP Safety Case - A Combination of System and Software Methods, in SRSS95, Proc. 14th IFAC Conf. on Safety and Reliability of Software-based Systems, Brugge, Belgium, 12 15 September 1995.
3. P.G. Bishop and R.E. Bloomfield, MJP Van der Meulen, “Public domain case study: An example application of the CEMSIS guidance”, v1.0, 26/3/2004, WP5 Deliverable, http://www.cemsis.org
4. P.G. Bishop and R.E. Bloomfield, “A Methodology for Safety Case Development”, Safety-critical Systems Symposium 98, Birmingham, UK, Feb 1998, ISBN 3-540-76189-6
5. T Kelly. “The goal structuring notation–a safety argument notation”, Proc. DSN 2004 Workshop on Assurance Cases, 2004
6. SE Toulmin, The uses of argument. Cambridge University Press, 1958
7. S Wilson, P Kirkham, “SAM User Manual”, University of York, December 1995.
8. S. E. Toulmin “The Uses of Argument” Cambridge University Press, 1958.
9. P. Bishop, R. Bloomfield and S. Guerra, “The future of goal-based assurance cases,” *Proceedings of Workshop on Assurance Cases*. Supplemental Volume of the 2004 International Conference on Dependable Systems and Networks, pp. 390-395, Florence, Italy, June 2004.