# A Methodology for Safety Case Development

Peter Bishop
Adelard, London, UK

Robin Bloomfield
Adelard, London, UK

## 1 Introduction

A safety case is a requirement in many safety standards. Explicit safety cases are required for military systems, the off shore oil industry, rail transport and the nuclear industry. Furthermore, equivalent requirements can be found in other industry standards, such as IEC 1508 (which requires a "functional safety assessment") the EN 292 Machinery Directive ( which requires a "technical file") and DO 178B for avionics (which requires an "accomplishment summary").

It is important that an adequate safety case is produced for a system. In regulated industries such as the nuclear industry, the need to *demonstrate* safety to a regulator can be a major commercial risk. For example the computer-based Darlington Reactor Protection System in Canada required around 50 man years of software assessment effort which was probably more than the effort required to develop the software. In addition the assessment delayed reactor start up so that many millions of dollars of income were lost. So the need to demonstrate safety can involve significant direct costs and indirect costs if the overall project is delayed.

This paper will outline a safety case methodology that seeks to minimise safety risks *and* commercial risks by constructing a demonstrable safety case. The safety case ideas presented here were initially developed in an EU-sponsored SHIP project [1] and was then further developed in the UK Nuclear Safety Research Programme (the QUARC Project [2]). Some of these concepts have subsequently been incorporated in safety standards such as MOD Def Stan 00-55, and have also been used to establish specific safety cases for clients. A generalisation of the concepts also appears in Def Stan 00-42 Part 2, in the form of the software reliability case.

## 2 The safety case structure

We define a safety case as:

"A *documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment*"

To implement a safety case we need to:

- make an explicit set of claims about the system
- produce the supporting evidence
- provide a set of safety arguments that link the claims to the evidence
- make clear the assumptions and judgements underlying the arguments
- allow different viewpoints and levels of detail

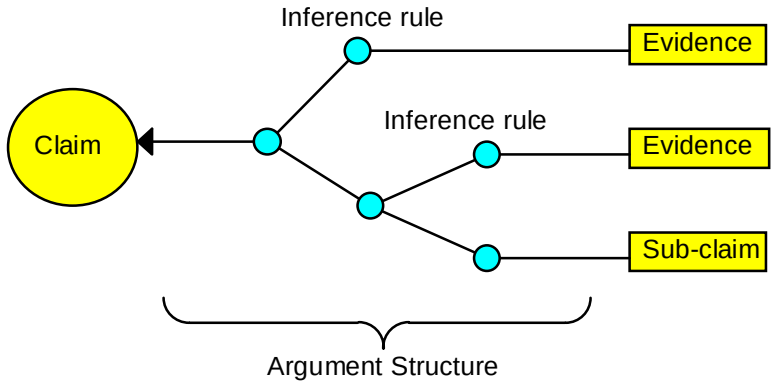
The following sections describe how we think a safety case should be structured to meet these goals.

## 2.1 Elements of a safety case

The main elements of the safety case are:

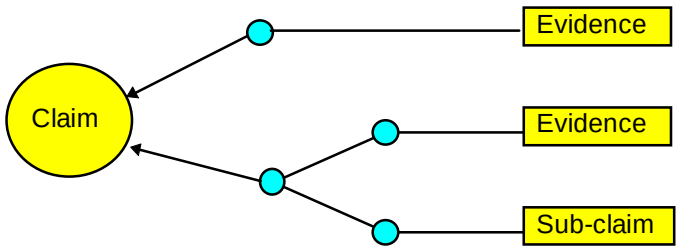
*Claim* about a property of the system or some subsystem.
*Evidence* which is used as the basis of the safety argument. This can be either *facts*, (e.g. based on established scientific principles and prior research), *assumptions*, or *sub-claims*, derived from a lower-level sub-argument.
*Argument* linking the evidence to the claim, which can be deterministic, probabilistic or qualitative.
*Inference* the mechanism that provides the transformational rules for the argument.

