Safety Case Construction and Reuse using Patterns

T P Kelly, J A McDermid

High Integrity Systems Engineering Group
Department of Computer Science
University of York
York YO1 5DD

E-mail: tpk|jam@cs.york.ac.uk
Phone: +44 0|1904 434728 Fax: +44 0|1904 4322708

Abstract

This paper presents an approach to the reuse of common structures in safety case arguments through their documentation as 'Safety Case Patterns'. Problems with the existing, informal and ad-hoc approaches to safety case material reuse are highlighted. We argue that through explicit capture and documentation of reusable safety case elements as patterns, the process of safety case construction and reuse can be made more systematic. For the description of patterns a safety case pattern language and a graphical pattern notation (based on the Goal Structuring Notation) are presented. Using this framework we briefly describe a number of example argument patterns. A fully documented example pattern is also included.

Introduction

The purpose of a safety case is to present the argument that a system, be it physical or procedural, is acceptably safe to operate. This argument should demonstrate how the available evidence concerning the system can be reasonably and defensibly interpreted as indicating compliance with the system safety requirements. As such, each safety case will ultimately be specific to a particular system - defined by both the details of the available evidence and safety requirements. However, amongst these 'specific' safety cases, patterns of argument emerge through, for example, common approaches to addressing a standard requirement or class of requirements, typical combinations of argument and accepted interpretations of specific types of evidence.

Informal reuse of safety case material is already commonplace, especially within stable and well-understood domains, e.g. aerospace engine controllers. However, this type of uncontrolled and often ad-hoc reuse can fail to fully exploit opportunities for reuse, and can in some cases be potentially dangerous. This paper describes the problems of informal safety case reuse and introduces the concept of *Safety Case Patterns* as a means of explicitly and clearly documenting common elements found between safety cases.

The Problems of Informal Safety Case Material Reuse

Much of existing safety case material reuse is triggered by, and centres around, the safety engineers responsible for the production of the safety case. It is not uncommon for an engineer, having recognised a similarity, to plunder a previously developed safety case to help in the development of a safety case in a new project. In some cases, the engineer may believe certain elements of the two projects to be sufficiently similar to actually "cut-and-paste" parts of the original documentation and subject them only to minor review and modification.

The central role of people in the reuse of safety artefacts is often crucial: many existing safety cases fail to clearly present the intent and rationale of the safety arguments and safety processes. Such safety cases cannot easily be read and understood in a way that permits re-application of principles propounded. They require interpretation. To understand the intent of the safety case can take many readings. To understand the rationale behind elements of a safety case can require a form of 'reverse engineering'. Safety cases with these properties are not readily amenable to reuse. Therefore, the safety engineers who worked on a safety case form an important 'missing link' in any attempt to gain value from it in future safety case developments. However, a number of potential problems arise where people are the **principal** medium for cross-project reuse of safety case artefacts, including:

- **Artefacts being reused inappropriately**
  If the original context of a safety case artefact is not fully recognised the artefact may be applied inappropriately in another context. An argument of safety from one context that is **not** applicable in the context in which it is reused can create a false or misleading picture of a system's safety. Such reuse can

carry "hidden assumptions" from the original context that are inconsistent with the application context. This danger is obviously greatest with the extreme "cut-and-paste" reuse.

- **Reuse occurring in an ad-hoc fashion**
  Reuse is dependent entirely on the engineer's ability, firstly, to *recognise* the potential to reuse some artefact and, secondly, to *recall* the appropriate information. Consequently, reuse often occurs in a fairly random, opportunistic, fashion and cannot be said to be exploited systematically. Opportunities to reuse an artefact may be wasted.
- **Loss of knowledge**
  A *total* reliance on people to achieve cross-project reuse is an admission that project documentation is insufficient to support systematic reuse. A danger is that particular people, the company 'experts', become a bottleneck on any project. Without documentation of their experience or expertise, they become a critical resource in an organisation. They effectively act as an 'index' into the organisation's existing documentation. If such people leave an organisation, disproportionately large amounts of the organisation's 'corporate memory' is lost and, as a result, less reuse is possible.
- **Lack of Consistency / Process Maturity**
  Without explicitly recognising and documenting the repeatable elements of safety case development there can be no assurance that these elements are being used consistently. If an element is not consistently applied, it is difficult to argue that the associated development process is mature. It is also difficult to argue how this process has been, and will be, improved and evolved over time.
- **Lack of traceability**
  Informal reuse is often invisible in the final safety case produced. No record is kept of reuse from existing documentation. This lack of traceability can lead to problems in maintaining the safety case. For example, if it were found that a particular *reused* safety argument was unsound (e.g. in the light of contradictory operational evidence), it would be necessary to locate all uses of that argument in order to update all appropriately. With no record of where it was reused this could be an extremely difficult task. Reuse has the potential to propagate one error many times. To deal with such potential situations requires adequate visibility and traceability of the reuse process.

