Safety Case Development: Current Practice, Future Prospects

S P Wilson, T P Kelly, J A McDermid
HISE Group, Department of Computer Science, University of York
York, England

Abstract

Safety-critical and safety-related systems are becoming more highly integrated and continue to increase in complexity. In parallel with this, certification standards for such systems are becoming more stringent, requiring more extensive and more detailed analyses. Safety cases, therefore, are themselves growing in size and complexity and are becoming increasingly costly to produce. It has become necessary to re-examine how and why safety cases are built in order that we might provide a means for managing their inherent complexity and reduce production costs.

In this paper, we examine some of the key issues in current industrial safety case development, in particular:

- **The Purpose of the Safety Case** - examining how stakeholders place demands upon the content and style of the safety case;
- **Safety Analysis Techniques** - examining the problem of ensuring consistency and completeness of results;
- **Safety Case Production** - examining how and when safety cases are produced through the development life-cycle;
- **Safety Case Structure** - examining how the reasoning and evidence aspects of the safety case are combined;
- **Safety Case Maintenance -** examining the need and support for safety cases that can be more readily maintained and reused.

We propose to address these issues through the use of a goal based notation for more effective structuring, a data model to tightly integrate the safety analysis techniques, and a process model to integrate the safety case activities into the overall development process. We demonstrate our approach using an integrated example from the automotive industry.

1 Introduction

Safety-critical and safety-related systems are becoming more highly integrated and continue to increase in complexity. In parallel with this, certification standards for such systems are becoming more stringent, requiring more extensive and more detailed analyses. Safety cases, therefore, are themselves growing in size and complexity and are becoming increasingly costly to produce. It has become necessary to re-examine how and why safety cases are produced in order that we might provide a means for managing their inherent complexity and reduce

production costs. We need to be very clear about the purpose of a safety case, it's structure, the analysis techniques it relies upon, it's development and maintenance, and the problems engineers face in these areas.

1.1 The Purpose and Content of a Safety Case

There is no definitive statement of what constitutes a safety case, and there is certainly variation in presentation and content across industries. In general this is dependent on legislation (from the Health and Safety Executive for instance) and the (emergent) national and international standards, see, for example, [6, 7, 8, 12, 13]. We do, however, believe that there is a commonality of purpose to all these variants of safety cases and offer the following working definition, [18]:

The purpose of a safety case is to present a clear, comprehensive and defensible argument supported by calculation and procedure that a system or installation will be acceptably safe throughout its life (and decommissioning).

Two key words from the above definition are "argument" and "supported": typically a safety case will contain *high level argument* and *supporting evidence*. The high level argument (HLA) sets out the principles on which the design is based and reasons why the design should satisfy the safety requirements; the supporting evidence (SE) provides detailed analysis of the implementation to show that the design has the predicted properties, and hence that the system meets its safety requirements. For example:

- ***HLA*** *- a two lane architecture is sufficient to meet the safety requirements, so long as the lanes fail independently.*
- ***SE*** *- the results of hardware reliability calculations, showing acceptable failure rates, and common mode failure analysis, confirming that the lanes are independent.*

The HLA also shows how the various parts of the SE relate, and how in combination this evidence proves key safety properties of a system.

1.2 Current Practice (and Problems)

In this section we set out our perceptions of current practice in the area of safety cases and safety assessment in general; our observations are based on experience of the aerospace, automotive, nuclear and rail industries.

Increases in the level of detail required by certification standards and the complexity of systems are causing safety cases to grow dramatically in size, making it increasingly difficult to maintain an overall picture of system safety. That is, the sheer volume of low level analysis and supporting evidence for a large system can jeopardize the clarity of the high level argument.

A safety case is normally produced as a single linear document summarizing and linking the results contained in other deliverable documents which contain the

detailed evidence (SE). Maintaining the links across document boundaries for consistency and completeness is difficult, and in an environment without effective tool support, it can only be achieved by rigorous human review.

Presenting safety cases in a purely linear format also makes it very difficult to structure the high level argument for a safety case and to show the relationships with the supporting evidence. It is hardly surprising therefore, that safety case reasoning is not always clear and it is difficult to have confidence in its completeness. It can also be difficult to trace from high level requirements to design decisions, and assessment results. In studying industrial safety cases we have invariably found it necessary to annotate and cross reference them, in order to "pull out" the structure and argument. Often the result was the discovery of a convincing argument – but the onus on discovering the argument was on the reader.

Returning to human aspects, it is often people who hold safety cases together. On a project there are usually key individuals who have in depth knowledge of the system and its safety. It is that knowledge, the assumptions and justifications underlying design decisions, that are omitted from the case, or whose contribution to the overall safety argument is unclear.

There are typically a number of "stakeholders" (individuals and organisations) involved in the production, review and use of a safety case. Typically they have differing, and often conflicting requirements for the safety case - this can lead to ambiguity in the purpose of the safety case, and confuses the reasoning and the evidence that it conveys. Key stakeholders include developers, assessors, customers, regulators, operators, victims, designers of peer systems and so on. For example a customer will be concerned with functionality and cost traded off against safety, whereas regulators will be concerned primarily with safety (this simplifies the situation somewhat, but does make the point that the stakeholders have differing priorities).

Classical safety analysis techniques (which can be used to determine the supporting evidence) typically only address one aspect of system safety. Even on a single project, there is therefore a need to use a number of them. The techniques are not independent, indeed there is significant propagation of results between them, and there are also a number of consistency and completeness rules that should hold. Despite this, the relationships between individual pieces of analysis for a system are often handled on an ad-hoc basis, and we have seen surprising inconsistencies between these analyses on practical safety cases.

The calculations used in safety analyses often are not made explicit, i.e. the formulae or the assumptions are not given, e.g. simplification of probability calculations because the numbers are small. When the data that was supplied for the original calculation changes it becomes necessary to determine the formula or decide upon the assumptions that can be made again. Without the formulae it is very difficult for reviewers of the safety case to have confidence in the final conclusions.

Safety case development is often treated as a post-construction presentation concern, and consequentially it often fails to capture the engineering judgment and experience applied in the development that is necessary for easier retrospective understanding and maintenance. Yet complex safety critical systems are used over a long period of time, and are subject to changing requirements, changing environmental conditions and challenges from operational experience and data.

Without effective structuring it is very difficult to assess the impact of such changes on the safety case.

Safety should be a key driver in the design of systems. Early safety analysis (for example Preliminary Hazard Analysis) often leads to derived requirements and design decisions. Yet it is difficult to link the results of the early analysis to those requirements and decisions; usually only the hazard log records such information.

For companies that build a number of systems within a certain domain, there is often repetition of information across the safety cases for such systems. Yet reuse of this information is seldom handled in a systematic or cost effective manner. There is also commonality between the systems that different companies build, although commercial pressures reduce the likelihood of such knowledge being pooled.

