Improving the Production and Presentation of Safety Cases through the use of Intranet Technology

Rupert Brown

Logica UK Ltd., Byron House, Business Park 4
Randalls Way, Leatherhead
Surrey, KT22 7TW, England

E-mail : brownrw@logica.com

Abstract

A Safety Case requires that a large number of links are made between requirements, analysis and design documentation, and the safety analysis itself. Working in a paper based environment leads to a documentation set that is large, expensive to produce, complex to follow and difficult to maintain. This paper examines how basic intranet technology may be used to capture links between documents and potentially make easier the development, maintenance and understanding of a Safety Case.

1. Introduction

Safety Cases enable technological progress. They make the seemingly unacceptable, possible.

Thankfully today there is a body of legislation designed to protect the public from structures and systems which may kill or injure. The legislation is policed by a variety of authorities within individual countries. A Safety Case contains the evidence showing the probability that a system will kill or injure people, or destroy property, is acceptably low. If a policing authority accepts the evidence, then it can be said that the Safety Case has enabled the use of the new system within society.

A Safety Case does more than enable. It also directs. The Safety Case shows how safety considerations have driven a system's design to incorporate features which mitigate the risks of hazards occurring. To do this the Safety case has to consider the whole design, including the way the system is used, and its performance in use. This means that the Safety Case will need to refer to any number of requirements, and design and test details described within a great array of technical documentation. This cross-referencing is in part the reason why they tend to be complex. There is a need to find ways of producing and presenting Safety Cases in a more controlled and accessible manner.

Complexity means expense, and to produce a Safety Case takes time, money and expertise. A Safety Case adds cost to a project, and the maintenance of a Safety Case is an overhead. In a safety-related system, the production of a Safety Case is a vital part of the project output, but with budgets becoming ever tighter, there is a need to find ways of producing Safety Cases at less cost.

This paper will examine how the new technology of intranets, HyperText Markup Language (HTML), the World Wide Web (WWW) and web browsers may be used to make the production of Safety Cases more direct, their maintenance simpler with the possibility of some automation, and their presentation more exciting and revealing. In short, this paper hopes to show how these new technologies may help to make a better job of Safety Cases.

2. What is a Safety Case?

A Safety Case may also be known by the names Safety Argument, Safety Assessment Report or Safety Justification. Whatever the name, the purpose and contents are the same.

The Hazards Forum safety-related systems guide[1] states that the Safety Case has two purposes, namely :

1. To justify to others the confidence which designers and intending purchasers have in the safety of the system,
2. To provide evidence that, though an event may occur which was not foreseen or considered when the system was designed,nevertheless all reasonably determinable safety-related concerns were considered and dealt with properly. This may provide an important legal defence.

A Safety Case does this by :

1. Identifying the important issues affecting safety in a given system,
2. Identifying the ways in which those issues have been addressed,
3. Providing the evidence which demonstrates that the issues were addressed adequately.

The 'Safety Lifecycle'[1] is the framework for the management of safety during the development of safety-related systems. The primary concern is the management of risk following the ALARP principle[2]. The process of building a Safety Case begins with the identification of a project's safety requirements. Safety requirements may be stated explicitly, may have to be extracted from some project

requirements document or even derived from scratch, considering the system's intended purpose.

A project's requirements set, including the safety requirements, must be tracked through the lifecycle of the project. Requirements are allocated to the high level design objects and processes which realise those requirements. In turn the objects and processes are refined into software or hardware components which satisfy their allocated requirements. In this way, the project components that satisfy the safety requirements are identified. These components must be developed and controlled in an appropriate manner.

During the project lifecycle, safety techniques are used to direct and assure the project design. The analysis of failure modes is used to identify single points of failure and areas of potential weakness within the design. Reliability, availability and maintainability (RAM) analysis add an extra dimension to the Safety Case.