The use of these elements is illustrated in the figure below:




Note that "evidence" can be a *sub-claim* produced by a subsidiary safety-case. This means that there can be a relatively simple top-level argument, supported by a hierarchy of subsidiary safety cases. This structuring makes it easier to understand the main arguments and to partition the safety case activities.

It is also possible to have two (or more) independent arguments supporting the same claim, as illustrated below.




By using independent evidence (and possibly different styles of safety argument), the claim can be more robust, i.e. it can tolerate flaws in a single argument.

## 2.2 Types of claim

The safety case is broken down into claims about different attributes for the various sub-systems, e.g.:

| | |
|---|---|
| reliability and availability | usability (by the operator) |
| security (from external attack) | fail-safety |
| functional correctness | accuracy |
| time response | robustness to overload |
| maintainability | modifiability, etc. |

Note that the attributes listed are only examples and further attributes may be safety-relevant. Conversely, for some applications not all attributes need be safety-related, e.g. time response would not be safety-relevant for off-line stress analysis programs, but it would be necessary to have accuracy and functional correctness.

## 2.3 Types of argument

Different types of argument can be used to support claims for the attributes:

*Deterministic* application of predetermined rules to derive a true/false claim (given some initial assumptions), e.g. formal proof of compliance to a specification, or demonstration of a safety requirement (such as execution time analysis or exhaustive test of the logic)

*Probabilistic* quantitative statistical reasoning, to establish a numerical level (e.g. MTTF, MTTR, reliability testing)

*Qualitative* compliance with rules that have an indirect link to the desired attributes (e.g. compliance with QMS standards, staff skills and experience)

The choice of argument will depend on the available evidence and the type of claim. For example claims for reliability would normally be supported by statistical arguments, while other claims (e.g. for maintainability) might rely on more qualitative arguments such as adherence to codes of practice.

## 2.4 Sources of evidence

The arguments themselves can utilise evidence from the following main sources:

- the design
- the development processes
- simulated experience (via reliability testing)
- prior field experience

The choice of argument will depend in part on the availability of such evidence, e.g. claims for reliability might be based on field experience for an established design, and on development processes and reliability testing for a new design.

## 2.5 Example Arguments

Some example arguments for different claims, using different types of evidence, are shown in the following table.

| Attribute | Design Features | Assumption /Evidence | Subsystem Requirements | Claim |
|---|---|---|---|---|
| Functional Correctness | Partitioning according to criticality<br><br>Design simplicity | Assumption that segregated functions cannot affect each other | Subsystem integrity level<br><br>Functional segregation requirements | Claim that the composite behaviour of the critical functions implements the overall safety function |
| Fail-safety | Use of functional diversity<br><br>Fail-safe architectures | System Hazard Analysis<br><br>Fault Tree Analysis | Fail safety requirements for subsystems<br><br>(response to failure conditions) | Claim that safety is maintained under stated failure conditions, assuming the subsystems are correctly implemented |
| Reliability /availability | Architecture, levels of redundancy, segregation<br><br>Fault tolerant architectures<br><br>Design simplicity | Reliability of components, CMF assumptions<br><br>Failure rate, diagnostic coverage, test intervals, repair time, chance of successful repair<br><br>Prior field reliability in similar applications | Hardware component reliability<br><br>Software integrity level<br><br>Component segregation requirements<br><br>Fault detection and diagnostic requirements<br><br>Maintenance requirements | Reliability claim based on reliability modelling and CMF assumptions, together with fault detection and repair assumptions<br><br>Reliability claim based on experience with similar systems |
| Response Time | Design ensures overall response time is bounded | Assumes sub-system time budgets can be met | Time budgets for hardware interfaces, and software | Claim that overall system design can meet target time response |