These problems stem from the key issue of **documentation**. The process of safety case reuse must be explicitly recognised and documented in order to control and support it. This involves identifying and abstracting reusable elements from existing safety cases; documenting them with information defining their characteristic function(s), applicability, record of applications, etc. Once such things are "down on paper" they can be evaluated, exploited, evolved and traced. The following section introduces the concept of recorded patterns as a means of documenting the common elements of safety case construction.

Patterns

The use of patterns as a means of describing common elements (or 'themes') of complex structures was first documented in the field of building architecture. Christopher Alexander, in his book, "The Timeless Way of Building" [1], argues that "Beyond its elements each building is defined by certain patterns of relationships amongst its elements". Alexander shows how patterns can be used to abstract away from the details of particular buildings and capture something essential to the design (the principles underlying the building; the reasons why elements of the building are successful or unsuccessful) that can then be used elsewhere. In later books, "A Pattern Language" [2], and, "The Oregon Experiment" [3], Alexander describes in more concrete terms how patterns can be documented and applied using pattern languages.

Influenced by Alexander's work, over the last five years the concept of patterns and pattern languages has received increasing interest from software designers [4,5,6]. Designers have turned to patterns as a means of capturing the repeatable and successful elements of a software design. Many have been disappointed with the unfulfilled promise of traditional component-based (compositional) reuse and believe that successful reuse lies is the ability to describe higher level software structures [7]. These structures communicating how components are combined to achieve certain functions, the principles of interfacing components, etc. The attraction of patterns is that they offer this means of abstracting fundamental design strategies from the details of particular designs.

Informal analysis of a number of safety cases suggests that patterns provide an appropriate level of abstraction to make safety case artefacts reusable without significantly reducing the benefit per application. Reuse of the specifics of safety cases (e.g. particular pieces of evidence) will largely be unsuccessful, as between different safety cases these are likely to change. However, reuse of the general principles of safety cases (i.e. the *why*s and *how*s of the construction of the safety case) is likely to be more successful as these

are more constant between safety cases. Patterns can provide a means of describing these general principles, structures and processes of the safety case.

To describe safety case patterns requires a pattern language. The following section proposes a safety case pattern language, adapted from those described for software design patterns.

A Safety Case Pattern Language

As with the design patterns of Gamma et. al. [8], safety case patterns are an attempt to capture solutions that have evolved over time. A safety case pattern should be a simple and efficient solution to a particular problem, whether it be the execution of a safety process or the construction of a particular safety argument.

In [8] a pattern format is proposed for describing design patterns. This format has been adapted for the description of safety case patterns:

Pattern Name and Classification	The pattern's name should convey the essence of the pattern succinctly. A good name is vital because, with use, it will become part of your design vocabulary.
Intent	A short statement that answers the following questions: What does the pattern do / represent? What is it's rationale and intent? What particular safety issue / requirement / process does it address?
Also Known As	Other well known names for the pattern, if any.
Motivation	A scenario that illustrates a safety issue / process and how the elements of the goal structure solve the problem. The scenario will help you understand the more abstract description of the pattern that follow.
Applicability (Necessary Context)	What are the situations in which the safety case pattern can be applied? What information is required as context for the pattern to be successful (necessary inputs to the pattern). How can you recognise situations in which the pattern can be applied.
Structure	A graphical representation of the pattern using the extended form of the goal structuring notation. The representation can describe a product or a process style goal structure. Where the structure indicates generality or optionality it should be clear how the pattern can be instantiated.
Participants	The elements of the goal structure and their function in the pattern.
Collaborations	How the participants collaborate to carry out the function of the pattern.
Consequences	How does the pattern support its objectives? What are the trade-offs and results of using the pattern? For a product oriented pattern - what are the principal arguments put forward? For a process oriented pattern - what are the outputs of the activities described?
Implementation	What pitfalls, hint or technique should you beware of when using the pattern? What degrees of flexibility are there in following the pattern?
Sample Text	Text fragments that illustrate how you might describe the pattern in the final safety case / safety plan.
Known Uses	Examples of the patterns application in existing safety documentation should be cited. If possible examples from two different applications should be shown.
Related Patterns	Safety Case Patterns that are related to this pattern, e.g. with the same motivation but different applicability conditions (e.g. different standards, different systems). For a process orientated pattern, related product (argument) patterns. For a product orientated pattern, related process patterns.

The principal differences between the format for design patterns and format for safety case patterns are firstly, the use of the **Goal Structuring Notation** (GSN) [9] rather than Rumbaugh's **Object Modelling Technique OMT** [10] to graphically describe the structural details of the pattern, and secondly the use of

sample text (for the eventual safety case) rather than sample source code. The use of GSN for pattern description is described in the next section.