All this effort is distilled into the end product and a large document set. It is the author's experience that a £5 million project is quite capable of generating over a hundred individual document items, many of which will exist in more than one version. A typical software based, safety-related project has the following document structure - note that the list below is based on the DOD-STD-2167A standard.

1. Project Plans
    a. Project Management Plan (PMP)
    b. Software Quality Programme Plan (SQPP)
    c. Software Development Plan (SDP)
    d. Software Quality Evaluation Plan (SQEP - split out from SDP)
    e. Software Configuration Management Plan (SCMP - split out from SDP)
    f. Formal Qualification Testing (FQT - split out from SDP)
    g. Software Safety Programme Plan (SSPP)
2. System Level documents
    a. Operational Concept Document (OCD)
    b. System/Segment Specification (SSS)
    c. System/Segment Design Document (SSDD)
3. Then for each subsystem:
    a. Software Requirements Specification (SRS)
    b. Software Design Document (SDD)
    c. Software Test Plan (STP)
    d. Software Test Description - procedures (STD)
    e. Software Test Description - cases (STD)
    f. Software Test Results (STR)

4. And there is more!
    a. Software Users Manual (SUM)
    b. Software Programmers Manual (SPM)
    c. Computer Resources Integrated Support Manual (CRISD)
    d. ... etc ... etc ...
5. Then the Safety Officer chips in with:
    a. Overview of System Safety (the way in to the Safety Case)
    b. RAM Analysis (RAM)
    c. Failure Modes, Effects and Criticality Analysis (FMECA)
    d. Fault Trees (FT)
    e. List of Risks (LoR) or Hazard Analysis (HA)
    f. Hazard Analysis Report
    g. V & V Report

Examples of how the safety case documents might cross-reference other project documents are:

1. The Software Safety Programme Plan will require that components which realise safety-related requirements have to be handled in special ways. This handling will involve configuration management, with the procedures described in the Software Configuration Management Plan, and testing with procedures described in the Formal Qualification Testing document.
2. Hardware configurations will be described in the System/Segment Design Document. The proof that these configurations meet RAM requirements will be in the RAM document.
3. Fault trees will be based on hardware and software designs contained in a number of design documents.

It is possible to find many more examples.

It is probably not possible to remove these cross-references, although careful document design may reduce their number. However, intranet technology brings with it the means to better exploit and manage these cross-references.

3. Writing about real safety-related systems can be a problem

To demonstrate how intranet technologies may improve Safety Cases, an example is required. The example has to be based on a safety-related system. It would have been preferable to use a real safety-related system, but this has deliberately not been done. It is the author's own painful experience that it can be difficult to get permission to discuss real safety-related systems in a public forum.

Should a safety-related system fail, and kill or injure, the designers, owners and operators face an investigation to determine liability. If they are found to be negligent in some way, then they may have to pay compensation and fines running into millions of pounds. Also, there is an increasing threat of jail terms to individuals within the designer, owner or operator organisations. It is simply not in their interests to open up their systems to public scrutiny, however well meaning that scrutiny may be.

This is a real problem for the community of engineers who work in the area of safety. Learning from failures is as natural a way of working as using maths. Yet because of concerns about liability, the benefits that may be derived from learning about a failure are limited because of restrictions on information. If we can establish a no blame international arena, free from the threat of prosecution, where failures may be discussed in detail, it would be of benefit to us all. It is possible to find excellent examples of reports into safety-related incidents. The risks forum at `comp.risks` is one well managed source, as is the Safety Critical Systems Club.

This paper uses a fictitious system, a space ship. Because it is fictitious, any aspect of it may be debated at length, and nobody is going to get prosecuted.

4. Intranet Technology

An intranet is a computer network within an organisation that allows members of that organisation to share information. An intranet is typically made up of the following elements:

1. The network itself which will be based on Ethernet or an equivalent technology, and provides communication paths between all its points of access,
2. The points of access which are typically PCs or workstations with the necessary hardware to connect to the network. It is possible to define two types of network user - individuals using personalised machines, and servers which act as data repositories,
3. Server software which handles user requests for data and which is usually associated with the server machine,
4. Access software which allows users to access the data available on a network and which is usually resident on the users machine,
5. The data itself which must be in a format which can be interpreted by the access software.