Further we note that some industries are moving towards a situation of phased safety related submissions throughout the life cycle (either internally or to regulators). This encourages production of the safety case in step with the development, and reduces the economic risks incurred by companies developing safety critical systems.

As a final point we should emphasize that we are not saying that the majority of current safety cases are flawed. The above problems might conceal flaws in safety cases or, more likely, lead to flaws being introduced during maintenance. In our experience the skills of safety engineers normally result in sound, if not obviously compelling safety cases. However, the issues identified above are certainly issues of cost; they need to be addressed if the development and evolution of safety cases is to be made more cost effective.

1.3 Future Prospects

We believe that the problems outlined earlier can be tackled effectively on three fronts.

- **Improving Reasoning and Scrutability** - we propose that through the use of *goal structuring principles* we can give safety cases a better structure; explicitly link requirements to analysis results and evidence; make rationale, assumptions and justifications explicit; provide an abstraction for handling complexity; and link together diverse strands of work.
- **Improving Consistency** - we propose a *data model* to integrate the results of safety analyses. Using this model and consistency rules we demonstrate how the results of several safety analyses can be integrated to assure consistency and completeness of the safety case.
- **Improving the Process for Development and Evolution** - we propose a *process model* that encourages a more evolutionary development of the safety case, in parallel with system development. Using this model, and through explicitly recording applied engineering judgment and experience, we believe that safety cases can be more readily maintained.

Note that the true benefits of our method can only be realised with effective tool support, and indeed we have developed a Safety Argument Manager (SAM) to support the concepts outlined in this paper. SAM is an evolving prototype tool

which consists of a number of specialized editors for building goal hierarchies, system models, and offers facilities for performing analyses over these models. The integration of the models and analyses is provided by a database implementation of the data model, which the editors can read and write to. We will refer to SAM throughout this paper.

1.4 Structure of the Rest of the Paper

We cover improving reasoning and scrutability in section 2, improving the consistency of supporting evidence is described in section 3, and improving the process for development and evolution is given in section 4. In section 5 we illustrate our principles with an integrated example from the automotive industry, and finally in section 6 we give some concluding remarks, together with an indication of our future work.

2 Improving Reasoning and Scrutability of Overall Case

In Section 1 we identified a number of problems which principally relate to the structuring of safety cases. The use of a hazard log gives a structure for a safety case once it has been developed. However, we believe that there is a need for a structuring principle which provides more assistance in the construction of the safety case.

We have developed a goal structuring notation, with which it is possible to express safety requirements as goals. Demonstration that these goals have been met is achieved by further decomposition or by direct appeal to supporting evidence, e.g. the results of individual safety analyses. The notation is also rich enough to capture assumptions, justifications, proof in the general sense, and rationale. In our approach a goal structure is built which forms the HLA of the safety case to which evidence, argument, and calculations are attached. The goal hierarchy that is built can be viewed as the **spine** of the safety case to which the **flesh** (models of the system and supporting evidence) is attached. Goal Structures provide a graphical notation with an underlying logical interpretation.

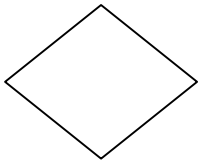
Early work on goal structuring is reported in [10] and later in [18].

2.1 Components of Goal Structuring

We first give a brief introduction to some of the basic building blocks of the goal structuring notation.

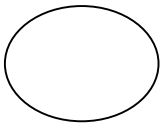
A **goal** is a requirement, target or constraint to be met by the system or parts of it. Goals are intended to be sufficiently general so that they can also capture process issues, e.g. *develop*

to the requirements of Interim Defence Standard 00-55, [12]. We use the term **goal hierarchy** to refer to the collection of goals produced by hierarchical decomposition into sub-goals. A Goal is usually "owned" by some stakeholder.

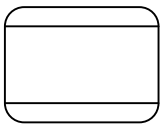


A goal is couched in terms of some **model** of the system, or its environment. A goal may often be expressed over a number of models. We do not constrain the form that the model may take; it may be a plant schematic, a process description, or an architectural model, etc. The goal will refer to objects within the models it is expressed over, e.g. *the auxiliary feed pump is to be switched on when....* It provides a basis for linking the safety case and supporting analysis to the design.

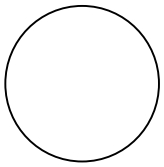
A goal (or set of goals) can be solved by a **strategy**, which breaks a goal into a number of sub-goals. The satisfactory solution of the sub-goals then entails the solution of the original goals. A strategy can be regarded as a rule to be invoked in the solution of goals. If we have a general strategy which is based on a number of assumptions then we will also have an obligation to demonstrate that these assumptions can validly be made. There will be situations in which goals are conflicting, and a strategy needs to be introduced to make a trade-off, e.g. *achieving safety in potentially dangerous situations through shutdown and maintaining a high availability by keeping the plant up and running*.



Strategies often need some **justification** for their use. It may be that the strategy is laid down in some standard followed by the developers, e.g. *ASME Boiler and Pressure Vessel Code Section III recommends that it be done*, [1]; it may be common practice; or it may be a more elaborate argument as to the validity of the use of the strategy. Alternatively a justification may call upon evidence from analysis of the model or be a structured proof.



It is also useful to identify **constraints**. A constraint is used to restrict the way in which goals can be solved. For example a common safety requirement is *No single point of failure shall lead to a hazard*. If this constrains the solution of a goal then any solving strategies and associated sub-goals should respect this constraint, i.e. no single failure at *any* level should lead to a hazard. Eventually, say at a low enough level of abstraction, we would wish to discharge the constraint, by showing that the failure assumptions encompassed any failures which could occur at a higher level.



Finally, some goals may be solved directly by what we term **solutions**, rather than by decomposition into sub-goals. **This is where the high level argument links to and uses the supporting evidence.** Solutions will be individual pieces of analysis, evidence, results of audit reports, or references to design material including models. In fact we are not restrictive at all of the form that solutions can take.

There are a few other points to make on Goal Structuring: **choices** (**meta-strategies)** record the situation where there are alternative strategies for the solution of a set of goals, and **Criteria** are used to decide whether or not a goal has been satisfactorily solved, i.e. they give measures and procedures for assessing goal satisfaction. For choices the justification will record the rationale for the choice of strategy in terms of the criteria. There are also various relationships which can be expressed relating to requirements conflicts, independence, orderings and so on.

