Keywords: Safety cases, safety case architectures, multi-view safety cases, certification.

Abstract

Due to high levels of complexity in the design and operation of safety-critical systems, the size and complexity of safety cases continues to grow. This presents considerable challenges to the development, review and maintenance of safety cases. The independent review into the Nimrod crash in 2006 pointed out the dangers of poor practices in safety cases. It noted that the UK Health and Safety Executive (HSE) has also found a number of problems with safety case practices in its role as regulator. In the past, the area of software architecture has been plundered to provide techniques that aid safety case construction and presentation. This paper argues that this can continue to bear fruit, and demonstrates how the principles of multi-view architecture can be used to produce multi-view safety cases. Multi-view safety cases have the potential to filter information of interest to stakeholders, thus reducing complexity and increasing comprehension of the safety argument. Modifiability and extensibility could be improved, as changes to the safety case can be reviewed more easily through relevant stakeholders’ views.

1 Introduction

Safety cases offer a means through which system safety can be argued and documented, either through text or through a graphical representation. They have become increasingly the means through which system safety is demonstrated to regulators, particularly in the nuclear, defence and rail sectors. Recently, the requirement for a safety case has been considered in emerging national and international standards. For example, a safety case is required for compliance with the new automotive functional safety standard ISO 26262 [1]. It is also a recommended practice in the US Food and Drug Administration’s draft guidance on the production of infusion pump systems [2]. The move towards safety cases presents a departure from highly prescriptive safety standards in which certification, particularly of software-intensive systems, is often obtained by compliance with predefined objectives and processes.

A safety case contains "a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment” [3]. Graphical notations for the representation of argumentation have achieved some success in improving the documentation and communication of safety arguments. Nevertheless, there are still issues related to the presentation and management of large-scale and complex safety-critical systems, making their safety arguments incomprehensible to many stakeholders interested in the safety of these systems (e.g. the certification authorities, design engineers and operators).

These issues are partly attributed to the fact that most safety cases address a number of stakeholders with the same safety argument. These stakeholders often have different safety concerns regarding the safety of the system (e.g. systems safety, hardware safety, software safety or operational safety). In this paper we investigate how the comprehension of the safety case can be improved when argument structures related to each group of stakeholders are addressed in separate, yet interlinked, safety argument views.

Further, we discuss how changes to the safety case (due to, for example, new data, new regulations, or change of use), could be assessed more rapidly, and with greater confidence, when each change is assessed according to each stakeholder in turn. That is each stakeholder could assess whether any sub-claim within their view has been affected, whether it impacts on the overall safety argument, or whether additional evidence is required to support the altered sub-claims.

Our preliminary investigation shows that multi-view safety cases hold the promise of providing a means for appropriate abstraction for the safety argument and evidence. Safety case stakeholders such as engineers, operational staff, users and quality professionals can view those aspects of the safety case that are of interest to them, with information not of interest removed from their view.

This paper is organised as follows. Section 2 outlines some of the recent challenges to safety cases, and in particular how complex systems can result in incomprehensible arguments. Section 3 refers to a number of software architectural techniques that have been adapted in the past for use with safety cases. Section 4 introduces the concept of multi-view architectures, while section 5 proposes how multi-view safety cases could be created. Section 6 looks at how multi-view safety cases could be implemented in practice, while section 7 looks at how the implementation could be validated.
2 Challenges to Safety Cases

Safety case document size and complexity can be great, and continue to grow [4][5][6]. Ever more complex systems are being built, and these provide continuous challenges to safety. Weaver and Kelly [7] have argued that if the safety argument is not of a manageable size, or grows to a degree that it is impossible to understand, even if factually correct, confidence in the safety argument will be reduced. They note that it will only be possible for an assessor to reach a conclusion on the merits of a safety case if the size of the safety argument is of a manageable size.

Present safety case construction sometimes results in a rambling argument whose structure may not be apparent to the reader [8]. This can make it difficult for stakeholders to ascertain if they have completed their obligations to the safety case.