The first three elements are outside the scope of this paper. Points 4 and 5 are discussed in the following paragraphs.

4.1 Access Software - Web browsers

User access software is essentially a web browser. The two most common examples in use around the world today are Netscape's Navigator[3] and Microsoft's Internet Explorer[4]. Both are simple to acquire and often available for free. For a user that is familiar with a windows environment, they are easy to learn how to use.

A web browser is a window into an intranet. It enables a user to navigate to data, and then display that data. Historically browsers were limited to text, but their increasing sophistication means that complex and colourful graphics can now be included with the text, along with sound and animations.

4.2 Data Format - HTML

The data format of the intranet world is HyperText Markup Language[5]. HTML was developed by Tim Berners-Lee while he worked at CERN. HTML is a series of markers that are inserted in ASCII text. The markers are instructions which are interpreted by browsers. The browsers format the ASCII text following the marker instructions.

There are markers defined for the whole range of typical text processing functions. Text may be specified as being a heading, as being written in bold or italics and blocks of text may be defined as a paragraph. There is a fantastic amount of information on the details of HTML on the Internet[5].

One of the most useful features of HTML is its ability to link text or diagrams in one place to text or diagrams somewhere else. The two pieces of text may be in the same document, in different document files in the same directory, in different directories or even on different machines at the opposite ends of the earth. So long as the browser is given the correct address, the linking should work. This idea is the key to making and managing cross-references.

5. The Vision

The easiest kind of document to read and understand is one that is contiguous and self contained i.e. may be read from start to finish without having to refer out to other documents. As has been discussed above, it is difficult to produce a self contained Safety Case because of their very nature. But HTML linking can make the referencing out to other documents easy and quick, and browsers will also remember the path that has been followed through a document set.

When reviewing a safety case, it is usual to want to follow an audit trail through the analysis to see how the Safety and System Engineers have constructed the

safety argument. With a paper based project documentation set, this would involve sitting at a large desk near the filing system and laboriously searching out references from fault trees and hazard logs, to design and requirements documents. This kind of operation takes the auditor a long time and fills up a lot of desk space.

Imagine being able to sit at one terminal and navigate the entire documentation set without even having to be aware of real underlying file system, or the physical location of files. The job of assessing the Safety Case becomes a straightforward case of understanding the information presented on the screen, rather than a fight with the medium and the project's record indexing system.

6. HTML Linking and Application

The syntax to create a link to a piece of text or a diagram is:

1. start the anchor with <A (include a space after the A)
2. specify the document you're linking to by entering the parameter HREF="filename#identifier" followed by a closing right angle bracket (>)
3. enter the text that will serve as the hypertext link in the current document
4. enter the ending anchor tag: </A> (no space is needed before the end anchor tag)

The browser highlights the text or image serving as the hypertext link with colour and/or underlines to indicate that it is a *hypertext link* (often shortened to *hyperlink* or *link*).

For example, imagine there is a heading Other Design Considerations in the file design.htm. A link to this heading in some part of the safety case document would look like:

```
... and the evidence that the design complies with
requirement is given in section <A
HREF="design.htm#ODC1">
Other Design Considerations</A>
```

To create the anchor point ODC1, in the design.htm file the heading Other Design Considerations would be written as:

```
<A NAME="ODC1"><H2>Other Design Considerations</H2></A>
```

where the <H2> specifies Other Design Considerations as a level 2 heading. Much more information on HTML can be found on the WWW[5]. A good way to learn HTML is to use a browser's File-View Source option to see the HTML behind a good web page.