2.2 Using Goal Structuring to Achieve Clear Safety Arguments

Now that we have set out the building blocks of the Goal Structuring Notation we can give some basic principles for their use in the construction of safety cases:

1. Couch the customer's, and regulator's safety requirements as top level goals to be met by the system (record the stakeholders for the goals);
2. Couch requirements set by codes of practice, standards, and acceptable risk levels as top level goals to be met by the system (note that some of these goals may meet some of the requirements identified in step 1); some of the requirements in applicable codes and standards may best be cast as constraints;
3. Break down the top level goals into sub-goals and show how design and analysis decisions meet the requirements set in steps 2 and 3 by use of strategies (when the safety case is produced in step with the development then the safety case can be used to record the decisions as they are made);
4. Provide justifications of strategies and state any assumptions made, relating to requirements, environmental conditions, usage restrictions, independence assumptions etc.;
5. Where alternative designs have been considered make this explicit by using choices, and make explicit the criteria for choosing between the alternatives;
6. Make explicit which parts of the system a goal relates to by linking goals to models (note that the sub-goals inherit the models of its parent unless otherwise specified);
7. Plan the detailed analyses of the system so that their contribution to higher level goals is clear;
8. Carry out the detailed analyses to show that the requirements are indeed met; this is where the HLA links to the SE.

Following these principles for safety case construction should not only make the construction easier, but the assessment of the case simpler too, since the assessor can see the reasoning behind the case, and the evidence it rests on. For example, if the

assessor doubts the validity of piece of low level evidence, or if they question an assumption then the effect on the overall argument should be more apparent. Of course if the argument itself is flawed then the assessor has a greater chance of detecting this. Perhaps more importantly, the developer is more likely to see the flaw and remove it prior to submission to the assessors.

3 Improving Consistency of Supporting Evidence

A great deal of the supporting evidence underlying a safety case is obtained through the use of safety analysis techniques. There are a wide number of these in current use and even on a single project it is necessary to use a number of them - there is no single technique powerful enough to be used in isolation. These techniques are not independent. Indeed there is a large amount of data propagation between them, and various completeness and consistency rules that should hold between the analyses. Despite this, the relationships between the analyses tends to be handled in an ad-hoc fashion, with the main linkage being (at a rather coarse grain) through the hazard log. If there are inconsistencies between the analyses then the safety case itself is likely to be inconsistent and therefore not trustworthy. It is therefore of much benefit to rigorously state the data flow, and consistency and completeness rules, that relate the various techniques used in the production of safety cases, and provide support for checking or enforcing them.

We have developed a data model, [19], to underpin the Safety Analysis Process, such that it is possible to map the inputs and outputs for each technique onto the data model. An implementation of the data model can therefore act as a medium for the data passing between analyses, but it also leads to a more cohesive view of the supporting evidence underlying a safety case, it allows generalization and simplification of the original rules, it enables the capture of more complex rules, and it allows certain types of requirement to be evaluated. [16] provides an alternative framework for method integration (principally requirements notations) by formulating pairwise rules between different notational *viewpoints*, but we prefer our centralised approach for the reasons outlined above.

There are a large number of safety analysis techniques in common use, and in our work we have attempted to take a set providing coverage in a number of categories.

- **System Modelling and Design:**
  Physical and Functional decomposition, Reliability Data
- **Hazard Identification Techniques:**
  Hazard and Operability Studies (HAZOPS)**,** see [9,2],
  Functional Failure Analysis (FFA)
- **Causal Analyses:**
  Fault Tree Analysis (FTA), see [15]
- **Consequence Analyses:**
  Failure Modes Effects and Criticality Analysis (FMECA), see [4],
  Event Tree Analysis (ETA)
- **Risk Assessment:**
  Risk Tables mapping likelihood and severity to a measure of risk

[14] provides comprehensive descriptions of most of these techniques, although a slightly different terminology is used.

The data model itself appears in Figure 1. At the centre of the model is a **Condition**, which describes some "state of affairs in a system" – it is a generalization of state and event (this generalisation is necessary in safety analysis as states and events are often treated the same way, e.g. they can both be inputs to gates in Fault Tree Analysis). Conditions are strongly linked to **Models** of the System and possibly **Components** within those Models. For every Condition it is *possible* to record the **Causes** & **Likelihoods**, and the **Consequences** & **Severities**. **Faults**, **Failures**, **Hazards**, and **Accidents** are just special cases of Conditions: this gives equality of treatment, i.e. we can carry out cause and consequence analysis on a condition regardless of its type - this is very important for hierarchically designed systems, i.e. we do not want to be restricted to only applying causal analysis (e.g. Fault Tree Analysis) to hazards.

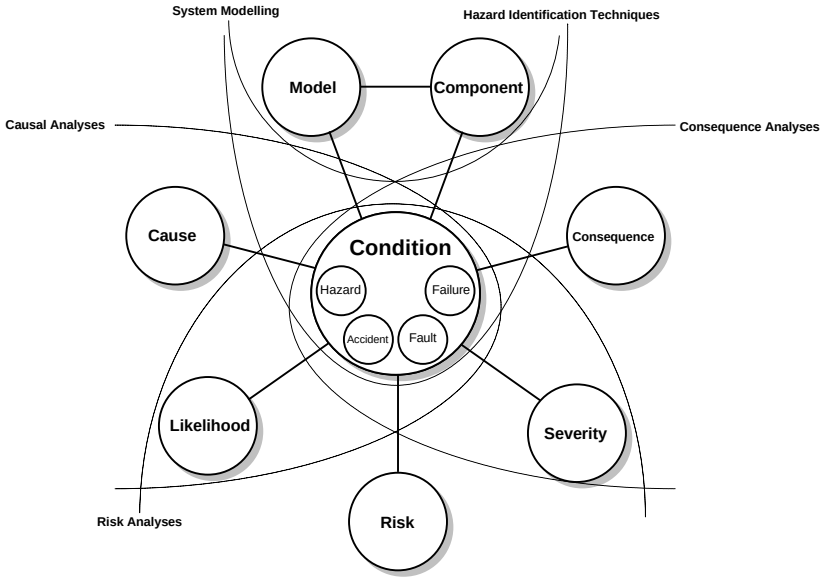


Figure 1: A Data Model to achieve Consistency of Supporting Evidence

The arcs on the diagrams indicate the relationships between the types of techniques and the data model:

- **System Modelling and Design** techniques fill in the **Models** and **Components** of the data model, and details of primitive **Failure** modes (as conditions) of those models and components.
- **Hazard Identification** techniques fill in potential **Hazards** of the System, taking as input the **Model** of the system (and its **Components –** Flows for HAZOP, and functions/processes for FFA) under investigation.
- **Causal Analyses** fill in **Cause** and **Likelihood** for some, imported, top level **Condition** (usually a Hazard), and read in details of **Fault** Events (and their **Likelihoods**), and other normal **Condition**ing Events in order to do the calculation of the **Likelihood** for the top level **Condition**.