A key element of the pattern format when applied to safety cases is the notion of pattern **applicability**. Applicability defines under what circumstances the pattern can be legitimately applied. For example, descriptions of applicability could indicate which standards the pattern adheres to, the level of design detail required or the assumption of system behaviour. Applicability of a safety case pattern is perhaps more closely tied to the structural description of the pattern than with design patterns. The goal structure representation of the pattern may specifically require certain elements to be present in the goal structure into which the pattern is placed (e.g. a **context** entity, **model, assumption or constraint**).

A difference in emphasis between the design patterns of Gamma and safety case patterns is that, in addition to the pattern rationale being documented by **intent** and **motivation,** the rationale behind a safety argument or safety process should also be embedded in the elements of the structural description using the goal structuring notation (through the conventional use of the GSN elements - **strategies**, **assumptions** and **justifications**).

Safety case patterns are intended to describe **partial** solutions and will not typically describe the complete structure of a system safety argument. It is expected that a collection of patterns will therefore emerge over time forming a 'recipe book' of safety arguments and processes, a number of which would be used together to aid in the construction of the safety case.

Structural Pattern Description using the Goal Structuring Notation

The Goal Structuring Notation (GSN) [9] was developed for the description of safety arguments, relating the breakdown of safety requirements to arguments based upon available evidence. Figure 1 shows an example goal structure, illustrating the key elements of the notation.