# 3 Implementing the safety case

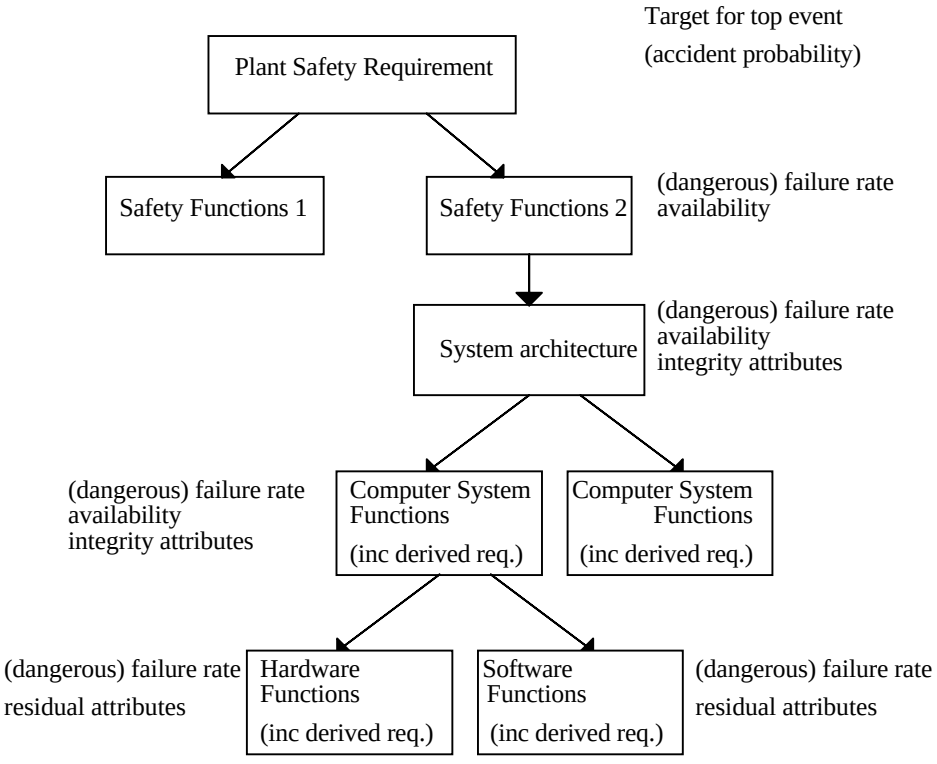
So far we have only discussed the structure of the safety case, and now we need to consider the process for producing the safety case. Many of the problems in producing an acceptable safety case arise from an attitude that regards the safety case as a "bolt-on" accessory to the system (often produced after the system has been built). At this stage it is often discovered that "retro-fitting" the supporting safety case is both expensive and time consuming.

We recommend a different approach where the safety case is considered throughout the project. In our approach we advocate:

- Integration of the safety case into the design and development process
- "Layered" safety cases, i.e. a top-level safety case with subsidiary safety cases for subsystems
- Traceability between system and subsystem levels
- "Design for assessment" which takes into account the costs and complexity of the safety case as well as the design.