Nevertheless, certification standards continue to require greater levels of detail. The increasing volume of detailed analysis and supporting evidence pose problems maintaining a high level view of system safety [5].

Further, Wilson et al note that the number of stakeholders involved in safety cases may lead to conflicting requirements, leading to a lack of focus in the safety case, as well as confused reasoning. They argue that there must be clear reasoning to all stakeholders, despite their differing interests. They note that as systems become more complex, and the associated safety cases become larger and more difficult to manage, the issue of how safety case maintenance can be aided becomes more taxing.

The issues of safety case structure and safety case analysis are also cited by Wilson et al as key issues in safety case development. They point to the challenges to the safety case that will appear over a lengthy period of time, including changes to the environment, changes to the regulatory framework, changing requirements and new operational data and experience. They go on to argue that it is difficult to assess how such changes will impact the safety case if there is no effective structuring in place.

Modularising a safety case offers the potential of reducing the size of the safety argument. However, this presents fresh challenges in overcoming issues with boundaries between safety case modules.

Re-use of mature arguments is a method of aiding safety case comprehension. However, there are questions as to whether such an approach is always appropriate, given that it may not always be clear to the assessor if the argument has been fully understood and is fully appropriate for the system being assessed.

The Haddon-Cave review [9] into the loss of an RAF Nimrod aircraft in Afghanistan in 2006 has highlighted serious concerns about existing safety case practice. It has also noted that the HSE found a number of problems in safety cases in its role as a regulator. Some of the problems recorded by the HSE that were noted in the report include:

- Safety case contains assertions rather than reasoned argument
- Unjustified and implicit assumptions
- Some major hazards have not been identified, and consequently never studied
- Ownership of the safety case not always clear.

The Nimrod report itself was critical of safety cases, claiming that the safety case regime had lost its way, with safety being replaced by what it calls ‘paper safety’.

3 The Adoption of Architectural Techniques to Safety Cases

The similarities between what software architecture and safety cases are trying to achieve has been observed by Kelly [10], and many principles from Software Architecture have been successfully adapted to aid the managing of complexity in safety cases. Examples of software architectural principles that have been adapted to use with safety cases are discussed below.

Modularity. Modularity offers a useful way to manage complexity. Indeed, the same mindset is involved in both architectural modularisation and safety case modularisation, and it can be useful for the modularisation of safety cases to follow the same requirements as modular architecture [11]. Further, some studies have shown how a complex safety case can be replaced with a number of modular safety cases which correspond to the modular structure of the underlying architecture [10]. The advantage of this approach is that it improves the evaluation of change (due, for example, to maintenance or modification) and reduces the recertification effort.

Architectural Patterns. The use of patterns had occurred in several domains before becoming established in software architecture. While details are likely to change from one instance to another, there is frequently commonality between safety cases on structures of argument [12]. In particular, this is true of safety cases within the same domain. These structures offer the possibility of re-use, reducing the effort in producing a safety case, and also of capturing knowledge gained on previous projects.

Scenarios. Scenarios are used to test how an architecture responds in terms of its quality attributes (e.g. maintainability, modifiability, performance, reliability and security). Scenarios test architectures by visualizing the demands placed on the architecture by the scenario, typically through the viewpoint of a stakeholder. The use of scenarios could also be used to test safety case architectures [6].

Tactics. The coarse granularity of patterns offers an impediment to precise reasoning of the safety properties [13]. Tactics are associated with patterns, but provide a more fine-
grained approach to architecture. For example, Wu et al have defined a hierarchy of safety tactics justifying the safety rationale of safety goals considered in a safety case [13].

**Architectural Evaluation Methods.** Software architectures are evaluated to assess how well the proposed architecture meets quality goals such as maintainability, availability, scalability and performance. One of the methods currently used for architecture evaluation, the Software Architecture Analysis Method (SAAM) has been used in the safety case domain [6].