- **Consequence Analyses** fill in the **Consequences** and **Severities** (and possibly **Likelihoods**) for some initiating **Condition**, reading in the **Likelihood** of that Condition and other contributing **Conditions**.
- **Risk Analysis** simply fills in a measure of the **Risk** for a condition, taking as input the **Likelihood** and **Severity**.

Within the SAM tool we have provided a database implementation of the data model, and provided a set of Modelling and Safety Analysis editors with special operations on this database. We can now give brief illustrations of how individual editors interact with the database.

- **Data Propagation**: A Fault Tree Analysis reads in the Likelihoods exported by a FMECA for use as Fault and Conditioning events within the Tree, and uses them in calculating the Likelihood for the top level Condition.
- **Consistency and Completeness Rules:** Within a FMECA, check that all Failure Modes for all Components have been considered, and their likelihoods are consistent with those produced by the system modelling and design.
- **Automation:** Generate template FMECAs, HAZOPS, and FFAs based on the models of the system under investigation – focusing on components, flows, and functions respectively.
- **Requirements Evaluation**: For example, check that the Risk for all hazards is at least Acceptable.

Collins and Dent, [3], are building a tool for managing reliability, safety and other concurrent engineering data, but we are not aware of any attempt by them to formalise the relationships between the analysis techniques.

4 Improving the Maintenance Process

The safety justification process for a system, like most other engineering processes, is naturally iterative. A portion of the system is developed, then analysed, evaluated in the context of relevant safety requirements, and necessary adjustments fed back into the development process. Change, therefore, forms an inevitable part of the development and justification process. Change to the safety case is also highly probable after system deployment in order to reflect, for example, maintenance changes made to the system or observations of degraded performance. In order to improve the safety case development process it is therefore necessary that change and maintenance in the safety justification process is well supported. We propose a process model for this in Figure 2.

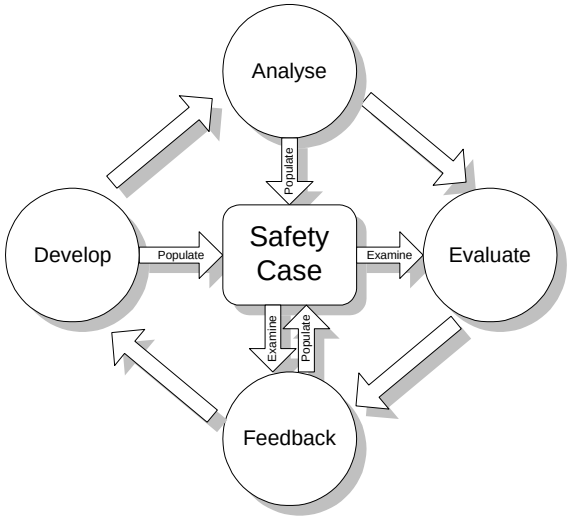


Figure 2: Evolutionary Development of the Safety Case

In Figure 2 we show that a portion of the system is **developed** and that this activity populates the safety case with safety related aspects of the development (e.g. safety codes used in the design process). The system is then **analysed**, e.g. by testing or using safety analysis techniques, and the results of this activity also used to populate the safety case (e.g. observed mean time between failure – MTBF). The available evidence from these first two activities is then used to **evaluate** the system in the context of relevant safety requirements expressed in the safety case (e.g. acceptable MTBF). Necessary adjustments are then **fed back** into the development.

Ensuring that the safety case captures the process in this way, allows us to replay the rationale behind the justification activity. Recording the rationale at this level will result in more information being held in the safety case than is necessary or perhaps desirable for the certification process. Support, therefore is provided in SAM, for extracting subsets of the recorded goal structures for presentation in the certification safety case.

The key difficulty involved with change applied to the complex, interdependent structure of the safety case is in knowing what elements are and are not affected by a particular change. At present, with limited or no support for analysing the impact of change, there is a tendency to rework more of the justification than is necessary.

The capture and management of the underlying rationale of the safety case plays an important role in improving handling of change. Without knowing the underlying rationale, accurate judgments concerning the resilience of the safety case to change are made extremely difficult. For example, without knowing the assumptions on which on an argument is based we cannot predict the effect on that argument of the a change that challenges those assumptions. Therefore, we prescribe the use of the goal structuring notation described earlier in order that the goals, strategies, models, assumptions, justifications, solutions and their interdependencies may be made explicit. Using this notation we have a means for scoping areas affected by a particular change. First, we can identify those elements of the goal structure

immediately challenged by a change. Having done this, using the linked dependencies of the goal structure, the challenge can then be 'traced into' the body of the safety case. Change applied to the safety case can be said to be of one of two forms:

- **Internally sourced change** - change resulting from additional analysis, testing or operational experience that challenges the validity, correctness or consistency if the safety case
- **Externally sourced change** - change enforced from outside of the system that challenges the goals (targets and methods) used in the justification process.

Internally sourced change typically challenges leaf elements of the goal structure, i.e. the models, solutions, assumptions and justifications. For example, degraded performance can be considered as a challenge to previously correct models of the system, operating experience can contradict the assumptions and justifications used in order to meeting goals.

Externally sourced change has a more widespread impact and can challenge any or all of the goals used in the goal structure. For example, if we consider a change made to a safety assessment standard, this can have wide implications for the acceptable methods and targets associated with the safety case.

Having identified the immediate challenge posed by a change, whether internally or externally sourced, we must then propagate this change through the goal structure. Using the explicit dependencies of the goal structuring notation, this process is made relatively mechanical. For example, having made a change to a model we must ask whether goals articulated over that model are still valid. Expert judgment is still required in order to answer such questions, however using goal structures we can more easily determine the questions to be asked.

5 Examples

In this section we present a detailed example, based on a real case study carried out within the High Integrity Systems Engineering (HISE) Group at the University of York, which follows the principles set out in section 2.2, although we do not consider stakeholders. This work was carried out as a fully integrated example with the SAM tool and we will mention some of the facilities provided by SAM during the exposition. In this example we show the development of a safety case in parallel with the design process.

It should be stressed that the example is primarily intended to illustrate the concepts we are developing, and it should not be viewed as a definitive statement on the design or safety analysis of computerised braking systems.

5.1 System Concept and Top Level Goals

Consider an experimental Computer Aided Braking System (CABS): the top level requirement for the system could be to "*Provide Safe Computer Aided Braking System*" (G1). We can capture this top level requirement as a goal, see Figure 3, and

link it to a conceptual model of the CABS, in its environmental context – "*CAB System Concept*" (M1). This conceptual model may set environmental constraints on the system, or constraints on usage – these may then be exploited by the designers of the CABS, but any reliance upon them should be made explicit in the goal structure. We attach a justification of the need for a CABS because of "*Higher Speed Limits relative to human reaction time*" (J1). In reality this justification may need to be much more comprehensive, as there is often a danger of introducing new technology for the sake of it.