## 3.1 Example of a layered safety case

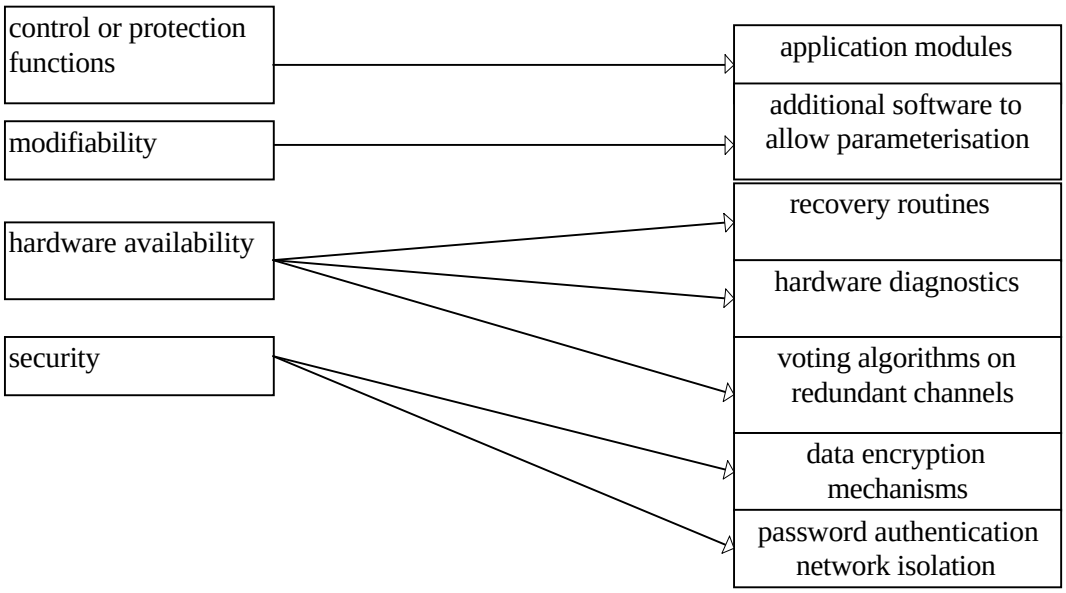
An example of a layered safety case is shown below. This starts at the top-level with the overall safety target (a worst case accident rate). This top-level requirement is progressively transformed into *derived requirements* for subsystems.




## 3.2 Traceability between levels

As shown in the figure above, the top-level requirements are transformed into derived requirements. Initially these might be attributes such as "security" or "maintainability", but at a

more detailed level of implementation these requirements will be converted into design requirements that are implemented in one or more subsystems. It is important that there is traceability between these levels so that there is a clear link between the design features and the safety attributes. The subsidiary safety cases for the subsystems should identify the design features and present arguments to support claims that they implement the safety attributes. The traceability between levels is illustrated in the figure below:




### 3.3 Design for assessment

"Design for assessment" integrates the production of the safety case with the design of the system. Typically, some candidate design options will be identified and a preliminary safety case will be constructed. This will normally be an iterative process, which involves the identification of hazardous subsystem states (e.g. through some form of hazard analysis), and appropriate countermeasures (elimination, reduction and failure mitigation). The design and safety case are then assessed to establish whether:

- the design implements the safety functions and attributes
- the design criteria are satisfied
- the design is feasible
- the associated safety arguments are credible
- the approach is cost-effective

In this assessment process, the costs of implementing the safety system *and* the associated safety case should be considered during the architectural design phase. This analysis should also include a consideration of the *long-term* safety risks and lifecycle support costs (e.g. changing the safety functions, changing the hardware, maintaining the equipment, maintaining the associated safety case, etc.).

By integrating the safety case into the design, the feasibility and cost of the safety case construction and maintenance can be evaluated in the initial design phase. This should help exclude unsuitable designs and enable more realistic design trade-offs to be made. It is difficult to be specific about the choice of appropriate design and safety case options that are likely to be

both cost effective and convincing, but some general "rules of thumb" for minimising costs and risk are:

- use a simple design (eases analysis)
- avoid novelty (use established designs or components with known performance)
- ensure supporting evidence is readily available

## 3.4 The safety case life-cycle

As noted earlier, the safety case life-cycle should be an integral part of the overall system development, and this should continue throughout the lifetime of the system. Quite often, the safety case will include assumptions about the behaviour of components (e.g. reliability or fail-safe bias) which are plausible but unverified at the time the system is accepted. In this case there may be a conditional acceptance of the system, and certain "areas of concern" may need to be explicitly monitored during actual operation. The main stages of safety case evolution are listed below.

