University of Leicester

Doctoral Thesis

Safety Requirement Patterns for High Consequence Arming Systems

Author:
Dan Slipper

Supervisors:
Dr. Alistair A. McEwan and
Dr. Wilson Ifill

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Engineering

in the

Embedded Systems and Communications Group
Department of Engineering

June 2015

Declaration of Authorship

Some of the material presented in this thesis has previously been published in the following papers:

1. Slipper, D., Ifill, W., Hunter, G., Green, R., Johnson, R., & McEwan, A. A. (2012). Towards Tool Support for Design and Safety Analysis of High Consequence Arming Systems Using Matlab. In Enterprise, Business-Process and Information Systems Modeling (pp. 393-405). Springer Berlin Heidelberg.

2. Slipper, D., McEwan, A. A., & Ifill, W. (2013, July). "Modelling and analysing Defence-in-Depth in arming systems". In System Science and Engineering (ICSSE), 2013 International Conference on (pp. 303-308). IEEE.

3. Slipper, D., McEwan, A. A., & Ifill, W. (2013, October). "A framework for specification of arming system safety functions". In System Safety Conference incorporating the Cyber Security Conference 2013, 8th IET International (pp. 1-7). IET.

UNIVERSITY OF LEICESTER

Abstract

College of Science and Engineering
Department of Engineering

Doctor of Engineering

Safety Requirement Patterns for High Consequence Arming Systems

by Dan Slipper

This thesis details research investigating issues with the way in which safety requirements (often termed assertions) are written for the specific application of high consequence arming systems. Existing methods for deriving such requirements focus on the approach through which these systems are designed. Currently this is based upon three main concepts: isolation, incompatibility and inoperability. These are often referred to as the 3I's, and are used in combination with a fourth I of independence. The issue motivating this research is that there is no rigour in the manner in which these are written and no methods exist to ensure completeness of the resultant requirements set.

A systems engineering approach has been adopted to perform this research and considers the needs of stakeholders involved in specification of arming system safety requirements, from these requirements of the project are derived. A solution has been presented in the form of a set of 8 templates which allow repeatable specification of assertions, along with a set of 12 patterns which cover realistic and commonly used relationships between these templates. The template assertions are based upon a state machine format and adopt a novel view of the 3I's where attenuation, incompatibility, state changes and race are used to specify lower level and more detailed requirements than the existing methods.

Application of the new approach to real industry projects showed that it identified assertions which were missed using the current state of the art methods. Through use of modelling it has also been demonstrated that the new approach produces a complete set of assertions which, when implemented correctly, provide protection against detonation in a given environment. This approach is intended for use alongside existing methods to produce a set of requirements which meet all regulatory needs, inclusive of independence, something which this approach does not consider.

Acknowledgements

I would like to thank the sponsors of this research, the Atomic Weapons Establishment and the Engineering and Physical Sciences Research Council. I would also like to thank my supervisors, Wilson Ifill and Alistair McEwan for their time, support, and reviews of the work throughout this process. I would also like to thank the members of the teams I have worked with at AWE during my studies for their help along the way. Finally, my thanks go to everyone who has supported me through the process, namely my wife Rachel, my family, friends, house mates and colleagues. This would not have been possible without all of your encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Systems Engineering

The term *system* is typically used to describe anything which consists of multiple interacting components which work together to achieve a common goal, or provide a capability. If such interaction does not exist between a number of elements, what is described is not a system but rather a set of individual components or elements [1]. The behaviour of a resultant system can be described as emergent; none of the elements alone can achieve the required function or capability, however, the sum of the individual parts is able to. System elements can be mechanical, computers, facilities or humans - each providing functions which contribute the aims of the system. Within this thesis focus will be upon a specific type of system - the high consequence arming system, more detail of which will be discussed in Section 1.5. Many aspects must be considered when systems are designed (or multiple existing elements are integrated together) to ensure that they fulfil the desired capability. This is achieved through the discipline of Systems Engineering (SE), which The International Council on Systems Engineering (INCOSE) [69] defines as:

> ...an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem.

[1] A definition similar to this is presented in the standard ISO15288 [72], the systems and software engineering life cycle processes

Many factors can have an impact upon the realisation of a successful system. SE is used to identify and mitigate as many risks as possible early in the development process. Rather than considering the behaviour of each element of the system in isolation, Systems Engineers take a view of the entire system - considering not only the desired technical behaviour of the system during operations but also, cost and schedule, performance, test, training and support, manufacturing and disposal.

This involves interaction with many stakeholders (people who have an interest in the success or failure of a system), these can be the customers to whom a system will finally be delivered, those enforcing regulations or standards relevant to the system, end users, manufacturers, funding organisations, and anyone providing training, support or maintenance. The needs of stakeholders should all be considered early on through requirements elicitation.

1.2 The V-Model

Many process models for realisation of a system exist, as summarised by Kasser and Hitchins [78] these different models have common stages around identification of needs, realisation of the system and verification. Arguably these are covered by the V-model, which is widely recognised within systems and software engineering. An example of this model is shown in Figure 1.1 (taken from [43]).

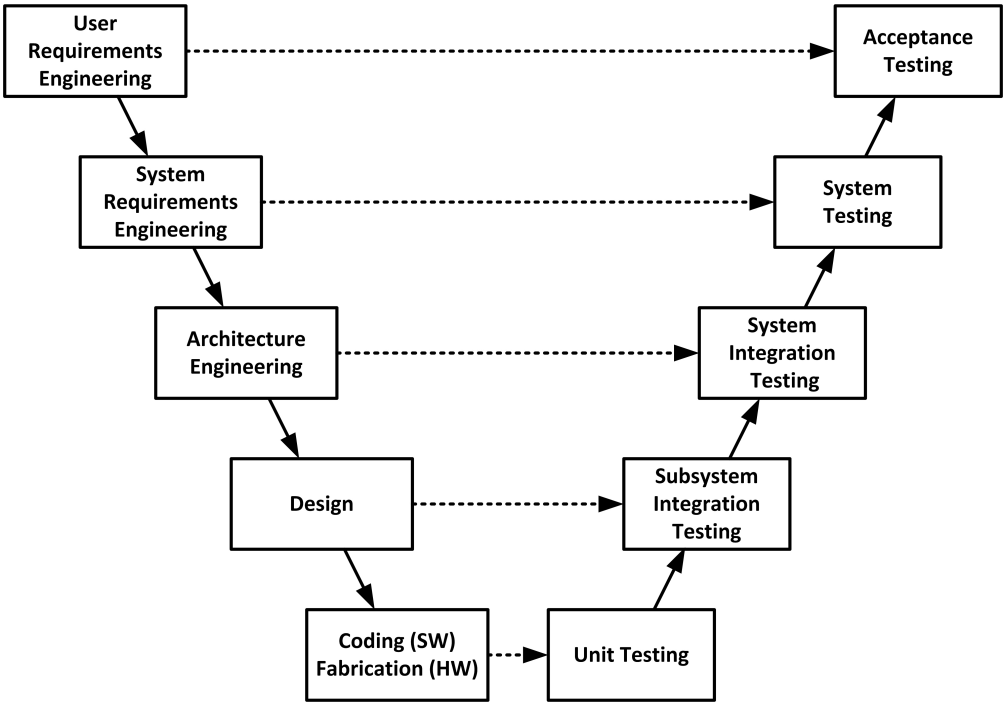


FIGURE 1.1: An example of the stages within the V model.

This linear process forms two sides of a V, the stages along the leading edge can include: gathering stakeholder needs, identifying requirements upon the system, partitioning them to elements of the system through design and results in implementation of the system parts. The second half of the V involves integration of individual sub-systems to form the system, whilst performing testing of each level against its requirements. The process is completed with confirmation that the needs of the stakeholder have been met by the system, through acceptance testing. The term *verification* describes tests used to ensure that the system (or sub-system) has been 'built right', i.e. that it fulfils the requirements upon it. Similarly, the term *validation* refers to confirmation that the system does the 'right thing' to fulfil the stakeholder needs. At the system level this would typically be performed through a confirmation review or by demonstration, however, Bahill and Henderson argue that this can be done at the requirement level [10] by ensuring that each requirement is feasible, implementable, correct and that a requirements set together is consistent. The notion of a consistent and complete set of requirements is pertinent to this thesis, in which the author will demonstrate how use of patterns and templates for writing safety requirements can support a systems engineer in meeting these traits of a good requirements set.

Although the V model in Figure 1.1 shows each step is performed in a one-way and linear fashion, in practice these stages can be performed concurrently and may involve multiple iterations. When applied to individual engineering disciplines (such as safety, reliability, availability etc.), an entire iteration of the model may be performed from that perspective alone. The problems tackled within this thesis are specific to the perspective of safety of high consequence arming systems. Sections 1.3 and 1.5 will provide an overview of system safety as a part of systems engineering and the unique approach taken to design for safety of high consequence arming systems.

1.3 System Safety

When considering the design of safety-critical or high consequence systems, extra functionality (beyond that necessary to meet its functional requirements) may be required in order to achieve adequate safety. In this subsection definitions are provided for safety and both these types of system of concern. Systems engineering is discussed with the perspective of safety in mind, considering verification of safety requirements and how the whole problem through life must be considered.

Safety is freedom from unacceptable risk. Dependent upon the type of system being put into service, risks defined as unacceptable will vary, for example safety-critical systems can be described [46] as:

> "A computer, electronic or electromechanical system whose failure may cause injury or death to human beings"

Similarly high consequence systems can be defined [26] as those:

> "where failures can cause catastrophic results. These results can include loss of life, loss of resources (i.e. money), or even loss of credibility"

Therefore safety-critical systems are, by definition, a subset of the type high-consequence systems, since high consequence systems also concern the risks to human life. Given such consequences of failure, safety must be a driving factor from early in the systems engineering process.

1.4 Safety in the Systems Engineering Process

Failure to consider safety requirements until late in the systems engineering process can result in either: an increased project cost (due to redesign to meet safety requirements), or worse, an unsafe system design could be put into service. Due to this overlap between safety and systems engineering, Fowler and Pierce [45] propose that safety should be considered as a perspective upon Systems Engineering and highlight a statement by Leveson, that "until recently, system safety was always part of the system engineering group. Over time and with ignorance, this interaction has faded." The author of this thesis agrees that this may be the case in some industries, however, argues that for the high consequence arming system industry (discussed in Section 1.5), this relationship has been maintained and remains a driving factor of systems engineering activities. The research presented within this thesis contributes a proposed improvement to this current practice.

System safety life-cycles (such as IEC 61508 [66] and MIL-STD-882E [32]) typically involve hazard identification, analysis and risk assessment (using methods such as Hazard and Operability Study (HAZOP) [70] or Failure Modes and Effects Analysis [31]) which result in requirements upon the system to avoid hazardous events. These requirements are implemented by design of mechanisms to reduce the risk or impact of such hazards. Dependent upon the nature of the system these safety mechanisms can be designed into a physical product or may be part of an enabling system. Protecting against the risks present throughout the entire system life cycle will require consideration of all of the stages, design, production, testing, transportation, operation, maintenance and disposal (decommissioning) - whilst considering the worst-case environments or scenarios that

could be seen in each. For each environment, a number of safety functions can be put in place to mitigate hazards. Different barriers can be used for risk reduction and can in many cases be achieved by adding functional requirements (e.g. to meet the requirements of a safety standard). Additional functions to achieve safety requirements result in an increase in complexity of the system since components, entire subsystems or redundant systems can be introduced into a design.

A number of methods may be applicable to mitigate hazards presented by a system, the hierarchy of hazard controls is shown in Figure 1.2.

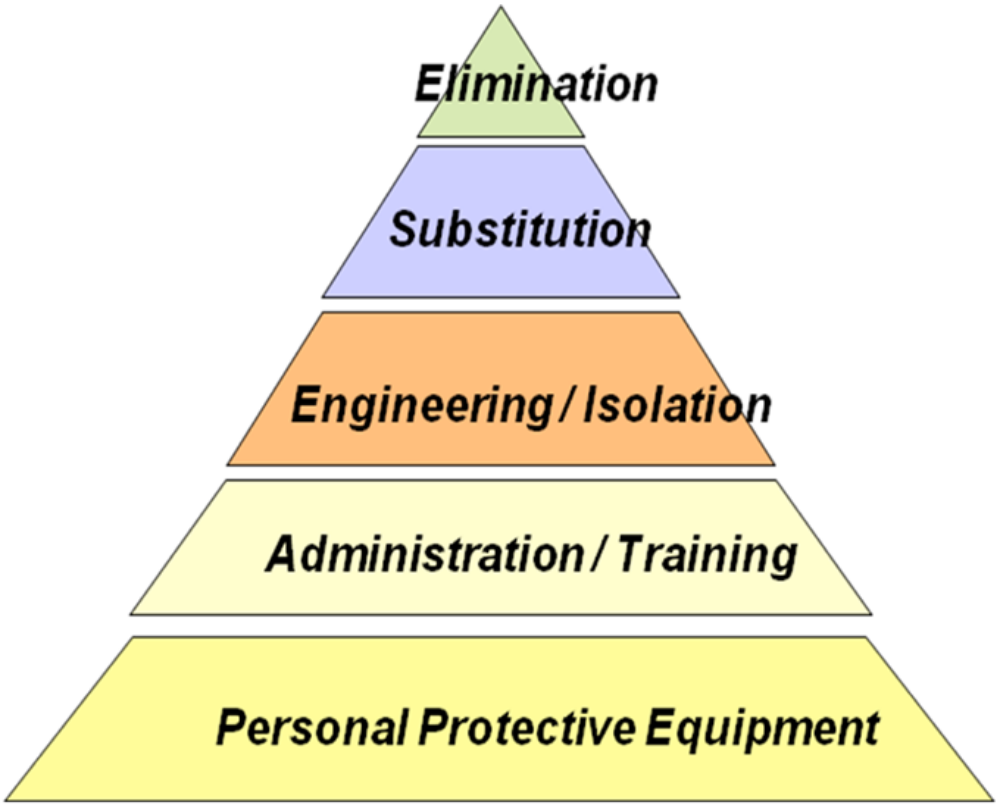


FIGURE 1.2: The Health and Safety Executive (HSE) hierarchy of controls for risk reduction.

Typically the first method for achieving safety is to eliminate any conditions where hazards exist. In many systems it is the desired function which introduces this hazard, for example aircraft are designed to fly, weapons are designed to be fired and nuclear reactors require a chemical reaction in order to produce power. Where the hazards can not be removed they must be mitigated through use of extra features, or by modifying the way in which the required functions are performed. If this can not be achieved through engineering within the system design itself, hazards are then addressed through training, warnings or personal protective equipment. Across different disciplines the methods in place to achieve adequate safety will vary drastically from functional barriers through to warnings to operators. Similarly the terminology can vary, for example the terms *safety*

feature [127], *barrier* [62] and *safety function* [57] all refer to a feature of the system being used to implement risk reduction.

A distinction can be made between two types of engineering/isolation hazard mitigation: the use of *active* or *passive* safety functions. Active safety systems are those which monitor against a given condition and react in order to avoid it, examples could be: cooling systems within a nuclear power plant or object detection and braking of a vehicle. Both of these examples require either an automated or human response (without technology to aid it, object detection and braking would be down to the driver). Reliability of the monitoring and control system becomes a safety critical aspect of the system, however, it is difficult to assure that these systems will necessarily perform upon demand. Conversely, passive safety systems are those which do not require any kind of control loop in place for them to perform their function. The response of such systems will be automatic (given certain conditions) and are defined by first principles (fundamental laws which underpin the behaviour). These can fall into two types, firstly, those which have no-response and will always perform their function in all conditions (e.g. a robust physical barrier). Secondly, there are those functions which have a change of properties, a common example is a capacitor losing the ability to hold charge after exposure to a high temperature (as presented by Kidder in Appendix C, page C-2 of [81] amongst others). Inability to charge the capacitor then results in inability for the system using it to function, subsequently not producing the unsafe conditions which are to be mitigated. Therefore by these definitions passive safety is typically the reverse of active safety[2], necessitating non-functionality of the components used for system operation rather than depending upon the reliability of additional monitoring systems. The use of passive safety functions is a foundation of the design of nuclear weapons, a high consequence system which is the specific topic of interest throughout this thesis.

1.5 High Consequence Arming Systems

1.5.1 Overview

The system of interest within this thesis is a certain part of a nuclear deterrent system. The entire system comprises of a delivery mechanism, (for example gravity bombs or ballistic missiles which includes the warhead element) and any enabling systems (i.e. aircraft or submarines, transport systems, facilities). The scope of the thesis is limited

[2] Bennett and Summers emphasise in [12] that the terms active and passive safety systems may vary in the different industries which use them (i.e. the automotive industry recognise passive safety differently to that of the nuclear weapons industry).

to the interests of the sponsoring company for the research, the Atomic Weapons Establishment (AWE) [8] who are involved in providing and maintaining the warhead aspect of the system. Of particular interest within this thesis is safety of the arming chain, there is not discussion of the nuclear package of the system, however, more detail upon that aspect of the system can be found in [101].

Due to the nature of such systems and the potential consequences of monetary loss, loss of life, environmental impacts and political standing (as emphasised by Spray and Cooper in [129] and also by Fetter and Hippel in [41]) safety becomes a driving factor in their design. So much so that Spray and Cooper state in [130] that:

> "In the U.S. nuclear weapons program, the consequences of an accidental nuclear detonation are so overriding that safety is unequivocally given precedence among competing considerations"

Another concern with these system is that they are developed to go into services for a long period of time, for example Nikolic notes in [99] that weapons systems within the U.S. stockpile have been in service since 1979, the article also discusses the Life Extension Program (LEP) which could keep old designs in service even longer than ever initially anticipated.

1.5.2 Systems Engineering Process

Due to the potential consequences if an incorrect design were put into service, the systems engineering process for high consequence arming systems requires the aspects of safety, security, reliability, quality and use control to be considered during system design. These aspects combined are referred to as system *surety*. Randall presents a high consequence system surety process in [108] which was developed at Sandia National Laboratories (SNL) in the 1990's as part of a revolution in engineering of high consequence systems. Although the process is generalised for all aspects of surety of a high consequence system, it is applicable when considering just the aspect of detonation safety in arming systems, as discussed by Ekman, Werner, Covan and D'Antonio in [37]. Although all aspects of surety are important, D'Antonio, Cooper, Spray, Caldwell and Covan presented the Pentagon /S/ process in [4], an approach developed with emphasis upon the safety-critical features of a system in order to maintain their performance throughout the entire system life-cycle (with particular influence over engineering and production).

As previously described in Section 1.1 the process begins with gathering the requirements of users and other stakeholders. In the case of UK nuclear weapons a major stakeholder

is the regulating body. Due to the consequences of unauthorised use of such systems strict regulations are enforced, detailing requirements upon the system for operation and adequate levels of safety and security. The regulations for the UK nuclear weapon programme are presented in JSP538/372 [96, 97] which cover the entire system life cycle.

In these regulations, adequate safety is defined in terms of the likelihood of detonation, which must be $\leq 10^{-9}$ per weapon lifetime. The likelihood of detonation is composed from contributions from the likelihood of weapon system failure and the likelihood of a particular environment occurring. The necessary contribution from the system can be combined with the probability of the system seeing one of the following three environment types:

Normal - scenarios which are expected to be seen for operation. For these environments there should be no degradation in operational reliability and the likelihood of occurrence is 1.

Specified Abnormal - scenarios which it is feasible that the weapon may be exposed to, at which point operational reliability is no longer expected. Examples are fire, crush and lightning. The likelihood of occurrence is $\leq 10^{-3}$.

Severe Abnormal - scenarios which are more severe and less probable than specified abnormal environments e.g. multiple simultaneous specified abnormal events. The likelihood of occurrence is $\leq 10^{-6}$.

It may be noted that the Ministry of Defence has stated in [63] that "No Trident warhead has experienced either a specified or severe abnormal environment. The Trident nuclear warhead system was designed against robust environmental standards that are now captured in JSP 538. In achieving approval for in-service use for Trident, trials and assessments of components and special build warheads against those standards were undertaken and passed". A number of individual *Lines Of Defence (LOD)* can be used to reduce the likelihood of detonation, which should be independent of each other. In JSP538 [96] it is noted that using LOD analysis is:

> "an approach used to present a structured deterministic argument to demonstrate that sufficient protection is provided. LOD analysis is a qualitative method that critically assesses the effectiveness of control measures in preventing an accident event sequence leading to undesired consequences."

These are used to achieve defence-in-depth, which will be discussed in more detail in Chapter 2. Underpinning each of the lines of defence are a number of *assertions* (a

term commonly used to describe arming system safety requirements), which collectively describe why detonation will not occur given a particular environment. These assertions are made in terms of Isolation, Incompatibility, Inoperability; commonly referred to as the 3I's (or 4I's if an additional assertion of Independence is included). These are merely acknowledged for the current discussion, but are visited in more detail in Chapter 2.

The system surety process has been considered from the perspective of safety in order to achieve the needs of such safety regulations, the design and analysis stages of this process will be considered throughout the next subsection.

1.5.3 Current Approach to System Safety

Ekman, Werner, Covan, and D'Antonio [37] present a unique perspective of the system surety engineering process previously discussed [108], describing each stage of the process with the focus solely on safety. Figure 1.3 (taken from Ekman et al. [37]) presents this system surety engineering process.

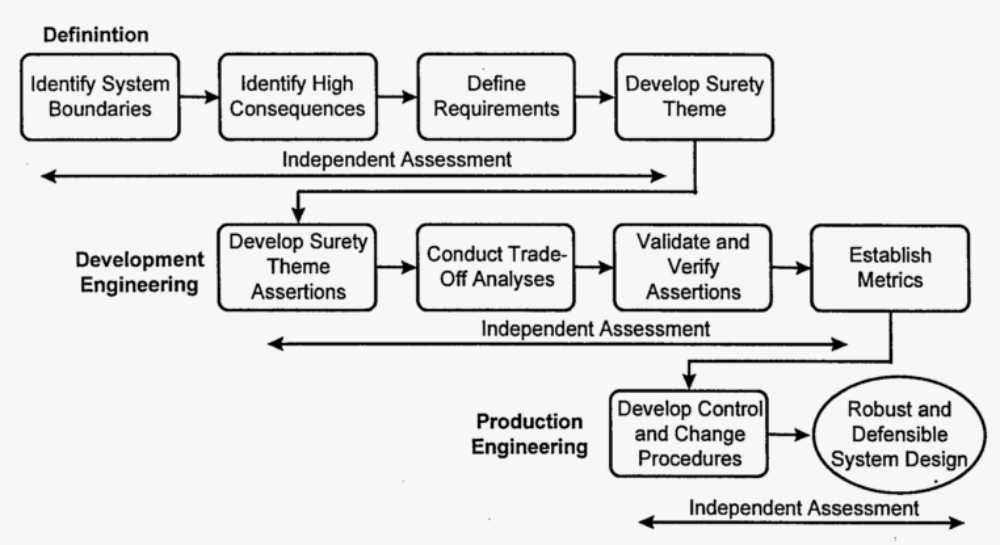


FIGURE 1.3: A process for surety engineering.

Safety of a high consequence arming system is achieved by developing functions into the architecture of the system itself, therefore, safety is an inherent and critical aspect of the *design* of the system. Safety requirements can limit the components which can be used to support passive safety, this can scope down the number of architectural options available for trade-off. When applying systems engineering principles with focus on the perspective of safety much of the design work (and analysis of design options) will be performed early in the process. This is achieved by defining a *safety theme* to fulfil safety requirements, a safety theme is defined by Caldwel and D'Antoniol in [19] as "a blueprint, or plan, of how to incorporate safety into a weapon" or by Dvorack, Jones,

Carlson, Wolcott and Sanders in [35] as "a plan of how safety requirements will be satisfied".

The 'develop surety theme' stage of the surety engineering process is not discussed in detail by Randall, D'Antonio or Ekman [4, 37, 108] and no tools or approaches are defined. Comparatively recent work by Johnson presented in [74] introduces two methodologies for combined design and analysis of a safety theme. The methodologies Johnson presents support the derivation of the safety critical components within an architecture by identifying the minimal cut sets through which energy can flow in order to reach the detonator. Assertions can be made about the behaviour of components such that the complete design meets system level requirements for defence in depth.

1.5.4 Issues with the Current Approach

Once a safety theme has been developed it can be listed as logical groups of assertions, each defined by one of the 3I's. Johnson presents assertions in [74] as both tables of which of the 3I's a component will adhere to and also as a list of inequalities between inputs and outputs of energy flowing around the system (i.e. incompatibility). Similarly, Dvorack et al. present an example safety theme in [35] in which no precise assertions are included and the safety theme is presented as a description of the system.

What is apparent is that there is no direct translation between these two methods of formalising the assertions nor a defined format in which these assertions can be captured. This presents a lack of precision in the definition of these assertions, which may have an impact if they are not written in a form that can be interpreted simply throughout the entire life-cycle (especially considering the potential lifetime of such a system and that assertions must still be met even if changes are made as part of a life extension program).

A second issue with the current approach is that no approach has been defined to verify that all of the assertions together fulfil the safety requirements. Attempts to do so would also suffer due to the lack of precision in defining the assertions.

1.6 Research Topics

The issue highlighted in Section 1.5.4 of lacking precision in the way assertions are written and difficulty in demonstrating that a set of requirements is sufficient are motivating problems for the research discussed in this thesis. These issues are of interest to the sponsoring company for the research and build upon research interests of the research

institution (more detail of which is described in Chapter 3, Section 3.2). The author believes that the issues of interest can be addressed with the application of requirements engineering techniques whilst following a systems engineering approach to the research.

1.6.1 Hypothesis

The contributing part of research detailed in this thesis has been performed in order to test the hypothesis that:

> It is possible to repeatedly produce sets of precise safety assertions about arming systems through use of an approach based upon patterns and requirements engineering techniques.

1.6.2 Contribution

The contribution of this thesis is an approach to the specification of assertions about arming system safety functions allowing repeatable specification to such projects. This is achieved through the following:

- Firstly, a new set of patterns describing the relationships between types of assertions is used to specify a safety theme.
- Secondly, safety assertions identified using these patterns are specified using a new set of template assertions, designed to support the design philosophy of arming system safety functions.
- Thirdly, demonstration that the new approach produces a more complete set of assertions than existing approaches.

1.6.3 Thesis Structure

Further background information of the historical approaches and unique aspects of high consequence arming system design and safety are given in Chapter 2, this describes the state of the art on top of which the contribution is built. The research contributing to this thesis has been performed using a systems engineering approach and follows the structure of the V model (as show in Figure 1.1). Further detail of the research methodology and how this is achieved is given in Chapter 3, along with a brief description of how the research problem was formulated through initial modelling work. Chapter 4 is a literature review of requirements specification techniques and the chapter closes

with selection of appropriate techniques to meet the needs for the project. Chapter 5 describes the detail of the contributions, a set of template assertions and a set of reusable patterns for defining a safety theme. The requirements of the project are verified in Chapter 6. Finally, a discussion of the research work, conclusions and future work are described in Chapter 7.

Chapter 2

History and Background Information

2.1 Introduction

In this chapter, both historical and state of the art theory and techniques to achieve safety of high consequence arming system are presented. Evidence is provided to support the problems introduced in Chapter 1: imprecise specification of safety requirements and repeatable generation of a complete safety theme. Finally, an example safety theme is introduced which is referred to through the thesis.

2.2 Background on High Consequence Arming Systems

High consequence arming systems, namely the firing system within a nuclear weapon, are the system of interest throughout this thesis. Within this section the following are discussed: the function of such systems, example components used to achieve this desired system function, the safety requirements enforced by regulators, incidents that have occurred to motivate such regulations, and finally the methods and philosophy through which adequate system safety is achieved.

2.2.1 System Function and Consequences

High consequence arming systems have a unique conflict between reliability and safety requirements, they must have the ability to function when necessary (except in abnormal events as discussed in Chapter 1, Section 1.5.2). Due to their consequence, they must

not function unless requested. Reliability of such systems is necessary to keep credibility of a countries defences, therefore the system must be able to perform its functions when called upon. This requirement of operation is needed to maintain capability of a deterrent system.

Operation of a high consequence arming system consists of a number of stages, this sequence of operation is discussed in general terms within the literature review. No specific system type will be discussed within this thesis. The typical arming sequence, as shown in Figure 2.1 (which is adapted from Hansen [56]), requires: a human interface or authorisation before the sequence begins, electronic logic which will trigger the arming sequence, an energy source (which may well be inside the system itself), this energy will then trigger the detonator which cause an explosion and then yield. Within this thesis, scope will only be concerned with the section between the energy source and the detonator (as shown within the red dotted line).

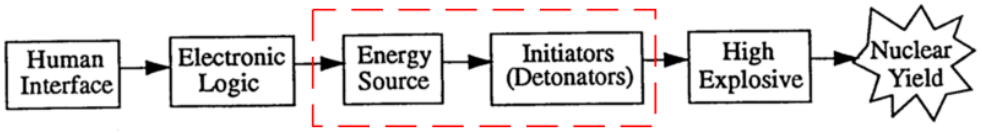


FIGURE 2.1: The sequence of events required for detonation.

Hansen also shows variations of Figure 2.1 within [56], where safety devices have been included at different points of the arming chain: either between the energy source and detonators, or alternatively prior to yield. The former is of interest within this thesis. The stage of the sequence where safety devices are included is often referred to as *safety and arming*. Fowler discusses these features in [47], stating that such elements are in place to stop the system entering the armed state during shipping, handling and storage. This requires that the entire life cycle of the system is considered when incorporating safety features into a design.

Another factor which impacts such concerns is the lifetime of these systems. There are high costs and difficulty in design and manufacture of such systems, followed by the costs of transportation, enabling systems, operation, maintenance and finally disposal. Due to these costs, the process is not often repeated and weapons may be required to remain in the stockpile for a long period of time (Medalia and Nikolic note in [93, 99] that this could be a number of decades). Whilst they are in service ageing of components within the system must be considered, in terms of both system safety and reliability. Miller, Brown and Alonso report in [95] that testing of an aged B43 weapon showed that it only produced half its expected yield, demonstrating that aging reduces the reliability of the system to operate to its full potential. Concern within this thesis is primarily upon the

behaviour of safety devices over the lifetime of the weapon (considering their ability to provide protection in worst case conditions).

2.2.2 High Consequence Arming System Accidents

Early designs of high consequence arming systems were not subject to such strict regulations upon safety as those in service today. Safety features were included within early weapon designs, achieved by the safety and arming portions of the system discussed in previous subsection 2.2.1. Elliott explains in [38] that for early designs the radioactive element of the system could be inserted prior to use and therefore was usually not in place. In this configuration the nuclear package was known as *seperable*.

Unfortunately these early designs were not as predictable in off-normal environments as designers had anticipated. Plummer and Greenwood discuss four particular accidents in [104]: Palomares, Thule and Goldsboro in the 1960's, and a latter incident in Damascus was in 1980. Other incidents of lesser impact are presented by the Department of Defence in [30], these are documented from as early as 1950. The incidents in Palomares and Thule were the more significant of those published, with them both resulting in radiological dispersal. Medalia reported in [93] that there has never been an unauthorised detonation of a US nuclear device.