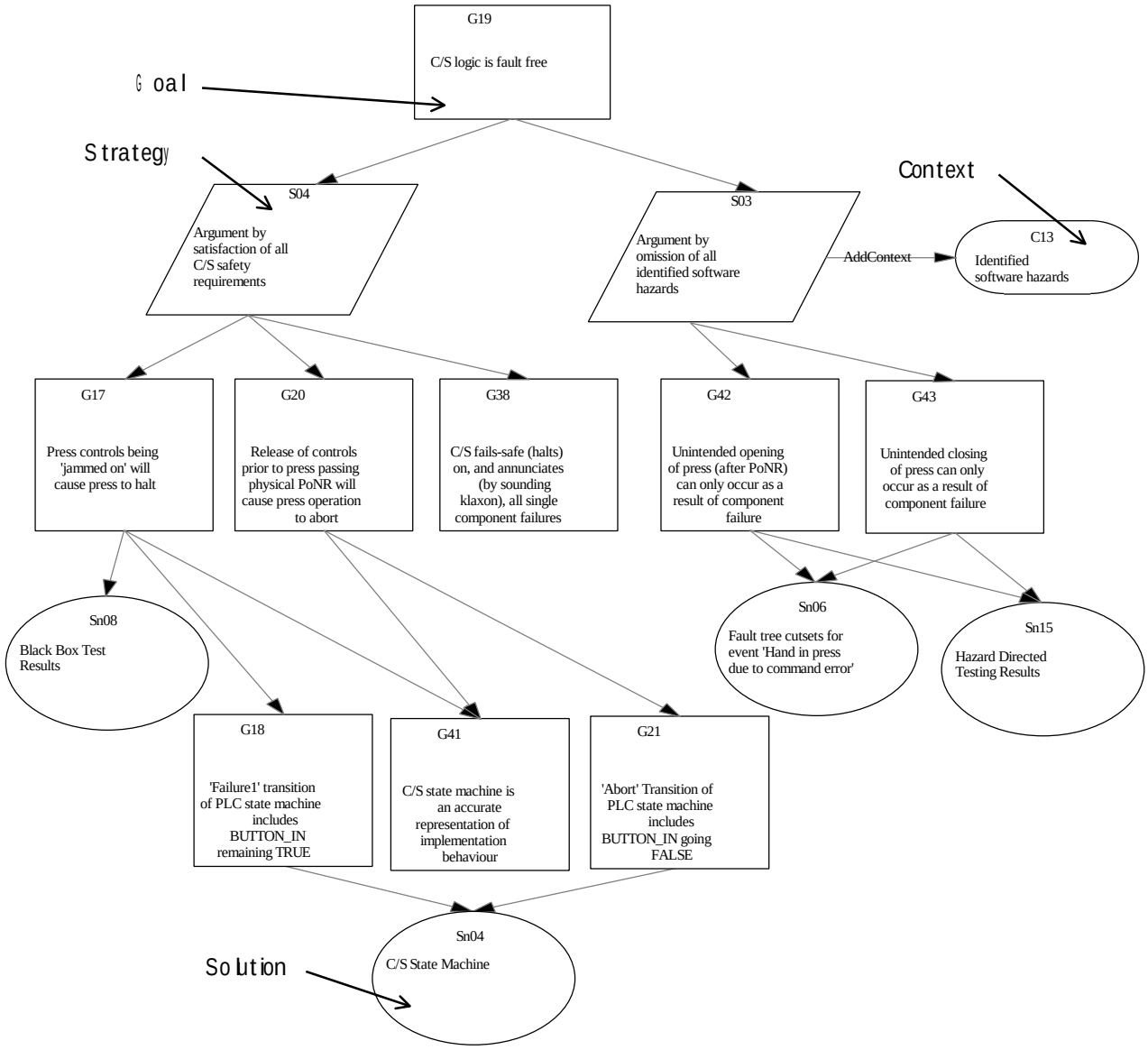


Figure 1. An Example Goal Structure

In the structure shown in figure 1, as in most, there exist 'top level' goals – statements that the goal structure is designed to support. In this case, *"C/S (Control System) Logic is fault free"*, is the (singular) top level goal. Beneath the top level goal or goals, the argument is broken down into sub-goals, either directly or, as in this case, indirectly through a strategy. The two argument strategies put forward as a means of addressing the top level goal in this structure are *"Argument by satisfaction of all C/S (Control System) safety requirements"*, and, *"Argument by omission of all identified software hazards"*. These strategies are then substantiated by five sub-goals. At some stage in a goal structure, a goal statement is put forward that need not be broken down and can be clearly supported by reference to some evidence. In this case, it is shown that the goal *"Unintended Closing of press after PoNR (Point of No Return) can only occur as a result of component failure"*, is supported by direct reference to the solutions, *"Fault tree cut-sets … "* and *"Hazard Directed Testing Results"*.

In its existing form, GSN can be used to express details of a *specific* safety argument, e.g. as shown in figure 1. However, in order to express *patterns* rather than simply *instances*, and perform the equivalent role for safety case patterns that OMT performs for design patterns, GSN must also be capable of representing generalisations of goal structures. For this reason, a number of extensions have been made to GSN to support entity and structural abstraction over the existing elements. Figure 2 shows a simple goal structure pattern that uses these extensions. (This pattern is described in the following section.)

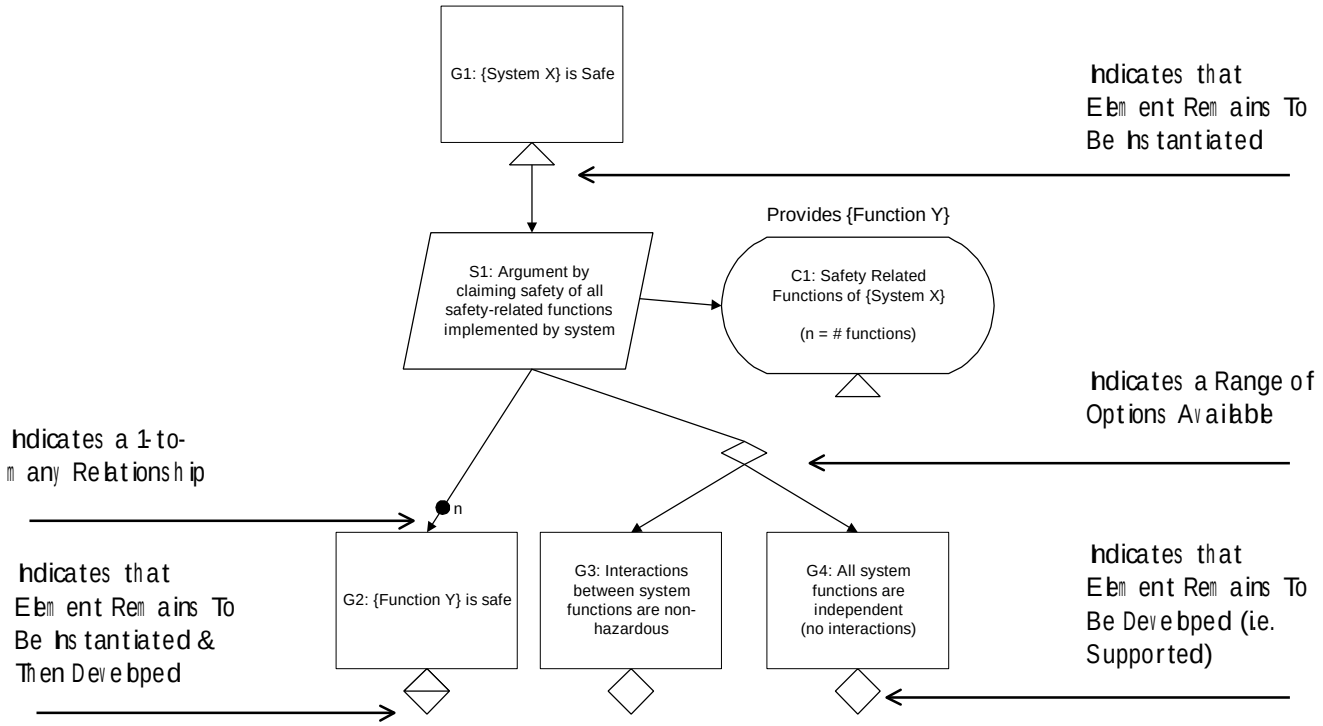


Figure 2. Extensions for Structural Abstraction

Example Safety Case Patterns

Patterns can emerge at many different levels in the safety argument and at varying degrees of specificity. At the highest level it is possible to identify a number of basic argument structures that are used to decompose ill-defined system safety requirements. For example, against the ultimate top level requirement …

"{System X} is safe"

… two possible argument approaches could be applied:

- **Hazard Directed Argument**
- **Functional Decomposition Argument**

Figure 3 shows the GSN pattern (without accompanying text) representing a hazard directed argument.