One example that makes good use of linking is a fault tree whose elements are linked to descriptions, perhaps within a list of risks document. The following is one possible implementation of an HTML fault tree to do with hitting things in space:

1. Collision destroys R.S. Salamander
1.1----IF--There is a space obstacle *Go to risk*
1.2----AND-There is no collision avoidance system *Go to risk*
1.2.1-----IF--The system fails to detect obstacle *Go to risk*
1.2.2-----OR--The manoeuvring system doesn't react *Go to risk*
1.3----AND-Forcing field unable to repel obstacle *Go to risk*
1.4----AND-Hull cannot withstand impact *Go to risk*

This is implemented using the HTML below. The file RSS_RISK.HTM contains the list of risks, with each risk given a unique identifier. The *Go to risk* is intended to show that more information on that fault tree element is available in the list of risks document.

```
<B>
<FONT SIZE=+1>
<TT>1. Collision destroys R.S. Salamander</TT></FONT></B><BR>

<TT>1.1----IF--<A NAME="TREE1_1">There is a space
obstacle</A></TT>
<I><A HREF="RSS_RISK.HTM#RISK1_1">Go to risk</A></I><BR>

<TT>1.2----AND-<A NAME="TREE1_2">There is no collision
avoidance system</A></TT>
<I><A HREF="RSS_RISK.HTM#RISK1_2">Go to risk</A></I><BR>

<TT>1.2.1-----IF--<A NAME="TREE1_2_1">The system fails to
detect obstacle</A></TT>
<I><A HREF="RSS_RISK.HTM#RISK1_2_1">Go to risk</A></I><BR>

<TT>1.2.2-----OR--<A NAME="TREE1_2_2">The manoeuvring system
doesn't react</A></TT>
<I><A HREF="RSS_RISK.HTM#RISK1_2_2">Go to risk</A>
 </I><BR>

<TT>1.3----AND-<A NAME="TREE1_3">Forcing field unable to
repel obstacle</A></TT>
```

```
<I><A HREF="RSS_RISK.HTM#RISK1_3">Go to risk</A></I><BR>

<TT>1.4----AND-<A NAME="TREE1_4">Hull cannot withstand
impact</A></TT>
<I><A HREF="RSS_RISK.HTM#RISK1_4">Go to risk</A></I>
```

7. The Red Shift Salamander II example

The version of this paper published in the symposium proceedings unfortunately does not contain the Red Shift Salamander II example. However, if you do have access to a web browser and the WWW, then the example can be found at `http://www.fam.aust.com/coolblue`. If you are reading this using a browser, you are already there.

The Red Shift Salamander II pages include examples of some of the documents that are found in a project documentation set. These include the Overview of System Safety, the Operational Concept Document and the System/Segment Design Document.

Hopefully the example demonstrates how quick and easy it is to navigate through links between documents, and how HTML and modern web browsers can make the presentation more colourful and exciting. This form of presentation has several advantages over conventional paper based forms:

1. There is no effort involved in following up a reference. The link is made automatically by the web browser. This should mean that more links are followed, and thus checked, resulting in an increase in confidence that the Safety Case is correct and complete,
2. It is more difficult to make idle references. Because links require HTML markers to be inserted in the text of documents, people working on those documents have to be sure about what it is they are referencing. This should reduce the number of references made in the hope that they "sort of satisfy the need of the moment",
3. The colours and graphics should make the documents more interesting than their monochrome paper equivalents, and thus better hold the attention.

8. Working in practise

Documentation can greatly add to the costs on a project. It is obvious, but worth stating again that the most efficient way of working is to:

1. From the start be very clear about what needs to be said and what needs to be recorded, and then to do nothing more or less. This is especially true with Safety Cases where it is easy to "overcook" documentation,
2. Make sure everybody on the project knows which bits they have to write and ensure that no two people think they have to write the same bit. It is not unusual to find ten people writing ten different system design summaries for ten different documents.