Safety of the arming system in isolation could be argued and some of the scenarios which occurred were purely due to the delivery mechanism, which for old US systems was use of gravity bombs dropped from aircraft. Such hazards presented by the delivery mechanism needed to be considered early in life and mitigated against.

Lessons learnt from the incidents caused a revolution in the safety of high consequence arming systems. Although safety had always been a concern, the safety features incorporated into the system either overlooked the potential environments that could be seen or relied too heavily upon the unlikely nature of abnormal environments.

2.2.3 Safety of High Consequence Arming Systems

2.2.3.1 Assured Safety

The approaches to assessing safety as previously described in Chapter 1 Section 1.5.3 fall into the categories of either: probabilistic assessment or deterministic assessment. Both of these methods rely upon the concept of *assured safety*, designing the system such that its behaviour will be achieved in a predictable way. Assured safety is achieved through use of passive safety functions and Enhanced Nuclear Detonation Safety (ENDS).

2.2.3.2 Enhanced Nuclear Detonation Safety (ENDS)

ENDS was implemented as a means to achieve a predictably safe response from systems when exposed to abnormal environments. Much of the development of the theory behind ENDS occurred at the same time as a shift in safety methodology in the Chemical Process Industry. Ekman et al. [37] present three typical methods of reducing risk of the hazards of a system:

Protective to reduce or eliminate the hazard.

Preventative reducing the likelihood of initiating events which will cause the hazard.

Mitigative to minimise effects in the event that consequence occurs.

ENDS introduces multiple protective and mitigative measures into the arming chain used to stop any chain of events resulting in detonation of the system. It may be possible to use preventative measures to reduce risk in the wider system (i.e. human operations of facilities).

To provide protection and mitigation within the arming system, it must be designed with devices to achieve such in place. Arguments about why the system will behave in such a predictable manner are based upon a *first principles* approach, which Dvorack et al. define in [35] as one which:

> "makes use of the fundamental characteristics inherent in the physics and/or chemistry of a material in order to provide a predictable response of a component when subjected to specific environmental stimuli".

Predictable behaviour is achieved through use of one of the three I's:

Isolation - a vulnerable component or undesired condition of the system is protected from initiating events via a robust energy barrier. The barrier would be expected to withstand between normal and specified abnormal environments.

Incompatibility - if energy were able to reach the vulnerable component within a system (i.e. the detonators), ideally such energy would be incompatible with the vulnerable component. This concept is readily extended to information. It will be shown that the removal of safety breaks to achieve detonation when required is unlikely by information sources that can reach the safety devices, such as strong links, unless it is intended (as will be discussed in Section 2.2.3.3).

Inoperability - for components of the system which provide a function, removal of the ability to perform this function in abnormal environments can aid the assurance of safety.

The principles used for ENDS have been presented here in an abstract way, achieving this behaviour can be performed through use a multitude of different component options. Many of these common components used for implementation are discussed in the following subsection.

2.2.3.3 Safety Devices

Components used to achieve such behaviour are specifically designed to meet the needs of high consequence arming systems, commercial off the shelf components are unlikely to behave in a predictable manner upon which safety can be assured with sufficient margin, since they will be developed for a range of uses (potentially across different industries). Behaviour of the devices described in this section are expected to be underpinned by at least one of the 3I's as previously presented.

The first common component used to control flow of energy between areas of the system architecture are Exclusion Region Barriers (ERBs). These are robust barriers which are used to electrically isolate (or divert) external energy from outside of an area termed an Exclusion Region (ER). In abnormal environments, any components within the same ER are deemed capable of coupling electrically, regardless of whether a functional relationship exists between the components (i.e. there is no design mode connection but in accidents this is possible). Multiple ERBs are often used to partition the system into separate regions.

Since energy must be able to travel freely between regions during the operation sequence (in order for detonation to occur) portals into such regions are controlled through components referred to as Strong Links (SLs). The challenges in implementing such a component are twofold. Firstly, it must be designed to be robust such that in this specified abnormal environments it should isolate external energy (with the exception of in extremely high temperatures in which case the detonation system is design to become irreversibly inoperable removing the need for ERB protection when it is predicted to fail). Secondly, the SL must be able to remove this isolation in order to allow energy to pass through in the event of authorised operation. A stimulus must therefore exist which will remove the isolating element of the strong link (which can be simply seen as a robust, controlled switch). This stimulus must be unique and not naturally generated in any abnormal environment. Hence, stimulus generated in normal and specified abnormal environments will be incompatible with the stimulus required to command the

removal of the isolating element. Much work has been performed upon analysis of such signals to determine patterns which are unique (such that they cannot be inadvertently generated by natural phenomena). In [24, 128] Cooper and Spray discuss the theory which underpins this argument of incompatibility between the environment and such UniQue Signals (UQS).

SLs can be implemented through various physical means, some of which are listed in a table in [103]. Two parts of the SL of interest are the *energy control barrier* and the *discriminator*. The energy control barrier is the intended path of energy through the strong link component. The discriminator receives serial input of the individual events which make up a UQS (usually 24 events), each event has an impact upon a mechanism within the device. Upon receipt of the final event the energy control barrier is opened. The discriminator can also behave as a locking mechanism if an incorrect UQS were entered. SL design must ensure that assured safety is maintained until the final event is received. This principle is termed"no progress towards arming". However, throughout the thesis assured safety of a SL will always refer to the situation prior to receipt of *any events* within the UQS.

Failure modes of a SL also need to be considered, for example, in extreme temperatures safety can no longer be assured. In this situation a second component, referred to as a Weak Link (WL) is used. As the name suggests, a WL is one which is designed to break or become inoperable given a certain stimulus. Once this has occurred the component should no longer provide its function (which must be part of the arming chain). An important aspect of the relationship between the SL and WL is that there is a thermal race between the two components. To assure the safe behaviour of the two together, a WL must be designed to fail before the SL does. An example is presented in [29] where the WL is assured up to 450 °F and the SL is required to maintain isolation up to 1100 °F. Scenarios where multiple SL/WL pairs are used within the same system must also be considered (analysis of the risk of failure of these race condition are discussed by Helton et al in [58–60]).

Another device used to support the 3I's design principles is the Lightning Arrestor Connector (LAC). Traeger and Ehrman explain in [134] that a LAC is designed as an interface between the electrical aspects of the arming system and the control and guidance of a missile. Electrical energy will need to pass between these regions during operation, however, hazards such as lightning are a risk. LACs are designed in such a manner that a connection between regions is possible up to a certain threshold of energy level, above the threshold the physical properties of the component will divert harmful energy (however sacrificing the ability to function in the future). A final concern is the design of the detonators and explosive element of the system itself. Conventional

explosives are susceptible to impacts or fires, use of which means the system could not be assured to be safe in abnormal environments. Insensitive High Explosives (IHE) are designed not to detonate in such environments. Elliot discusses in [38] how precise timing of the explosive alongside the activation of other elements of the system limit the likelihood of a detonation which will result in yield. IHE is used to avoid the scenario where detonation occurs prematurely and results in dispersal of radioactive material.

A number of safety device types have been discussed, without specific implementation detail of how such devices are designed. Typically they are implemented through electromechanical systems, due to the predictability of these systems in abnormal environments, where the behaviour of electronic and software systems cannot be so accurately determined (according to Caldwell and D'Antonio in [19] and D'Antonio et al [5]). This is not to say that electronics will not play a part of the system. Such components or subsystems can be used, however, system safety will simply not be dependent upon those particular components of the system and they will be deemed as a potential hazard.

The method through which a safety device is implemented is not of particular concern within this thesis, however, adequacy of such components must be verified to meet the appropriate safety assertions upon them. In isolation, none of the device listed here would be capable of meeting system safety requirements, therefore, groups of devices are required to provide protection for all possible scenarios in which detonation could occur.

2.2.3.4 Safety Subsystems and Defence-in-Depth

When components within a system are combined into a grouping which contribute to a common function they are referred to as a subsystem. This is true for the features of a system which contribute to safety. *Safety subsystems* are a logical grouping of components with associated assertions about system safety which contribute to prevention of an undesired consequence. Like lines of defence safety subsystems are based conceptually around a particular I (of the 3Is) and they are as far as practicable independent of other safety subsystems.

The regulations set out in JSP538 [96] require the likelihood of unintended detonation per life cycle phase to be $\leq 10^{-8}$. If the design has one single safety system which meets this system level requirement, failure of this one system is a single point of failure. Events which would cause such a failure should be very unlikely, however, it is difficult

to probabilistically assure that it will not occur. For example 'black swan' events, as described by Taleb and Donnell [34, 131]. [1]

Proving a probability of occurrence of $\leq 10^{-8}$ is difficult, therefore, the safety analysts use the idiom of "don't put all of your eggs in one basket". This design philosophy requires use of a number of safety subsystems, each of which has a lower probability of failure (e.g. $\leq 10^{-3}$ which can be determined with higher confidence than $\leq 10^{-9}$). Given failure of one safety subsystem there should still be others which provide confidence of system safety. By grouping safety assertions through multiple safety subsystems they can be combined to address a particular environment or threat and each individually can be assured with a lower probability. Characteristics of a safety subsystem according to Jones [76], are that each is:

- composed of individual elements which work together to prevent an undesired catastrophic event
- independent from other safety subsystems. This is twofold, in that:
  - The elements making up one safety subsystem should not be part of a second safety subsystem
  - Failure of one safety subsystem does not have a domino effect which would cause another to fail
- able to remain in a safe state in normal and abnormal environments, with the exception of operation
- has a realistic level of assurance associated with it

Safety should not be overly dependent upon any one safety subsystem of the number that are defined according to Jones in [75], who also in [76] discusses difficulties in balancing the number of safety subsystems against the probability of failure of each. If a low number of high probability safety subsystems is used, each is harder to verify. On the other end of this spectrum many safety subsystems with a low probability are easier to prove, however, coupling between these subsystems may make it difficult to prove that they are independent.

The UK regulators refer to the term *lines of defence* in JSP538 [96] as opposed to safety subsystems. These are determined *including* any protection designed into the warhead itself. These can be "appropriately designed facilities, storage and transport containers and fragment barriers in magazines, storage areas and assembly facilities" which are

[1] A black swan was a term used to describe something that is considered practically impossible, since they were not believed to exist - that was until their discovery in 1697.

used to provide protection where hazards occur. It should also be noted that in total three LOD are required, which should function independently and ideally be passive, physical or engineering LOD. Additionally, the number built into the warhead should be maximised (hence a number of safety subsystems are used to achieve this).

A similar concept is also used within the nuclear power industry, where *defence in depth* is used "to ensure that a single failure, whether equipment failure or human failure, at one level of defence, and even combinations of failures at more than one level of defence, would not propagate to jeopardize defence in depth at subsequent levels. The independence of different levels of defence is a key element in meeting this objective" [71]. The key difference between weapon safety features and nuclear power safety features is that the former is avoiding operation occurring, whilst the latter requires reliable functioning of cooling systems. In Benjamin's [11] view, nuclear weapon safety is about avoiding unwanted electrical pathways, where for nuclear reactors, focus is upon conditions which could interrupt existing pathways for fluid flow.

2.2.3.5 Safety Themes

Many definitions of a *safety theme* exist throughout literature, the author of this thesis favours that used by Caldwell and D'Antonio in [19], who state that:

> "The safety theme is a blueprint, or plan, of how to incorporate safety into a weapon"

In the opinion of the author this captures the essence of a safety theme in a simple way. Defining that it is a plan of how safety will be achieved, without describing in detail the actual methods of implementation. Caldwell's definition then continues to describe the relationship between the safety theme and safety subsystems. Figure 2.2 depicts a Domain Model using the Unified Modelling Language [115] which represents the author's perspective of the relationships between a safety theme, safety subsystems and assertions. Within this figure, diamonds marked upon relationships represent a composition - each of which are marked with multiplicities.

In the nuclear weapons industry a safety theme is the first step of developing a *safety case*, which according to Kelly [79] "should communicate a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context". Safety cases can have many forms through life, starting with preliminary safety cases [80] (most like a safety theme) and then developing that into interim and operational safety cases. These later evolutions are used to capture evidence of how each safety argument is

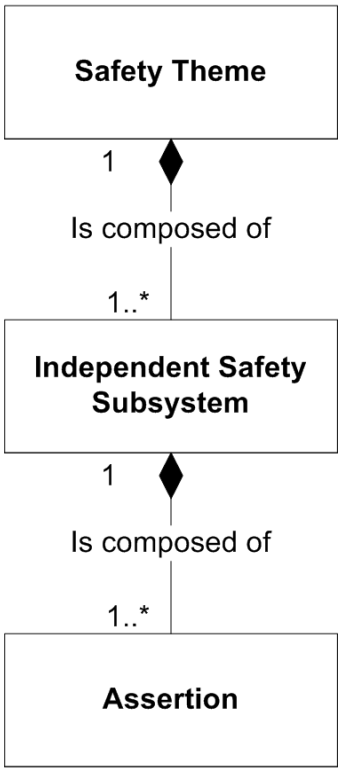


FIGURE 2.2: Relationships between a safety theme, safety subsystems and assertions.

achieved in an implementation and how this meets system level safety requirements and objectives. The safety theme concentrates on the design, whereas the case will justify a life cycle operation for a particular design configuration, so there may be many cases through out the weapon system's life.

Best practices used in safety case development (some of which are listed by Maguire [89] to be HAZOP, Fault Tree Analysis, Event Tree Analysis, Zonal Analysis and Failure Modes and Effects Analysis) are not directly applicable to development of a safety theme due to their unique nature, however, safety cases are used in certifying safety of facilities and operations (e.g. a safety case for the transportation of a nuclear weapon is presented in [100]). In the next section the current best practice approaches for safety theme development and an example safety theme are presented.

2.3 Developing a Safety Theme

Safety themes are not usually created from scratch, many years of design using ENDS have resulted in preferential architectural layouts for the system. Many of which will be compared by performing a trade study (using systems engineering tools such as Pugh's

decision matrix [106] [2]or Saaty's Analytical Hierarchy Process [116]). This approach allows systems engineers to rank potential solution options as part of the "Architecture engineering"' stage of the V-model (as described in Figure 1.1). A safety theme will ideally be defined from early in the systems engineering process. As noted, many architectural design options will be defined for trade and therefore a basic method for comparison of safety of these options is necessary. As Figure 2.2 described, a safety theme is a composition of multiple safety subsystems, each of which will be expected to withstand certain environments. These safety subsystems are a further composition of many assertions of the behaviour of elements of the system, underpinned by the 3I's principles.

In this section an example safety theme is presented, followed by discussion of the approach through which it was designed and analysed. Attention is then applied to the way in which safety assertions are specified. As discussed in the introduction, this has been deemed as imprecise and is a motivating factor for the work presented in Chapter 5.

2.3.1 An Example Safety Theme

The example safety theme discussed throughout this thesis was originally presented by Johnson in [74]. This represents the most detailed published example of a safety theme where both the assertions and individual safety subsystems have been discussed. Much detail about the components used for multiple system architectures were presented by Hansen in [56]. In comparison to Johnson's example, Hansen's safety themes are lacking the requirements upon safety devices.

2.3.1.1 Function

Figure 2.3 shows the three components necessary for the reliable operation of the system: a power source (in this case described as Low Voltage (LV) energy, a Firing Unit (FU) and finally the Detonator (DET). Components have been depicted in a similar manner to SysML internal block diagrams [48], i.e. as boxes with ports for input or outputs where appropriate (these should be assumed to be with the inputs on the left, outputs on the right unless direction is shown). This format is used throughout the thesis. The intended operation of such components would be that LV is stepped up by the FU, resulting in energy which is compatible with the DET (for simplicity in this case it is assumed to be High Voltage (HV) energy). For this to remain an abstract example

[2]an example use of which is shown in Chapter 4, Table 4.1

the numerical ranges which define LV and HV are not of concern, however, Johnson exemplifies an assured DET 'no-fire Voltage' of around 500V.

FIGURE 2.3: Components required to perform the function of an arming system.

2.3.1.2 System Topology

Using a number of the safety devices discussed in Section 2.2.3.3, Johnson described the layout of a system as an abstract topology. Exact three-dimensional detail has not been presented, however, the different ERs and components which interface between the regions have been included. Figure 2.4 depicts the topology of the entire arming system, also considering the threats that would exist in the outside world (e.g. HV energy). For this simple example only electrical threats are considered. Other threats such as high temperature and shock are not discussed.

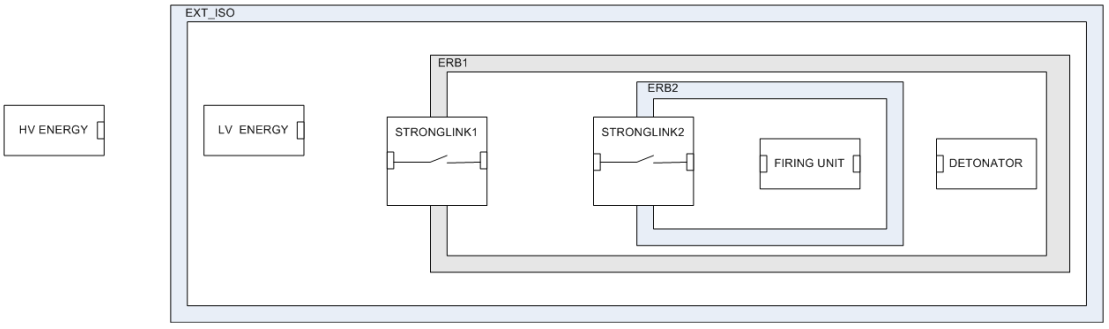


FIGURE 2.4: The layout of elements of a simple arming system, including safety devices.

2.3.2 Design Approach

The system consists of the typical components required to provide reliable functionality when necessary and incorporates a number more components to provide safety. Firstly, the DET and FU are enclosed within an ERB (ERB1 in Figure 2.4), SL1 is necessary to allow initiating energy into the region. These components provide isolation from the threats of LV and HV energy, however the two components together only provide one layer to protect energy from reaching the detonator, as shown in Figure 2.5. These features together provide one complete line of defence and assertions upon these components will make up one safety subsystem.

To achieve the requirements of multiple safety subsystems, a second set of independent arguments must be provided. Since the safety achieved by ERB1 and SL1 cannot be

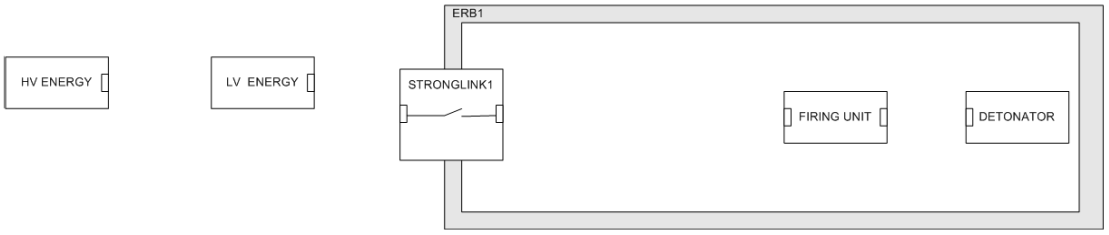


FIGURE 2.5: Safety subsystem 1 of the example system.

depended upon by this second safety subsystem (to ensure independence) there are still potential routes through which LV and HV energy can reach the DET, requiring more features or attributes in place to assure safety.

Firstly, because safety subsystem 2 must be argued completely independently of safety subsystem 1 the LV energy exists within the same region as the DET (ignoring the isolation features of safety sub-system1 as seen in Figure 2.6). Incompatibility between the DET and the LV energy source must be argued. Other compatible energy sources (except lightning) exist in the wider environment such as test equipment. These threats are managed by procedures and facilities so for example procedures ensure compatible test equipment is never connected into the system of interest or that the facility protects against HV energy sources coupling into the system of interest. Johnson notes that this is achieved through 'external isolation' (EXT_ISO), however, no implementation method is defined. Since the FU is within the same region as the DET and LV energy which can be made compatible, isolation is asserted through the addition of ERB2 (and SL2 to allow operation). These will isolate energy to a level such that any residual energy is incompatible with the FU.

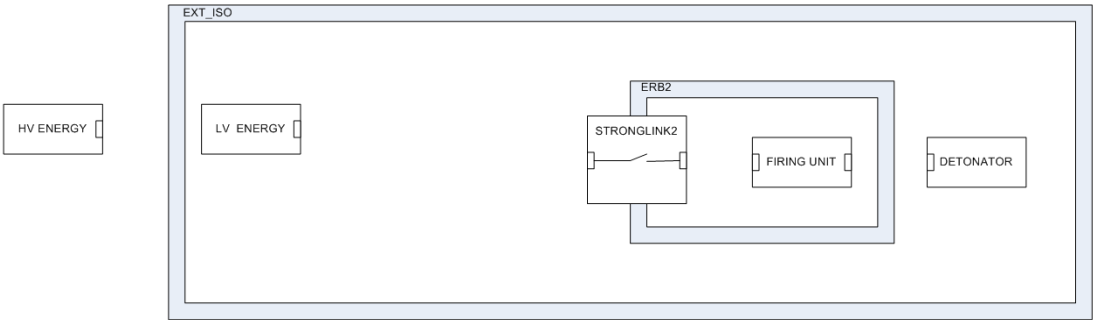


FIGURE 2.6: Safety subsystem 2 of the example system.

In cases where more threats or conditions exist (e.g. the threat of high temperature in abnormal environments), each must be completely protected by each safety subsystem. These environments have not been presented by Johnson in the example safety theme, however, are considered briefly in the architectures presented by Ekman et al. [37] and Li, Li, Suo and Xiao [87].

Johnson's approach to deriving assertions is to define the system layout (in Figure 2.4) as a network, as show in Figure 2.7. For each safety device used along these paths one of the 3I's are used to specify why energy cannot flow along that path. As previously noted ERB1 and SL1 are part of safety subsystem 1, ERB2, SL2, EXT ISO and the DET are part of safety subsystem 2.

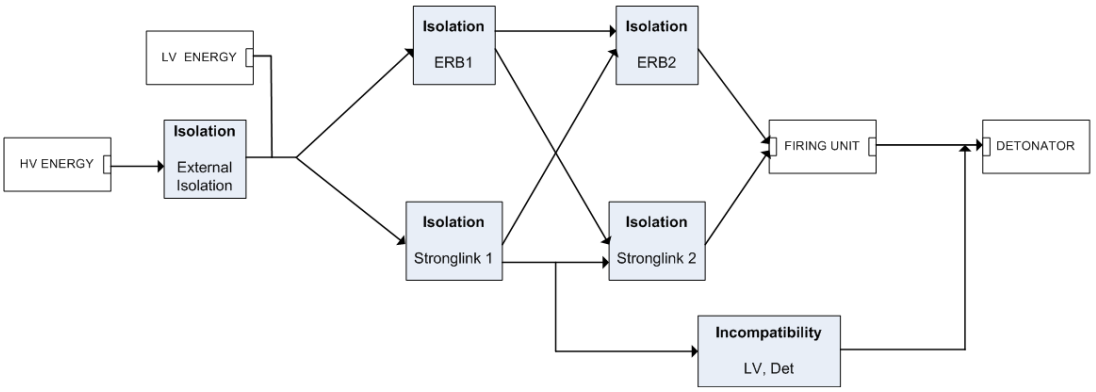


FIGURE 2.7: Logical layout and brief description of assertions in the example safety theme.

2.3.3 Writing Assertions

Johnson summarises the assertions upon components in a number of forms throughout [74]. At the highest level of abstraction, assertions can be summarised as the following:

1. SB1 [3] provides isolation
2. ERB1 provides isolation
3. SB2 provides isolation
4. ERB2 provides isolation
5. Det is incompatible with LV energy
6. HV is externally isolated

The assertions shown here are referred to throughout this thesis as top-level assertions. Like any requirements decomposition (e.g. exemplified by Hull, Jackson and Dick in [64, p. 47]) a top-level assertion can be achieved through a combination of related sub-assertions (referred to as lower level assertions throughout the thesis). Johnson makes reference to lower level assertions in [74], however, there are a few subtle issues in the way these assertions are written.

[3] Johnson refers to a Safety Break (SB), which is equivalent to the term Strong Link (SL) as adopted within this thesis

Figure 2.8 shows a decomposition of the top-level assertion 'SB1 provides isolation'. What becomes clear in these assertions is that the different states of the SB component are not defined and that the assertion is not specific in defining which components it interfaces (e.g. there is no definition of 'nearby electronics').

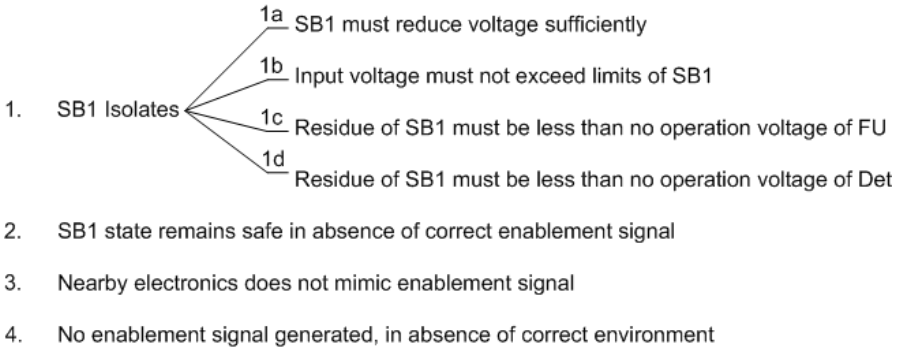


FIGURE 2.8: A decomposition of the assertion 'SB1 provides isolation'.

These issues are considered in the remainder of this thesis where the precision of assertions is addressed in order to generate a complete and precise safety theme.

2.4 Conclusions

In this chapter a number of background topics have been covered including: the purpose, safety regulations and theory behind development of an arming system, along with the current approaches to design and analysis of a safety theme including an example architecture and assertions. This background will be used throughout the remaining chapters of the thesis which explain the research approach, available options for the new approach, implementation of the new approach and finally demonstration, verification and validation of the requirements upon the project.

Chapter 3

Research Method and Approach

3.1 Introduction

This chapter describes the approach taken to the research within the thesis whilst highlighting the main decisions which steered the work throughout the project. Reference is made to the research methodologies used and the reasoning behind each selection is presented. The thesis is structured to describe a systems approach to meeting needs of the sponsoring company of the research. This chapter will introduce an initial study which steered the research work, provide an overview of the V-model followed through the research and present the outcomes from the initial stages of the V-model.

3.2 Initial Research

At the start of the research with the sponsoring company a research area was identified through semi structured interviews with members of the safety design and analysis group of the company. Through these discussion prior art was identified, tool support had been developed to support Johnson's analysis approach to determine potential paths through which energy could flow in a system (which could lead to detonation). This work has since been published as part of this research work, as seen in Appendix B, Paper 1 B.1. The paper describes the tool which had been developed in Matlab. This approach was beneficial to provide an automated approach model behaviour, however, it suffers from a number of issues:

- safety subsystems are not considered;

- failure to meet assertions and independence is not identified (i.e. design faults, ageing or abnormal environments);
- the individual assertions upon components are not traceable to the model;

These issues and common research interests between both the sponsoring company and the Department of Engineering within the University of Leicester (e.g. Ifill and Evans verifying hardware with formal methods [40], McEwan specifying and verifying control algorithms in [92] and prior research under this EngD by Slipper and McEwan to investigate re-engineering from formal models [122]) motivated development of an approach to model and verify properties about safety subsystems and their independence using model checking techniques. The process algebra Communicating Sequential Processes (CSP) [61] was selected for this due to the prior knowledge within the research group and availability of the model checking tool FDR2 [44]. An overview of how modelling and model checking are performed using CSP is presented in Appendix A. The research performed to apply this modelling and verification approach to arming systems is described detail in Appendix B, Paper B.2.

An outcome of this preliminary research was that issues were identified with the precision of how assertions were expressed as part of a safety theme. A case study to model a full scale industry project demonstrated difficulty in interpreting assertions in order to correctly verify that they have been met. This was due to there being no repeatability or precision in the way the assertions were formulated. Results from this initial research along with further discussions with the safety analysis team within the sponsoring company resulted in identification of a smaller research area to become the focus of this EngD thesis. The approach of how this issue has been tacked through the thesis is shown in the next section.

3.3 Overview of the Approach

The next chapters of the thesis are set out around the format of a V-model (where applicable, following stages of the technical processes from ISO 15288 [72]). These steps involve: identification of stakeholder needs, development of a solution and its integration, verification that the solution is correctly implemented, transition to its use and finally validation that the stakeholder needs have been met. An overview of this V is shown in Figure 3.1, the steps of which are described in this section. Those greyed out are not applicable stages for the project.

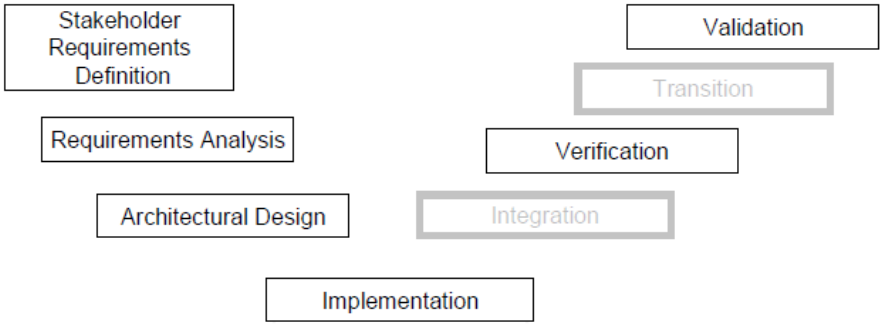


FIGURE 3.1: The stages of the V model followed through this thesis, adapted from ISO 15288

3.3.1 Stakeholder Requirements Definition

3.3.1.1 Overview

When any systems engineering project is undertaken it is important to consider all of those who have a stake in its outcome. In the case of this EngD the primary stakeholders include, but are not limited to: the author, the research institution, funding bodies and the sponsoring company. Secondary stakeholders may be other researchers within the field. For this project, the scope has been limited to stakeholders who are the direct beneficiaries of the research output, i.e. those involved in either specifying or using safety assertions through the life of a typical engineering project. This involved consideration of the entire project life and the stakeholders involved.

3.3.1.2 Inputs

No inputs are required for this stage as it is the first step of the process.

3.3.1.3 Process and Research Methodology

The nature of this research degree is that the thesis author was embedded within the sponsoring organisation as a research engineer, in accordance with the guidance of the Engineering Doctorate scheme provided by the Engineering and Physical Sciences Research Council (EPSRC) (see [39, p. 8] point 40). By the nature of the research degree, the entire project can be described as an *ethnographic study*, which Hammersley and Atkinson [55, p. 3] define as:

> "the researcher participating, overtly or covertly, in people's daily lives for an extended period of time, watching what happens, listening to what is said,

> and/or asking questions through informal and formal interviews, collecting documents and artefacts – in fact, gathering whatever data are available to throw light on the issues that are the emerging focus of inquiry"

This allowed the author to be actively involved within projects in the organisation, which meant stakeholder identification and derivation of their needs became apparent throughout the study. Requirements of the stakeholder were identified through discussions with the engineers involved in developing a safety theme (which followed a similar process to the D'Antonio et al's [4] Pentagon /S/ process and also through review of literature (both published and internal to the company). This gave an indication of the key stakeholders for the project which allowed their individual needs to be captured.