In this pattern, the implicit definition of 'safe' is 'hazard avoidance'. The requirement G1 is addressed by arguing that all identified hazards have been addressed (S1). This strategy can only be executed in the context of some knowledge of plausible hazards, e.g. identified by Hazard Analysis. Given this information (C1), identifying n hazards, n sub-goals of the form G2 can be constructed. The argument then develops from this 'hazard avoidance' goals.

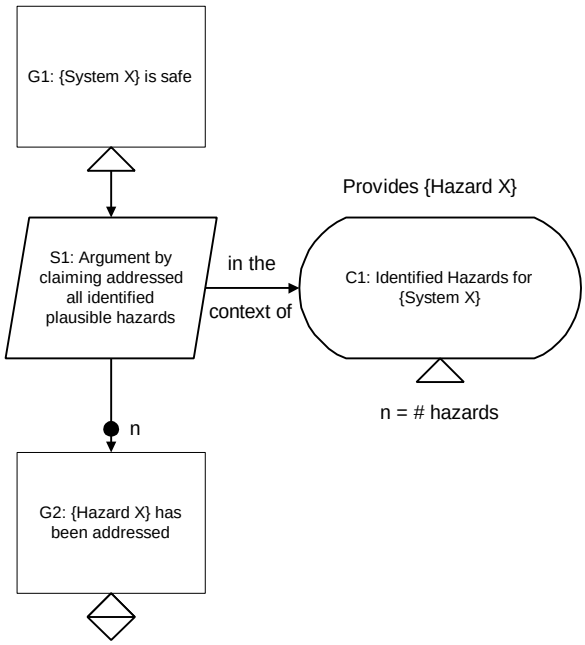


Figure 3. Hazard Avoidance Pattern

In the previous section, Figure 2 shows the GSN pattern (again, without accompanying text) representing a functional decomposition argument. In this structure, the top level goal of system safety (G1) is re-expressed as a number of goals of functional safety (G2) as part of the strategy identified by S1. In order to support this strategy, it is necessary to have identified all system functions affecting overall safety (C1) e.g. through a Functional Hazard Analysis. In addition, it is also necessary to put forward (and develop) the claim that either all the identified functions are independent, and therefore have no interactions that could give rise to hazards (G4) or that any interactions that have been identified are non-hazardous (G3).

At lower levels in the safety case argument, patterns also emerge. For example, when arguing the safety of software it is often common to claim a level of software integrity from an appeal to having used best practice tools, techniques and methods during development and testing. Other common argument structures emerge from the use of particular techniques. For example, to support the claim that a particular software condition cannot arise, a pattern could be identified showing the typical use of either *formal verification*, *Software Fault Tree Analysis (SFTA)*, or *black box testing* – each strand of argument having its own associated arguments to develop (e.g. that the formal specification is an accurate representation of the final target code (for formal verification), that sequential composition has been appropriately represented (for SFTA), that sufficient coverage achieved (for testing) .

Figure 4 shows an example pattern that could be found in the lower levels of a safety case argument.

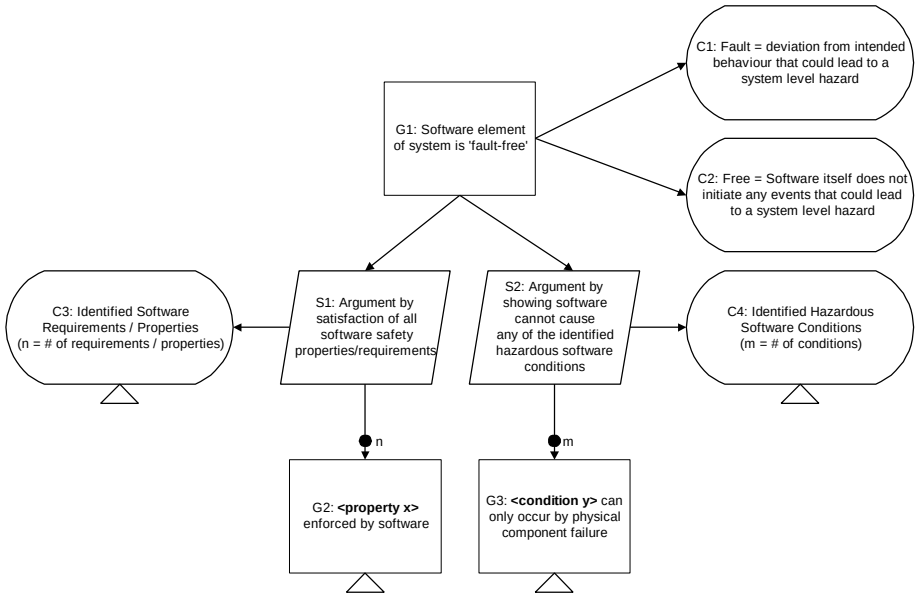


Figure 4. Fault Free Software Pattern.