- Safety functions and top-level safety attributes identification
- System architecture and outline safety case identification
- Preliminary assessment of design options:
  - costs and risks (implementation and safety case)
  - long-term support
- Progressive elaboration of the design and safety case in parallel:
  - safety case requirements part of subsystem specifications
  - reviews of subsystem safety cases
- Integration into final safety case
- Long-term support infrastructure plans
- Approval
- Long-term monitoring and audits
  - areas of concern
  - support processes
  - gathering field evidence to support assumptions
- System updates and corrections

## 3.5 Safety case contents

The safety case is "living document" which evolves over the safety life-cycle. Since it records the safety argument, the basic structure should remain broadly similar over time, but the status of the evidence will change. For example, planned levels of test coverage are replaced by test evidence on the achieved level of coverage. In practice of course the safety case could be split into several documents (e.g. covering specific subsystems) and would also refer to supporting documents (e.g. design documents, analysis reports, test reports etc.).

We consider that the following items should be included in the safety case document:

- Environment description
  external equipment, interfaces, failure modes, hazardous/safe states, potential changes

- PES safety requirements
  safety functions, reliability and other safety attributes, anticipated changes
- PES system architecture
  subsystems, interconnections, subsystem derived functions, integrity levels, design constraints, evidence
- Planned implementation approach
  PES system architecture safety argument
  (at least one safety argument for each requirement)
  identify all design assumptions used in the argument (e.g. claim limits, failure modes)
  identify supporting evidence and analyses (e.g. SHA, HRA, RAMS)
- Subsystem design and safety arguments
  (similar to the main safety case)
- Long term support requirements
- PES maintenance and operation procedures
  safety case support infrastructure
- Status information
  safety case evidence, design assumptions, outstanding concerns for subsystems, unresolved hazards
- Evidence of quality and safety management
  results of QA audits, safety audits, evidence that identified problems are resolved
- References

Note that evidence of quality and safety management is included because it is important to have confidence in the validity of the underlying evidence supporting the development of the system and its safety case.

# 4 Concluding remarks

We have presented an outline of our safety case methodology. Our safety case approach places the main emphasis on claims about the behaviour of the system (i.e. functional behaviour and system attributes) and suitable arguments to support those claims. The structuring ideas (using claims, argument and evidence) are quite simple, but they should allow quite complex safety cases to be constructed which are both understandable and traceable. The approach also allows for multiplicity of argument approaches.

To implement the safety case we have advocated the integration of safety case development into the design process. By including the safety case and its possible costs in the design trade-offs, unsuitable designs can be avoided together with their attendant costs in safety case construction, project delays and long-term support overheads.

The layered structure of the safety case allows the safety case to evolve over time and helps to establish the safety requirements at each level. For large projects with sub-contractors, this “top-down” safety case approach helps to identify the subsystem requirements and the subsystem safety case can be made an explicit contractual requirement to be delivered by the sub-contractor.

We have developed this methodology over several years. Initially the ideas were the product of research studies, but they have subsequently been adopted in standards and for the development

of safety cases for specific systems. The approach has evolved during this period, but the evolution is largely through extensions to the methodology (including long-term support) rather than changing earlier ideas.

We have also been working on developing a suitable computer-based support environment for the supporting documentation using commercially available Web browser technology [3], and further work is planned for recording and linking the safety case argument to the supporting evidence within such an environment.

While the methodology is likely to evolve further, we believe that our current safety case methodology provides a good basis for safety case development.

## References


[1] The SHIP project (ref. EV5V 103) was carried out with financial support from the CEC in the framework of the Environment Programme, sub-theme: Major Industrial Hazards.

[2] The QUARC2 project was funded by the UK (Nuclear) Industrial Management Committee (IMC) Nuclear Safety Research Programme under Scottish Nuclear contract PP/74851/HN/MB with contributions from British Nuclear Fuels plc, Nuclear Electric Ltd, Scottish Nuclear Ltd and Magnox Electric plc.

[3] Adelard Linkbase Tool Demonstration, http://www.adelard.co.uk/