3.3.1.4 Outputs

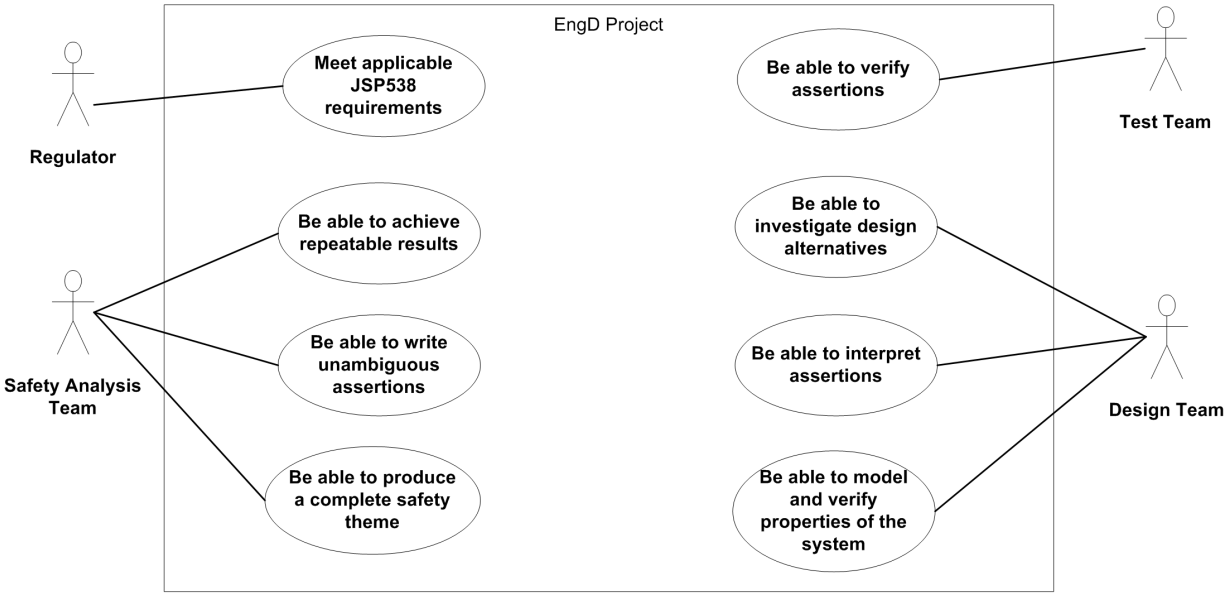


FIGURE 3.2: A use case diagram of stakeholder needs from the EngD project.

Figure 3.2 shows a UML use case diagram, which includes each of the relevant stakeholder as *actors* around the boundary. Each *use case* within the diagram details the individual needs of each stakeholder written as their desired use of the project output. All parties who would be directly impacted by the result of this project have been included on the diagram. An ideal system design would meet the needs of all stakeholder, however, prioritisations can be made where necessary. In this case, the opinions of the safety analysis team and the regulator are the most influential.

3.3.2 Requirements Analysis

3.3.2.1 Overview

The requirements analysis stage of a project should determine the boundary of the system being implemented, the functions that it should perform and identify any constraints upon the system. The system in question in this thesis describes the research output and does not represent a *physical* system. Output from this stage is a set of requirements for the project which as a set must posses overall integrity.

3.3.2.2 Inputs

To perform the requirements analysis stage of the process the needs of key stakeholders are required (as identified in the previous section).

3.3.2.3 Process

Once stakeholder requirements have been identified, requirements upon the system itself are derived in order to fulfil the needs of each stakeholder. Each of the requirements upon the system should be traceable back to a stakeholder need. This process was performed by identifying which of the needs could be achieved through common aspects of the system.

3.3.2.4 Outputs

Figure 3.3 shows a SysML requirements diagram which includes the requirements upon the project. This is presented in format where individual requirements are linked to use cases (from Figure 3.2). Demonstrations of this approach (for example by Soares and Vrancken [123]) show that the main benefit of linking the use cases to requirements is that traceability is demonstrated between the use cases and the requirements, clearly identifying gaps. Alternative methods, such as the use of a requirements management tool (e.g. IBM Rational DOORS [65]) could be adopted to manage such traceability.

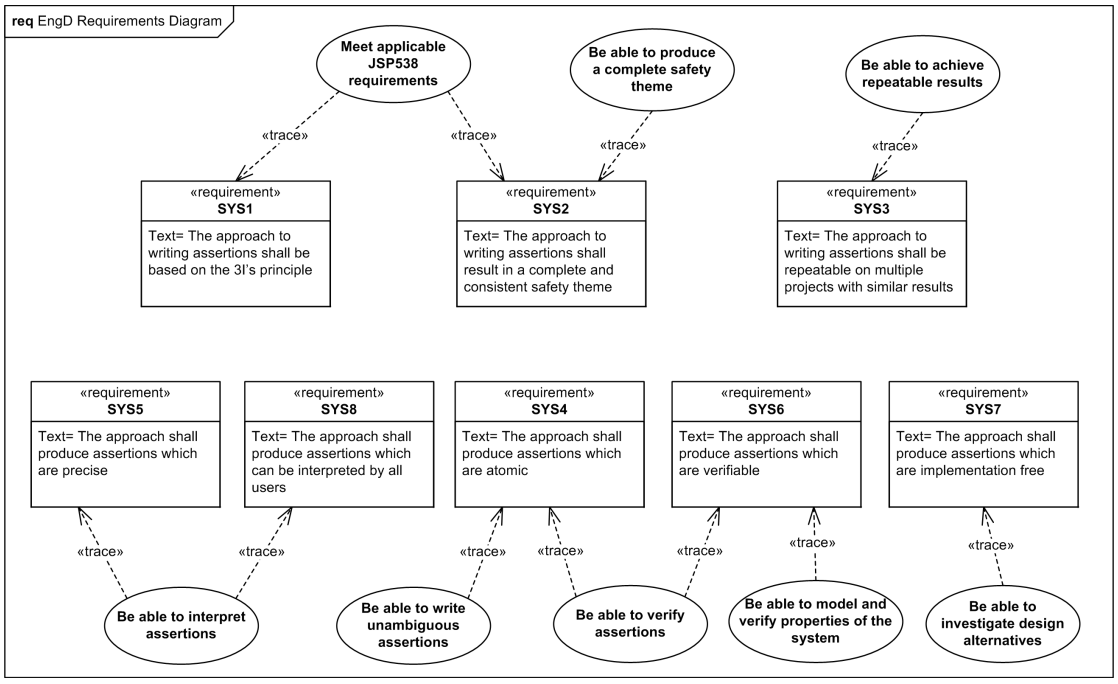


FIGURE 3.3: SysML requirements diagram of the project requirements, linked to stakeholder needs.

3.3.3 Architectural Design

3.3.3.1 Overview

The process of architectural design, according to ISO 15288 [72], is to define "areas of solution expressed as a set of separate problems of manageable, conceptual and, ultimately, realizable proportions". In comparison to the scale of this project, large engineering projects often require the system to be decomposed into many smaller, manageable sub-systems.

3.3.3.2 Process

Decomposition of a system into sub-systems can be based upon common functions or physical attributes. In the case of this project, the opportunities for decomposition are limited and were identified from analysis of the project requirements.

3.3.3.3 Outputs

The requirements for the project shown in Figure 3.3 are pitched at two different levels. Requirements upon:

Individual assertions - this group of requirements relate to the way in which assertions are written. Each assertion should be specified in such a way that the following requirements are satisfied: SYS4, SYS5, SYS6, SYS7 and SYS8.

Groups of assertions forming a safety theme - this group of requirements relates to the complete safety theme. The entire set of assertions when composed together should satisfy the following requirements: SYS1, SYS2 and SYS3.

These motivate two research aims for the rest of this thesis which require different implementations.

3.3.4 Implementation

3.3.4.1 Overview

In this stage of the V model potential solutions are identified for each of the two research areas identified through architectural design. The advantages and disadvantages of these solutions are compared against the requirements of the project. An appropriate approach was selected for each problem and then an implementation was developed through an inductive case study approach.

3.3.4.2 Inputs

The project was separated into the two research areas identified in the previous section as the architectural breakdown of the problem. Requirements upon the project were used from subsection 3.3.2. Existing safety themes from industry and literature were also require as an input used to develop the implementation.

3.3.4.3 Process and Research Methodology

Initial identification of the options available resulted from literature review within the research area, considering the different methods through which requirements are typically formulated. The best option to meet the requirements was identified through use of a decision matrix. A thorough literature review and discussion of the options is shown in Chapter 4, Section 4.2.3.

The author used an inductive research approach in order to meet the requirements. This built upon the research within the initial study discussed in Section 3.2 in which a full scale industry project was used to test a formal modelling technique. This same project

and other examples from literature were used to identify a theory which formed the foundation of the research. The approach taken is shown in Figure 3.4 (which is taken from Blackstone [14]).

FIGURE 3.4: The stages of inductive research.

As described, the *gather data* phase used existing safety themes from industry and literature. These case studies were used to identify solutions for both research areas, by considering: how assertions were written and how these were combined to produce a complete safety theme. Eisenhardt examines case study based approaches in [36], noting that one of the main difficulties in generating theories from case data is deciding an appropriate number of examples to work with, where he recommends a minimum of 4 cases. A safety theme is decomposed into safety subsystems, which in themselves provide multiple independent layers to analyse. The industry case study provided three safety subsystems for analysis and two further published examples by Johnson [74] and Li [87] were considered.

This *analysis stage* involved review of each of the safety themes and grouping of their assertions based upon the 3I's and common recurring patterns. Using the methods selected from Chapter 4 the analysis of this data is presented in the form of template assertions and a set of patterns of how these templates fit together. These *theories* were developed, both for individual assertions and for the entire safety theme, through iterative development of these theories they were reviewed with the major stakeholder of the project.

3.3.4.4 Outputs

To arrive at the decision of how the research goals would be achieved the options were identified through literature (reviewed in Chapter 4) and the most applicable was selected through use of a decision matrix (shown in Chapter 4, Table 4.1). The main contribution from this stage was the development of a set of patterns and templates for specifying a safety theme, full detail of which is shown in Chapter 5.

3.3.5 Verification

3.3.5.1 Overview

In this stage of the process the theory developed in Section 3.3.4 is tested through a deductive approach. This approach is used to confirm that the implementation meets the requirements upon the project (i.e. the correct approach has been developed).

3.3.5.2 Inputs

The requirements of the project and the outputs of the implementation stage are required to verify against each other.

3.3.5.3 Process and Research Methodology

The theory was reviewed with the safety analysis team to ensure the template assertions and patterns were all viable and acceptable to the stakeholder who would use them. To demonstrate whether the theory was valid (and to verify the project requirements) the approach was applied to reverse engineer three different safety themes. The results of which have been published (as seen in paper B.3). A deductive approach was used to show that the derived theory could be applied to multiple cases and a hypothesis could be proven. Figure 3.5 shows the stages of the approach (taken from Blackstone [14]).

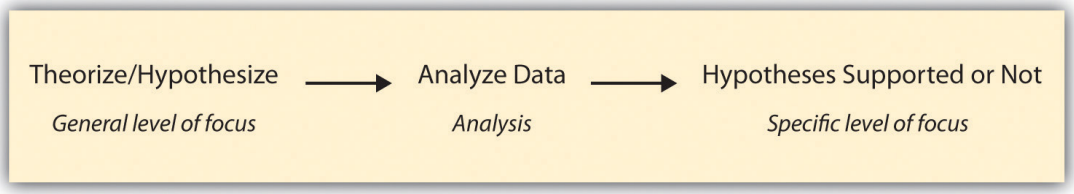


FIGURE 3.5: The stages of deductive research.

The *analysis* stage was performed using safety themes from three industry projects. The safety theme from each was developed by a different member of the safety analysis team, this provided some stylistic independence in the way assertions were written and the approach taken to develop the complete safety theme.

Further to application of the approach to these industry case studies to identify any new assertions, the example system presented by Johnson was used to verify that the patterns produced complete safety subsystems. To do this the first stage of the approach detailed in Appendix B Paper B.2 was revised to include template model checks which mirrored the template assertions which have been developed.

3.3.5.4 Outputs

Requirements associated with both of the architectural sections of the project are demonstrated with these examples presented in the Appendix and Papers and we discuss how this demonstrates that project requirements are met.

3.3.6 Validation

3.3.6.1 Overview

This final stage of the approach is validation that the outcomes of the project met the needs of the stakeholders (i.e. did we build the right thing?).

3.3.6.2 Inputs

Inputs to this stage are: the list of stakeholder needs presented in Figure 3.2 and the approach developed to meet these needs as as result of the implementation stage.

3.3.6.3 Process and Research Methodology

Since it has not been possible to follow the full lifecycle of an industry project to determine if all stakeholder requirements have been implemented through testing, confirmation from the research sponsor was sought. A discussion of the individual stakeholder requirements is shown in Chapter 7.

3.3.6.4 Outputs

The sponsoring company has provided a letter of confirmation that the output of the research met the needs of the stakeholders, this is presented in Appendix D. The letter also confirms that the approach has been applied to industry projects, where these can not be presented in detail in the thesis.

3.4 Conclusions

In this chapter the needs of the project have been identified in detail using a V-model format. The needs of each stakeholder for the project have been identified, from which

project specific requirements have been derived. These requirements fell into two categories, those upon assertions and those upon an entire safety theme. Sections 3.3.4 through to 7.7 describe the approach taken to implement a solution to these needs, justifying the use of deductive and inductive research methodologies together. The detail of the implemented solution is covered in detail through the remainder of this thesis.

Chapter 4

Literature Review and Concept Selection

4.1 Introduction

This chapter is a literature review of requirements specification methods and best practice in requirements engineering. Two main sections of literature are reviewed, these refer to the two areas of the project identified from the system architecture section of Chapter 3 (Section 3.3.3): writing individual assertions and compiling sets of assertions. A number of potential solutions from literature are identified for both of these areas in turn, the merits and drawbacks of each option are identified and compared against the requirements for that aspect of the project. Finally, the chapter concludes with a discussion and conclusions of the approach that have been adopted for use in this project.

4.2 Writing Individual Assertions

In this section, the requirements for the project upon individual assertions are re-visited whilst considering the best practice in requirements engineering. Different potential solutions are identified in Section 4.2.2 and the most appropriate method to meet these needs is selected through use of a trade study as shown in Section 4.2.3.

4.2.1 Project Requirements for Individual Assertions

From decomposition of the project into the two parts five requirements were deemed to be applicable to individual assertions. In order to fulfil these requirements whilst writing of assertions, it is necessary to identify what the traits of a 'good' requirement are and how some of these issues have been tackled within the wider area of requirements engineering.

The following factors are heavily referred to throughout literature [42, 51, 67, 114, 125] and are deemed characteristics of good requirements. These have been linked to the requirements upon the project identified in Chapter 3, Section 3.3.2:

SYS4 - Atomicity - each requirement is an individual statement they are not composite requirements?

SYS6 - Verifiability - is it possible to show that a requirement is met by an implementation?

SYS7 - Implementation free - requirements should state the intent of part of a system to be designed, not state how it will be achieved

SYS5/8 - Unambiguous - requirements should only have one meaning or interpretation

These traits can be determined about each individual requirement (or assertion). Writing requirements in an imprecise way may cause problems to be introduced and ideally the good traits of requirements shown above should be adhered to when writing assertions in a safety theme.

4.2.2 Requirements Specification Techniques

Requirements can be specified in a number of different ways, each of which is discussed within this section. The different types are: natural language, constrained natural language, diagrammatic notations and formal specification. These different methods increase in rigour as they transition from natural language through to formal specification, alongside this increased rigour the complexity of their use also increases.

4.2.2.1 Natural Language

Natural language is the way which most people will think and be able to communicate what is required of a system. It seems most natural for people to communicate in written

language that we all understand. Natural language is usually the only common platform between all of the potential stakeholders that are interested in a system.

A recent study by Luisa et al. [88] showed that 79% of projects analysed (in the software engineering domain) used natural language to capture their requirements. This is not surprising since natural language presents the most flexibility in the way requirements can be written, however, this flexibility means this technique can suffer from issues this research aims to solve. Alexander and Stevens describe the anatomy of a good requirement in [3], which includes the following:

1. User type - who benefits from the requirement
2. Result type - what should be achieved
3. Object - what object(s) will be affected
4. Qualifier - any conditions

This anatomy defines parts which would be ideally presented in a requirement. Some assertions provided by Johnson in his example safety theme in [74] do not follow this form. For example an assertion shown in Chapter 2, Section 2.3.3 that "No enablement signal generated, in absence of correct environment" does not define the user type or what should be affected by the requirement and this assertion could be improved.

Denger et al. have identified in [27] that two approaches that can be used to help improve the way in which requirements are written. The first approach is to *detect* imprecision through the use of tools which scan requirements for common traits. The second method is to *prevent* the introduction of issues when writing the requirements. The former method has resulted in development of tools such as the Quality Analyzer for Requirements Specification (QuARS) which has been demonstrated to have identified vague or undefined elements in requirements. Lami reports in [84, p. 26] that in a case study applying QuARS, 16% of defective requirements the tool identified were not identified by human inspection. Tjong et al [132] support the second of these two approaches, noting that improvement of requirement should ideally be during capture and writing, rather than detecting after they are written (e.g. scanning with tools). No structure is enforced for natural language requirements, therefore, use of template requirements, syntax based approaches or frameworks are used to produce (or re-write) a requirements specification. These methods are discussed in the following subsection.

4.2.2.2 Constrained Natural Language

From results of the study in [88] Luisa et al. identified that 79% of software project requirements were written in natural language. 16% of the requirements capture methods were then characterised as constrained (or 'structured') natural language. These have had a significant interest both across the disciplines of systems engineering and software engineering. Examples include the Easy Approach to Requirements Syntax (EARS) as presented by Mavin, Wilkinson, Harwood and Novak in [91], boilerplates presented by Hull et al in [64], the Volere requirements specification template [111], a number more are summarised by Dick and Llorens in [33] on the topic, where they summarise that using templates provides the following benefits (amongst others): a uniform manner of writing requirements, ensuring essential characteristics are present, and easy identification of conflicting requirements.

Toro, Jiménez, Cortés and Bonilla also note in [133] that "filling in blanks in pre-written sentences... is easier and faster than writing a whole paragraph", which supports the claim by Robertson and Robertson that their Volere requirements specification template has been used in "organizations worldwide by saving significant time and money for their requirements activities" [7]. Findings from use of EARS, as described in [90, 91], show that use of the syntax templates results in an increase in the number of requirements due to compound requirements being broken into atomic ones (which is one of the characteristics described in Section 4.2.1).

4.2.2.3 Formal Specification

Formal specification can be defined as "the expression, in some formal language and at some level of abstraction, of a collection of properties some system should satisfy" [85]. Formal languages are those used to abstractly describe the properties of a system in a precise way by using both: an alphabet of symbols and also a set of rules describing how they are used together (i.e. a syntax). Comparatively to the systems engineering domain, such methods have had a higher uptake in the software engineering world, with developments of: the Z notation [126], the Specification and Description Language (SDL) [17], the Vienna Development Method Specification Language (VDM-SL) [102], the B method [1] amongst others.

Benefits for use within the software engineering domain are that these specification languages often have rules from which they can be translated into a computer program, model or simulation. This becomes particularly useful as the sequence of specifying, proving and then implementing a system use a common language which can be understood. For example McEwan demonstrated in [92] that control requirements could

be specified using a formal language, translated into a implementation then properties verified about the software using simulation. This is a beneficial approach which allows industrial scale problems to be taken from a specification through to a model using appropriate tool support (e.g. the B-toolkit as presented by Boulanger in [15, p. 142]).

What is discussed less in literature is the use of specification techniques for systems engineering problems (as opposed to software engineering projects), however, a comparatively recent development is the Compass Modelling Language (CML) [135]. CML is being designed for systems engineering rather than software with focus upon the specification and modelling of behaviours of 'systems of systems' which are out of scope for this thesis.

Hall [54] notes that formal methods of specification are useful for identifying errors, due to the rigorous nature and constrained language used. This removes some of the lexical ambiguity that natural language can cause. Lamsweerde notes in [85] the conflicting needs for formal specification languages, in that their expressive power must be able to capture all of the needs of a requirements analyst simply, whilst on the other hand they should be communicable between different stakeholders (some of whom will not be fluent in mathematical notations). Bowen reports in an overview of formal methods use in industry [16] that uptake has been lacking for such approaches, however, this is not to say they have not been used. It is argued that the expected level of expertise required is often deemed a factor that limits their use (i.e. needing a formal methods guru on call), whereas the likes of Sommerville [124] argue that all engineering disciplines require mathematical formalisms and that requirements specification should not be an exception. To refer to the study [88] by Luisa et al. once again, only 5% of those projects surveyed used formal specification of their systems.

4.2.2.4 Graphical Notations

Requirements can also be captured using graphical notations, the most common of which are the Unified Modelling Language (UML) [115] and its extension into the Systems Modelling Language (SysML) [49]. Diagrams of interest when specifying requirements are the UML Use Case diagram and the SysML Requirements Diagram (as used through Chapter 3).

Use Case Diagrams

Use case diagrams are used to specify all of the necessary functionality of a system from the perspective of different stakeholders (shown as actors). Interactions between the

actors and the system are defined using a number of scenarios which describe what the users or other people/systems interacting with that one being specified would expect to see from the system. Larman [86, p. 81] provides a number of guidelines for writing use cases, a key observation is that use cases should be specified with the system as a black box. Use cases are intended for capturing functional requirements of a system, which is a limitation of the approach for capturing assertions in terms of the 3I's. For example an assertion of incompatibility would not be captured best as a use case as it could not be argued as a behaviour of one component.

Requirements Diagrams

UML has evolved into SysML which contains two new diagrams over standard UML, the Requirements and Parametric Diagrams. The requirements diagram is designed to bridge the gap between natural language requirements. According to Soares and Vrancken [123] a basic requirement stereotype will consist of a requirement text and an ID. Individual requirements also have relationships between them, for example a requirement upon a single component could have been derived from a user need, or related requirements can be decomposed into a number of sub-requirements.

The use of these approaches is often integrated into a Model Based Systems Engineering approach, where rather than capturing the requirements and design of a system in documents they are captured in a model. Using graphical notations such as UML do not themselves force a given structure upon the way individual requirements are written, unlike the structured natural language approach.

4.2.3 Concept Selection

The decision matrix is a tool introduced by Pugh [106] to support engineering decisions and selection from a number of alternatives through a quantitative approach. The approach requires the needs from the system (which become rows of the matrix) and the options available (which become the columns). To manage a large number of options, every option is compared against the first of the options to maintain a common baseline for comparison. Burge explains in [18] that this is due to human ability to handle complexity when a large number of options is presented. In the matrix in Table 4.1, the first column refers to the current state of the art approach to use natural language to specify assertions. The rows of the matrix show the project requirements identified in Chapter 3. Within the table, S refers to the baseline, + refers to a better solution over the baseline and - refers to a worse solution than the baseline.

TABLE 4.1: A decision matrix used for selection of an assertion specification method.

ID	**Project Requirement**	**Natural Language**	**Constrained Natural Language**	**Formal Specification**	**Graphical Notations**
SYS4	Atomic	*S*	+	+	S
SYS5	Precise	*S*	+	+	S
SYS6	Verifiable	*S*	+	+	S
SYS7	Implementation free	*S*	S	S	S
SYS8	Interpretable	*S*	+	-	S
Total		0	+4	+2	0

The comparisons shown in Table 4.1 are discussed through the next sections and leads to a proposal of which method should be taken into implementation of the solution.

Natural Language forms the current state of the art practice for writing assertions and issues have been identified with this current practice, therefore it has been used as the baseline for comparison (hence all have been marked as S).

Constrained Natural Language Boilerplate and requirements syntax approaches have shown to decompose larger compound requirements into smaller atomic statements which are easier to understand and therefore verify. Such approaches remove lexical ambiguity by writing requirements in a set way, written language is still used - which means the assertions would be more interpretable to different stakeholders than formal specification.

Formal Specification Formal methods are useful for rigorous specification and ensuring that all requirements are fully understood and also ensure that requirements are atomic (each can independently be tested), they can be verified through formal model checking techniques and enforce precision. A negative point is that formal models are not necessarily interpretable by all stakeholders that would use such requirements through the system life. A further note is that a model-based specification would not integrate well into a document centric process (e.g. parts of the Pentagon /S/ process [4]).

Graphical Notations Use cases do not seem applicable for use with assertions, their main use is for functional requirements of a system. Requirements diagrams seem more appropriate, however, still requires use of natural language and therefore would only add benefits of modelling relationships between requirements (not solving the issues of precision).

None of the options provide a direct solution to the issue of capturing implementation specific detail into assertions. This is something that needs to be enforced during use of a new approach. Overall it appears that Constrained Natural Language templates are the most appropriate method for limiting the way in which requirements are written, this allows current document centric (rather than model based) systems engineering process be supported.

4.3 Completeness of Requirements Sets

4.3.1 Project Requirements for the Entire Safety Theme

Many of the requirements of the project have been covered with addressing how assertions are written individually, however, there are three remaining requirements which require consideration of the set of assertions as a whole. These requirements are:

SYS1 - The approach to writing assertions shall be based on the 3I's principle;

SYS2 - The approach to writing assertions shall result in a complete and consistent safety theme;

SYS3 - The approach to writing assertions shall be repeatable on multiple projects with similar results;

To achieve these requirements, the following literature review section considers ways in which a sets of requirements are typically developed in order to demonstrate their completeness for a given application. SYS1 requires that the approach is developed upon, which will build upon the state of the art knowledge in weapons safety as discussed in Chapter 2.

In order to achieve the requirement SYS2, the factors which impact a *set* of requirements should be considered, such as:

Consistency - are all requirements achievable?

Completeness - are all necessary requirements identified and documented? As a whole, do they all do the necessary job?

Priority - are some more important to be achieved than others?

With respect to issues on requirement priority, in some cases an ideal solution for a set of requirements may not be possible, for example with weapons systems there are significant trades between reliability and safety, e.g. limited ability do perform pre-use self checking for safety reasons as noted by Bierbaum and Wright in [13]. In this situation, Cooper and Spray explain that priority is given to safety [130]. An ideally reliable high consequence arming system would not have strong links or exclusion region barriers preventing energy from allowing its necessary function to occur. This thesis focusses on only assertions in a safety theme, where failure to meet one will result in a weakness in one safety subsystem. Therefore, all assertions have equal priority and must be met without compromise. Due to this, only approaches which can be used to achieve consistency and completeness in a requirements set are discussed through the next section.

4.3.2 Concept Identification

Throughout literature a number of different approaches have been developed to ensure completeness when tackling a problem. This can be achieved through a structured approach to determine the requirements or design of a system, or to verify that a system will behave as desired through modelling. As described in Chapter 3, Section 3.2, an initial study was performed as part of this research into modelling methods and from the result of this it is considered as a possible method for meeting the project requirements. This section of the literature review addresses methods for writing a complete set of requirement right first time, this includes: ontology, patterns and decomposition.

4.3.2.1 Ontology

According to Gruber [52] an ontology can be defined as "a set of representational primitives with which to model a domain of knowledge or discourse". They are often used in software engineering to reduce the scope of requirements or design to a limited yet specific vocabulary which has been identified to describe all of the possible parts of that system. Gasevic et al. [50, p. 45] explain that ontology goes much further than just terminology, but also includes: classifications, taxonomy, hierarchy and constraints. Use of ontology in requirements engineering is a shift to incorporating domain specific knowledge during requirements capture, this involves utilising knowledge of the domain to define the problem space rather than only using this knowledge in the solution space. These two areas can often be disperate and recent work by Kossmann et al. in [83] and their prior publications encourage use of ontology-driven requirements engineering. Using this approach in the engineering process can help describe multiple projects in a

similar manner and removes the need for requirements to be iteratively evolved through a project as the ontology will be developed to capture mature requirements concepts. Such results have been shown in a number of areas, e.g. Kossmann, Wong, Odeh and Gillies in the aerospace domain [82], Shibaoka, Kaiya and Saeki through the elicitation of requirements for a feed reader [120], where an improvement in quality of requirements was shown through use of an ontology. Siegmund, Thomas, Zhao, Pan and Assmann note that by using such an approach it is possible to "quickly identify where (these) requirements are inconsistent and incomplete" [121].

In terms of meeting the requirements for this aspect of the project (as noted in Section 4.3.1), ontology is a potential option which would address the issues of ensuring completeness of a safety theme if an ontology were designed with the correct amount of detail. Use of an ontology would ensure that safety themes were specified in a repeatable manner (especially if combined with the use of a requirements syntax as selected in Section 4.2.3). The major issue that would arise in using ontology for writing assertions during development of a safety theme is the dependency upon the architecture of the system, which can vary drastically between different options as exemplified by Hansen in [56]). This dependency would limit the use of an ontology to strictly define the primitives of *every* possible system.

The author developed a domain model to represent the knowledge within the area prior to the research within this thesis, as published in Paper B.1 as Figure 1. This figure shows the relationships between the constructs which have been modelled to represent the system, its topology, and the threats within an environment.

4.3.2.2 Patterns

Rising [110] defines a pattern as "simply a form of documentation", however, this documentation is observed and formed from many projects. Rising suggests such patterns should be subject to a minimum of three applications to be proven useful [110]. Patterns have been developed for use in a range of disciplines: Pont identifying them for time-triggered embedded systems in [105], Cloutier has identified in [21] that patterns have been used in various areas of the systems engineering process, including requirements writing (which Meszaros and Doble described in [94]), use cases (which Adolph, Bramble, Cockburn and Pols describe in [2]) and architecture (which ranges in applications from Cloutier, Muller, Verma, Roshanak, Hole and Bone's work on system architecture patterns [22] and Cloutier and Verma's enterprise architecture patterns [23] through to systems of systems, as presented by Kalawsky, Tian, Joannou, Sanduka and Masin in [77]).

According to Cloutier [22, p. 4], patterns have no fixed format in which they are documented. This is beneficial in that no knowledge of a specific notation (e.g. UML or SysML) is required to interpret them, allowing multiple stakeholders to understand, use, and even contribute to the patterns. In terms of their use for capture and re-use of knowledge it is argued that using patterns can reduce innovation in engineering, a valid opinion, which may not be desirable in certain industries. For high consequence arming systems the scope for innovation in safety is limited to methods of implementing such devices reliably (as discussed towards the end of Chapter 2 Section 2.2.3.3). Therefore, the author argues that the underpinning safety philosophy (based around the 3I's) remains unchanged and would be a good candidate for such a method. In comparison to an ontology, a set of patterns is less strict and can define a limited set of re-usable constructs applicable to a given system rather than claiming to define the entire domain. This can be seen as a benefit or a drawback depending on the perspective. For a first-pass in identification of patterns and recurring constructs in a safety theme this is beneficial, however, if further rigour were required these patterns could be developed into an ontology of the domain.