In this pattern, the claim that the software element in a system is 'fault free' (G1) is supported by two main strands of argument (S1 and S2). First, over a list (C3) of identified hazardous software conditions (e.g. *"Controller demands speed greater than maximum safe speed"*) the *m* sub-goals of the form G3 are expressed, to argue that these hazards can only occur through physical component failures. Second, over a list (C4) of identified software requirements (e.g. *"Operation will not start if operator detected near machinery"*) the *n* sub-goals of the form G2 are expressed to argue that these properties are enforced in the software. In order that this pattern will be appropriately applied, the context of the pattern is made clear through the elements C1 and C2 - both defining key terms in the top level claim.

The example patterns given here are deliberately general, as they can be readily understood and have wide applicability across technologies and regulatory contexts. However, in well understood and stable domains it is possible to identify argument patterns at a greater level of specificity. For example, in the civil

aerospace sector common arguments are often developed against particular individual regulations (in Europe from the Joint Aviation Requirements, in the U.S. from the Federal Aviation Requirements) - e.g. capturing what is an acceptable approach to arguing that "*Thrust Reverser will not deploy during flight*".

An example of a pattern complete with supporting text is provided as an appendix to this paper. This pattern presents an approach to arguing satisfaction of the ALARP (As Low As Reasonably Practicable) Principle at the highest level in a safety case.

Using Patterns in Argument Construction

It is intended that, over time and within individual domains, collections of safety case patterns will be developed. These collections will be used as 'recipe books' for future safety case developments. When faced with particular requirements to support, engineers will then be able to retrieve and execute the approach as defined by the corresponding pattern. As well as potentially saving development effort, using patterns in argument construction in this way addresses many of the identified problems of informal reuse:

- **Artefacts being reused inappropriately**
  Through documentation of artefacts as patterns, including documentation of applicability and clear description of the required context (both in the text and in the structural pattern) - inappropriate use of material is made less likely.
- **Reuse occurring in an ad-hoc fashion**
  Through the development of a core 'recipe book' of patterns, opportunities for reuse can be more easily identified and exploited.
- **Loss of knowledge**
  Documentation through patterns, especially including the supporting text, helps to explicitly capture the knowledge developed within an organisation.
- **Lack of Consistency / Process Maturity**
  Through the development of a core 'recipe book' of patterns, the consistency of approach between developments can be more readily encouraged and supported.
- **Lack of traceability**
  Through the more explicit reuse of material as patterns, ease of recording traceability information (e.g. documenting those (versions of) patterns used within a new development) is improved.

Conclusions

There is potential for reuse of material between safety case developments. This is borne out by the levels of informal reuse instigated by safety engineers. However, there are a number of deficiencies with such an ad-hoc approach. Documentation of common safety case argument structures as patterns provides a suitable medium through which to foster systematic artefact reuse and aid in the development of new safety cases.

References

[1] 'The Timeless Way of Building', Christopher Alexander, Oxford University Press, New York, 1979

[2] 'A Pattern Language', Christopher Alexander, Oxford University Press, New York, 1977

[3] 'The Oregon Experiment', Christopher Alexander, Oxford University Press, New York, 1975

[4] 'Patterns and Software Development', Kent Beck, Dr. Dobbs Journal, 1993 vol. 19 no. 2 pp 18-23

[5] 'Patterns', Grady Booch, Object Magazine, 1993 vol. 3 no.2

[6] 'Object-Oriented Patterns', Peter Coad, Communications of the ACM, September 1993 vol. 35 no. 9 pp 153-159

[7] 'Reusing software: issues and research directions', H. Mili, F. Mili and A. Mili IEEE Transactions on Software Engineering, June 1995 vol.21 no. 6 pp 528-62

[8] 'Design Patterns: Elements of Reusable Object-Oriented Software', Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Addison-Wesley, December 1995

[9] 'SAM User Manual', Stephen Wilson, Pete Kirkham, University of York, December 1995

[10] 'Object-Oriented Modeling and Design', James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, Prentice-Hall, 1991