**Software Contracts.** Individual component failures pose relatively low risks in safety-critical systems [14][15]. Safety is a system property, and a comprehensive and defensible argument cannot be composed by listing the individual safety arguments of the components [16]. In the setting of a software component contract, if a component continues to meet the contract requirements on its side, then the overall system functionality is expected to be maintained. In a similar way, and in order to promote loose coupling and thereby minimise the impact of changes between interrelated safety case modules, the concept of argument contracts was proposed for safety cases [16]. Argument contracts preserve the overall integrity of the modular safety case when the internal details of one or more argument modules are modified. This is mainly because these contracts are specified using the interfaces of the interrelated argument modules rather than using the internal details of these modules (hence protecting interdependent modules from changes to the internal details of the arguments contained in these modules). Essentially, a safety case contract captures a 'rely-guarantee' relationship between two argument modules.

### 4 Multi-view Architectures

Software architectures serve two main functions. It provides a design plan of a system, and it is an abstraction that facilitates managing the complexity of a system.

Most modern software architectures are complex, and it is not feasible to represent the entire software structure in one document. The information present within the architecture is of interest to a wide range of stakeholders, each of which will have their own particular interests, or concerns. Anything that lies outside these concerns only serves to obscure the concepts that the stakeholder is really interested in. To address this, architectural views were proposed where each view reveals certain information on the architectural implementation, while reducing the amount of information provided on other aspects, or even ignoring some aspects altogether [17].

That is, it is not the quantity of details that becomes a barrier to understanding, but rather the number of details that must be considered at the same time [18]. A software architecture will have a number of stakeholders, and each of these stakeholders will have specific interests in the architecture, or concerns. Multi-view architectures allow for the separation of these concerns, and to allow stakeholders view only those aspects of the architecture which are of interest to them.

A number of multiview architectures exist. Some of these have a fixed number of views, while some permit the architect to produce whatever views are of interest for a particular application [17][19].

### 5 Multi-view Safety Cases

To communicate the safety argument of a complex system within a safety case, the advantages of using a graphical notation are well understood. Because of its widespread adoption within safety-critical sectors, the Goal Structured Notation (GSN) [9][20] has been used as the basis for multi-view safety cases. GSN is a graphical notation for the representation of arguments in terms of basic argumentation elements such as claims, context and evidence. Arguments are created in GSN by linking these elements using two main relationships, ‘supported by’ and ‘in context of’ to form a goal structure. A goal structure represents a recursive decomposition of goals (i.e. claims) typically using GSN strategies, until sub-goals can be supported by direct solutions (i.e. evidence).

One of the benefits of using a graphical notation such as GSN is that the structure can be used as a focus for discussion between different stakeholders who can have different viewpoints on the validity of some of the safety arguments [21]. For example, a system designer may rely on software testing to provide sufficient evidence that a design goal has been met, but a test manager may not have the resources to achieve the level of testing required in the time available. Further, a user may not be prepared to pay the cost required of a particular hardware solution, preferring to meet a safety goal through the adoption of operational procedures instead.

Safety cases represented in GSN can also facilitate discussions on argument alternatives and trade-offs between stakeholders. Safety goals may be delivered by software or hardware, through process or product design, or through operational procedures or product design.

The principle advantage of using multi-view safety cases is that comprehension of the safety argument is facilitated by abstracting those argument elements that are of interest to particular stakeholders. Safety case stakeholders include safety professionals and regulators, architects, hardware engineers, software engineers, testers and validators. Through the use of views, design decisions and trade-offs can be aided.

One of the dangers of a multi-view approach is that a safety case view may only show partial arguments, and the overall argument may be missed. We propose two features to help to guard against this:

- An additional extension will be provided to GSN to provide an indication to the stakeholder that further argumentation is present in the overall argument, but
The argument addresses the justification of certain product elements (hardware and software) as well as the process by which these elements were developed. If a process view of the safety case was to be selected, the extract in Figure 3 would highlight those aspects of the safety case that are of interest to a process stakeholder, where a process view is as depicted in Figure 4.

6 Implementation

The first step in implementing multi-view safety cases is to identify and group the stakeholders. As the intention is to make this approach applicable across a broad spectrum of domains, it was also important that the implementation be capable of addressing additional viewpoints.