4.3.2.3 Requirement Decomposition

One approach adopted within the sponsoring organisation utilises the concept of a "success tree". This is the inverse of a fault tree, as described in [28], and can be used qualitatively to illustrate the build up of events which are necessary together to achieve a top level "success event". By using AND gates, it is a useful method to capture which requirements together fulfil an event higher in the tree and can be used to identify any obvious logical gaps. A recent paper by Johnson [73] builds upon this approach and introduces a method designed for specification of safety requirements which combines the use of natural language requirements alongside the formal logic of propositional analysis. This introduces rigour when considering requirements and the way they are worded and decomposed. This propositional analysis method has been developed alongside the research described in this thesis. At the time of concept selection, only the success tree approach was in use, this was deemed the current practice for visualising how a number of assertions could be combined to fulfil the requirements of a single safety subsystem (i.e. a number of trees would exist for a safety theme).

4.3.3 Concept Selection

From the approaches discussed in the previous section each is considered in terms of how applicable it would be for meeting the requirements upon this aspect of the project.

Basing the approach on the 3I's principle - Regardless which of the approaches discussed in the previous section is selected, each would handle the 3I's in a different way. Therefore, it would be difficult to compare how relevant each would be and concept selection should be limited to comparison of how well the other project requirements are met.

Completeness and consistency - Each of the methods previously discussed have their own benefits for ensuring a safety theme is complete. Since each application will have a different architecture, it is difficult to determine using knowledge of the domain in a manner which allows re-use. Patterns could be derived which are applicable for different architectural layouts and could therefore be selected based upon the architecture of the system which they are being applied to. Using formal requirements decomposition is a useful method to ensure that requirements are consistent throughout the set, however, this alone would not echo the constructs of a safety theme as well as patterns or ontology.

Repeatable results between projects - Using methods such as patterns and ontology are the most appropriate for providing a repeatable framework that could be used for multiple different projects. As discussed above, using a set of patterns is likely to be more flexible in terms of selecting the appropriate constructs for a number of different architectures.

Ontology may be too elaborate to be a solution for the requirements presented in this project. It could become a further development into keeping the same terminology and structures from project to project. Patterns are deemed to be the most useful option for capturing repeatable aspects of a safety theme in an ad-hoc manner. This allows the focus of the research to be upon the detail in the patterns and the approach to identifying and applying the patterns than the formalism of the patterns themselves.

4.4 Conclusions

In this chapter, two distinct areas have been considered: methods of specifying individual assertions and methods of producing a complete safety theme from project to project. For the former, constrained natural language has been selected as an option for writing assertions in a repeatable way. For the latter, patterns seem the most appropriate method of identifying re-usable constructs (in terms of constrained natural language assertions) which can be combined to develop a complete safety theme. The next chapter describes how these two approaches can be combined in order to address the requirements upon this project.

Chapter 5

Implementation of Assertion Templates and Patterns

5.1 Introduction

The focus of this chapter is upon the implementation stage of the V-model as described in Chapter 3, Section 3.3.4. At this point the requirements for the project have been defined (SYS1-8), the project has been split into two parts (regarding the individual assertions and the entire safety theme), and finally the best methods for tackling the problem have been selected. This chapter describes the major aspect of research contribution of the thesis, which reflects the two parts of the project.

The first contribution of this research work is presented in Section 5.2.1, a set of template requirements which are presented using structured natural language. These are designed to be applicable to support specification assertions in a manner which meets the requirements of the project set out in SYS4, SYS5, SYS6, SYS7 and SYS8. The second contribution is a set of patterns which describe the relationships between these different requirement templates in order to provide a complete set of atomic assertions which use the 3I's as an underpinning safety philosophy, these are designed to be used based upon the system architecture. The patterns are designed to meet the project requirements SYS1, SYS2 and SYS3. Discussion in the next section begins with the way in which top-level assertions are presented and describes how the author has developed the template assertions from here.

5.2 Writing Individual Assertions

As discussed in the previous chapter a set of assertions is captured as a safety theme, a blueprint of how safety features will be apportioned around the system. This can be viewed at two different levels of abstraction, firstly as top-level assertion which explain how the 3I's can be used to loosely define the ideas behind a safety theme. Secondly, lower level assertions can be used to make up a single top-level assertion. These lower level assertions can be categorised into a limited number of types.

5.2.1 Top-Level Assertions

The 3I's principles are used as a foundation of the assertions about the safety of the system. Typically an overarching argument will be used to describe how safety will be achieved. This is often attempted using isolation first and where isolation cannot be used, or has been used in excess, incompatibility and/or inoperability can be used as they are independent. Each of these types will now be discussed and exemplified. The following decomposition of an isolation assertion is not a novel contribution of this thesis, it has been presented by Johnson in [74]. This approach to decomposition of a top-level assertion into a number of lower-level, more realistic assertions has been a motivating factor for the work described in this section.

5.2.1.1 Isolation

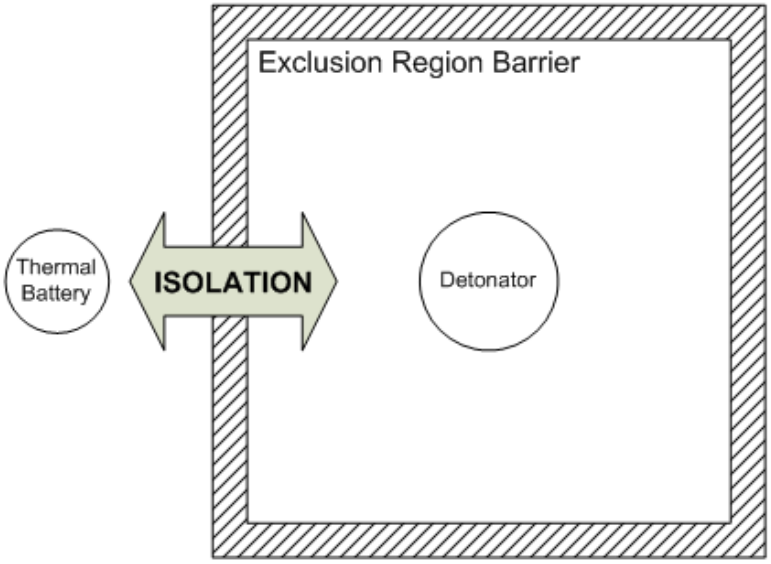


FIGURE 5.1: Example of an Exclusion Region Barrier (ERB) being used to isolate components.

Assertion The ERB shall isolate the Thermal Battery from the Detonator.

The assertion shown here is a typical top level assertion of isolation. This involves three components, a component which generates a hazard (i.e. Thermal Battery, the details of which has been presented by Guidotti and Masset in [53]), the isolating component (the Exclusion Region Barrier) and the vulnerable component (the Detonator), as shown in Figure 5.1. Ideally isolation would be perfect, resulting in none of the energy from the hazard generating component reaching the vulnerable component. In reality this is not physically possible and a number of low level assertions are used in combination to achieve isolation.

The ERB achieves isolation through *attenuation* of energy produced by the Thermal Battery. Energy must be attenuated to a low enough level such that any residual energy that passes through the ERB is incompatible with the Detonator (the detail of 'incompatibility' is discussed further in Section 5.2.1.2). Isolation is achieved through use of two assertions: attenuation and that it produces an output which is incompatible.

A further argument of incompatibility can be included in order to assert that the hazard generating component is not able to produce hazards above a threshold which cannot be attenuated (or where the ERB could become permanently damaged). This results in three assertions being used to meet the top-level assertion of isolation, as shown in Figure 5.2.

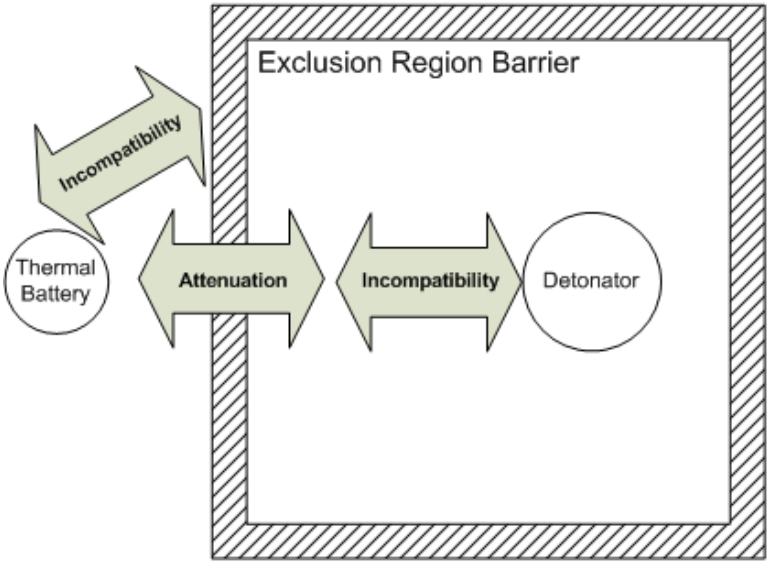


FIGURE 5.2: Lower level assertions required to achieve isolation.

There must be a method to allow the energy of the thermal battery into the system when operation is required, therefore a Strong Link (SL) is used to provide a portal through which energy can enter the inner exclusion region. The SL is used to isolate energy

between two regions until use is authorised. This requires the SL to have at least two *states*. In the first of these states the SL provides isolation, to do this it must attenuate electrical energy from passing through to the inner region in a similar fashion to the ERB. When this is achieved the component can be seen to provide *assured safety* and therefore its behaviour can be relied upon. The second state is one where authorised operation of the system is requested, in which case the strong link must be opened to provide an electrical connection between the outer and inner regions.

Since the SL is able to unlock, removing isolation, it must be asserted that it does not do so unless in an authorised operational role.[1] To do this one must consider the conditions in which it will actually isolate (i.e. only when the SL is in the isolating state) and also the events which could cause the component to leave such a state, (e.g. a strong link must receive the correct UQS to the discriminator for it to unlock and lose assurance of safety). It is also important to consider whether these events causing state change are compatible with the physically local components or the wider environment of the system, ideally it would be possible to assert that incompatibility exists between any potential threat and a compatible UQS. Figure 5.3 shows the SL as an isolating component, its state change events must be incompatible with the outputs of the thermal battery. Similarly to the example of the ERB, when in the isolating state the SL must attenuate and any residual energy output into the exclusion region must be incompatible with the detonator.

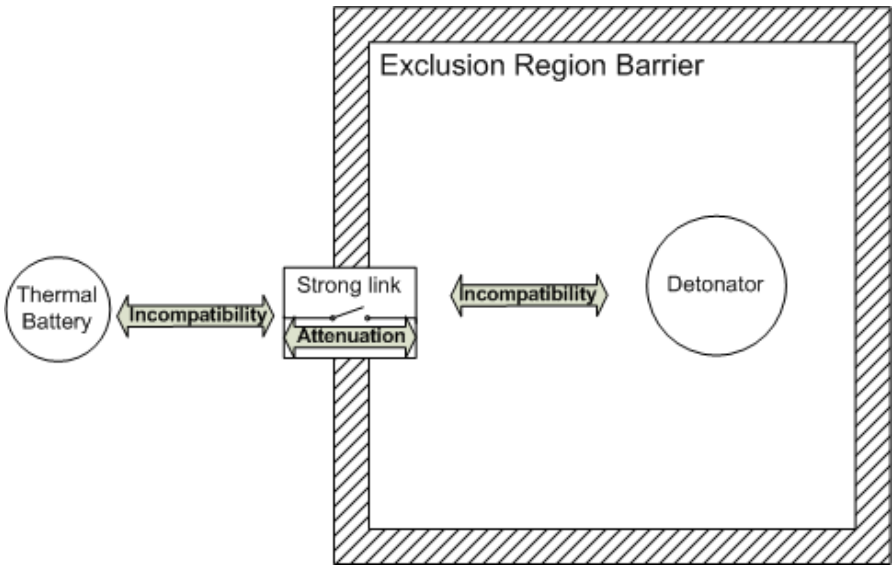


FIGURE 5.3: Example of isolation achieved by a strong link.

[1]System safety will only be assured based upon the weakest of a group of components, having a weak SL would mean the argument of the ERB is weakened since it can be bypassed via the SL.

5.2.1.2 Incompatibility

Incompatibility is an assertion made between two components: a hazard generating component or propagating a hazard and a vulnerable component. Figure 5.4 shows the example of a Thermal Battery and a Detonator. A vulnerable component is seen to have more than one state, one where incompatibility is argued and a second which is the result of 'compatibility'. If a component can change to a state in which the level of safety assurance is lower than that in its current state, the state change event will be referred to as a *vulnerability*, a term used throughout this thesis. Types of state and the associated levels of safety assurance will be discussed in Section 5.2.2.

FIGURE 5.4: An example of incompatibility of components within the same region.

Assertions of incompatibility are often required for components within the same exclusion region, since it is difficult to assert that coupling of the two components is not possible in abnormal environments [2]. An incompatibility assertion specifies the desired behaviour of both components involved. For example (using the components in Figure 5.4) the Thermal Battery must not be able to produce a signal which is compatible with the Detonator, whilst similarly, the Detonator must not become compatible with the output of the Thermal Battery. It is important to consider whether conditions exist where the two components are compatible. If so, each component can be seen to have multiple states (i.e. a compatible state and an incompatible state).

A pattern emerges from the decompositions of top-level assertions about isolation and incompatibility. Most components can be seen to have multiple states. The conditions of state change and behaviours in each state contribute to achieving the top level safety assertion.

5.2.1.3 Inoperability

Inoperability is used when a component (required within the arming chain) becomes unable to function in order to achieve safety requirements. It has already been noted in Chapter 2 that the most common example is a charging capacitor within a Firing Unit being used as a thermal weak link, where the Firing Unit can become inoperable

[2] In fact, it is assumed that in abnormal environment such coupling *is* possible within the same ER and can conceivably bring components together that are not in the same region.

and no longer hold charge. The result of this inoperability is that the Firing Unit cannot produce energy which is compatible with the Detonator, as shown in Figure 5.5. The inoperability assertion can be decomposed into a more realistic assertions of: incompatibility in a given state and also the conditions of when such a state change can occur.

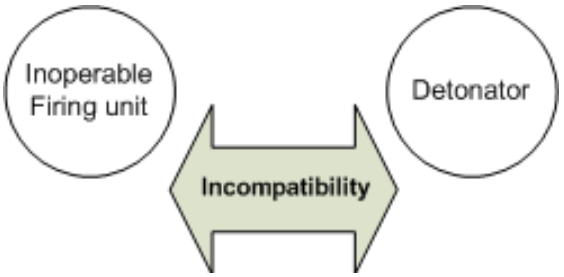


FIGURE 5.5: An example showing inoperability as incompatibility in an inoperable state.

A weak link component is initially in an operable state and then enters an incompatible state due to effects of the environment. Some inoperable components exist where the opposite is possible, two examples are relevant: firstly the previously discussed example of a Thermal Battery requires activation before it produces its full range of output. The initial state of this component could be described as inoperable. Plummer and Greenwood list a number of types of stronglink energy control device in [103], which Elliott summarises in Table II of [38]. An example is a magnetic barrier, used within some strong links where a transformer steps up voltage when the strong link is unlocked, when the SL is locked the transformer would be seen as inoperable.

5.2.1.4 Race

Although it is not noted as one of the 3I's, the idea of a *race* between two components changing state is considered safety-critical. Where strong links and weak link components are used, they must fail in a given order. The weak link must change to a state where safety is assured *before* the strong link changes to a state where safety is not assured. This way the system always has one component upon which safety can be assured.

5.2.1.5 Example

To give an overview of how top-level assertions are made, defining the behaviour of components around the system, the author refers to an example which has been presented in two prior publications. Figure 5.6 is taken from Ekman et al. in [37, p. 11, Figure

2] but the same architecture is also presented by Li et al. in [87, p. 5, Figure 4]. Top level assertions are useful for providing the vision of how safety is achieved by the different components. For this vision to be useful, a detailed list of assertions must be documented, building the foundation of specifications for each component within the system.

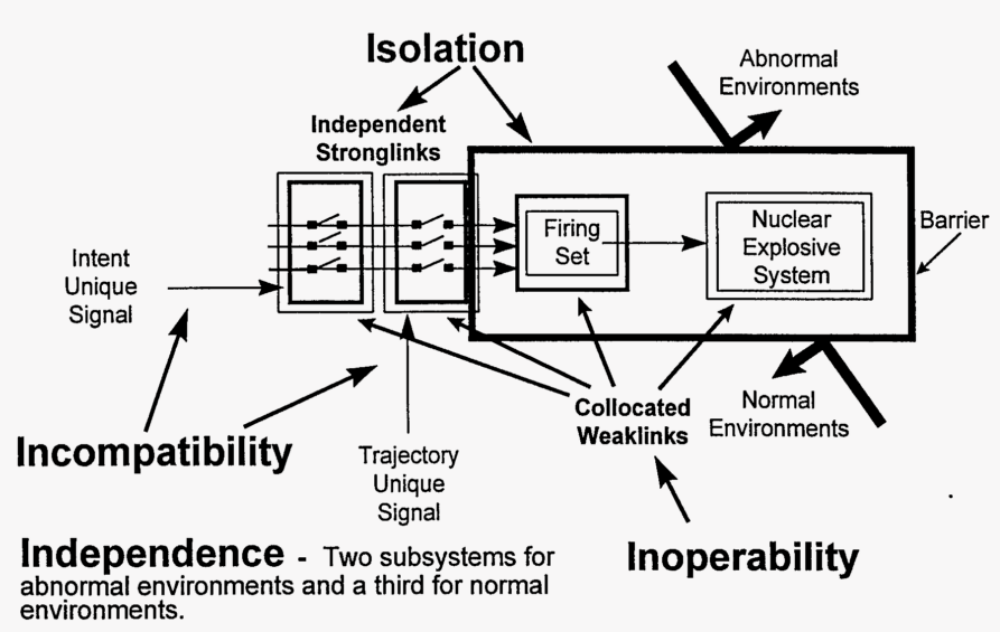


FIGURE 5.6: A published system architecture and top level assertions.

What is clear from the example in Figure 5.6 is that the original 3I's are used as the foundation of a system safety argument. Neither of the publications presenting this example follow up the top level assertions with a full list of detailed assertions about the system. The closest to that in any prior publication is that of Johnson in [74].

5.2.2 New View of the 3Is

Throughout the previous subsection the four types of top-level assertion were discussed. These varied in whether one or more components were necessary to achieve their required behaviour (for example isolation is performed by a single component, where incompatibility is a behaviour between two different components). As discussed throughout this section, a number of commonalities have been observed through these four top-level assertions, leading to a new lower level interpretation of them.

5.2.2.1 Lower Level Assertions

When considering the realistic behaviour of components (rather than idealistic or perfect behaviour), the author has observed that *all* the top-level assertions decompose into a number of lower level assertions (beyond the decomposition of isolation presented by Johnson [74]). Combinations of these low-level assertions can be composed to assert the 3I's. Four lower level assertions were referred to throughout the previous subsection: attenuation, incompatibility, state change and race. Table 5.1 shows the relationship from the original 3I's and race, to these four new low level assertions.

TABLE 5.1: The relationships between the 3Is, race and the low level assertions.

Top-level \Lower level	**Attenuation**	**Incompatibility**	**State change**	**Race**
Isolation	x	x	x	
Incompatibility		x	x	
Inoperability		x	x	
Race			x	x

From Table 5.1 it is clear that lower level assertions about state change are used to underpin all of the top level assertion types. This requires a new view of system components, where each can be seen as a state machine.

5.2.2.2 State Machine View

Realistically, all components of the system which contribute to the safety theme will have two or more states, each falling into one of the following categories:

Safety assured state - one where the component will contribute to system level safety assurance. E.g a SL in the isolating state.

Functional state - a state which is necessary in order for the system to provide its function, e.g the Firing Unit in an operable state.

Failed state - one where safety cannot be assured, either due to damage to the component or it cannot be assured to have safe behaviour, e.g. an ERB in a damaged state.

Undesired state - the ultimate consequence of the system, which is necessary for operation but avoided from the perspective of safety. E.g. a detonator once in a fired state.

The states described above have been presented in decreasing order of safety assurance, with the exception of the Functional and failed states which can be seen as having

an equal level of safety assurance. State changes can be seen as either an increase or decrease in safety assurance (e.g. an ERB failing is seen as a decrease, whereas a FU capacitor becoming inoperable is an increase). A strong link can potentially have a number of states, as shown in Figure 5.7.

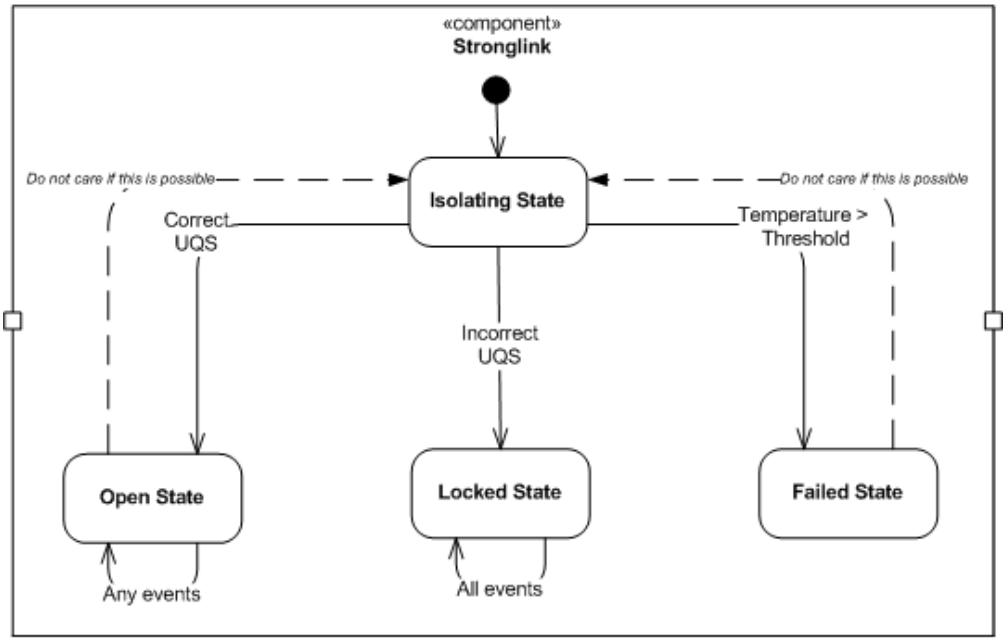


FIGURE 5.7: Example states of a strong link.

The strong link has 4 states: the Isolating State (in which safety is assured), the Failed State (in which safety is not assured), the Open State (in which safety is not assured) and finally the Locked State (in which safety is assured). Transitioning events are defined for each state change after the initial state has been entered, where a given stimulus causes this state change it has been labelled appropriately. In the locked state the component will remain in this state with all events. In the open states and failed states it may be possible to return to the isolating state, however, this is an increase in safety assurance and is positive.

5.2.2.3 Completeness of the State Machine Model

Typically the parts are required to form a complete *finite* state machine are:

1. A finite set of states
2. A start state
3. An input alphabet

4. An output alphabet
5. Transition functions
6. Output functions

In terms of the state machine view of a component as described previously in Section 5.2.2.2, dependent upon whether the component falls into the category of a source, sink or propagator/transformer of energy it may have a set of inputs and/or outputs (for a sink the output set is effectively empty, for a source the input set is effectively empty). A start state of the component should be defined and all other states should be known (see Figure 5.7). Transition functions are the events labelled to represent a change from one state to another, the output function is the mapping between any inputs and outputs.

5.2.2.4 Identifying Viable Lower Level Assertions

Table 5.2 shows each of the low level assertion types and the potential conditions that can arise for each type. For example when asserting attenuation, one must consider whether it is achieved at all times or conditionally as only some scenarios will be realistic. The *used* column in Table 5.2 shows whether each type of low level assertion is reasonable to use within a safety theme, it shows that three types are realistic and eight others which are feasible. This provides an exhaustive list of lower level assertions which would be used when developing a safety theme. In the next subsection these limited assertion types are presented as templates, which constrain the way in which the assertion must be written.

TABLE 5.2: Potential types of low level assertion which could be used.

Assertion	**Conditions**	**Description**	**Used**
Attenuation	Always	The component is assumed to be perfect which is not realistic	No
	In one state	The component also has failed or functional states	Yes
	For range of input	The component may only attenuate for some hazards, e.g. LAC	Yes
Incompatibility	*All* states of a hazard generating component are incompatible with *all* states of the vulnerable component	The vulnerable component has no weakness which could be exploited (i.e. no compatible state)	No
	All states of the hazard generating component are incompatible with *at least one* state of the vulnerable component	Acceptable if the hazard generating components are always incompatible	Yes
	At least one state of hazard generating component is incompatible with *all* states of the vulnerable component	The vulnerable component has no weakness which could be exploited (i.e. no compatible state)	No
	At least one state of hazard generating component is incompatible with *one* state of the vulnerable component	Conditional incompatibility (often used for inoperability)	Yes
State change	Initial state	Describes which state the component starts in	Yes
	State change conditions	Describes the ways in which the component can change state	Yes
	Irreversible state change	The component is unable to return to any previous states once in a safety assured state	Yes
Race	Order of state changes	Components must change state in a given order	Yes

5.2.2.5 Template Assertions

TABLE 5.3: Exhaustive set of assertion templates.

Low level assertion	**Template #**	**Assertion Template**
Attenuation	1	When (Element X) is in (State S) it shall attenuate outputs of (Element H *(in one state)*)
	2	When (Element X) is in (State S) it shall attenuate outputs of (Element H *(in one state)*) between (Thresholds T1 and T2)
Incompatibility	3	Output of (Element X) shall be incompatible with the vulnerabilities of (Element Y) when (Element Y) is in (State S)
	4	When (Element X) is in (State Sx) its output shall be incompatible with the vulnerabilities of (Element Y) when (Element Y) is in (State Sy)
State change	5	(Element X) shall initially begin in (State S)
	6	(Element X) shall only change from (State Sx1) to (State Sx2) given stimulus (V)
	7	Once (Element X) is in (State Sx2) it shall not change state again given any stimulus.
Race	8	(Element X) shall change from (State Sx1) to (State Sx2) before (Element Y) shall change from (State Sy1) to (State Sy2)

A contribution of this thesis is a syntax for writing assertions for a safety theme. Table 5.3 presents an exhaustive set of templates which cover all four low level assertion types. This table is an extension from one of the authors publications, Paper B.2 shown on page 128, specifically Table 2. An addition to the table since its publication is the inclusion of an initial state assertion. The templates are presented in a similar format to the EARS templates developed by Mavin et al. [91]. The EARS templates themselves could not be used in this instance because they would not capture behaviour of all of the 3I's.

5.2.2.6 Discussion

The templates are provided to force assertions to be specified in a precise manner, in order to meet the project requirements upon the individual assertions (as defined in Chapter 3 in Figure 3.3). The templates are formulated in a way such that they can be interpreted by both humans reading a specification document (meeting SYS8) and building upon the initial research work of this project the assertions have been developed in a manner that allows them to be modelled formally for automatic analysis (meeting SYS6 and SYS5). Many attributes of good requirements were described in Chapter 4, Section 4.2.1, each of which are discussed with reference to these new assertion templates.

Atomic Only one component (or pair of) is defined per assertion (meeting SYS4) e.g. incompatibility between a number of components would require the use of a template for each.

Verifiable Each of the assertions can be tested against either a model or a physical implementation of the system (meeting SYS6).

Implementation free Types of component will be defined as part of developing a safety architecture, however, no specific detail about how they will achieve such requirements is written at this stage of the life cycle. None of the template assertions refer to *how*, they only specify *what* (meeting SYS7).

Unambiguous It is clear which component performs the actions, no other interpretations can be made from this e.g. Element X attenuates. The conditions are made explicit (e.g. X is incompatible with Y when Y is in State S. Therefore this is the only time it can be assured for this assertion, other claims may exist to assert incompatibility in a different state). This meets the requirements of SYS5.

The assertions in the templates contribute towards one aspect of the second set of project requirements (upon an entire safety theme) by addressing the trait of consistency.

Consistent In no way can each individual template assertion contradict itself (however uses of a combination together may result in impossible scenarios, however, it should become clear if a component specification includes two contradicting templates, which meets aspects of SYS2)

One may note that when using an assertion template for attenuation, the level to which energy must be attenuated by is not explicitly stated. This is because other assertions will be used in conjunction with the attenuation assertion to form a complete specification. Energy must be attenuated to a level where the residual output is incompatible with any vulnerable components, hence a measure of performance of an attenuation assertion is not included. These lower level assertions are not used individually and relationships between them are necessary. They are composed together to form a single top level assertion. In Section 5.3 these relationships are investigated and patterns are identified.

A final addition to the templates in Table 5.3 is the use of a 'Via statement'. When asserting incompatibility between components there may be concern that other components within the same exclusion region may be able to transform the energy within the region to something compatible with a vulnerable component, sometimes this can be intentional (e.g. the arming chain discussed in Section 2.2.1 on page 14) or on other

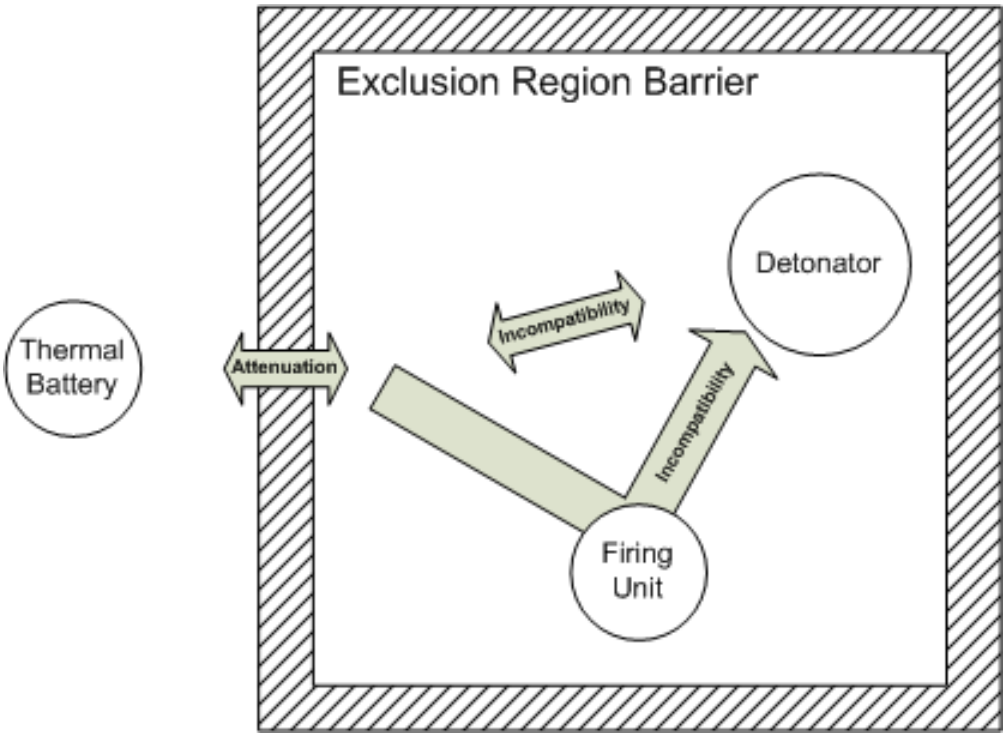


FIGURE 5.8: An example where a via clause is necessary in an incompatibility assertion.

occasions it may be undesired. The example in Figure 5.8 shows the output of the ERB (when in the isolating state) must be incompatible with the detonator (what is not shown is that this requires the detonator to be in an operable state). The output of the ERB must also be incompatible with the detonator if energy flows via the Firing Unit. Therefore template #3 can be used with an additional clause to state the flow of energy through the chain.