SAM provides a Goal Structuring Editor which allows the user to build models and connect them to relevant parts of goal structures, the structure in Figure 3 provides an illustration of this. Also note that once the model has been connected SAM provides a means of navigating directly from the goal structure to the attached model.

Given this top level Goal, we can then adopt a number of derived design goals to make a safe CABS. We do this first by introducing a strategy S1, which must respect two constraints, C1 and C2: to "*Follow MISRA Guidelines*" and "*Follow Company Quality Procedures*" (SAM currently draws constraints without bars on the top and bottom of the box, but labels them as constraints). MISRA Guidelines: The Development Guidelines for Vehicle Based Software have been produced by the Motor Industry Software Reliability Association, [14]. We now have the following design principles:

- G1.1 *"Provide Alternative Braking Mechanisms at different levels of sophistication":* with the simplest algorithm as the "fall-back" algorithm;
- G1.2 *"CABS should be Fault Tolerant"*: this is decomposed further into two sub-goals:G1.2.1 "*CABS should be able to detect faults*" and G1.2.2 "*CABS should be able to switch to redundant channels*"
- G1.3 "*CABS should be highly available,*"
- G1.4 *"No single point of failure should lead to a hazard."*

Note that G1.4 could have been expressed as a constraint, but for this relatively small example it is simpler to treat it as a goal. Also the goals are all qualitative. G1.2.1 and G1.2.2 could have been quantified, e.g. a proportion of detected faults could be given. In practice we would expect goals to be introduced qualitatively, and later made more precise with criteria (not illustrated here).

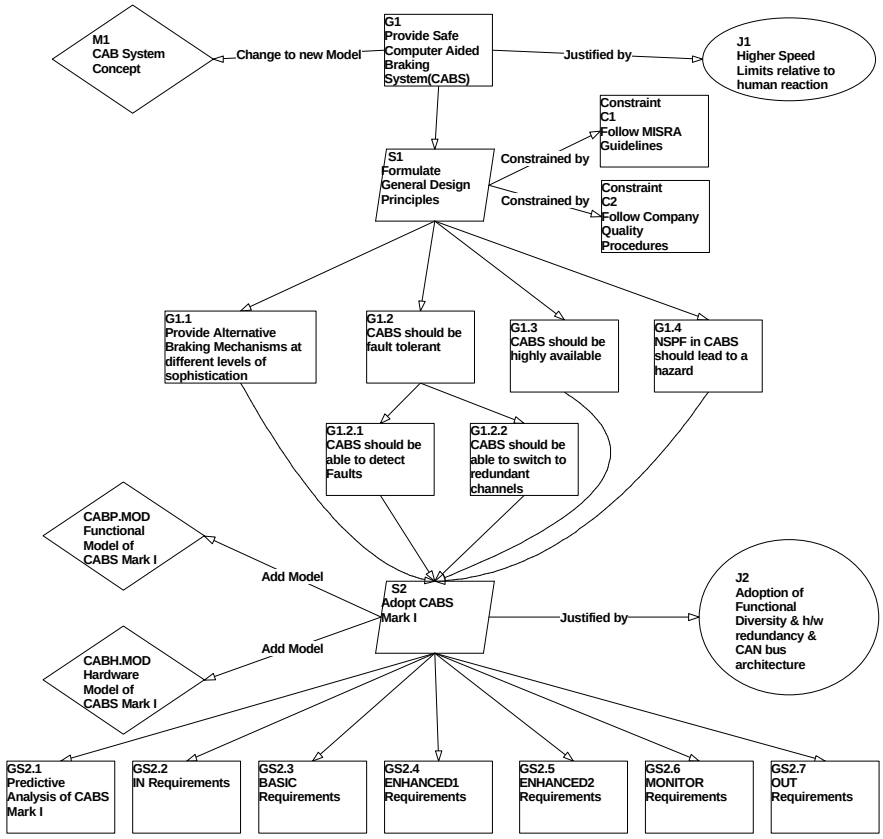


Figure 3: Top Level Goal Structure for Computer Assisted Braking System

5.2 Adoption of Design for CABS

We then factor all of the derived design goals into a strategy to "*Adopt CABS Mark I*" (S2), and attach Functional and Physical Models for this chosen design. We also attach a justification for the strategy *"Adoption of Functional Diversity and Hardware Redundancy and CAN Bus Architecture should lead to a Safe CABS"* (J2). The Functional Model for the system is given in Figure 4. The models should of course be developed with the high level design principles in mind.

The CABS has sensors (*Sensor_values*) to get readings on brake pedal pressure, wheel rotation, axle load, and brake pressure feedback. The *IN* process will take in these sensor readings, and apply various range checking and conversions before passing them onto the other processes in the system, via *Pedal*, *All_sens1*, and *All_sens2*. Note that *IN* also informs *MONITOR* (via *Data_sent*) that it has passed these values on to the algorithms (see below), so that *MONITOR* knows when to expect output from the algorithms. There are three algorithms to decide on the Braking pressure to apply: *BASIC*, *ENHANCED1*, and *ENHANCED2*. Each of these

processes will pass their results to the *MONITOR* process, and assuming the availability of the results, MONITOR will give preference to *ENHANCED2*, then *ENHANCED1*, and finally fall back to *BASIC*. *ENHANCED2* provides the most sophisticated Braking algorithm, followed by *ENHANCED1* and then *BASIC*. *MONITOR* will send the brake pressure to apply to the *OUT* process which will convert this into appropriate values to send to the actuators (via *Actuator_control*). Note that the OUT process gets feedback from the sensors (*Feedback_sens*) to ensure that the applied braking pressure does not increase or decrease too dramatically. *ENHANCED2* provides full anti-lock breaking on each wheel, *EHNANCED1* provides anti-lock breaking based on axle loads, and *BASIC* provides brake pressure proportional to pedal pressure and travel (the same for each wheel).

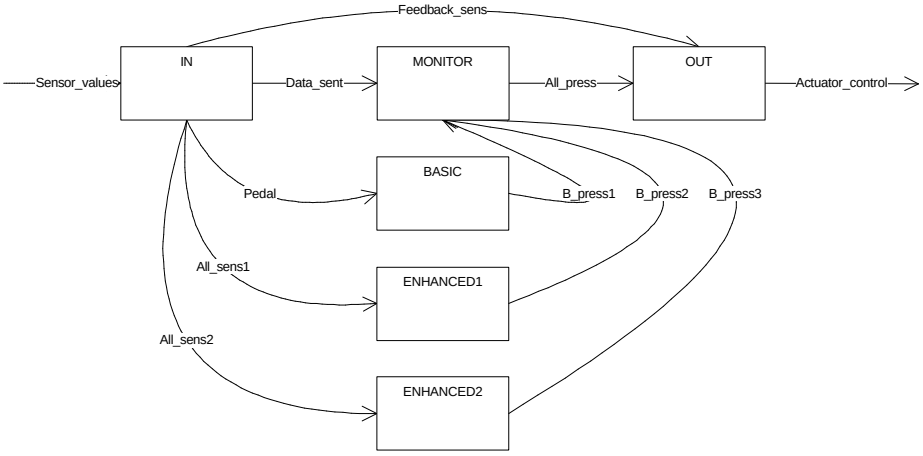


Figure 4: Functional Model of CABS Mark I

Note that in the goal structure in Figure 3, each of the processes introduced in the Functional Model have requirements imposed on them, relating to functionality and performance, for example "*IN should provide Range Checking of sensor values*". Due to space constraints, however, we have elided the detail for Figure 3 and written "*IN requirements*", "*BASIC requirements*", etc. SAM provides support for capturing more details about the goals which do not appear on the diagrams, i.e. summaries appear in the graphical structures, the more detailed information is available through forms and tabular views of the data.