In addition to providing for stakeholders, an added value to the implementation would result if it could aid in decision-making by permitting the user to switch between orthogonal views such as hardware versus software, product versus process, cost versus safety, safety argument versus confidence argument.

It is possible for each GSN symbol to be associated with more than one view, so each symbol needs an additional attribute to signify which view it is associated with. We have implemented this attribute as an array of Boolean characters. Each position within the array would be associated with a view, and each Boolean value representing whether the symbol is applicable to a particular view or not.

Because of the divergent manner that a software tool could be expected to operate, complexity both in operation and design could be significantly reduced by restricting the options of the tool user to certain defined operations. We have implemented this restriction through the use of a state machine.
A state transition will be accompanied by a new GSN output together with a menu update indicating the views available for that state.

Thus the implementation of a multi-view safety case in GSN will be comprised of the following elements:

- The original GSN
- An array of Boolean characters attribute associated with each GSN element representing which views are represented.
- A state machine used to guide the user through the building of views. In each state, the user will have the option of selecting certain views, depending on the GSN tool state.

To illustrate the uses of the abovementioned elements, consider a simplified case where a user wishes to view the software confidence argument view. On opening the safety case document, the user will be presented with a list of possible views for selection.

<table>
<thead>
<tr>
<th>INIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SOFTWARE</td>
</tr>
<tr>
<td>2. HARDWARE</td>
</tr>
<tr>
<td>3. PROCESS</td>
</tr>
<tr>
<td>4. OPERATIONAL</td>
</tr>
<tr>
<td>5. VALIDATION</td>
</tr>
</tbody>
</table>

Figure 5: Initial Views Presented to User

Selection of any of these options will highlight the associated argument view to the user (as illustrated the figure 5). The user now has the option of selecting other views in turn, or of generating and presenting the selected view via another user interface control.

This selection will also cause a transition in the state machine, and new view options will then be presented. If the software argument view is selected in this way, the view options will then be as appears in figure 6.

<table>
<thead>
<tr>
<th>SOFTWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SAFETY ARGUMENT</td>
</tr>
<tr>
<td>2. CONFIDENCE ARGUMENT</td>
</tr>
</tbody>
</table>

Figure 6: Available Views on Selecting Software View

Separation of the safety argument and confidence argument has been proposed by Hawkins et al [8], with the intention of giving both arguments greater clarity of purpose by separating them. By using this separation in conjunction with views, each stakeholder can see with greater clarity if there is any degree of what Hawkins et al refer to as assurance deficit within their area of concern.

7 Evaluation

Work is currently underway on evaluating the concept of multi-view safety cases through two different methods:

- A number of safety cases from the public domain have been chosen to test how the method will cope with different aspects of the implementation;
- The results will be reviewed by a panel of peers. A questionnaire will be provided which, \textit{inter alia}, will seek opinions on whether such an approach will aid comprehension of the safety argument.

In each case, stakeholders will be chosen and stakeholder views will be selected through traversing the state machine and visualising the resultant argument in GSN. Test cases will be selected to test different aspects of how multi-view safety cases would operate in practice. In each test case, on opening the safety case the full GSN will appear, with a number of views available for selection.

The test cases will be used to illustrate how each view will filter out information that is not of interest to the relevant stakeholder. In addition, the use of the proposed new GSN symbol will communicate to the viewer that the full safety argumentation is not complete where the symbol is present.

8 Conclusions

One of the recommendations arising out of the Nimrod Review was that safety cases were in future to be made “more focussed, proportionate, and relevant” [6]. By allowing the safety case to address individual stakeholders, multi-view safety cases have the potential to become more focused and relevant.

Multi-view safety cases have the potential to aid comprehension and communication of the safety argument. The increased clarity promised by this method could lead to the deficiencies identified earlier becoming more identifiable, and hopefully will lead to improvements in these areas.

Further, the assessment of changes to the safety case brought about by additional operational data, or by operational changes requiring the system to be used in a new way, can be facilitated through viewing the impact of the changes through the viewpoint of individual stakeholders.

References