> "When the *ERB* is in the *Isolating State* its output, *'after flowing via the Firing Unit'*, shall be incompatible with the vulnerabilities of *the Detonator* when *the Detonator* is in the *Operational State*"

5.3 Creating a Complete Safety Theme

In the previous section a set of template assertions were presented which fell into categories of the four low level assertion types: attenuation, incompatibility, state change and race. Table 5.1 described a mapping between the top level assertions (which are currently used as best practice for specification) and these four lower level types. This mapping alone does not provide enough detail for the author of a safety theme to translate a top level assertion into the appropriate templates in a way that allows a complete

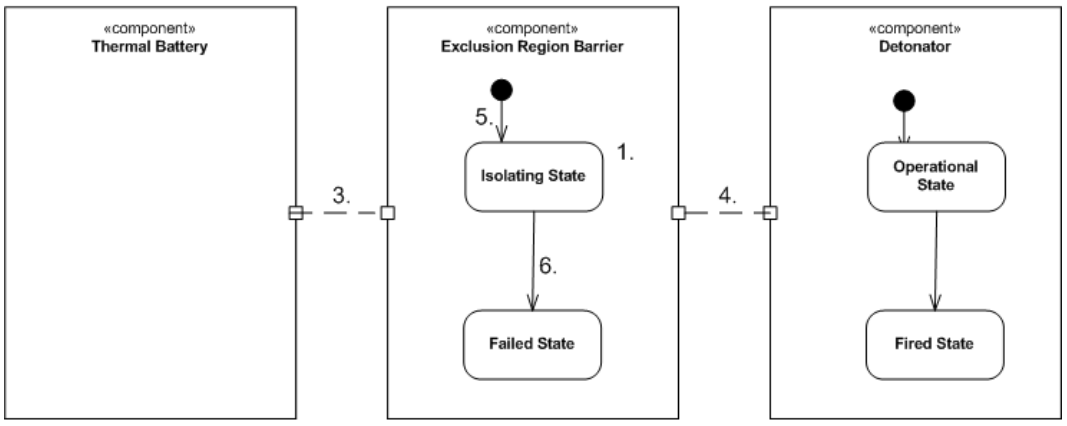
safety theme to be developed with repeatable results, as they need to be used in combination a particular way. In this section a set of patterns is presented which allows a safety theme author to specify a top level assertion in terms of template low level assertions.

The patterns describe relationships between lower level assertions in order to achieve one assertion at the top level and are grouped by the four top level assertions. In total 12 patterns have been identified: 4 for isolation, 3 for incompatibility, 4 for inoperability and 1 for race. In the following subsections each of these patterns are described with examples of how each pattern relates to the topology. The patterns only describe *realistic* architectural options and do not include any which are not achievable.

5.3.1 Isolation

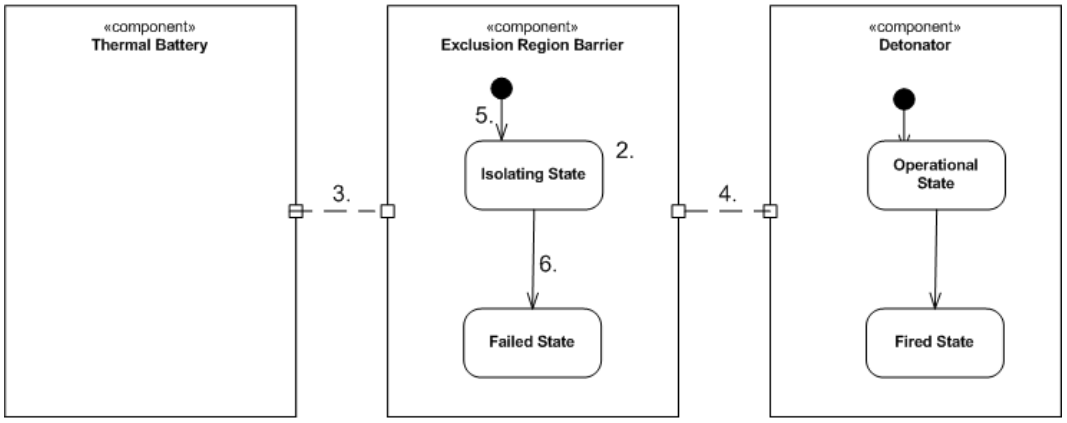
When asserting isolation it should be done so with realistic expectations. As previously discussed, no component will be able to perfectly isolate energy such that no residual energy will be able to flow from one region to another. Therefore a number of low level assertions are used in combination to assert this in a realistic manner. Table 5.4 shows the 4 patterns used for isolation which show different combinations of low level assertions that must be used in conjunction in order to make an assertion of isolation which considers all of the factors that fall within the scope of the isolation assertions and no further. It could be possible to consider the conditions upon other components which help contribute to safety beyond that assertion, however, this approach is only used to translate top level assertions into lower level assertions, not identify all assertions necessary within the system.

TABLE 5.4: Patterns to define the possible isolation assertions.

Pattern Number	**Description**
P1	This pattern is used when the isolating component does so in one state (potentially for all hazards). This requires use of the following templates: **T1** - Attenuation is argued in one state. **T4** - The attenuated output should be incompatible with the vulnerable component. **T5 and 6** - The components' initial state and state transitions are considered. **T3** - Any external hazard generating components are always incompatible with the vulnerabilities of the isolating component.
e.g. The ERB provides isolation in a given state. Always incompatible with the hazard generating component.  	
P2	This pattern is used when the isolating component does so in one state between certain thresholds of hazard (e.g. a LAC). This requires use of the following templates: **T2** - Attenuation is argued in one state between certain thresholds. **T4** - The attenuated output should be incompatible with the vulnerable component. **T5 and 6** - The components' initial state and state transitions are considered. **T3** - Any external hazard generating components are always incompatible with the vulnerabilities of the isolating component.

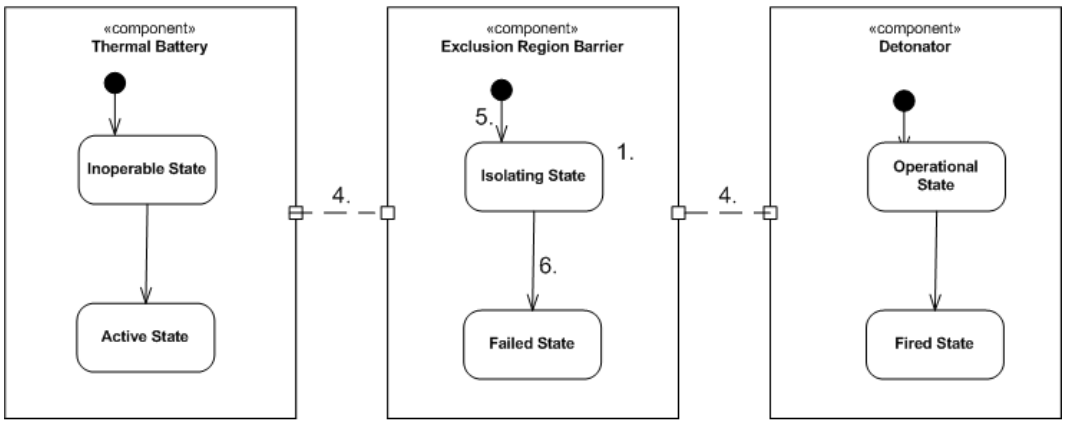
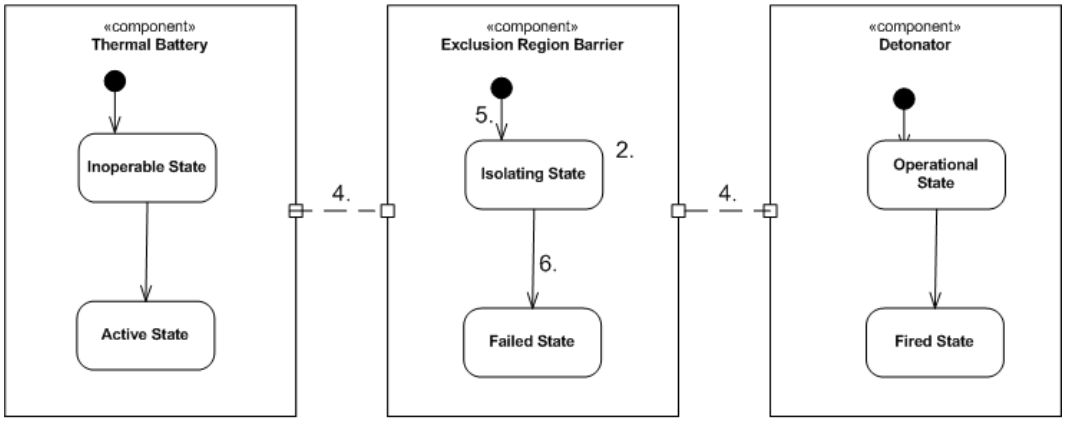
Continued on next page

Table 5.4 – *Continued from previous page*

Pattern Number	**Description**
e.g. The LAC provides isolation in a given state (up to a certain threshold). Always incompatible with the hazard generating component. 	
P3	Used when the isolating component does so in one state (potentially for all hazards). Any external hazard generating components are incompatible with the vulnerabilities of the isolating component, when the hazard generating component is in a particular state. This requires use of the following templates: **T1** - Attenuation is argued in one state. **T4** - The attenuated output should be incompatible with the vulnerable component. **T5 and 6** - The components' initial state and state transitions are considered. **T4** - Any external hazard generating components are incompatible with the vulnerabilities of the isolating component in a given state.
e.g. The ERB provides isolation in a given state. Incompatible with the hazard generating component in one state.	

Continued on next page

Table 5.4 – *Continued from previous page*

Pattern Number	**Description**
	
P4	Used when the isolating component does so in one state between certain thresholds of hazard (e.g. a LAC). Any external hazard generating components are incompatible with the vulnerabilities of the isolating component, when the hazard generating component is in a particular state. This requires use of the following templates: **T2** - Attenuation is argued in one state between certain thresholds. **T4** - The attenuated output should be incompatible with the vulnerable component. **T5 and 6** - The components' initial state and state transitions are considered. **T4** - Any external hazard generating components are incompatible with the vulnerabilities of the isolating component in a given state.
e.g. The LAC provides isolation in a given state (up to a certain threshold). Incompatible with the hazard generating component in one state. 	

Pattern 1 in Table 5.4 shows the example of an Exclusion Region Barrier which isolates energy from a Thermal Battery reaching a Detonator. Using Pattern #1 to make an assertion that the ERB isolates in one state results in the following template assertions being used:

(a) Template 1 - When the ERB is in the Isolating State it shall attenuate outputs of the Thermal Battery

(b) Template 4 - When the ERB is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

(c) Template 3 - Output of the Thermal Battery shall be incompatible with the vulnerabilities of the ERB when the ERB is in the Isolating State

(d) Template 5 - The ERB shall initially begin in the Isolating State

(e) Template 6 - The ERB shall only change from the Isolating State to the Failed State given stimulus >Threshold

The example shown is one of the more commonly used patterns, alternatively pattern 2 could be used where assertion template #4 would be selected instead of #3 if the thermal battery could only be argued to be incompatible in one state. Less commonly used patterns would be #3 and #4, which can be used to describe attenuation by a Lightning Arrestor Connector (as described in Chapter 2, Section 2.2.3.3).

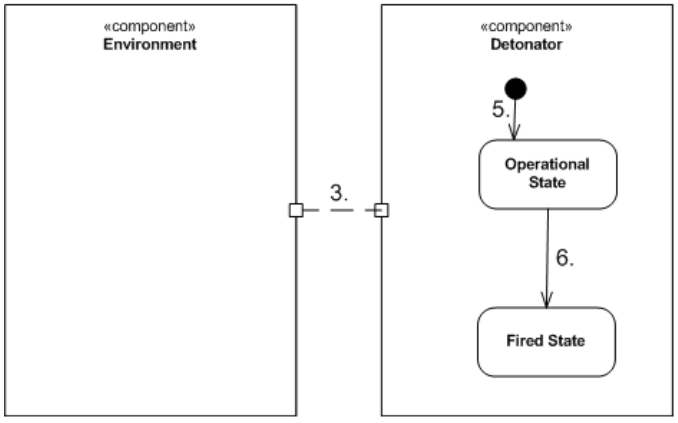
Multiple uses of the same template may be required (depending upon the architecture), with the exception of the initial state for each component. In this case for patterns 1 through 4 the safety theme author may be required to assert incompatibility between the vulnerabilities of an ERB from multiple hazard generating components, whilst also asserting that its output is incompatible with the vulnerabilities of other component inside the exclusion region. Similarly a multiplicity exists when a component has more than one state. The exception to these multiplicities is that each component will only have one initial state, however, every state change of the component must be defined when multiple states exist (for example a strong link could have states for: isolating, functional, failed or locked).

5.3.2 Incompatibility

Incompatibility forms the foundation of the 3I's, it is used as a low-level assertion for each of the 3I's with the exception of race. Although the other top-level assertions

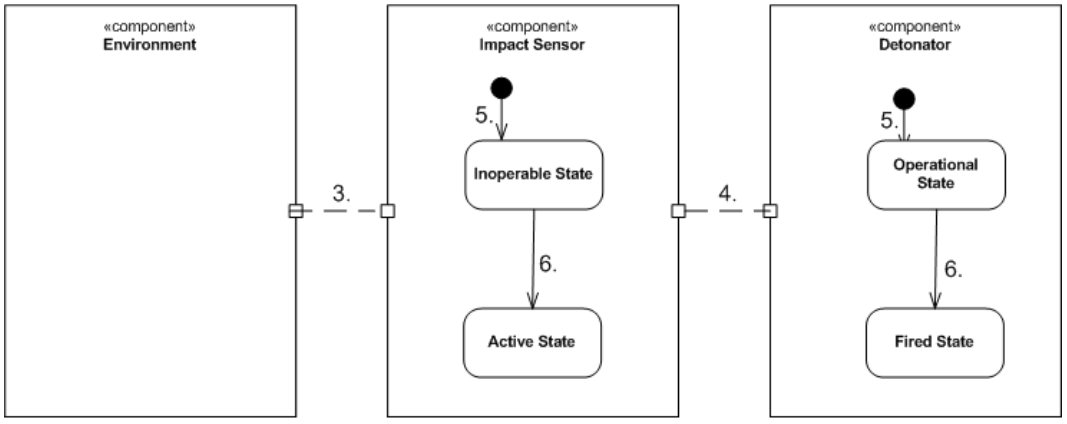
utilise incompatibility it is often asserted alone. The ideal use will be that certain hazard generating components within the system are always incompatible with vulnerable components. The vulnerable components will again be expected to have more than one state and vulnerabilities which cause this undesired state change. Three patterns have been identified for incompatibility assertions, these have been described in Table 5.5.

TABLE 5.5: Patterns to define the possible incompatibility assertions.

Pattern Number	**Description**
P5	This pattern is used when two components are always incompatible. This requires use of the following templates: **T3** - Incompatibility is argued between two components for all states of the hazard generating component. **T5 and 6** - The components' initial state and state transitions are considered.
e.g. Energy produced by the Environment is always incompatible with the vulnerabilities of the Detonator.  	

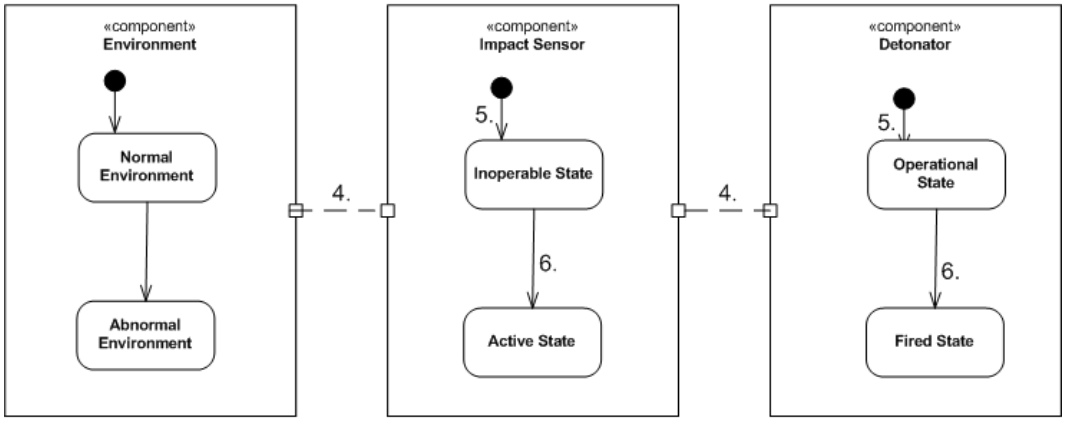
Continued on next page

Table 5.5 – *Continued from previous page*

Pattern Number	**Description**
P6	This pattern is used when two components are incompatible when the hazard generating component is in one state. This requires use of the following templates: **T4** - Incompatibility is argued between one state of a hazard generating component and the vulnerable component. **T5 and 6 x2** - The initial state and state transitions of both the hazard generating component and the vulnerable component are considered. **T3** - Any other hazard generating components in the environment are always incompatible with the vulnerabilities of the hazard generating component.
e.g. The Impact Sensor is incompatible with the vulnerable component in one state.  	

Continued on next page

Table 5.5 – *Continued from previous page*

Pattern Number	**Description**
P7	This pattern is used when two components are incompatible when the hazard generating component is in one state. This requires use of the following templates: **T4** - Incompatibility is argued between one state of a hazard generating component and the vulnerable component. **T5 and 6 x2** - The initial state and state transitions of both the hazard generating component and the vulnerable component are considered. **T4** - Another hazard generating component in the environment is only incompatible with the vulnerabilities of the hazard generating component in a single state (e.g.only in normal environments, not in operation).
e.g. The Impact Sensor is incompatible with the vulnerable component in normal environments.  	

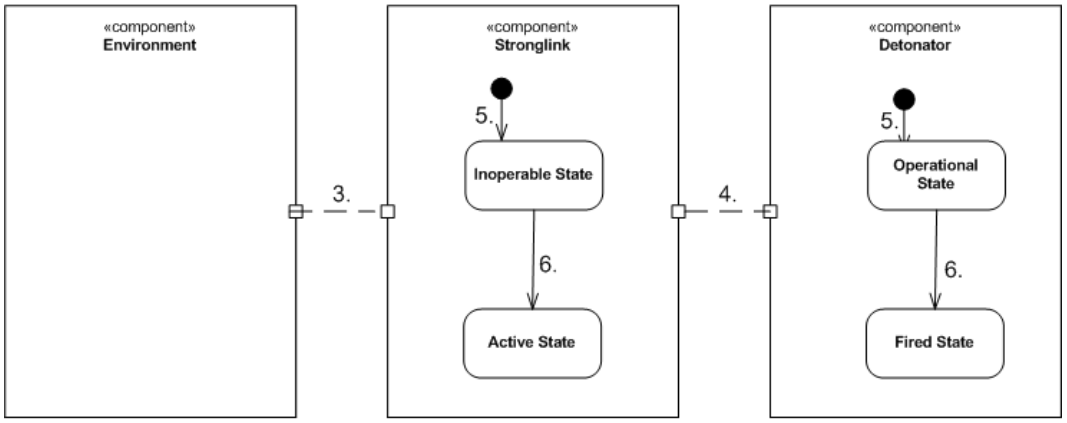
Patterns 6 and 7 are demonstrated with an example where an Impact Sensor (IS) which is expected to produce an output upon the shock of missile impact (i.e. it has a state where shock has been experienced leading to it producing an output and its initial state is that no compatible energy is produced). This example was identified from a publication by Hansen [56].

5.3.3 Inoperability

A state change of a Firing Unit component from a functional state to a safety assured state is a common example of an inoperability assertion. This type of assertion can also be used when the vulnerable side of a pair of components becomes inoperable, for example Li et al [87] and Ekman et al [37] present a safety architecture where the detonator contains a thermal weak link, typically due to Insensitive High Explosives as discussed by Elliott [38]. Elliott also presents many types of strong link, one of which is the interrupted transformer model where energy is transferred between regions via transformer coils. Hansen [56, p. 8] presents this type of strong link in more detail describing the timing of when the transformer is made operable. This means the strong link is typically inoperable and will become operable when necessary for authorised operation.

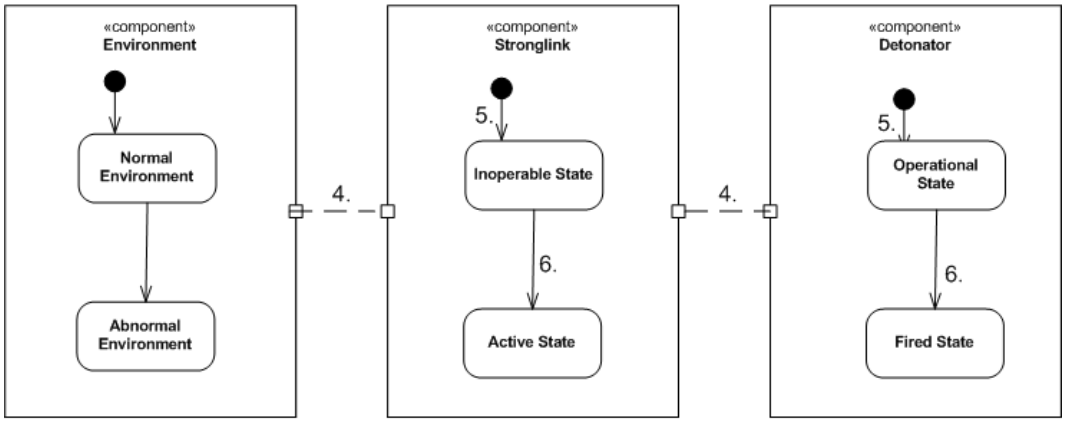
In terms of viewing the system as a state machine, inoperability can either be achieved by initially being in a safety assured state (e.g. an inoperable strong link) or it can become inoperable given an abnormal environment (e.g. capacitor weak link or detonator) and transition from a functional state to a safety assured state. Each of these options are presented in Table 5.6 as a set of 4 patterns. The first three patterns (#8-10) relate to a scenario where the hazard generating component becomes inoperable, the final pattern (#11) describes as scenario where the vulnerable component changes from a "functional state" to a "safety assured state". Additional patterns for inoperability exist, but provide repetition of the three patterns used for incompatibility and therefore they have not been repeated in this list. This is not a surprising result as Table 5.1 showed that the two approaches overlapped in the way they asserted state change and incompatibility templates.

TABLE 5.6: Patterns to define the possible inoperability assertions.

Pattern Number	**Description**
P8	This pattern is used when the hazard generating component is initially inoperable. This requires use of the following templates: **T4** - Incompatibility is argued between two components whilst the hazard generating component is in the inoperable state. **T5 and 6 x2** - The initial state and state transitions of both the hazard generating component and the vulnerable component are considered. **T3** - All environments are argued to be incompatible with the stronglink.
e.g. A stronglink is incompatible with the vulnerabilities of a detonator when the correct UQS is not presented from the environment.  	

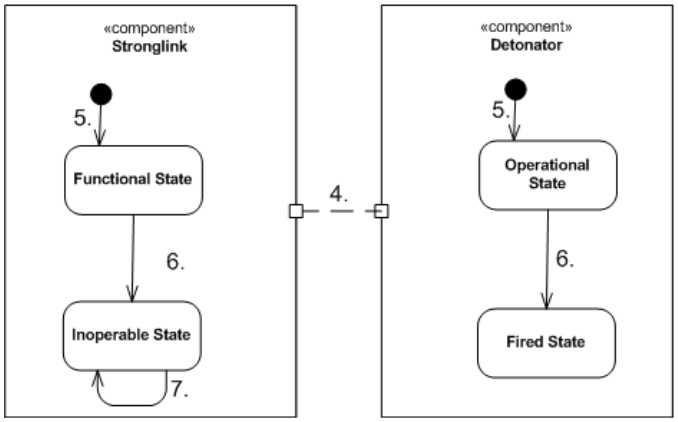
Continued on next page

Table 5.6 – *Continued from previous page*

Pattern Number	**Description**
P9	This pattern is used when the hazard generating component is initially inoperable. This requires use of the following templates: **T4** - Incompatibility is argued between two components whilst the hazard generating component is in the inoperable state. **T5 and 6 x2** - The initial state and state transitions of both the hazard generating component and the vulnerable component are considered. **T4** - Only normal environments are argued to be incompatible with the stronglink.
e.g. A stronglink is incompatible with the vulnerabilities of a detonator when the correct UQS is not presented from the environment.  	

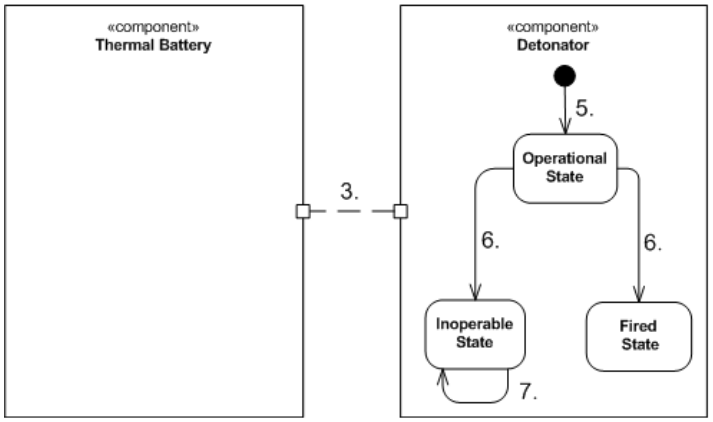
Continued on next page

Table 5.6 – *Continued from previous page*

Pattern Number	**Description**
P10	This pattern is used when the hazard generating component becomes inoperable and remains inoperable. This requires use of the following templates: **T4** - Incompatibility is argued between one state of a hazard generating component and the vulnerable component. **T5 and 6 x2** - The initial state and state transitions of both the hazard generating component and the vulnerable component are considered. **T7** - Once the vulnerable component is in its inoperable state it will remain in that state such that safety can be assured.
e.g. A stronglink is permanently incompatible with the vulnerabilities of a detonator if an incorrect UQS is recognised.  	

Continued on next page

Table 5.6 – *Continued from previous page*

Pattern Number	**Description**
P11	This pattern is used when the vulnerable component becomes inoperable and remains inoperable. This requires use of the following templates: **T3** - Incompatibility is argued when the vulnerable component is in the inoperable state. **T5 and 6** - The initial state and state transitions of the vulnerable component are considered. **T7** - Once the vulnerable component is in its inoperable state it will remain in that state such that safety can be assured.
e.g. A detonator becomes permanently incompatible with the thermal battery given certain conditions.  	

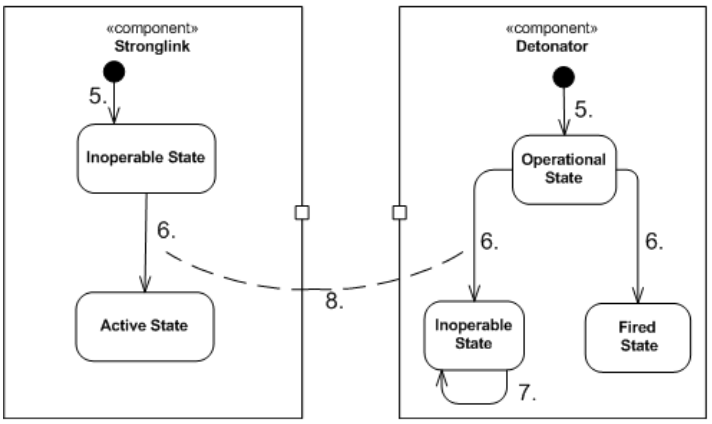
It is possible that multiple inoperable components will be designed into a safety architecture. Covan and Cooper note in [25] that many different inoperable components may be required to address different environments, such as thermal or shock (the detail of each of these environments can be found in a publication by Sanders [117]). It is common that an inoperable weak link (or perhaps multiple weak links) can be used in conjunction with a strong link to assure safety in scenarios where components may fail. This requires an assertion of race.

5.3.4 Race

A race will occur between two components when one is required to assure safety before the failure of another. When describing system components as a state machine this is a race between two state changes. The most common example is that a weak link within a firing unit (charging capacitor) will become irreversibly inoperable before a strong link fails to provide isolation. Helton et al have been actively working on such problems, identifying the probability of loss of assured safety given a number of scenarios where strong links and weak links are used in combination [58–60]. Although Helton et al are considering multiple strong link and weak link combinations, each assertion specified using the approach presented in this thesis will have an individual assertion per strong link/weak link pair. The pattern for race conditions is shown in Table 5.7.

The pre-conditions for a race assertion are that one component will start in a safety assured state and the other component is not in the safety assured state. The post-conditions of the state changes (given the environment or any other compatible stimulus) is that both components have changed state (i.e. the first component is no longer in the safety assured state and the second component is now in a safety assured state). A stage between these conditions must have occurred where *both* components are in a safety assured state. Similar work has been presented by Johnson [73], where the sequence of events for a safe access to a high-voltage handling system are discussed.

TABLE 5.7: A pattern to define race assertions.

Pattern Number	**Description**
P12	This pattern is used when two components must make state changes in a certain order to provide assured safety. This requires use of the following templates: **T5 and T6 x2** - The initial state and state transitions of both the hazard generating component and the vulnerable component are considered. **T7** - One component must transition to and then remain in a safety assured state. **T8** - The order in which the state transitions should occur is asserted.
e.g. The Detonator will change to an inoperable state before the stronglink will enter an active state.  	

5.4 The Role of Topology in Specification

It is important to consider where potential threats lie within the system during specification. The simple assertions discussed have typically been exemplified with a single hazard generating component local [3] to the component(s) to which an assertion is associated with. In reality there may be a number of regions between components (the assertions about which are in other safety subsystems and therefore their behaviour cannot be depended upon), or furthermore there may be global hazards which can affect the entire system (potentially simultaneously). An example of such global events would be temperature.

[3] local meaning they are within the same exclusion region, or a component bordering regions.

As discussed prior to presenting the patterns, each of the low level assertions has a multiplicity associated with them, with the exception of the initial state and race templates. It is not possible to specify all assertions necessary for a system without having the topology defined, as the exact details will depend on the layout of the system and components. The patterns have been designed to support this specification and show where consideration must be made as to the number of components to be argued as incompatible. Similarly the number of states per component will vary and therefore a number of state change assertions may be required. When considering the design mode arming sequence, components within the same region must be considered and the 'via' statement may be required to assert incompatibility after energy has passed through other components.

5.5 Conclusions

In this chapter two research contributions have been presented. A set of template assertions has been presented which can be used to structure assertions in a repeatable manner which is designed to improve their precision. These have been limited to a set of 8 assertion templates which capture all of the commonly used assertion types. Patterns have also been presented which provide groupings of these assertions templates which can be re-used for common scenarios based upon the topology of the system. One limitation of these patterns is that they are only designed for use with a single safety subsystem and not the *entire* safety theme, however, other methodologies have been developed within the sponsoring company which support analysis of independence between safety subsystems. Within this chapter the detail of the implementation has been presented, in the next chapter the patterns and templates are verified against the project requirements.