5.3 Analysis of Design and Derived Safety Requirements

We now have a requirement to perform a "*Predictive Analysis of CABS Mark I*" (GS2.1). GS2.1 is referenced in Figure 3 but the decomposition of the goal is presented in Figure 5. In reality SAM maintains a contiguous Goal Structure with support for breaking large hierarchical structures up into fragments for printing and inclusion in document based safety cases (this is the very facility we have used in

producing this paper). We adopt a strategy of performing a HAZOPS over the functional viewpoint of the system, and carrying out an FTA for each of the hazards identified: "*Do HAZOPS and FTA for each hazard"* (S3). The justification for this strategy references the quality plan – "*See Quality Procedures section X*" (J3), or it could be to the appropriate section of the MISRA Guidelines. Note how the actual HAZOP is attached as a Solution to Goal GS2.1, it can also be quickly found by clicking on it within the SAM Goals Editor.

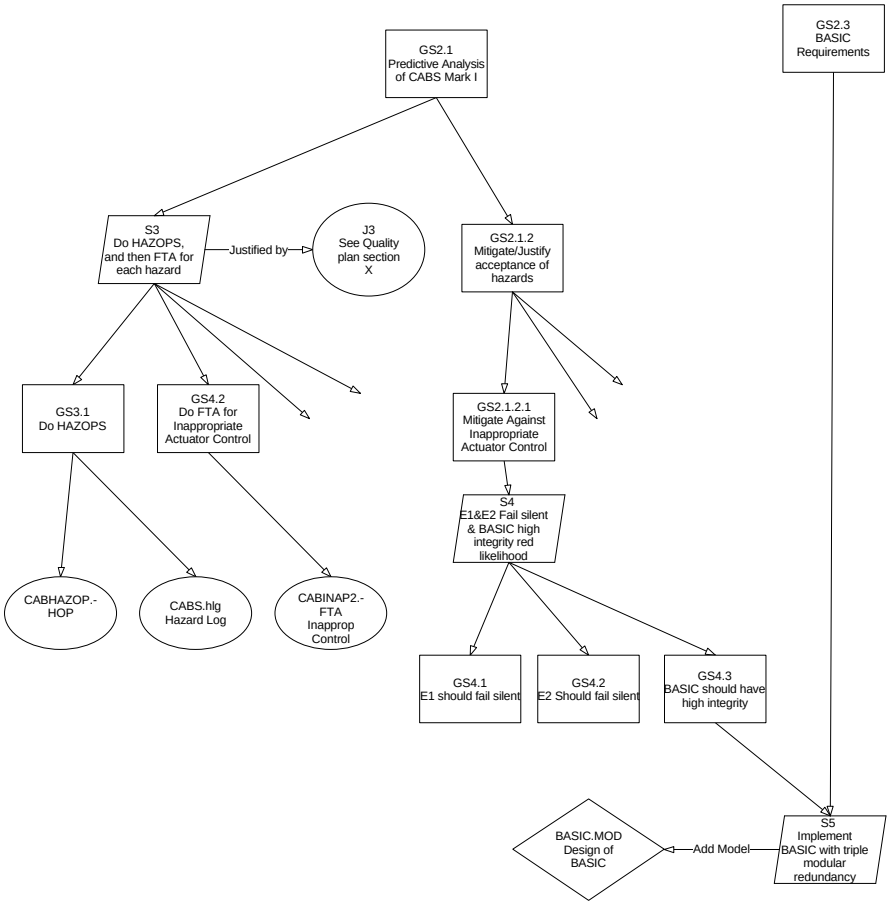


Figure 5: Predictive Analysis for CABS and Derived Safety Requirements

The functional model of CABS Mark I forms the basis for the HAZOP. Using SAM we can request generation of a template HAZOP, following the principles defined in [11]. SAM does this by reading all of the flows and their flow types, reading all of the properties of interest for the flow types, reading all of the deviations that should be considered for the properties, and finally producing a list of flows with types, properties and deviations. By doing this it is possible to ensure the consistency and completeness of the analysis with respect to the Model of the system. If the model changes it is possible to re-generate a template of the HAZOPS, but preserving any

effects which have been entered. A fragment of the deviations to consider appears in Figure 6, in reality there are a large number of deviations to consider (11 flows * 6 guide words = 66) and having SAM generate this does save a considerable amount of time. Note that all of the flows in our example are *data* flows, and so they do not have multiple properties defined. In other examples where one is considering fluid flow for example, it is useful to identify properties such as temperature and rate, and produce guide words specifically for those properties (SAM provides a special editor for doing this).

Flow	Guideword	Effect
Sensor_values	Vu	-
Data_sent	O	-
Data_sent	C	-
Data_sent	E	-
Data_sent	L	-
Data_sent	Vd	-
Data_sent	Vu	-
All_press	O	-

Figure 6: Fragment of Deviations to be Considered in HAZOPS

Note that the guide words chosen for this example are Omission (O), Commission (C), Early (E), Late (L), Value detectable (Vd), and Value undetectable (Vu), these are defined in [11].

Within the HAZOP editor it is possible to fill in the effects for the deviations: a set of hazards identified by the HAZOP process appears in Figure 7. Full details of the hazards (description and severity for example) can be entered – these appear in Figure 8. The dashes appearing in Effect entries are unused fields of Conditions: conditions can be formally linked to models via part (a component or a link), input/output, and property – in this example we only formally link to model CABSM1. We have illustrated how different deviations can lead to the same effect. Note that, rather unsurprisingly, most of the hazards were thrown up by consideration of deviations on *Actuator_control* – the ultimate output of the CABS.