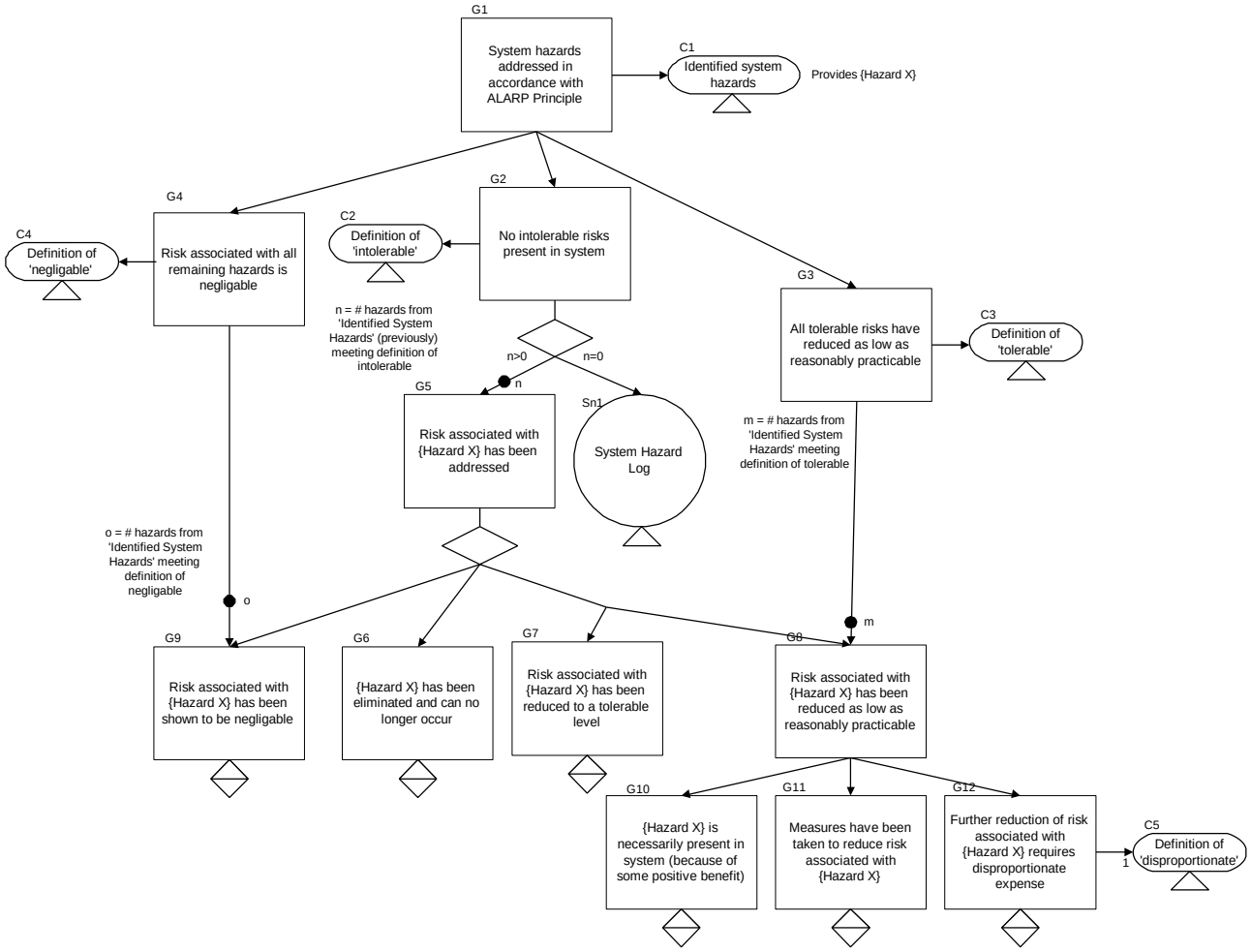
Appendix: ALARP (As-Low-As-Reasonably-Practicable) Pattern

ALARP (As-Low-As-Reasonably-Practicable) Pattern

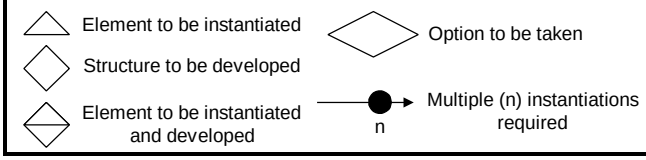
Author	Tim Kelly		
Created	04/02/97 10:41	**Last Modified**	05/02/97 09:47

Intent	This pattern provides a framework for arguing that identified risks in a system have been sufficiently addressed in accordance with the ALARP principle.
Also Known As	• Risk Reduction Argument Pattern
Motivation	This pattern was developed for two reasons: • To argue compliance with the ALARP principle at the highest level when addressing system level hazards. • To provide a more structured approach to presenting a 'Hazard Avoidance' argument (See Hazard Avoidance Pattern) by showing differing treatment of hazards according to their associated risk.