Chapter 6

Verification of Assertion Templates and Patterns

6.1 Introduction

In this chapter we discuss whether the implementation of assertion templates and patterns detailed in Chapter 5 meet the requirements upon the project which were defined in Chapter 3. First of all, the methods through which verification can be performed are discussed. The verification strategy for each project requirement is considered and these are performed in groups where an overlap is identified. The expected and actual results of the verification is listed for each test. In some cases the detail used for verification has been included in Appendix C to simplify the structure within this chapter.

6.2 Verification Methods and Strategy

The Systems Engineering Body of Knowledge (SEBOK) explains in Section 4 (system verification) that [107] "to verify a system (product, service, or enterprise) is to check its realized characteristics or properties against its expected design characteristics". In this case the approach developed throughout this thesis is being verified to ensure it meets the project requirements (as derived from stakeholder needs). A number of means are also defined within the SEBOK which can be used to achieve this, these are:

Inspection - which involves examination to check properties of the system or its parts.

Analysis - which involves use of logical reasoning, modelling, simulation or calculations to verify properties about the system if a real system can not be tested.

Analogy or Similarity - which involves use of evidence from similar verification activities (e.g. testing of similar systems).

Demonstration - which is used to exemplify correct use of the system or approach in a real life scenario.

Test - which involves quantitative testing of the system to determine whether requirements are met under real or simulated tests.

Sampling - which involves verification of system characteristics through a number of sample systems, parts or scenarios.

Good practice prior to verification is to define a verification strategy for each requirement that will be verified. Table 6.1 lists the verification criteria for each requirement upon the project and the different testing methods previously listed are considered for each requirement.

TABLE 6.1: Verification strategies for each project requirement.

Req ID	**Requirement**	**Verification Strategy**
SYS1	The approach to writing assertions shall be based on the 3I's principle.	*Inspect* assertions developed through use of the approach to determine which of the 3I's they relate to.
SYS2	The approach to writing assertions shall result in a complete and consistent safety theme.	*Inspect* assertions developed through use of the approach to ensure they are not contradictory. *Model* an example system and verify that if components are designed to meet all assertions that the safety theme is complete.
SYS3	The approach to writing assertions shall be repeatable on multiple projects with similar results.	*Demonstrate* use of the approach on multiple projects to verify that similar results are identified.
SYS4	The approach shall produce assertions which are atomic.	*Inspect* assertions developed through use of the approach to ensure they can not be decomposed into multiple assertions.
SYS5	The approach shall produce assertions which are precise.	*Inspect* assertions developed by applying the approach and compare them against assertions within a published example to identify how precision has been improved.

SYS6	The approach shall produce assertions which are verifiable.	*Demonstrate* that assertions developed using the approach can be tested through use of modelling.
SYS7	The approach shall produce assertions which are implementation free.	*Demonstrate* use of the approach on real projects and that the resulting assertions do not include implementation specific detail.
SYS8	The approach shall produce assertions which can be interpreted by all users.	*Demonstrate* that assertions can be interpreted as natural language or translated into a model check.

The inspections and demonstrations listed in Table 6.1 utilise both a published safety theme (using the example presented by Johnson in [74]) and also three real industry safety themes. For security reasons the detail of the industry safety themes has not been presented within the thesis, however, a generalisation of the results is presented in Appendix A, Paper B.3. The sponsor of this research has provided a letter (Appendix D) to confirm that these three projects were analysed.

6.3 Verification of the Project Requirements

6.3.1 Inspection of Assertions for the 3Is and Atomicity

6.3.1.1 Overview

Where possible populated templates from the example safety theme shown in Appendix C have been used for inspection of each template in turn to check whether each assertion is related to the 3I's and that they are written in an atomic manner. This has been shown in Table 6.2. Example assertions have been created for use to verify the assertions where templates have not been used in the example safety theme.

Table 6.2: Examples used for verification of SYS1, SYS4 and SYS8.

Template Used	**Assertion ID**	**Assertion Text**	**Applicable "I"**	**Atomic?**	**Interpretable?**
1	1a	When ERB1 is in the Isolating State it shall attenuate outputs of LV energy	Isolation	✓	✓
2	-	When ERB1 is in the Isolating State it shall attenuate outputs of LV energy up to a threshold of TBC V	Isolation	✓	✓
3	2e	Output of LV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State	Incompatibility	✓	✓
4	3d	When ERB2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State	Incompatibility	✓	✓
5	4g	SL2 shall initially begin in the Isolating State	X	✓	✓
6	5e	EXT_ISO shall only change from the Isolating State to the Failed State given stimulus of >HV	X	✓	✓
7	-	Once SL1 is in the Locked State it shall not change state again given any stimulus.	X	✓	✓
8	-	The FU shall change from the Operational State to the Failed State before SL1 shall change from the Isolating State to the Failed State	X	✓	✓

6.3.1.2 Verification Results

Table 6.3 shows the expected and actual results of verification of requirements SYS1, SYS4 and SYS8.

TABLE 6.3: Result from verification of SYS1, SYS4 and SYS8.

Requirement	Expected Result	Actual Result	Pass/Fail
SYS1	Assertions developed through use of the approach should be based upon the 3I's.	Assertions developed through use of the approach fall into the 4 categories defined as lower level assertion types. These do not all strictly map to the 3I's as inoperability is lost as a direct assertion type. The assertions related to the state machine format which maps to all of the 3I's. The use of templates resulted in the discovery that the 3I concepts over lap when the atomic assertions are considered.	FAIL
SYS4	Assertions developed through use of the approach should be atomic and can not be decomposed into smaller results.	Each template assertion is written as an individual statement and can not be decomposed further.	PASS
SYS8	Assertions developed through the approach should be interpretable by the safety analyst and also by the model based verification team.	Assertion templates developed through the approach have all been written in a way which allows any stakeholder to interpret them without ambiguity. In this inspection we do not consider whether it is possible to model a system which meets each assertion, this is covered in Section 6.3.3.	PARTIAL PASS

6.3.1.3 Discussion

Although the individual assertions themselves do not cover all of the 3I's Table 5.1 in Chapter 5 shows how each of the lower level assertions can be mapped back to the top-level assertions. In many cases the template assertions can overlap which top-level

assertion they are used by. For example, if a SL were to isolate in a given state and also argued to be inoperable, the assertions upon state changes and the initial state would overlap.

6.3.2 Inspection of Assertions to Verify Consistency

6.3.2.1 Overview

In Appendix C Section C.4 the assertions upon each component have been grouped together to provide a list of characteristics which will influence the component's design. Upon inspection of the set of assertions upon ERB1 shown on page 151 it is clear that ERB1 has two states and should be incompatible with the arming chain. The initial state is the "Isolating State" and it will remain in this unless a stimulus greater than HV energy is seen. The external components producing LV energy and HV energy are both incompatible with this state change. Figure 6.1 shows how these assertions are related to ERB1 and its interfaces with other components.

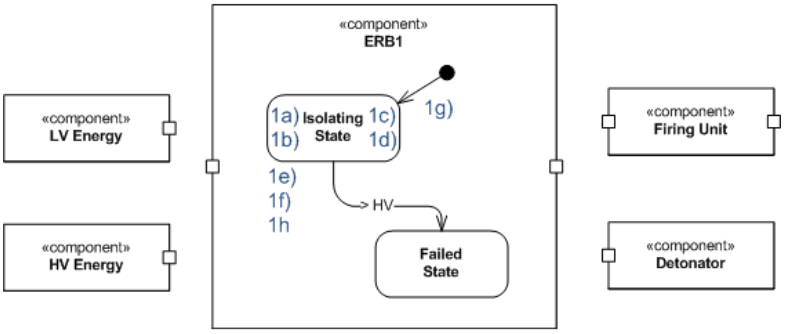


FIGURE 6.1: Assertions related to ERB1 and its interfacing components.

6.3.2.2 Verification Results

6.3.2.3 Discussion

Splitting the assertions by component helps identify if interfaces between components have been defined which can not be met. For example, if it had been asserted that ERB1 could change state with a stimulus greater than LV and had also been argued that the HV energy source would be incompatible with the state change of ERB1 (which would not be met).

TABLE 6.4: Result from verification of SYS2.

Requirement	Expected Result	Actual Result	Pass/Fail
SYS2	Assertions developed through the approach should be consistent.	Assertions developed through use of the approach are derived from a top-level assertion. It will become clear when writing assertions upon a component when assertions contradict each other. In the case of a component's initial state multiplicity is not allowed (as noted when the patterns were defined). If multiple initial states were defined in a component specification it would be apparent that this is not consistent.	PARTIAL PASS

This step of verification only results in a partial pass since the verification strategy defined for SYS2 involves verification through both inspection and model based demonstration. The entire requirement also considers completeness of a safety theme which can not easily be verified through inspection and requires use of a model.

6.3.3 Modelling Assertions to Verify Completeness

6.3.3.1 Overview

Manually checking for completeness of a safety subsystem or entire safety theme is a difficult task. The modelling work which was performed as initial research for this project is a useful tool for testing whether each safety subsystem provides complete protection if the assertions defined are met. In order to achieve this a model of the example system presented by Johnson has been developed. Two levels of model checking have been performed: firstly to ensure that each component meets the behaviour defined by each assertion upon it (which can itself comprise of multiple model checks). This is performed using the method presented in Paper B.2, Section III A, however, model checks are adapted to meet the assertions for the templates defined in this thesis (an example of these model checks is presented in Appendix C Section 5.3). Secondly, the models representing each component are composed into a model which represents the topology of a safety subsystem, upon which a model check is executed to determine whether all routes to detonation are covered by the assertions in that safety subsystem.

The model check for safety subsystems can be performed in two ways, the first is to create a model of each safety subsystem independently (i.e. just the topology of that safety subsystem). Modelling each safety subsystem this way makes the assumption that there are no dependencies between safety subsystems and that failure of a component in one safety subsystem would not have an impact upon the other. The second method is to model the entire system topology including all components (regardless of whether they are part of the safety subsystem under test), the components which are not part of the safety subsystem under test should be able to behave in their worst-case behaviour and not have an impact upon the completeness of the safety subsystem under test. The detail of the second approach is presented in Appendix B Paper B.2 Section III B. For discussion in this thesis, only the first approach is considered for simplicity as only the 3I's have been discussed throughout the thesis (not the fourth I of "Independence").

The purpose of this discussion is only to determine completeness of each independent safety subsystem. Use of the patterns is not expected to guarantee that safety subsystems are independent. To achieve this would require use of existing methodologies for design and analysis.

6.3.3.2 Model Check Results

In order to verify that the project requirements SYS2 and SYS7 have been met the model checking method has been applied to safety subsystem 1 from Johnson's example. Through application of the approach presented within this thesis 8 assertions have been identified which relate to ERB1 and 9 have been identified which relate to SL1. The model used for this verification is included in Appendix C Section C.5, along with annotations which explain the model. The desired results from a number of model checks and evidence of their execution (from model checker FDR2) are shown in Table 6.5. The full listing of these model checks is provided in Appendix C in Section C.6.

A final model check is also required to determine the completeness of the entire safety subsystem (once it has been verified that each component fulfils the assertions upon it). The expected result is that no paths exist where energy can flow from either the HV or LV source which result in compatible energy reaching the Detonator. This is model checked against a specification which allows any sequence of events except detonation to occur, if detonation can not occur the result is *true*. The result of executing this model check are shown in Figure 6.2. The detail for this model and specification is shown on page 164.

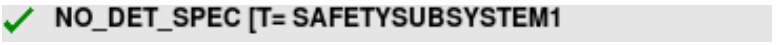


FIGURE 6.2: FDR output from model checking safety subsystem 1.

6.3.3.3 Verification Results

Table 6.6 shows the requirements applicable to this section of model based verification and the results of verification of project requirements SYS2, SYS4, SYS6 and SYS8 are shown. The "Lines of the Model" section refers to line where these model checks are performed in Appendix C Section C.6. Where appropriate the model is annotated for explanation.

TABLE 6.5: Results of model checking assertions.

Assertion	**Desired Result**	**Evidence of Result and Lines of the Model**	**Pass or Fail**
1a	At least one must fail	Shown in lines 246-250 ✓ OUTPUT_GT_NONE [T= NONE(LV_ENERGYHazards)[output<->input]ERB1_ISO ✗ OUTPUT_GT_RES [T= RES(LV_ENERGYHazards)[output<->input]ERB1_ISO ✗ OUTPUT_GT_LV [T= LV(LV_ENERGYHazards)[output<->input]ERB1_ISO ✓ OUTPUT_GT_HV [T= HV(LV_ENERGYHazards)[output<->input]ERB1_ISO ✓ OUTPUT_GT_VHV [T= VHV(LV_ENERGYHazards)[output<->input]ERB1_ISO	Pass
1b	At least one must fail	Shown in lines 252-256 ✓ OUTPUT_GT_NONE [T= NONE(HV_ENERGYHazards)[output<->input]ERB1_ISO ✗ OUTPUT_GT_RES [T= RES(HV_ENERGYHazards)[output<->input]ERB1_ISO ✗ OUTPUT_GT_LV [T= LV(HV_ENERGYHazards)[output<->input]ERB1_ISO ✗ OUTPUT_GT_HV [T= HV(HV_ENERGYHazards)[output<->input]ERB1_ISO ✓ OUTPUT_GT_VHV [T= VHV(HV_ENERGYHazards)[output<->input]ERB1_ISO	Pass
1c	True	Shown in line 259 ✓ OUTPUT_GT_VHV [T= VHV(HV_ENERGYHazards)[output<->input]ERB1_ISO	Pass
1d	True	Shown in line 260 ✓ DET_INCOMP_SPEC [T= ((ERB1_ISO[output<->input]FU)[output<->input]DET)	Pass
1e	True	Shown in line 262 ✓ ERB1_INCOMP_SPEC [T= LV_ENERGY[output<->input]ERB1	Pass
1f	True	Shown in line 263 ✓ ERB1_INCOMP_SPEC [T= HV_ENERGY[output<->input]ERB1	Pass

1g	Both tests true	Shown in lines 267-8 ✓ ERB1_INITIAL_SPEC [T= ERB1_UNDER_TEST ✓ ERB1_UNDER_TEST [T= ERB1_INITIAL_SPEC	Pass
1h	True	Shown in line 270 ✓ ERB1_SC_SPEC [T= ERB1_ISO_SC[|{input.veryhighv}|]STOP	Pass
2a	At least one must fail	Shown in lines 272-276 ✓ OUTPUT_GT_NONE [T= NONE(LV_ENERGYHazards)[output<->input]SL1_ISO ✗ OUTPUT_GT_RES [T= RES(LV_ENERGYHazards)[output<->input]SL1_ISO ✗ OUTPUT_GT_LV [T= LV(LV_ENERGYHazards)[output<->input]SL1_ISO ✓ OUTPUT_GT_HV [T= HV(LV_ENERGYHazards)[output<->input]SL1_ISO ✓ OUTPUT_GT_VHV [T= VHV(LV_ENERGYHazards)[output<->input]SL1_ISO	Pass
2b	At least one must fail	Shown in lines 277-281 ✓ OUTPUT_GT_NONE [T= NONE(HV_ENERGYHazards)[output<->input]SL1_ISO ✗ OUTPUT_GT_RES [T= RES(HV_ENERGYHazards)[output<->input]SL1_ISO ✗ OUTPUT_GT_LV [T= LV(HV_ENERGYHazards)[output<->input]SL1_ISO ✗ OUTPUT_GT_HV [T= HV(HV_ENERGYHazards)[output<->input]SL1_ISO ✓ OUTPUT_GT_VHV [T= VHV(HV_ENERGYHazards)[output<->input]SL1_ISO	Pass
2c	True	Shown in line 282 ✓ DET_INCOMP_SPEC [T= SL1_ISO[output<->input]DET	Pass
2d	True	Shown in line 283 ✓ DET_INCOMP_SPEC [T= ((SL1_ISO[output<->input]FU)[output<->input]DET)	Pass
2e	True	Shown in line 285 ✓ SL1_INCOMP_SPEC [T= LV_ENERGY[output<->input]SL1	Pass

2f	True	Shown in line 286 ✓ SL1_INCOMP_SPEC [T= HV_ENERGY[output<->input]SL1	Pass
2g	Both tests true	Shown in lines 290-291 ✓ SL1_INITIAL_SPEC [T= SL1_UNDER_TEST ✓ SL1_UNDER_TEST [T= SL1_INITIAL_SPEC	Pass
2h	True	Shown in line 293 ✓ SL1_SC_SPEC1 [T= SL1_ISO_SC[|{input.veryhighv}|]STOP	Pass
2i	True	Shown in line 295 ✓ SL1_SC_SPEC2 [T= SL1_ISO_SC[|{input.uqs1}|]STOP	Pass

TABLE 6.6: Results from verification of SYS2, SYS4, SYS6 and SYS8.

Requirement	**Expected Result**	**Actual Result**	**Pass/Fail**
SYS2	A component can be modelled to meet all of the assertions upon it to demonstrate that all assertions are consistent. When all assertions within a safety subsystem are met their composition can be demonstrated to produce a complete safety subsystem.	It has been demonstrated in Table 6.5 that all assertions are passed for ERB1 and SL1, therefore all assertions developed are consistent for that component. Figure 6.2 shows results of a test upon safety subsystem 1 of the example safety theme to demonstrate it is complete. This is a partial pass as other aspects of requirement SYS2 have been tested by inspection.	PARTIAL PASS
SYS4	Tests should be atomic. Only one attribute is tested for each assertion, regardless of how many individual model checks are necessary to achieve this.	Each template tests one trait per assertions and this can comprise of multiple model checks (depending upon the system model).	PASS
SYS6	The plausibility of each assertion can be verified through use of a model, prior to the lengthy development time for physical hardware.	Results from using a model to verify assertions upon two components in an example safety theme are shown in Table 6.5.	PASS

SYS8	The template assertions can be interpreted by the design team who can use formal modelling techniques to verify assertions.	Template assertions are written in such a manner that allows model checks to be performed to verify each assertion. This is only a partial pass as it does not consider how all users would interpret the assertions and not every requirement template has been demonstrated through model checking from the example (as some templates are not used).	PARTIAL PASS

6.3.4 Demonstrating Results from Multiple Projects

6.3.4.1 Overview

To verify SYS3 the approach has been applied to Johnsons's published safety theme example and safety subsystems from three industry projects to determine whether the results gathered are repeated. Within Paper B.3 in Appendix B the results are generalised as ten different safety subsystems. The approach used involved taking each of the assertions in the safety subsystem, determining which top-level assertion they were related to and then applying the appropriate pattern. Once the entire set of assertions had been identified the new assertions were compared against the original ones from the safety theme. Table 4 of Paper B.3 in Appendix B shows a generalisation of the types of mapping identified between these two sets of assertions.

The results were common for most safety subsystems (as shown in Figure 5 of Paper B.3), as in most cases the original assertions were decomposed into multiple lower level assertions (showing a one-to-many relationship). Where no mapping existed the assertion from the safety theme could not be mapped to one of the 3I's, in this case the safety theme under analysis included implementation specific detail about how a component would operate. This detail was not included into the safety theme developed through this approach. No many-to-one relationships existed when mapping from the original assertions in the safety theme to the template based assertions. This is further supporting evidence that project requirement SYS4 has been met as each of the new assertions is atomic and can not be mapped to multiple existing assertions. Finally, in cases no

mapping existed from the newly identified assertion back to the safety theme, in this case the assertion had been missed from the safety theme. Figure 6 in Paper B.3 shows that there were a number of these occurrences, which can be argued to support meeting project requirement SYS2, in that the approach helps support production of a complete safety subsystem.

6.3.4.2 Verification Results

Table 6.7 shows the expected results from the project requirement and the actual result identified through application to multiple projects. The result of whether the test has passed or failed is also listed.

TABLE 6.7: Results from verification of SYS2, SYS3, SYS4 and SYS7.

Requirement	**Expected Result**	**Actual Result**	**Pass/Fail**
SYS2	Assertions developed through use of the approach should be composed to produce a complete safety theme.	Results from application of the approach to multiple projects showed that in many cases new assertions were identified which were missing from the original safety theme. This helps support completeness of a safety theme, however, can not guarantee that the assertions combined covers every potential path of energy flow in the system. The approach is also only applied to one safety subsystems at a time and not the entire safety theme.	PARTIAL PASS

SYS3	Repeatable results should be identified from application to multiple projects.	Results from application of the approach to multiple projects showed that the only time the new assertion templates could not be used was where implementation specific detail had been included within the safety theme. The results showed a common trend in an increase in the number of assertions generated and that any which had been missed would be identified through the approach by using patterns.	PASS
SYS4	Assertions developed through use of the approach should be atomic and can not be decomposed into smaller assertions.	Application to multiple projects showed that no mapping existed where the new template assertions could be decomposed.	PASS
SYS7	Assertions developed through use of the approach should not include any implementation specific detail.	Assertions from each safety subsystem which had no counterpart developed through use of the approach were not related to the safety concepts necessary for a safety theme. This often included implementation specific detail which was not included as an assertion through use of the approach.	PASS

6.3.5 Comparing Precision of Assertions Against an Existing Approach

6.3.5.1 Overview

This section is intended to verify project requirement SYS5, which specifies that assertions developed through use of the approach should be precise. In order to verify this requirement assertions presented throughout the paper by Johnson in his example safety theme [74] are used for comparison against those developed using the approach presented in this thesis. Table 6.8 shows a comparison between these two assertions (where ID's are given for new assertions they reference those in Appendix C, Section C.3).

Overall the results show that the new assertions have been specified to be more precise, however, in some cases, (e.g. the first row of Table 6.8) multiple assertions are required to provide the entire picture, assertions 2a, 2d and 2e together provide the information. A second observation, from assertion 2d, shows that in the original assertion there is no consideration of how energy can be made compatible with the Detonator after flowing via the Firing Unit. Also, in many cases the precision is improved by considering the architecture of the system. Assertions are written for a specific layout and this should be a foundation of the safety theme, therefore assertions like 2e can be used to improve the vague statement that "nearby electronics" will not mimic the enablement signal. A final observation is that using the new templates it becomes clear when certain assertions are designed to be met, e.g. a SL only attenuates when it is in the 'Isolating State'.

TABLE 6.8: Verification of requirement SYS5.

Original Assertion	**Comparable New Assertion**	**Improvement in Precision?**
Safety break must reduce the voltage sufficiently	(2a) - When SL1 is in the Isolating State it shall attenuate outputs of LV energy	✓ (alongside 2d or 2e)
Input voltage must not exceed limits of safety break	(2e) - Output of LV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State	✓

Residue of safety break must be less than no operation voltage of FU	(2d) - When SL1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator, when the Detonator is in the Operational State	✓
Residue of safety break must be less than no operation voltage of Det	(2c) - When SL1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State	✓
Safety break state remains safe in absence of correct enablement signal	(2i) - SL1 shall only change from the Isolating State to the Open State given stimulus of UQS1	✓
Nearby electronics does not mimic enablement signal	(2e) -Output of LV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State	✓
No enablement signal generated, in absence of correct environment	*A further component of the "environment" should be included into the example to cover this assertion. However, (2e) is the closest example.*	✓
SB2 Isolates	*Multiple assertions would be identified from an applicable pattern.*	✓ by multiple assertions
ERB2 Isolates	*Multiple assertions would be identified from an applicable pattern.*	✓ by multiple assertions
Det Incompatible with LV	(6a) - Output of LV Energy shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State	✓

Nothing else in the system generates HV	*Multiple assertions like the previous line would be used to represent the system topology*	✓ by multiple assertions
System isolated from external HV	(5a) - When EXT ISO is in the Isolating State it shall attenuate outputs of HV energy	✓

6.3.5.2 Results from Verification

Table 6.9 shows the expected and actual result from verification of requirement SYS5.

Table 6.9: Results from verification of SYS5.

Requirement	**Expected Result**	**Actual Result**	**Pass/Fail**
SYS5	Assertions developed through use of the approach shall be precise and describe the context in which they are applicable.	Assertions developed through use of the approach overall provide an increase in precision in the way they are specified. In some cases multiple assertions must be considered together to provide the full picture of how well one must be performed (e.g. attenuation to a level which results in incompatibility). This provides an overlap with SYS2 in that the set of assertions must be complete.	PASS

6.4 Summary of Results

In this section the results of the verification activities is summarised. In most cases a single test could be performed to demonstrate that implementation of assertion templates and patterns met the project requirements. However, for three requirements multiple tests were performed which contributed to a single overall test result. In the case of SYS4, this requirement was verified three different ways, whereas for requirements SYS2 and SYS8 different aspects of each requirement were demonstrated per test. Table 6.10 shows a summary of results from verification of all of the project requirements. The section where each requirement is verified is referenced.

TABLE 6.10: A summary of results from verification of all project requirements

Project Requirement	**Result**	**Applicable Section(s)**
SYS1	FAIL	Section 6.3.1
SYS2	PARTIAL PASS (From multiple partial passes)	Sections 6.3.2, 6.3.3 and 6.3.4
SYS3	PASS	Section 6.3.4
SYS4	PASS	Sections 6.3.1, 6.3.3 and 6.3.4
SYS5	PASS	Section 6.3.5
SYS6	PASS	Section 6.3.3
SYS7	PASS	Section 6.3.4
SYS8	PASS (From two partial passes)	Sections 6.3.1 and 6.3.3

6.5 Conclusions

The first major result from verification to discuss is the failure of project requirement SYS1 - that the approach to writing assertions shall be based upon the 3I's principles. The approach developed through this research work has been underpinned by the concept of the 3I's, the test result showed that for each template it was not possible to directly reference one of the 3I's. However, it can be argued that there will always be traceability back to the 3I's since use of the patterns requires the safety analyst to determine which of the 3I's is being asserted. If this decision is traceable then the safety theme will still comply with the needs of the regulators who request use of the 3I's approach in JSP538 [96].

Another consideration which is pertinent to these results is for project requirement SYS2 - that the approach shall result in a complete and consistent safety theme. The verification results described in Table 6.10 were gathered from testing individual safety subsystems and not an entire safety theme. The partial pass status is because this has been achieved for safety subsystems and not the full theme. This approach has not been developed to consider the completeness of an entire safety theme and independence between set of assertions. This is a limitation upon the scope of the project, however, independence can by analysed through use of existing methods and this can be applied to safety subsystems which have been verified to be complete through use of this approach.

Overall the implementation meets the project requirements with the exceptions discussed above, however, where these have not been met the results have been justified. In the next section we consider whether the contributions to knowledge in this thesis fulfil the hypothesis of the research, how the stakeholder requirements from the project are met and compare the results of the project to existing literature.

Chapter 7

Discussion and Conclusions

7.1 Overview

The contribution presented within this thesis has been an approach to specification of assertions about safety of high consequence arming systems. Development of this approach has led to many individual contributions to knowledge, firstly a new interpretation of the 3I's has been presented within Chapter 5 which is underpinned by a state machine based approach. This new interpretation has resulted in definition of templates which constrain specification of safety assertions to a limited set, all of which are realistic, atomic and feasible. A final contribution is a set of re-usable patterns for specifying sets of assertions (based upon an original assertion in terms of the 3I's).

In Chapter 6 one of the methods for verifying that the project requirements were met was through demonstration of how the process algebra Communicating Sequential Processes can be used to model the flow of energy through components of an arming system and then how these components can be composed to form a system model. This model was used to verify whether assertions generated through use of the approach produced a complete safety subsystem and that assertions upon the different components of a system are consistent.

An application of the approach has been presented in Appendix C where assertions of one safety subsystem from an existing safety theme have been defined using templates derived from applicable patterns for the system topology. A model of these safety-critical components has also been shown in the appendix and the behaviours of the components and the entire safety subsystem have been verified through model checks. The components of the case study system were modelled in a way such that they would meet each of the assertions (which in this case was possible as all assertions were consistent).

7.2 The Wider Process

This approach alone is not intended to fulfil the entire systems engineering process, it is simply one step developed to aid the process (by utilising some of the philosophy of systems engineering). To use the approach the top-level assertions, system topology, and an understanding of the possible states of each component are required.

Derivation of assertions and design of the system are expected to be performed using the design methodology Johnson presented in [74], which would then be provided as an input into the approach presented in this thesis. The entire safety theme can then be maintained through life with use of the Pentagon /S/ process as presented by D'Antonio et al. in [4]. Detailed modelling and analysis can be performed using Finite Element Models and a model based safety assessment approach presented by Carlson and Jones in [20]. This shows that the approach described within this thesis is not a stand alone approach, it interfaces with a number of existing published techniques and has been developed to address the specific issue of precise specification of assertions.

7.3 Specification of Assertions

In Chapter 2 the issue of imprecise specification was discussed, the state of the art approach to definition of a safety theme presented no structured or repeatable format through which assertions should be defined. The work presented within this thesis introduces a format in which assertions should be written and relationships between different assertions. Johnson had acknowledged that the concept of isolation decomposed into lower-level assertions of incompatibility and attenuation. The author of this thesis has taken this decomposition and identified similar relationships for incompatibility, inoperability and race - whilst also introducing a state machine format.

Use of the templates makes individual assertions simpler to read and understand (in Paper B.2 it was acknowledged that assertions in an industry scale project were difficult to distinguish when trying to develop model checks). Use of patterns between assertions allowed existing safety themes to be defined in terms of these new templates (also identifying that some had been missed). A finding of interest was that a safety theme from one of the industry projects analysed contained implementation specific detail about *how* a component would achieve the assertions upon it.

ISO 15288 [72] suggests that the implementation phase of the life-cycle is used to transform "specified behaviour, interfaces and implementation constraints into fabrication

actions that create a system element according to the practices of the selected implementation technology". This suggests that until that point in the life-cycle the specification should not define *how* a system is implemented, but capture *what* the system should do.

Written natural and structured language are more helpful for documenting the system in a paper based form, whereas model based systems engineering (E.g. UML, SysML) and formal modelling (e.g. Z, VDM) require more expertise in understanding the assertions, which need to be understood by many stakeholders of the system throughout the life-cycle.

7.4 Modelling and Model Checking

An additional development through this research work, but not a core contribution of this thesis, is a method for modelling the system of interest using CSP. This is a development upon the current UK state-of-the-art approach in this area, which was presented in Paper B.1. This Matlab based tool inspired the approach to modelling the system topology presented in this thesis (i.e. identifying which components interface between exclusion regions). In comparison, the models developed in CSP are more abstract since they only address a limited set of ranges of energy level (e.g. lowv, highv etc.) rather than defining set values for input and output of energy. The CSP model, however, adds far more complexity by incorporating a method to verify independence between safety subsystems (as presented in Paper B.2 Section III C). A benefit of the abstract model and use of CSP is the compositional nature of processes. Model checks have been performed against the processes which represent each component, the behaviour of which are then re-used through composition into a system model. This means verified behaviour of each component is then known to exist within the system model.

Modelling of the topology is a tricky manual part of the approach, however, it forces the safety analyst to understand the ways in which energy is able to flow in and out of different regions. A benefit of this is demonstrated in Appendix C Section C.1.2 where it is noted that in the safety theme presented by Johnson, an extra interface exists between the region containing the Firing Unit and the region containing the Detonator. The impact of this interface being missed is that this design of the barrier between these regions is not safety-critical and may not be designed in a way which would isolate energy. This would have the resultant impact of failure of one safety subsystem.