Within the HAZOPS Editor there is an option to "export" these hazardous effects to the SAM database. SAM also provides a Hazard Log Editor, and we have attached a Hazard Log for the CABS to the goal structure in Figure 5. The Hazard Log Editor has an option to "import" all of the hazards from the database, these hazards may have been identified by a HAZOPS such as this, and also potentially from other sources such as FFA and FMECA if they had been performed. The hazard log also provides an indication of which piece of analysis discovered the hazard, and also which analyses have been subsequently carried out, for example FTA or ETA – it is possible to use the hazard log to navigate from hazards to all of the analyses associated with them.

Flow	Guideword	Effect
All_press	Vu	-
Actuator_control	O	CABSM1, -, -, -, No braking
Actuator_control	O	CABSM1, -, -, -, Lock up
Actuator_control	O	CABSM1, -, -, -, Inappropriate Control
Actuator_control	O	CABSM1, -, -, -, Tardy
Actuator_control	C	CABSM1, -, -, -, Unexpected
Actuator_control	C	CABSM1, -, -, -, Lock up
Actuator_control	E	-

Figure 7: Some Deviations and their Effects

Name	Description	Severity	Treatment
No braking	Complete Lack of braking	Cat	Hazard
Lock up	Lock up (1-4 wheels, 1-2 axles)	Cat	Hazard
Unexpected	Unexpected application / release of brakes	Cat	Hazard
Inappropriate Control	Braking response not proportional to demand	Maj	Hazard
Tardy	Too long from demand to brake effect	Maj	Hazard
Uneven	Pressures vary wildly with constant demand	Maj	Hazard
Unequal	1-3 wheels brake less or more than required)	Maj	Hazard

Figure 8: Details of Hazards Determined by the HAZOPS

Strategy S2 indicated that an FTA should be performed for each hazard, we illustrate this for the Inappropriate Control hazard, identified in Figure 8. In figure 5 we have a goal "*Do FTA for Inappropriate Control*" (GS2.2), and to this we attach a new Fault Tree (CABINAP2.FTA). Once again SAM provides support for navigating from the Goal Hierarchy to the solving Fault Tree.

The Fault Tree appears in Figure 9. Inappropriate Control can occur if a *wrong value* from *ENHANCED1* or *ENHANCED2* (abbreviated for convenience to E1 and E2 in Figure 9) **OR** *BASIC gets through*. A wrong value from *BASIC* can get through only if *ENHANCED1 failed to deliver value*, **AND** *ENHANCED2 failed to deliver value*, **AND** *BASIC delivered wrong value* (because of *OUT's ENANCED1*, *ENHANCED2*, *BASIC* priority ordering). A wrong value can be delivered by *BASIC* either because of *hardware failure* **OR** *BASIC calculating a wrong value*. A hardware failure could be caused by *memory corruption* **OR** *CAN Bus failure*. BASIC can calculate a wrong value either because of *sensor failure* **OR** *algorithm failure*. The diamonds in the Fault Tree are events that we choose not to develop any further, and ellipse events are assumed to be basic failures, box events are undesired events. It is usual to perform a Minimal Cut Set Analysis of Fault Trees which reduces them to disjunctive normal form (sum of products). Each conjunct will be a set of necessary and sufficient conditions to cause the hazard, which simplifies the mathematics if likelihoods are to be calculated.

We can get SAM to perform the cut set analysis, using the numeric labels of the Fault Tree Nodes we get the following cut sets:

$$2 \vee 3 \vee (5 \wedge 6 \wedge 10) \vee (5 \wedge 6 \wedge 11) \vee (5 \wedge 6 \wedge 12) \vee (5 \wedge 6 \wedge 13)$$

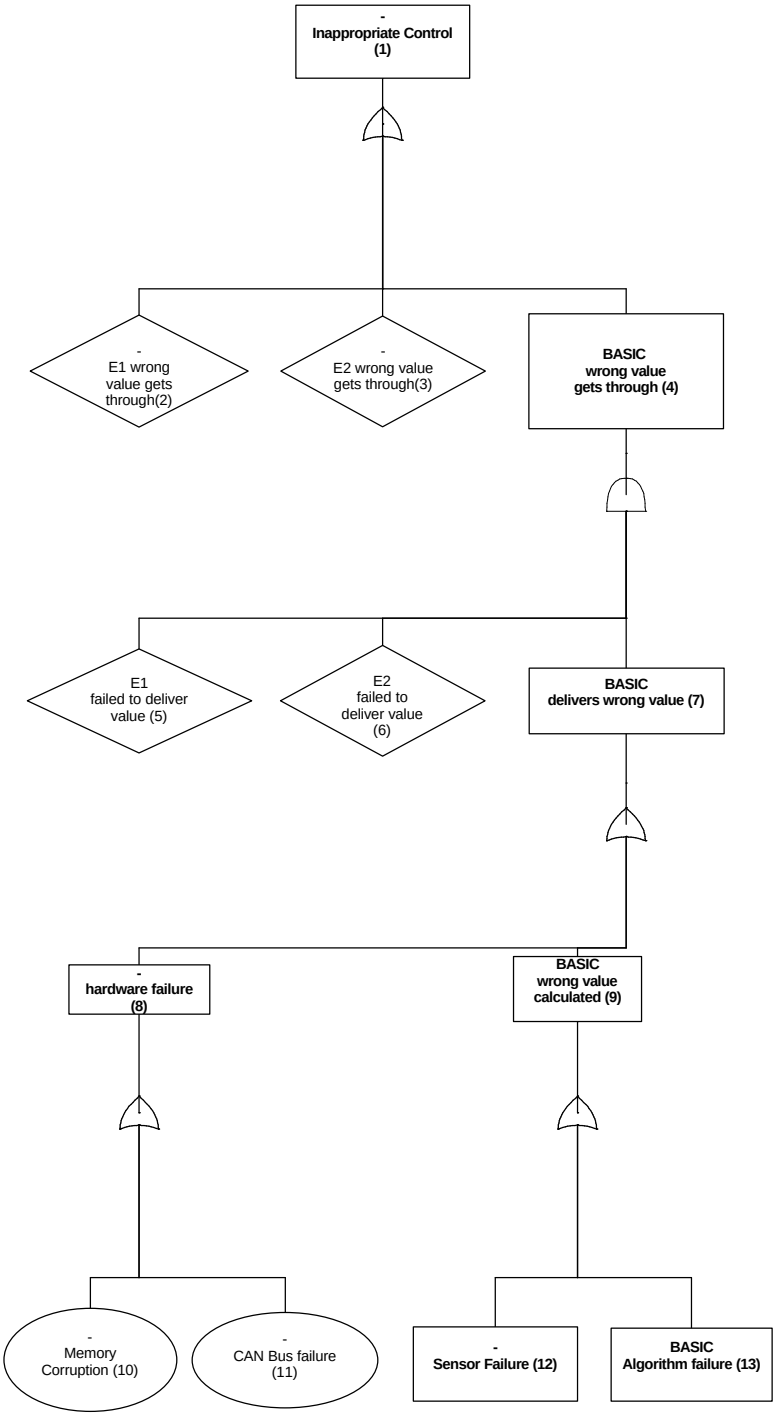


Figure 9: Fault Tree Analysis for Inappropriate Control Hazard

If we had probabilities for the primitive failures we could then calculate the likelihood for the Inappropriate Control Hazard. (Note that we are not recommending treating software failure probabilistically – merely observing that this can be accommodated, if so desired.)