Structure



Key



<table>
<tr><td rowspan="11">Participants</td><td>G1</td><td>Defines the overall objective of the pattern</td></tr>
<tr><td>G2, G3, G4</td><td>Defines targets for three classes of identified risks: negligible, tolerable, and intolerable</td></tr>
<tr><td>Sn1</td><td>Provided at this point to support the claim that no intolerable risks have (ever) been identified with the system</td></tr>
<tr><td>G6 or
G7 and G8</td><td>Claims either that hazard has been eliminated or associated risk reduced to a tolerable level and dealt with as a tolerable risk.</td></tr>
<tr><td>G8</td><td>Defines ALARP target for each identified tolerable risk</td></tr>
<tr><td>G10, G11, G12</td><td>Claims required to support ALARP target:
Hazard only acceptable if positive benefit achieved
Risk reduction measures have been taken up to the point where further measures would be disproportionate to benefit gained.</td></tr>
<tr><td>G9</td><td>Claim for each remaining hazard that associated risk shown to be negligible</td></tr>
<tr><td>C1</td><td>A context identifying all system hazards, including indication of associated risks (e.g. Risk Category from A, B, C, D).</td></tr>
<tr><td>C2, C3, C4</td><td>A workable definition of 'intolerable'/ 'tolerable'/ 'negligible' risks that can be used as a basis for selection from the list of hazards(e.g. Intolerable = Risk Category A, Tolerable = Risk Category B or C, Negligible = D).</td></tr>
<tr><td>C5</td><td>The ALARP principle relies on some understanding of when it is no longer cost-effective to spend further money on risk reduction. This element, a definition of cost-effectiveness, is therefore required.</td></tr>
<tr><td>Collaborations</td><td colspan="2">An important aspect of this pattern is that it divides and conquers the goal of hazard mitigation / elimination according to the level of risk associated with each hazard. There are three strands to the safety argument: one tackling intolerable risks, one tackling tolerable risk and one discounting negligible risks. To satisfactorily support the top level goal (G1) it is important that these three strands address all identified risks. The definitions of tolerable, intolerable and negligible (C3, C2 and C4 respectively) should therefore be so defined to cover and classify the range of possible levels of risks.

It should also be noted that the definitions of negligibility (C4) and disproportionate (C5) cannot be considered entirely independently. It would not make sense, for example, to force risk reduction to a level below that identified elsewhere as negligible.

As the goal structure shows, if the means of addressing a previously identified intolerable risk is to reduce it to a tolerable level, then the remaining risk must be tackled as for all tolerable risks. If the level of risk has been reduced to a negligible level, then the hazard must be tackled as a negligible risk.

It is important that the source of Identified System Hazards (C1) identifies the level of risk posed by a hazard in a way that permits sub-division into the classes of risk defined by C2, C3 and C4.</td></tr>
<tr><td>Applicability</td><td colspan="2">This pattern is applicable in contexts where the ALARP principle is accepted as the device for reasoning about the relative importance of risks and the cost-effectiveness of risk reduction.

In order to apply this pattern it is necessary to have access to the following contextual information:
C1: Identified System Hazards
(See Participants section)
C2, C3, C4: Definition of Intolerable / Tolerable / Negligible Risk
(See Participants section)
These definitions are typically provided by the appropriate regulatory authority, standards or through investigations by safety engineers, including discussions with customers.
C5: Definition of Disproportionate
(See Participants section)</td></tr>
</table>

Consequences	After applying this pattern, there will be a number of undeveloped goals of the form: • **G7: Risk associated with {Hazard X} has been reduced to a tolerable level** • **G9: Risk associated with {Hazard X} has been shown to be negligible** • **G6: {Hazard X} has been eliminated and can no longer occur** • **G10: {Hazard X} is necessarily present in the system** • **G11: Measures have been taken to reduce risk associated with {Hazard X}** • **G12: Further reduction of risk associated with {Hazard X} requires disproportionate expense**
Implementation	Implementation of this pattern involves first instantiating the contexts C1, C2, C3, C4. In the context of the list of hazards referenced by C1, the solutions to goals G2, G3 and G4 can be provided. If no tolerable risks were ever present in the system, then reference to the system hazard log (Sn1) is sufficient to support the claim G2. However, if any intolerable risks have been identified, it is necessary to claim (G5) that these have been resolved through complete elimination of the hazard (G6), or reduction to a tolerable (G7, G8) or negligible (G9) level. For each tolerable risk identified an argument must be constructed (G6, G10, G11, G12) to demonstrate that it has been addressed in accordance with the ALARP principles. Measures taken in risk reduction must be stated in support of G11. Some evidence / argument of the non cost-effectiveness of further risk reduction measures must be supplied in support of G12, in accordance with the definition given by C5. Evidence of risk analysis (probably based upon consideration of probability of occurrence) is required in support of each claim of hazards posing negligible risk (G9). **Possible Pitfalls** • Not providing complete coverage of levels of risk through definitions C2, C3, C4 • Expressing definitions C2, C3, C4 in a way that is difficult to apply to the information provided by C1 (and vice versa) • Not having a commonly agreed concept of when to stop attempting further risk reduction (C1) - this can result in a non-uniform approach to tackling risks where significantly different levels of effort are committed to risks at the same level.
Examples	TBD
Known Uses	See *Industrial Press Safety Argument*
Related Patterns	• Safe by Hazard Mitigation Argument