Although the model used for verification of the requirements upon the project (as presented in Appendix C) was only used to verify properties of each individual safety subsystem, a model check has also been defined to verify independence. Findings shown

in Paper B.2 from application of this approach to a model and verify a safety theme for an industry project hit limitations due to state space explosion issues with the size of the model. Future work to improve the scalability of this approach is presented in Section 7.8.

7.5 Testing the Hypothesis

In the introduction to this thesis, the following hypothesis was stated:

> It is possible to repeatedly produce sets of precise safety assertions about arming systems through use of an approach based upon patterns and requirements engineering techniques.

Based upon the results of the example used for verification and the applications to industry projects, this has been achieved through use of a set of constrained natural language templates for writing assertions. Not only are the assertions written in a form which can be understood simply, the amount of template assertions has been limited to a minimal set. This set has been derived such that it is expressive enough to specify assertions necessary upon components of the system without introducing any which are not physically implementable (as noted in Table 5.2 on page 61). Through use of these templates on four safety themes (in Paper B.3), the only occasion in which assertions did not fit these templates was when implementation specific detail has been included (it can be argued that these are not real assertions of a safety theme).

Another argument why precise safety assertions have been specified is that they can be translated into a machine checkable form. This has been achieved through definition of a set of template model checks which map directly to the assertion templates. This limits the types of model check that are used, removing the issues encountered when attempting to express some of the original assertions of the safety theme of an industry project (as noted in Paper B.2).

The scope of the patterns based approach presented in this thesis is limited to application to individual safety subsystems, this does not affect whether the hypothesis for this research has been met. Development of a complete safety theme can be achieved through Johnson's design and analysis method to define a safety theme prior use of the patterns.

7.6 Significance of Results

Through use of the approach it has been demonstrated that realistic assertions can be specified and that the behaviours of individual components can be verified to meet these through use of the modelling technique. This can provide a higher level of confidence in a safety theme over the current manual process and use of the existing Matlab tool, since the approach presented in this thesis considers verification of both the assertions upon each component and the resultant impact on each safety subsystem.

7.7 Validation of Stakeholder Requirements

The Project Management Body of Knowledge (PMBOK) [68] defines validation as "the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders". A letter from the research sponsor is included in Appendix D in which it is confirmed that stakeholder requirements have been met. In Table 7.1, paragraphs of this letter are referenced against the stakeholder requirements.

7.8 Further Research

7.8.1 Scalability of the Model Checking Technique

Further work would be required to allow a safety themes of any size to be specified, modelled and verified using CSP and FDR2. Alternatively, other modelling techniques could be considered which could further exploit the state based nature of how assertions are defined. The issue that accompanies such a change in modelling technique is that an alternative must support expressive reasoning about the model in order to test both: the individual assertions and independence between safety subsystems.

As noted in Chapter 3, the order in which this research work was performed meant that the template assertions and patterns were identified as a solution to the issues identified during development of the modelling and model checking stages. Had these stages been reversed, CSP may not have been selected as the ideal language to model the system with.

TABLE 7.1: Validation of the stakeholder requirements.

Stakeholder Requirement	**Validation**
Meet applicable JSP538 requirements	New approach is traceable to the 3Is - sponsor letter page 2, paragraph 1.
Be able to achieve repeatable results	Similar results have been identified from multiple projects, as summarised in the sponsor letter, page 1, paragraph 4. This is also agreed on page 2, paragraph 1.
Be able to write unambiguous assertions	In the sponsor letter page 2, paragraph 1 it is acknowledged that the research work and new view of the 3Is has become clearer than the original assertion types.
Be able to produce a complete safety theme	The new approach allows the completeness of a safety subsystem to be tested, as summarised in the sponsor letter, page 1, paragraph 5.
Be able to verify assertions	The sponsor expects that it would be possible to verify these requirements later in the system life cycle, as shown in the sponsor letter, page 1, paragraph 6.
Be able to investigate design alternatives	The new approach does not allow assertions to be written which contain implementation specific detail, as summarised in the sponsor letter, page 1, paragraph 4.
Be able to interpret assertions	The sponsor has acknowledged that assertions are both readable and can be modelled.
Be able to model and verify properties of the system	The sponsor explains that a technique has been presented through which a model can be created and properties of the system can be verified. As shown in the sponsor letter, page 1, paragraphs 5 and 6.

7.9 Conclusions

This thesis has addressed the lacking precision in specification of assertions presented in a safety theme. The resulting approach has involved specification of assertions in a constrained format, modelling of components and safety subsystems using Communicating Sequential Processes, followed by an approach to model checking of safety properties using FDR2. The approach has been demonstrated to be useful for deriving assertions upon components at a low level and in a complete form. Modelling helps identify whether a set of behaviours enforced upon a component are all physically possible and requires the topology of a system to be well understood and defined. Application of the approach to an example safety theme has unearthed an issue which could result in missing safety critical requirements (i.e. a port from the firing unit should isolate energy on its output). From the perspective of the author this work has provided a useful set of steps which could be used as either an entire approach or as stand alone processes. In combination

they can be used to prove that a set of assertions within a safety theme meet the needs of the regulatory body. The author believes that a safety theme is a focal point in the design of a high consequence arming system, as such it is crucial that it is developed correctly and communicated in a way which will support the entire systems engineering life cycle. Many parties within the organisation sponsoring this research have been interested in the potential adoption or further expansion of the work within this thesis. Potential developments are to redesign the modelling technique with use of compression techniques within FDR2. The research sponsor has confirmed in a letter of validation in Appendix D that the assertion specification approach presented in Chapter 5 is being used upon a current demonstration project. The outputs of this research have resulted in a subtle change in the way of thinking within the domain, with potential adoption of the specification technique as part of common practice. In summary, this research has resulted in a stepping stone for safety analysts to begin developing a safety theme in a more formalised manner.

Appendix A

CSP Overview

A.1 Creating CSP Models

A.1.1 Background

Communicating Sequential Processes (CSP) is a process algebra, first presented by Hoare [61], it is predominately designed for modelling and reasoning about concurrent computer systems. Its common uses are in the software domain, the most common of which would be to model and prove whether a software system meets its specification. A benefit is that CSP has translation rules from specification to a model (such as from Z [126]) and from the model to implementation code (such as occam [112] and Handel-C [9]). A benefit being that the model has already been proven to meet the specification prior to implementation.

Beyond the software engineering domain CSP has been used to model and reason about logical systems or concepts, such as: games (e.g. peg solitaire [113]), the logic of protocols [118], and even modelling biological phenomena such as the interactions between blood platelets (with use of aspects from the B language in [119]).

A.1.2 Modelling Features

A CSP model is composed of a number of processes, which engage in events. Processes may be able to choose between events it can engage in, and to compose processes into a larger system model the processes can be interleaved or synchronised upon certain events. Each of these aspects will be addressed throughout this section:

A.1.2.1 Events

The alphabet of the system, described as α, lists all of the possible *events* that the system is able to perform. This describes the system in its entirety (it is not able to perform any events which are not within α). Although the alphabet describes a limited set of events, this does not mean the system will actually perform them all, however these are the only events that will be seen when observing the system. In the traditional form of CSP presented by Hoare in [61] detailed timing of events was not considered, where necessary the order of start and end of particular events are used to describe the order in which they occur (allowing overlapping events to be recognised). An extension of Timed-CSP exists (as presented by Reed and Roscoe [109]), however, only a subset of the available features of standard CSP are used within this thesis, since the concepts within a safety theme are of an abstract nature and do not relate to detailed timing (only ordering) of events.

Events are defined in a CSP model as *channel* communications. Each event name is written as lower case, for example:

```
channel a, b, c, d
```

Events may also have suffix types, defined by datatypes which can be existing types (e.g. integers) or newly defined ones. For example:

```
Datatype Newtype = type1, type2
channel a, b : Int
channel c, d : Newtype
```

A.1.2.2 Processes

In order to describe the 'behaviour pattern of an object', as it was referred to by Hoare, the collections of events and the sequence in which they can be performed must be defined. To achieve this, components are modelled through the use of *processes*. A process is defined in a CSP model using a capitalised name, processes `A` and `B` are noted in the following example along with a special processes, `STOP` and `SKIP`.

$$A = a \rightarrow B$$
$$A = a \rightarrow A$$
$$A = a \rightarrow STOP$$

The first example is a process `A`, which engages in event `a` and then behaves as process `B` (the behaviour of which has not been defined in this example). Recursive behaviour can also be defined by a process which then behaves as itself (i.e. `a` then `A`). `STOP` is a process which represents a component which is no longer willing to engage in any events in alphabet α, therefore the process will *terminate*. This means the component will no longer be able to perform other events (which in some safety critical systems could cause failure to function). This can be referred to as *deadlock* and is referred to in Section A.2.1. A process named `SKIP` refers to successful termination of the process, where no more events are required to be performed.

A.1.2.3 Choice

When modelling component behaviour, it is possible that a choice between a number different behaviours is possible. CSP offers two different types of choice: external and internal. External choices are those where the choice is controlled by the environment therefore there is control over the events which could occur.

$$A = a \rightarrow A$$
$$\Box\ b \rightarrow B$$

An internal choice is a non-deterministic choice between events which is not controlled by the environment. The component may or may not engage in any of the events offered. For example:

$$A = a \rightarrow B$$
$$\sqcap\ b \rightarrow STOP$$

Throughout this thesis only external choice is used, this is to ensure that the only influencing factors of the system are the other components. Components are synchronised upon events, which allows them to engage in the same event simultaneously. Within this thesis synchronisation is used to represent the flow of energy around a system.

A.1.2.4 Synchronisation

Components are able to engage in events and operate concurrently (as with in real life). If components have different alphabets and can behave without interacting between the two (or more) of them, the component processes can be *interleaved*. This means both `A`

and `B` are able to engage in any events without communicating.

$$SYSTEM = A \,|||\, B$$

If the components are required to communicate (or both engage in the same event) the processes are synchronised upon such events. For example `A` and `B` may both be able to engage in event `a`. Event `a` will only occur, however, when both components are able to engage in this behaviour.

$$SYSTEM = A \,|[\, a \,]|\, B$$

When two components are required to engage in events which are not the same for both processes (for example the output of A is synchronised with the input of B) it is possible to use the linked parallel operator, as shown:

```
SYSTEM = A [a <-> b] B
```

Essentially, the linked parallel requires two different events (`a` and `b`) which will become synchronised together. To do this the events are renamed to be the same event, the result of this is that their names become hidden and cannot be seen in an output *trace* of the model. The event will be shown as a *_tau* event.

A.1.2.5 Traces

The events that can be performed by a model will have one or more sequences of events which it can perform (growing more complex the more choices and synchronisations available). This sequence starts with an empty set where no event has been performed and then includes every combination of events that can occur according the behaviour of the system. For example:

$A = a \rightarrow b \rightarrow c \rightarrow SKIP$
$B = c \rightarrow b \rightarrow a \rightarrow SKIP$
$SYSTEM = A \,|[\, c \,]|\, B$
$traces(SYSTEM) = <>, < a >, < a, b >, < a, b, c >, < a, b, c, b >, < a, b, c, b, a >$

Since both processes `A` and `B` terminate successfully with `SKIP` the set of traces of `SYSTEM` is complete. Process `B` would not be able to engage in any event until process `A` is also ready to engage in event `c`. If the behaviour of `A` and or `B` had been recursive, these

traces would still be valid, however, more options would become available once event `c` had occurred.

A.1.2.6 Example Model

An example model is shown in Appendix C, Section C.5. This uses the constructs described through this section to model the components of an example system as processes. This model is annotated with explanation of what each part of the model means. A description of the modelling technique is shown in Section II of Paper B.2.

A.2 Model Checking

It is possible to prove whether a model satisfies certain conditions using tool support. This involves a design model (i.e. the model of system or individual component of interest) and also a specification. Model describe a system in terms of event sequences in which it is able to engage, abstracted to the particular trait of the system of interest which in this case are the properties regarding safety. The specification will describe the desired behaviour of that component or system. A specification can be as detailed as necessary to perform checks of the behaviour of the system and can be written in either: a form that limits that a component or system such that it must *always* be able to perform certain events, or alternatively the specification may be written such that the component or system will *never* exhibit certain behaviours. Such investigation of the system is achieved through refinement checking. Throughout this thesis the model checker FDR2 [44] is discussed, Armstrong, Goldsmith, Lowe, Ouaknine, Palikareva, Roscoe and Worrell explain in [6] that it is the industry standard for model checking CSP and is still maintained.

A.2.1 Refinement

As discussed in Section A.1, a CSP model can be seen as a set of possible traces which it is able to perform. These describe all of the possible sequences of events which a process can engage in as a finite state automata, where all possible sequences of events are represented. A trace refinement check between two CSP processes requires a `SPECIFICATION` process (i.e. the ideal behaviour of the system) and a `DESIGN` process (i.e. what the system being built will do).

The specification is a *trace refinement* of the design if all of the possible traces within the specification are exhibited within the design. i.e. the design does not contain any

traces which are not in the specification. It is not necessary for *all* of the traces of the specification to be met by the design, it just may not exhibit extra behaviours than those specified. Such a model check is performed by use of the following *assert* statement:

```
assert SPECIFICATION [T= DESIGN
```

This type of model check is a guarantee that certain conditions are not broken. It is *not*, however, a method of testing whether other conditions are guaranteed (i.e. functional or reliability based criteria). This makes trace refinement beneficial for verification of safety properties about a system. Liveness properties are more difficult to determine and can be performed through use of further model checks for *failures*, *refusals* and *divergences*, all of which are discussed by Murray in [98]. These further model checks are acknowledged, but not used within this thesis as the simple properties desired of the system can be determined through trace refinement.

A.2.2 Deadlock

Another commonly used model check in FDR2 is the deadlock check. If a process behaves according to STOP, or is waiting to engage in a synchronised event that is not possible, the process can be described as deadlocked. This is not of concern with this model, since if no more events can be performed and safety properties still hold the system will not exhibit unsafe behaviour.

A.2.2.1 Model Checking Example

Model checks are performed on the example system in Appendix C, these are listed in Section C.6.

Appendix B

Research Publications

B.1 Paper 1

Towards Tool Support for Design and Safety Analysis of High Consequence Arming Systems Using Matlab

Dan Slipper[1], Wilson Ifill[2], Gordon Hunter[2], Roger Green[2], Richard Johnson[2], and Alistair A. McEwan[1]

[1] Department of Engineering, University of Leicester
[2] AWE, Aldermaston

Abstract. High consequence arming systems are designed to prevent unwanted external (or potentially internal) energy flowing to a critical component without authorisation. The hazard analysis of such systems can be a slow and difficult manual process, potentially repeated in various lifecycle phases or on multiple design options. This paper details a simulation tool under development at AWE to provide a fast and repeatable analysis process. The simulation generates a set of possible paths along which different energy types could potentially propagate through the system. Behaviour identified by the tool can support the design of the system and selection of an architecture where safety is assured whilst still providing reliability. We present an outline of the model development process, results from its use with a case study and demonstrate the advantages over manual analysis. A number of limitations of the current implementation are discussed, we then propose future work aimed at alleviating some of these issues.

Key words: safety analysis, matlab, simulation, propagation

1 Introduction

High consequence arming systems such as nuclear weapons pose a potential hazard throughout their operation. *High consequence systems* can be defined as 'those where failures can cause catastrophic results' [1], this can apply to many industries where the combination of different energy types within a system can be dangerous, for example the chemical process industry. However for our specific industry, (and the nuclear power industry) the catastrophic results would be dispersal or nuclear yield. Within both normal and abnormal environments there are system *hazards* (potential conditions that can cause injury [2]). Where possible these hazards should be removed from the system during the design phase to maintain assured safety, when necessary components present a hazard, the impact of such should be reduced through the implementation of safety features. The effects of external *insults* (physically measurable phenomena with the potential to detrimentally affect the system [3]) must also be considered to ensure that the system remains adequately safe through all environments.

The Nuclear Weapon regulations and design principles set out in JSP 538 [4] and JSP 372[5] provide requirements for design and state that in abnormal environments it is not required for the system to be operational. Evidence that these stringent requirements are fulfilled must be provided before a system can be commissioned for service. Live testing of a full nuclear weapon system is no longer viable due to the Comprehensive Nuclear Test Ban Treaty [6]. Therefore sufficient evidence from sub-system tests, design rationale and modelling must be provided to demonstrate that the implemented safety features are reliable and adequate.

A consideration whilst providing evidence of a safe system design is the behaviour of individual components. Many of the components can potentially affect energy passing through them and produce a hazard. These hazards can be caused by conversion of given energy into a another type, or a change of magnitude by amplifying or reducing it. Whilst sound engineering and scientific judgement is required to determine the effectiveness of different designs, the variety of threats to components coupled with the complexity of the systems can make it difficult to effectively determine whether all areas of potential concern to the engineer have been investigated.

Software based modelling methods can ease the analysis of these areas by providing the user with a fast, thorough and repeatable process. With use of an object-oriented approach [7] a model can be easily developed and modules re-used through the use of abstract classes. This paper introduces a Matlab [8] simulation tool designed as the initial step towards a methodology to aid design and analysis through modelling. The simulation automates a manual analysis technique used at AWE [3], by identifying the possible paths from an insult through to the critical component of a system. The paths along which energy could potentially travel are then analysed, producing an expected 'safe' or 'unsafe' result for each, dependent on whether appropriate safety features exist. The contributions of the paper are details of a simple, re-usable simulation technique built upon manual methods for analysis and case study evidence demonstrating how this method improves upon the existing process. The technique does not aim to provide proof or evidence that a path is sufficiently safe, however it has been developed to aid the safety analyst in identification of points of concern within a system design.

The contents of the paper are structured as follows; Section 2 describes the requirements of the simulation tool. Section 3 details the model components. Section 4 presents the results of an example application analogous to the high consequence system of interest. Section 5 presents a discussion of the results, highlighting the limitations of the tool which could be seen as potential for future work. Section 6 proposes how this future work will be achieved and development of an overarching methodology. Section 7 provides some conclusions about the tool and methodology.

2 Requirements and Context

The tool is intended to aid the design and analysis of a system to ensure it meets the appropriate safety requirements. In order to do this the safety analysis team identified a number of requirements for a tool, which shall:

- Address abnormal environments with non-design mode connectivity.
- Cater for electrical and non-electrical hazards and transformations.
- Only address loss of assurance of safety and not performance.
- Use a deterministic approach that combines critical safety functions and inherent hazards and vulnerabilities of the system.
- Produce a list of the unsafe pathways through the system (some of which could be potentially missed using the manual method).
- Aid design of the system and selection of appropriate safety features (or assertions that given scenarios cannot happen).

The structure of the tool is described in Figure 1 using the Unified Modelling Language [9] (UML). It can be seen from this diagram that the system can be modelled as: a group of abstract components, a defined topology, and a defining scenario.

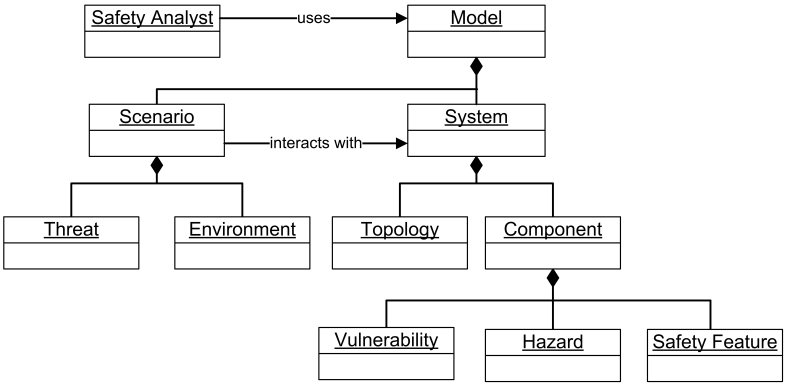


Fig. 1. UML model of the system context.

The safety analyst uses a system model to identify potential paths around the system (in a defined scenario). Energy is introduced to the system from an insult (or threat) and then passed between components by affecting vulnerabilities which in turn produce a hazard. Each component will have multiple response functions, each of which represents a vulnerability and could potentially present a hazard to other system components. The components within the system have a topology (or layout) which is ultimately defined by the scenario under test. The

requirements of this tool were to analyse abnormal environments in which all components could potentially interact with each other, this interconnectivity is only limited by the use of safety features, which can stop energy flowing within known limits. The safety features use the concepts of Isolation, Incompatibility or Inoperability as described further in [10], [11], [12], [13] and [14].

3 Modelling and Execution Process

The context shown in Figure 1 captures both the Matlab elements used to create a system model and its environment. The main elements of interest are the component models, insults and the system topology. When the model is executed it uses a path generator and insult propagator to analyse paths which are potentially unsafe. Each of these stages is described in more detail in the following sections.

3.1 Component Models

Each component (or sub-system) in the model will either be a required to provide a necessary system function, or to provide safety. For either scenario energy may be generated by, contained within, or passed through a component. Therefore each is modelled with an input and output port (P1 and P2 respectively), through which it can transfer energy to any connecting components via an insult vector (see Section 3.2). A component can have multiple responses, dependent on the type, magnitude and direction of energy present at either the input or output port (depending if it is forward flowing or feedback). Figure 2 depicts a component with two potential responses to an electrical insult.

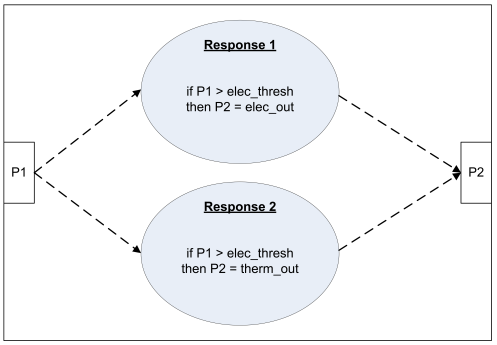


Fig. 2. An example Component with input ports and a number of internal response functions.

Although Matlab provides the ability to develop transfer functions for component responses, safety analysis is only concerned with the point at which a component can output energy, or in the case of a safety feature, the point that safety can not be assured. Figure 3 demonstrates how the behaviour can be transformed into a function where the component assures safety up to a threshold of X. At threshold Y the component is expected not to provide safety, therefore between X and Y there is a *potential* loss of assured safety.

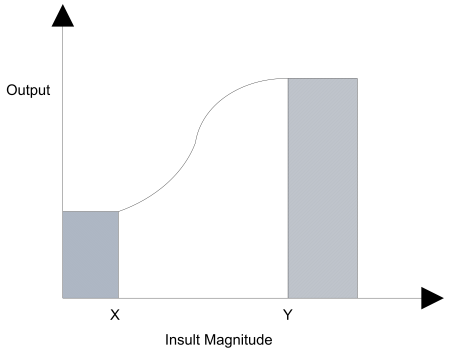


Fig. 3. Representation of a components safety assured response.

The software for these models is developed upon a rule-based approach [15], where conditional statements represent the safety assured response in Figure 3. A benefit of using an object-oriented design is that components have no hard coded values. Each instance of the component is created with its thresholds and output values as constructor arguments. For example component C1:

C1 = Component_A(240, 240, 100)

Would create an instance of Component A with an electrical input and output of 240V and a thermal output of 100 Degrees. Assigning values from the main program code allows repeatability by changing the values in a single place. In early system design stages (e.g. generating and testing architecture options) the information about component response functions would be difficult to provide accurately. The values assumed would be the estimated worst case for assured safety and based on expert judgement. The output from the model can aid the selection of appropriate components if margins of unsafe behaviour were considered. By using this iterative development process the model specification can be refined as further detailed design information becomes available, or by using results of trials on individual components. The tool allows us to model the system level response that emerges from a network of interrelated components.

3.2 Insults

To run a simulation an insult source should be provided. This is representative of the scenario of interest to the safety analyst and could involve multiple types of energy being provided at a given point. For this initial iteration of the model, the insults from the scenario can only be associated with a single component. In Matlab the insult is captured as an object containing a vector (or array) of values. This object is passed between components through p1 and p2 along the paths that are generated by the system (see Sections 3.4 and 3.5). The values in the insult vector are modified where appropriate as they pass through each component, until the the critical component is reached. The component then generates a 'safe' or 'unsafe' response. In the current implementation only a single critical component is modelled.

3.3 System Topology

To understand how system components interact the relationships between them should be modelled. This is to understand where the insult vector can be passed between components. In abnormal environments it is assumed that all components could potentially be connected to all others unless a safety feature removes this connectivity (such as an area isolated by a barrier). The system is modelled like a graph, with components represented as nodes and potential connections between them as verticies. These connections can be represented by an adjacency matrix [16].

Capturing all of the interactions in this way aids the modelling of the system in Matlab. Each row of the matrix would be translated into an appropriate line of code detailing where each component could connect to. For example:

```
I_p2 = {'A_p1'}; %Components are named with a letter. I is an insult
A_p1 = {'A_p2'}; %p1 and p2 represent input/output ports
A_p2 = {'B_p1'}; %Relationships shown by C = {'Connecting ports '}
B_p1 = {'B_p2'}; %Connecting p1 to p2 shows propagation
B_p2 = {'C_p1', 'D_p1'}; %Multiple connections can exist
C_p1 = {'C_p2'};
C_p2 = {'D_p1'};
```

3.4 Path Generation

To generate a list of all of the paths from the insult source through to the critical component, a breadth first search [17] of the network is performed. This algorithm identifies each connection from its current component then in turn repeats the process on each branching component. This generates a tree across the breadth of the graph rather than the depth. When a full chain is found that does not end with the critical component, it is removed from the list. The resulting paths are written to a file and provide the first part of the output. An example of the component search order is shown in Figure 4. Paths generated also show

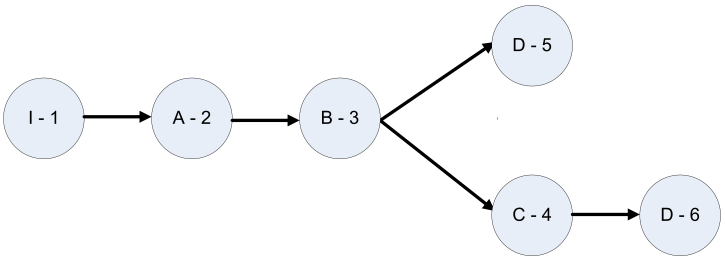


Fig. 4. Selection order from a breadth first search algorithm.

which ports the path passes through (e.g. component A would receive energy in through A_p1 and transmit it out of A_p2). The paths from the adjacency matrix connections previously shown would be:

```
I_p2    A_p1    A_p2    B_p1    B_p2    D_p1
I_p2    A_p1    A_p2    B_p1    B_p2    C_p1    C_p2    D_p1
```

3.5 Insult Propagation

To propagate the insult the simulation iterates through each of the paths generated, taking an insult magnitude that is input to the system and then calculating the change to this vector based on the component responses. The components in the system are instantiated by calling their constructor with the appropriate values (setting thresholds and outputs based on the components responses). This could look like:

```
i1  = I(50);          %electrical signal of 50V
a1  = A(40, 50, 80);  %threshold of 40V, outputs 50V, and 80 deg
b1  = B(..., ..., ...); %Appropriate types or values are given ...
```

The final component determines if it is unsafe based on the magnitudes held within the insult vector (if applicable). The resulting output is a list of statements of whether the paths are safe or unsafe, linking to the order in which they were generated.

4 Example Application

To demonstrate the use of this simulation tool and highlight the strengths in its usage, a simple case study has been devised. This case study is analogous to that of a high consequence arming system, where there is a potential hazard that should be protected from external energy sources. The selected case study is the design of a car. Modern cars have a number of safety devices both to protect the passengers and to reduce the effects of a potential impact (many are summarised

in [18]). The area of interest for the safety in this case study however, is that the fuel provides a constant hazard of fire or explosion. Although vehicles are now designed with safety of fuel tanks in mind and appropriate measures are considered, older vehicle designs have demonstrated these risks. This risk needs to be balanced along with other safety, performance and reliability requirements of the system.

An example is the Ford Pinto. This vehicle was designed with the fuel tank rear of the axle, according to [19] this was due to the limitation on boot space (a balance between performance and safety), this also limited the amount of crush space around the component. This design decision resulted in the fuel tank exploding upon impact (above a certain magnitude) to the rear of the vehicle. A number of components have been identified within the vehicle system, these could each have a number of responses to insulting energy and in some way affect the fuel tank. The connectivity of some components of interest are shown in Figure 5.

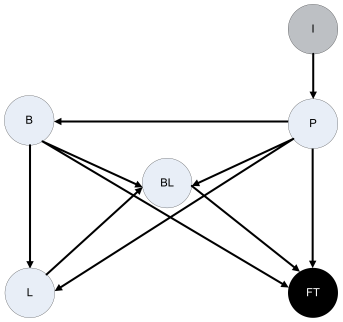


Fig. 5. Potential non-design mode connectivity of the system of interest.

For this example only 5 components from the system will be discussed, these are the: Fuel Tank (FT), Lighter (L), Brake Pedal (P), Brake Light (BL) and Battery (B). In reality, this combination of elements are unlikely to be co-located within the system, however in accident scenarios it must be ensured that they have a sufficient argument that a connection is not possible (either due to an isolating barrier, or incompatibility of energy). The scenario under test within the following section is a shock to the brake pedal mechanism, which in a drive by wire system would produce an electrical signal elsewhere.

4.1 Manual Analysis

The case study was analysed manually to identify the expected results and for verification of the model. Using the 5 components previously described, all potential paths through the system were generated by hand and compared against

the output from the simulation to validate its results. Manual analysis of the system took just under an hour, with the author having prior knowledge of the case study. The analysis highlighted 16 paths through the system, 2 of which were potentially unsafe for the scenario of interest. Notes under the arrows demonstrate the type and magnitude of energy transfer of a given type (e.g. el = electrical). Some paths are shown as an example:

$$I \underset{s=1}{\rightarrow} P \underset{el=12}{\rightarrow} L \underset{t=100,el=12}{\rightarrow} FT = UNSAFE$$

$$I \underset{s=1}{\rightarrow} P \underset{el=12}{\rightarrow} L \underset{t=100,el=12}{\rightarrow} B \underset{t=100}{\rightarrow} BL \underset{0}{\rightarrow} FT = SAFE$$

It was noted during manual path generation that possible paths could easily be missed, even when generating possibilities for a small system of 5 components. This becomes far more difficult with a larger set of components and also with more potential energy types.

4.2 Simulation Results and Analysis

The outcome of the simulation was that the 16 expected paths were generated and as with the manual analysis two of these were potentially unsafe. The simulation itself only takes a matter of seconds to execute, providing a fast method to get repeatable results. To change the component specifications and test a range of insulting values only required changes to the constructor of the insulting component. Changes to the structure are also simple and can be iterated based on results of an initial analysis.
Execution of the model with a scenario where the Brake Pedal is shocked has demonstrated that unsafe paths exist within the design. These unsafe paths are scenarios where thermal energy can propagate from the Lighter to the Fuel tank. For this particualr scenario it is possible to argue that the path identified has a very low likelihood of occurance due to the distance between components. To ensure safety however, the incorporation of a barrier with an area for electrical cables and the fuel line to pass through would remove all potentially unsafe paths *for this scenario*. Figure 4.2 shows the updates to the system, where passing the cable/fuel lines through a barrier limits some of the energy types that can be transmitted to the fuel tank.