In an HTML based system, once a piece has been written it would be stored in a file on a central server somewhere. Everybody in the team would be given the address of the piece, and then they are free to reference it, and put links to it through their documents at will. The ability to link to other pieces of text in this way should:

1. ensure that no duplication of effort occurs,
2. enable people to spot pieces of text that haven't been written but should be.

The corollary to that is if, at the end of a project, there is a piece of text that stands in glorious isolation, with no links to or from any other documents, the question has to be asked whether it was really needed in the first place.

This whole concept depends on a project using HTML as its documentation standard. That in itself may be quite a significant change for a large number of organisations. It implies that documentation is going to be created, used and maintained on-line, rather than in paper versions. That has implications for ways of working, the security of project information and the status of that electronic information both in terms of change control and its validity in the eyes of the courts.

The basic infrastructure requirements are:

1. An organisation has an intranet with the resources to maintain it, including system administration and configuration control.
2. Everybody within a project organisation has access to the intranet via a workstation, PC or network computer (NC) running a web browser.
3. Everybody within a project organisation has the capability to publish documents in HTML format - this can be achieved using a word processing package with an HTML filter, or a dedicated HTML editor.

A project would need to develop standards and protocols for using HTML. Specifically regarding the creation of anchors within pieces of text e.g. every heading or paragraph number is automatically given an anchor point.

In an HTML system diagrams may be a problem. Traditionally, the graphics of intranets are GIF and JPG bitmaps. Bitmaps are not a particularly good format for displaying the sort of complex system and software design diagrams produced by modern CASE and CAD tools. However, the technology of intranets and the WWW is developing at a fantastic rate. Everyday new and more sophisticated tools are coming onto the market which give greater control and flexibility over intranets. The problem of diagrams will be solved by the kind of web browser plug-in that is able to display vector based images, such as CMX produced by Corel[6].

A project would need a reliable tool set. HTML can be written using Windows Notepad which is how this paper was produced. It is foreseeable that a real project, with a large number of inter-related documents, would need more sophisticated tool support. Particularly important from the Safety Case point of view would be the control of changes. There is scope for a series of automatic change control and change notification tools i.e. the Safety Case documents are alerted when design or specification documents are changed. Given that HTML is ASCII and not some proprietary format, the same tools used to control source code could be used to control HTML. Equally important would be the automatic validation of hypertext links. Perhaps change control and link validation are good areas to develop Java[7] Applets [Note 1].

9. Progressing the idea

It is probably asking too much to go straight from a traditional word processor based document production system, to an on-line one based on HTML and web browsers. A better route would be to select some part of the project documentation set and produce that on-line as a pilot study.

Perhaps an appropriate place to start would be project design and coding standards; documents which once written are unlikely to change a great deal, but all the team would need to refer to on a regular basis.

In summary, a good way to consider an HTML safety case is to regard it as a view into a data repository. The HTML links within the safety case create and shape the view. The safety case is but one view required by a typical project - others might be required by the Quality Manager, the Project Manager, or the Design Authority. I hope this paper is a view into the future.

Notes

Note 1

At the 1995 Ada Language UK Ltd. conference in London, Tucker Taft gave an excellent tutorial on the Ada 95 language. During the tutorial he drew similarities between the Java language and Ada 95. Intermetrics[8] have developed an Ada 95 front end for the Java intermediate language interpreter. This allows Internet applets to be written in Ada 95, but work with HotJava browsers.

References

[1] Safety Related Systems - Guidance for engineers from the Hazards Forum, 1 Great George Street, London SW1P 4AA.
[2] Guidelines on Risk Issues, The Engineering Council, February 1993.
[3] Find Netscape at http://www.netscape.com
[4] Find Microsoft at http://www.microsoft.com
[5] Find HTML at http://www.w3.org
[6] Find Corel at http://www.corel.com
[7] Find Java at http://www.javasoft.com
[8] Find Intermetrics at http://www.intermetrics.com