We have a goal GS2.1.2 to "*mitigate or justify acceptance of hazards*" we provide "*mitigation against Inappropriate Control"* (GS2.1.2.1) as an example. We can determine how to mitigate against this hazard by examining the cut sets produced by the Fault Tree. We observe that if *ENHANCED1* or *ENHANCED2* provide a wrong value then the hazard will occur – this implies that we should make *ENHANCED1* and *ENHANCED2* "*fail silent"*. The remaining four causes relate to *BASIC* failing, and because of the ultimate reliance on *BASIC*'s value we should develop it to a "*high integrity*." We capture this through strategy S4, with its sub-goals GS4.1, GS4.2, and GS4.3. We provide an illustration of how we can factor in the new requirement on *BASIC* to its existing requirements via Strategy S5, and "*Implement BASIC with triple modular redundancy*". Thus we have illustrated how the safety analysis fits into the high level argument, and how it can be used to drive the detailed design.

5.4 Observations

We have been able to illustrate some, but not all, of the facilities offered by SAM (and which are under development). Other important current facilities are the hazard log, support for a variety of other graphical and tabular safety analyses, representation of assumptions and mechanisms for managing large scale graphical structures. We are investigating facilities for more effective management of large scale databases, including hierarchies, more explicit representation of criteria, including the ability to check the satisfaction of criteria,. As will be seen from the examples some work is also needed on presentation.

The example is necessarily fragmentary. A safety case for a true CABS would undoubtedly run to hundreds of pages (the example from which this is extracted is over 50 pages, without any FMEAs). Nonetheless we hope that it serves to illustrate the philosophy that we are advocating. We leave it to the reader to judge the potential of the approach for overcoming the problems laid out in section 1.2.

Finally, we believe that structuring the design and the assessment of the CABS in this manner, makes it much easier to assess the impact of change. Early work on change management is reported in [5]. Such structuring is also the first step to making safety case reasoning more amenable to reuse across different projects, and indeed this is an area of future work.

6 Conclusions

In this paper we have outlined the main problems encountered in the production of safety cases (*current practice*), and proposed a three pronged approach to solving them (*future prospects*): goal structuring to improve the clarity of High Level Arguments; the data model to improve the consistency of Supporting Evidence; and

a process model to improve the development and evolution of the safety case We have presented a detailed example illustrating a subset of these principles in use, and we have shown how the SAM tool can be used to assist in the development of safety cases.

Future research will focus on evolvable and reusable safety cases, extensions to the data model and the rules that can be captured. The potential for developing SAM into a commercial product is also being investigated.

7 Acknowledgments

The ideas presented in the paper are drawn from work being performed at the University of York as part of the ASAM-II (A Safety Argument Manager) project, grant number IED4/1/9212, and related work in the Rolls-Royce University Technology Centre (UTC). We are grateful to all of our project colleagues for their contribution to this work. We are also grateful to Mark Nicholson and David Pumfrey for allowing us to use their Computer Assisted Braking System case study, and also to Andrew Vickers for his comments. Any errors and omissions are of course our own.

Our definition of a Safety Case is based on that presented by John Stansfeld, of Lloyds Register of Shipping, at the "Workshop on Safety Case Management", University of York, 29th to 30th March 1994.

8 References

1. ASME. The American Society of Mechanical Engineers. Boiler and Pressure Vessel Code, Section III, Rules for the Construction of Nuclear Power Plant Components.
2. CISHEC. A Guide to Hazard and Operability Studies, The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd, 1977.
3. Collins R, Dent J N. A Practical Case Study of the Management of Reliability, Safety and other Concurrent Engineering Information. In proceedings of the Safety And Reliability Conference, Altrincham, 1994. pp 8/1-8/20.
4. Department of Defence. Procedures for Performing a failure mode, effects and criticality analysis, MIL-STD 1629A, 1980.
5. Fenelon P, Kelly T P, McDermid J A (1995). Safety Cases for Software Application Reuse. In the proceedings of SAFECOMP '95. Italy Oct 11-13th 1995 (to appear).
6. HSE. Safety Assessment Principles for Nuclear Plants. Health and Safety Executive 1992. HMSO Publications. ISBN 0 11 882043 5. 1992.
7. IEC. Technical Committee No.65: Industrial Process Measurement and Control, Sub-committee No. 65A: System Aspects. Draft International Electrotechnical Commission Standard (IEC) 1508 - Functional Safety: safety-related systems. 1995.
8. Joint Airworthiness Authority (JAA). JAR-25, Joint Airworthiness Requirements, Part 25. 1990

9. Kletz T. Hazop and Hazan: Identifying and Assessing Process Industry Hazards. 3rd Edition, Institution of Chemical Engineers. 1992.
10. McDermid J A. Support for safety cases and safety arguments using SAM, Reliability Engineering and System Safety, 43, 111-127. 1994.
11. McDermid J A, Pumfrey D J. A Development of Hazard Analysis to aid Software Design, COMPASS '94, Proceedings of the Ninth Annual Conference on Computer Assurance, June 94, 17--25.
12. Ministry of Defence. The Procurement of Safety Critical Software in Defence Equipment. Interim Defence Standard 00-55. 1995.
13. Ministry of Defence. Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment. Interim Defence Standard 00-56. 1995.
14. MISRA. The Motor Industry Software Reliability Association, MISRA Report 2: Integrity, February 1995.
15. NUREG. Fault Tree Handbook, W Veseley. Nuclear Regulatory Commission Washington D.C. 1981. NUREG 0942.
16. Nuseibeh B. A Multi-Perspective Framework for Method Integration, Ph.D. Thesis, Department of Computing, Imperial College, University of London, 1994.
17. Villemeur A. Reliability, Availability, Maintainability and Safety Assessment. Volumes I & II. J. Wiley and Sons. ISBN 0-471-93048-2. 1992.
18. Wilson S, McDermid J, Fenelon P, Kirkham P. No More Spineless Safety Cases: A Structured Method and Comprehensive Tool Support for the Production of Safety Cases. Institution of Nuclear Engineers Conference ‘95 2nd International Conference on Control and Instrumentation in Nuclear Installations, 19th-21st April 1995.
19. Wilson S P, J A McDermid. Integrated Analysis of Complex Safety Critical Systems. Computer Journal, Special Issue on Engineering Complex Systems 1995 (to appear late 1995).