Other scenarios also need testing to assess the overall system safety, as extra safety devices may need to be incorporated into the system to assure safety.

5 Discussion

The overall outcome from using the modelling technique is that the generation and analysis of paths with use of a software based tool is faster and more accurate than the current manual process. Removing the aspect of human error from

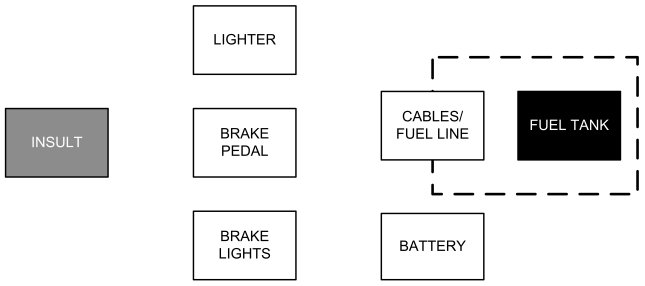


Fig. 6. Updated system with a new safety feature included.

the analysis process is an important motivation for this development. Use of the simulation tool allows repeatability of experiments with very minor changes to the code, which would require a full repetition of the analysis process if undertaken manually. Despite these advantages, the system is only in the early stages of becoming a useful tool to aid analysis and support design decisions. Ideally a number of system design options would be compared against each other to demonstrate which provides the highest assurance of safety. There are also many limitations on modelling the real world with the concepts captured through this technique, as will be discussed.

Component Responses The first limitation of the system is the component response function. These are programmed as a safety assured response and it is thought that there is little added value at this stage to have a detailed system response for a full range of possible inputs. In the current implementation an insult vector is passed from a single component to one other along the current path being analysed. In reality it is possible that multiple insults could be produced by connected components simultaneously.

Component Composition Within this implementation components are individual modules which cannot be combined to form a larger section of the system without programming their combined behaviour manually. It would be desirable to analyse the system at different levels of abstraction, allowing low level component models to be composed together to represent a sub-system model.

Distributed Insults Another limitation of the system in its current form is the way insults to the system are input into a single component. In some scenarios, the environments which the system may experience could involve multiple insults to different places. For example dropping the system could crush some components upon impact and provide a mechanical shock to others. The result of these parallel events would then propagate between the components.

Capturing Design Information The way in which component specifications are captured is a possible expansion of the technique. When developing a model

for a complex system, tools to capture the appropriate information are desirable (for example the adjacency matrix method). When capturing component responses, a structured representation of the system responses would be useful, for this the Unified Modelling Language (UML) is proposed, and discussed further in Section 6.

Design Decisions Once components required for the systems functionality have been identified, there could be a range of potential arguments or safety devices that could be used to assure system safety. Some of these options may not be the most efficient or cost effective, and part of the design process would be to select appropriate features to assure safety as efficiently as possible. Ideally the model would aid analysis of this, in its current form it only provides the set of potentially unsafe paths. Information of which components allow the most paths to pass through them or the connections closest to the critical component are of possible interest here.

6 Proposed Methodology

Future work is proposed to fulfil the limitations described in the previous section. Ideally a full methodology to aid design and analysis is desired and this should be supported by an appropriate simulation tool. This tool should aid the team selection of an appropriate safety architecture, and possibly extend to consider the reliability of a design. The use of simulation would provide the team with a robust, repeatable method of analysing the system and supporting the stringent safety requirements placed upon them. This methodology would consist of stages to capture the system information appropriately, with the use of multiple UML views. These models could then be (manually) translated into appropriate rules about the system in a language that supports simulation, concurrency and model checking. Tools of interest are Coloured Petri Nets [20] or Communicating Sequential Processes [21]. Both are being investigated as to their potential use. A process for development of such a model will be considered in order to avoid state space explosion problems, this process could involve abstraction of the system to a high level system representation, approaching the model subsystem at a time or removing know impossibilities before model development (such as removing energy types that are not compatible). All of this information will be captured in an appropriate format to present the safety analyst with a set of arguments about the system which assure safety.

7 Related Work

Related work has been published from Sandia National Laboratories where the combination of Fault Tree Analysis, Event Trees and Finite Element Models have been described in [22]. Failure modelling techniques exist with a similar concept of component specifications and responses when analysing software systems. The Failure Propagation and Transformation Notation (FPTN), developed

by Fenelon and McDermid [23], provides a notation for capturing component responses and analysing ways through which a component failure can propagate between components. This has been developed further into a calculus by Wallace [24], where different types of transfer of energy can occur (source, sinks, transformation or propagation). Our tool adds the path generation aspect to these existing methods, but utilises the existing concepts for propagation of insults. The methodology under development is expected to provide much more functionality for analysis of the system and to aid the whole design process.

8 Conclusions

The issues of speed, completeness and reliability of safety analysis with manual process can be improved through the use of software tools. Case study evidence has demonstrated that it is possible to analyse all paths through a system and identify which are of concern in a short time scale using the Matlab tool. The tool was designed to fulfil a number of requirements as stated in Section 2, the model is seen to have fulfilled these requirements, however some of the limitations we have described pose possible extensions beyond this initial requirement set. Future work aims to provide a full methodology to support design and analysis (considering reliability alongside safety), whilst with tackling some of the highlighted issues with the current tool.

References

1. J. Davis. Integrated Safety, Reliability, and Diagnostics of High Assurance, High Consequence Systems, May 2000. Electrical Engineering.
2. Department of Defense. Standard Practice for System Safety MIL-STD-882D. 1993.
3. C.R. Johnson. Methodology for Designing and Analyzing High Consequence Arming Systems. In J.M. Livingston, R. Barnes, D. Swallom, and W. Pottraz, editors, *Proceedings of the US Joint Weapons Systems Safety Conference 2009, Huntsville, Alabama*, pages 552–561, 2009. ISBN 9781617387142.
4. Ministry of Defence. JSP538 - Regulation of the Nuclear Weapon Programme, 2008.
5. Ministry of Defence. JSP372 - Approving Authority Management Arrangements for the Trident Re-entry System, 2011.
6. Comprehensive Test Ban Treat Organisation. Comprehensive Nuclear Test Ban Treaty, 1996.
7. T. Budd. *An Introduction to Object-Oriented Programming.* Addison-Wesley, 2002.
8. Mathworks. MATLAB - The Language of Technical Computing. `www.mathworks.com/products/matlab/`, 2011. [Online; accessed 15-December-2011].
9. Object Modeling Group. OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. Technical report, November 2007.
10. S. D. Spray. Deriving and applying generally applicable safety principles. In *International System Safety Conference*, Seattle, WA, September 1998.

11. J.A. Cooper and J.M. Covan. Predictable Safety in the Control of High Consequence Systems. In *3rd IEEE High-Assurance Systems Engineering Symposium*, Washington, DC, Nov 1998.
12. D. W. Plummer and W. H. Greenwood. The History of Nuclear Weapon Safety Devices. In *Conference: 34. AIAA/ASME/SAE/ASEE joint propulsion conference*, Cleveland, OH, Jul 1998.
13. G. Elliott. US Nuclear Weapon Safety and Control. In *MIT Program in Science, Technology, and Society*, 2005.
14. M.E. Ekman, P.W. Werner, J.M. Covan, P. E. D'Antonio, and E. Perry. A Thematic Approach to System Safety. In *Process Safety Progress 17:3, American Institute of Chemical Engineers*, 1998.
15. L.A. Zadeh. Commonsense Knowledge Representation Based on Fuzzy Logic. *Computer*, 16(10):61 –65, oct. 1983.
16. E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, 2010.
17. D. Jungnickel. *Graphs, Networks, and Algorithms*. Algorithms and computation in mathematics. Springer, 2008.
18. E. Fournier, T. Bayne, and J. Kot. Review of the State-of-the-Art in Fuel Tank Systems - Phase II. Technical report, May 2003.
19. G.T. Schwartz. The Myth of the Ford Pinto Case. *Rutgers Law Review*.
20. James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
21. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
22. D.D. Carlson and T.R. Jones. Model-Based Safety Assessments. In *Conference: Lockheed Martin systems engineering and software symposium*, New Orleans, LA, May 1998.
23. P. Fenelon and J.A. McDermid. An Integrated Tool Set for Software Safety Analysis. *J. Syst. Softw.*, 21:279–290, June 1993.
24. M. Wallace. Modular Architectural Representation and Analysis of Fault Propagation and Transformation. In *Proc. FESCA 2005, ENTCS 141(3), Elsevier*, pages 53–71, 2005.

B.2 Paper 2

ICSSE 2013 • IEEE International Conference on System Science and Engineering • July 4-6, 2013 • Budapest, Hungary

Modelling and Analysing Defence-in-Depth in Arming Systems

Dan Slipper and Alistair A. McEwan
Department of Engineering
University of Leicester
Leicester, UK
Email: ds151@le.ac.uk, aam19@le.ac.uk

Wilson Ifill
Directorate of Systems Engineering
AWE Aldermaston
Berkshire, UK

***Abstract*—Safety analysis of high consequence arming systems is complex, many arguments about the behaviour of a design are required to validate that the system fulfils its safety requirements. Manual analysis of such systems can miss potential paths of energy flow and this process becomes increasingly difficult when the concept of defence in depth is incorporated into the design. Utilising the process algebra Communicating Sequential Processes allows component specifications and system level safety specifications to be formalised. Model checking techniques can then be applied to ensure the design of each component meets their individual specifications and that when composed together achieve the required system level behaviour, demonstrating both system level safety and meeting the requirements of defence in depth. We present validation of the technique through the use of a small example representative of the systems of interest we are analysing. The approach is then demonstrated to identify potential problems in this example through various scenarios.**

I. Introduction

Defence-in-Depth (DiD) has been defined as "having multiple, redundant and independent layers of safety for the single, critical point of failure" by the IAEA [1]. It is used in the design of high consequence arming systems to assure they behave in a safe and predictable way, in both normal and abnormal environments. According to systems engineering standard ISO15288 [2], a system is "a combination of interacting elements". For DiD both desired and worst case behaviours of elements must be considered. Analysing the system to ensure that failure to meet one safety requirement does not impact upon others is therefore a difficult task. The coupling between elements is difficult to analyse manually and this process is dependent upon expert judgement, making results difficult to quantify.

Designing for DiD starts early in the system life-cycle, involving safety experts throughout the conceptual design phase of a project. Identifying an appropriate system architecture and specifications for system elements early in the life-cycle phases helps reduce the potential cost and likelihood of design re-work in later phases, it is generally known that later changes to a design have a higher cost, as Emes [3] reports.

Specifications for the safety of high consequence arming systems are documented in the conceptual design phase of the life-cycle and is referred to as a 'safety theme'. This document (described by Covan in [4]) details the measures in place to achieve safety criteria enforced by regulators, for example JSP538 [5] and JSP372 [6]. Such measures use the safety principles of incompatibility, isolation, inoperability and independence, referred to as the 4 I's. More details of these concepts can be found in many sources of literature [7], [8], [9], [10]. This literature also introduces a number of common components used to implement these safety functions, as will be discussed throughout this paper e.g. Exclusion Region Barriers (ERB), Stronglinks (SL) and Lightning Arrestor Connectors (LAC). Other system elements discussed throughout this paper are Detonators (DET) and Firing Units (FU). A safety theme consists of a number of 'claims' as to how each of the system elements should behave, based upon the 4Is principles. An example of isolation would be that:

```
The ERB shall isolate external energy from
the Detonator
```

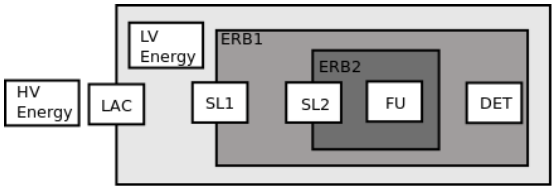
The specification for an example system has been provided in Table I, the architecture for this system is shown in Figure 1 a). This example was presented in paper by Johnson [11] to exemplify a manual design and analysis technique for DiD. Multiple groupings of claims are combined to achieve a system level safety requirement, which in this case is to prevent undesired detonation. Figures 1b and 1c show the groupings of claims from Table I (SS1_x and SS2_x). Such groupings are referred to throughout this paper as independent Safety Subsystems (SS).

TABLE I: Claims upon the safety subsystems. HV and LV refer to High and Low Voltage energy

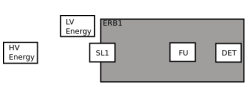
Group_ID	Claims
SS1_1	ERB1 shall isolate external energy from the FU and DET, residual energy shall not be compatible with the FU or DET.
SS1_2	SL1 shall isolate external energy from the FU and DET, residual energy shall not be compatible with the FU or DET.
SS2_1	ERB2 shall isolate external energy from the FU and DET, residual energy shall not be compatible with the FU or DET.
SS2_2	SL2 shall isolate external energy from the FU and DET, residual energy shall not be compatible with the FU or DET.
SS2_3	The Det is incompatible with LV energy.
SS2_4	The LAC shall reduce external HV energy such that it shall not be compatible with the FU or DET.

Any system element is safety critical if it has one or more claims associated with it. Circumstances can arise where these claims are not met, including; an environment in which the element is not expected to meet its claims (e.g. high temperature), operational needs which remove the safety function or finally,

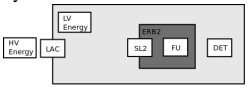
978-1-4799-0009-1/13/$31.00 ©2013 IEEE



(a) Whole system.



(b) Safety subsystem 1.



(c) Safety subsystem 2.

Fig. 1: 2 LODS and each set of claims

if an element has been designed incorrectly. We are most concerned with the circumstances in which an element has been designed incorrectly or is not expected to meet its claims, in which case it is expected to behave at its worst case. The impact of this worst case behaviour upon the whole system must be analysed to ensure the required levels of defence are still in place and do not cause an effect between one SS and another. Analysis techniques already exist within the high-consequence arming systems domain, both manual techniques (as introduced by Johnson [11]) and we have also introduced an automated path analysis software tool [12], [13]. These tools and techniques are useful for defining and analysing system designs, however the software tool does not support analysis of DiD and is not closely related to the regulations of the industry. The manual technique is beneficial however requires expert judgement and lacks repeatability, especially when small changes to a specification need to be analysed. The focus of the manual tool is for grouping the claims such that a complete set of safety functions exist to meet the system level requirements, and the software tool is an automated process for verifying that there are no unexpected paths leading to the undesired consequence given a set of detailed specifications.

Model based approaches have also been used within this industry, introduced by Carlson [14] and Dvorack [15], combining detailed finite element model with event sequences for analysis, however the analysis of DiD has received little attention in any of this literature. Research in the nuclear power industry however, does consider DiD. Lind has conducted exploratory research into modelling DiD as energy flow using Multilevel Flow Modelling [16], however further work is required before reasoning about such models will be possible.

To provide a repeatable and rigorous approach to modelling and analysis of arming system safety claims, we present a formal model based approach that supports analysis of DiD. In the rest of this paper we provide: an overview of model checking and how it is applicable for the problem described, an explanation of the proposed modelling and analysis approach, demonstration of this approach with a published example, lessons learnt from an industrial application and discussion of the benefits and weaknesses of the new method.

II. Modelling and Model Checking

Our new approach to analysis of DiD utilises model checking, a technique commonly used in computer science. Various model checkers exist, a few examples are SPIN [17] and NuSMV [18]. Although these are powerful tools, their drawbacks are that they lack the notions of composition, abstraction and hiding. We utilise Communicating Sequential Processes (CSP) [19] and the model checker FDR2 [20] which provide these desired features. A CSP model is a network of communicating processes, the behaviour of which is defined as a sequence of events which are used to communicate between processes via channels. We employ processes to describe behaviour of a design under test and also to represent a specification describing how the design should behave. A system level model is composed from these individual element models through synchronisation of common events. Assert statements are used to verify whether a design meets a given specification. For example:

```
assert SPECIFICATION [T= DESIGN
```

This model check demonstrates trace refinement ([T=), which exhaustively analyses all potential event sequences (traces) of the DESIGN process, and the SPECIFICATION process identifying any traces in which the design does not behave like the specification. If the design is not a trace refinement of the specification the model checker will provide appropriate counter-examples, which can be analysed to identify the issues.

When modelling a high consequence arming system we are interested in the flow of energy between elements, this could be in the form of electrical signals, shock or temperature. Since this method will be used at the conceptual design stage of the system life cycle, detailed behaviour models such as transfer functions will not be available and therefore energy levels are abstracted into ranges.

Every process in the system is an element of the type source, sink or propagator/transformer, as used by Fenelon in the Failure Propagation and Transformation Notation (FPTN) [21]. One of the differences between our approach and that of FPTN is that state changes are not discussed and the transfer function of an element may change between states.

Our approach is to model the behaviour of elements within the system as a set of rules for input and output, in a similar fashion to that presented by Roscoe [22] and utilised by McEwan [23] to model hazards and consequences.

```
GENERIC2(Set1, Set2, CauseSet1) =
let
  EX(CurrentSet) =
   ([] x : Energy @
     ([] y : ( { b | (A,b) <- CurrentSet,
           member(x, {A}) } ) @
      if   x==none then output.y -> EX(CurrentSet)
      else input.x -> output.y -> EX(CurrentSet)))
within
  EX(Set1)
  []
  ([] x : CauseSet1 @ x -> EX(Set2))
```

This code snippet presents a generic process that can be reused to model an element of the system with two states and transitions between them. The set CauseSet1 contains events

ICSSE 2013 • IEEE International Conference on System Science and Engineering • July 4-6, 2013 • Budapest, Hungary

able to cause this state change, we are mostly interested in failure to behave in the way set out in the safety theme. Rule sets for input and output in each state are shown, along with a transition set. For example ERB1:

```
erb1_1  = { ( residual,none) }
erb1_2  = { ( lowv,none) }
erb1_3  = { ( highv,residual) }
Set1   = Union( {erb1_1, erb1_2, erb1_3 } )

erb1_f1  = { ( residual,residual) }
erb1_f2  = { ( lowv,lowv)  }
erb1_f3  = { ( highv,highv) }
Set2   = Union( {erb1_f1, erb1_f2, erb1_f3 } )
CauseSet1 = {ss1_1}
```

Each element is modelled using these rule sets, in the case of sinks and sources, the energy level for input or output respectively are modelled as the *none* event. Set2 represents misbehaviour of ERB1 where it does not meet the claims upon it. Modelling failure modes alongside functional behaviour has been labelled an 'extended model', and has explored by Joshi [24] and the ESACS project [25] using the NuSMV model checker. A similar approach has been taken to failure modelling using CSP by Wu [26] to evaluate the effectiveness of protective measures in software architecture designs.

Processes modelling element behaviour can be composed to form a system model, this is achieved through the use of the linked parallel operator, allowing events to be defined as either inputs or outputs. Figure 2 provides an example with a source of energy (Element X), a propagator/transformer (Element Y), and a sink (Element Z).

Element Z is protected from energy which it is vulnerable to (produced by Element X), this is achieved by enclosing it within Region 2. Element Y represents a barrier around Element Z. Passing through Element Y is the only route from Region 1 into Region 2, and will attenuate energy to a level where it is incompatible with Element Z.

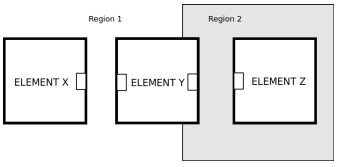


Fig. 2: Simple example of layout

Composing these elements together to form a system model requires the inputs and outputs of each element to be synchronised (at which point they can only engage in an event if both elements are able to perform it together). This approach is used to model the topology of the whole system, the simple example in Figure 2 would be modelled as follows:

```
SYSTEM = (ELEMENT_X [ output <-> input ]
 ELEMENT_Y) [ output <-> input ] ELEMENT_Z
```

The approach of modelling elements as a set of events representing energy flow and state changes allows the system to be analysed through multiple model checks. The next section details our approach for analysing system elements to ensure they meet their specification.

III. ANALYSING DEFENCE IN DEPTH

Our approach for analysing DiD consists of three stages; checking system elements to confirm that they meet the claims upon them in the safety theme, establish whether each SS protects against known threats and finally to ensure that these safety subsystems are independent fulfilling the appropriate DiD requirements. This entails model checks for each of the three levels of detail, the approach is bottom up and aims to provide traceable correctness between the claims upon each element and their resultant behaviour at the system level. Figure 3 provides and overview of the approach, each stage is discussed in more detail.

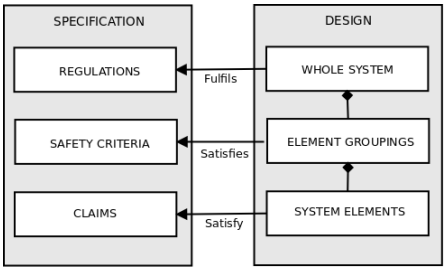


Fig. 3: Overview of the approach.

A. Checking Individual Claims

The first step of the approach is to specify the desired behaviour of each element, based upon the claims associated with them. Taking the example of SS1_1 from Table I, ERB1 must isolate compatible energy from the DET and FU when given LV inputs from external sources. A test case for this claim requires the threats external to ERB1 as an input and the vulnerable elements (either individually or as a firing chain) as an output. In order for ERB1 to meet claim SS1_1 the specification assures that the detonator is not triggered (by any output from ERB1, either directly or stepped up by the firing unit). In order to verify whether this claim has been met, we use a specification which states that any events can occur but *fire*. If the design under test (i.e. the ERB1 process) meets the specification, then the claim is satisfied.

```
TEST = (EXT_HV [] EXT_LV) [ output
<-> input ] (ERB1 [ output <-> input ]
((FU [ output <-> input ] DET) [] FU [] DET))
Vuln = {fire}
SPEC = [] x : diff(AllEvents, Vuln) @ x - > SPEC
assert SPEC [T= TEST
```

One observation from this is that the claim test case does not require a model describing the topology of the whole system. This is due to the nature of SS, at this point we are testing an individual element without considering the threats produced by elements from other safety subsystems behaving worst case.

This is considered in the second stage of the approach. It is clear from the example test case that there are dependencies between each of the models under test. In the example we tested whether claims upon ERB1 have been met by a design under test, however we introduce other elements of the system that could too have claims upon them. The architecture in Figure 1a has a claim upon the detonator, which will have its own individual test case to ensure the DET process meets its specification. Design of the system elements is a balancing act between each of these requirements. The complete set of test cases must be met in order to demonstrate a valid system. Figure 4 shows the FDR2 outputs where a set of claim tests have all been met, and a set where one has not.

(a) Passed all claims. (b) Failed claim 3 of SS2.

Fig. 4: Example of FDR2 output from met and failed claims.

When writing the specification for elements which are required to perform the system function (and may be detrimental to system safety) there may be trades between safety claims and functional requirements. This is out of scope of this paper, however the use of liveness properties in model checking may be appropriate in this situation.

B. Checking Independent Safety Subsystems

The second level of model checking ensures that each safety subsystem is complete and adequate to assure safety, regardless of whether claims from the other safety subsystems have been met or are behaving worst case. Worst case states can be modelled either as a realistic behaviour (for example an ERB failing may act like a wire and pass energy through without attenuation) or as a more extreme scenario (the ERB is able to amplify energy and produces an output higher than the input). Starting with extreme scenarios and using an iterative approach to design allows the analyst to gradually rule out unrealistic or improbable scenarios. Alternatively they can provide a specification which only shows realistic behaviour which they have confidence in being valid.

In order to test each SS, the complete system topology is required. This approach is similar to that used in the software based path analysis approach introduced by Slipper [12]. Figures 1b and 1c show two SS and their topology when elements associated with the other safety subsystem are ignored. We create a model of the whole topology where the SS under test does not change out of its normal state, achieved by modelling the whole system topology and synchronising state change events with SKIP, suppressing these events so they never occur. The other safety subsystems can behave as normal or as if any combination of claims have not been met. For example subsystem 1 under test:

```
SS1Claims = {ss1_1, ss1_2}
TEST = TOPOLOGY [| SS1Claims |] SKIP
Vulnerabilities = {fire}
SPEC = [] x : diff(AllEvents, Vulnerabilities)
    @ x - > SPEC
```

In this second stage of model checking we specify that each SS does not allow detonation to occur. Again, not considering liveness properties and presence of the normal sequence of events for operation. Model checks are necessary to demonstrate that all claims in the SS together provide complete protection against all applicable threats. If these claims together are not satisfactory, the model checker returns a counter-example detailing traces where the specification was not met.

C. Checking the Whole System

The final step of our approach is a model check to verify that the safety subsystems are actually independent of each other, and that failure to meet claims in one does not have a detrimental impact upon the others. This coupling is an aspect lacking in the existing techniques for analysis in this domain. The following specification is used as part of this check:

```
SS(Ss) =
  let
      P(SSClaims) =
      [] x : SSClaims @ x
          -> P'(diff(SSClaims, {x}))

      P'(SSClaims) =
      SKIP
      []
      not empty(SSClaims) &
          [] x : SSClaims @ x
          -> P'(diff(SSClaims, {x}))
  within
      P(Ss)
```

This specification describes the desired behaviour of a single safety subsystem. Initially any event from the set Ss can be performed (this represents any claim in the given subsystem) after which, any of the remaining events can be performed in any order, or at any point the process can SKIP. Essentially, one or more claims of the independent safety subsystem shall be performed. This process is used to model every safety subsystem, and they are then synchronised as interleaving parallel processes, followed by the undesired consequence (see SPEC):

```
SS1Claims = {ss1_1, ss1_2}
SS2Claims = {ss2_1, ss2_2, ss2_3, ss2_4}
SS1 = SS(SS1Claims)
SS2 = SS(SS2Claims)
UNDESIRED = fire -> SKIP
SPEC = (SS1 ||| SS2); UNDESIRED
```

The SPEC process represents the specification for the whole system. We require at least one claim from each safety subsystem to fail before the undesired consequence occurs. The model checker will return counter-examples if the specification has not been met, highlighting any relationships between the claims, be this that the claims overlap multiple subsystems, or coupling causes failure of one subsystem to fail another. More SS can be used in the SPEC process, as many as necessary can be interleaved with the same effect.

ICSSE 2013 • IEEE International Conference on System Science and Engineering • July 4-6, 2013 • Budapest, Hungary

IV. Results

The approach to modelling and analysis of DiD has been described through the running example in Figure 1, taken from [11]. We defined appropriate specifications for each claim in Table I and modelled the elements such that they satisfied these specifications. This demonstrated that when each system element is designed to meet the appropriate claims, they together provide system level safety. Figure 5) shows that claims were met at every level.

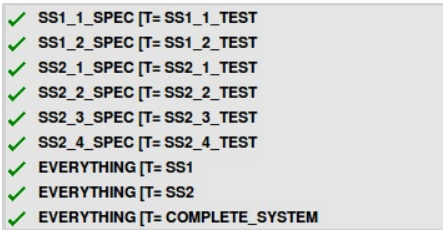


Fig. 5: FDR2 output from a safe system.

A number of scenarios have been used to demonstrate how incorrect claims or designs are identified using this approach. The first example demonstrates one claim which has not been met within this example. In safety subsystem 2, claim 3 states that "The detonator shall be incompatible with LV". The model of DET was modified to make it compatible with LV, thus not fulfilling the specification. The model checks were performed and the results indicated that although the system may still be safe (i.e. no detonation) it does not meet the regulations. It was also reported that the claim was not met and therefore that SS alone did not protect against DET, as shown in Figure 6.

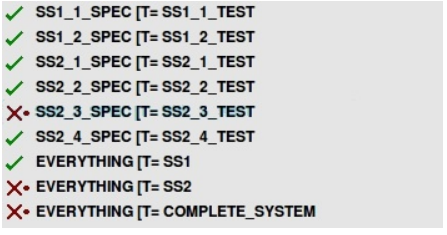


Fig. 6: FDR2 output when a single claim is not fulfilled.

A second scenario was tested, where the detonator was again made compatible with LV, (ss2_3 failed) and also SL1 fails to isolate LV energy (ss1_2 failed). In this scenario one claim from each SS has not been met, the appropriate model checks demonstrated that both of the subsystems have failed and that this has an impact upon the system level safety. Results are shown in Figure 7.

This approach has been applied to an industrial case study to assess its applicability on a larger scale. A promising result of this is that 99% of claims analysed were able to be modelled.

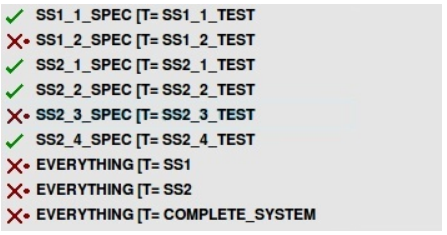


Fig. 7: FDR2 output when a multiple claims are not fulfilled.

A number of claims were questioned whilst developing the model, due to subtle ambiguities in the claim (and to help the modeller fully understand the context). Throughout the modelling 23% of the claims were questioned. One particular claim was found to be described incorrectly. Utilising the model found that this was incorrect and that it would lead to failure of the safety subsystem, although to resolve the issue only took a minor change to the documentation. A negative result from this large scale case study was that the final stage of analysis could not be performed on a full system model due to state space explosion.

V. Discussion

We have demonstrated that our model based approach aids the analysis of specifications for safety critical components within an arming system, both with a simple published example and in an industrial case study. The novel result over other tools in this area is that existing tools used path analysis to identify routes that energy could flow and did not consider how each element would behave if their claims were not met. This approach alone does not appear to solve all of the problems that will arise during such a development, as a number of claims in the industrial case study needed clarification before they could be modelled. This is a weakness in the process that could be further addressed in order to support a model based approach. Although the approach does not address the issues of completeness and ambiguity in a safety theme, it does analyse the complex problem of coupling based on worst case behaviour, and would be a useful tool for testing out possible solutions in a formal and repeatable manner. These aspects demonstrate that the model checking approach is more rigorous and mathematically grounded than that of the software based and manual path analysis methods previously discussed. The use of independent safety subsystems is also part of a larger probabilistic approach, where the probability of the undesired system level consequence is required to be $\leq 10^{-9}$. Probability of consequence is calculated from the probability of failure of the independent safety subsystems (as discussed in more detail by Jones [27]). The probability of failure of each claim could be assigned and totalled up at the system level, however the focus of this paper has been to introduce the model checking approach to analysis of DiD. An issue encountered during the industry case study is that as the model increases in complexity and size, state explosion

becomes an issue. In certain situations models can take hours to complete (if it all). Currently, this is a limitation upon the size of model that can be handled by a typical workstation to perform execution within a set time, or given only limited computer resources. Further work will be required to either identify ways in which the model can be compressed, or we will need to experiment with variations on the approach to handle larger models.

VI. Conclusion

We have demonstrated that our model based approach can be used to support repeatable analysis of DiD, it has been shown to be applicable to the problems of concern within the high consequence arming systems industry. Although modelling of inputs and outputs of energy is abstract, detailed information and concrete values for a transfer function are not likely to be available at the conceptual design stage of the system life cycle. Providing an overview of how safety claims will be apportioned before detailed design of components allows safety criteria to be included into their specification and allows this to be proved early in the life cycle.

Model checking large examples has proven tricky and there may be a limit where state explosion reduces the benefits of this method. We have found that with removal of a few elements the model runs, therefore we are close to being able to handle large scale models. Further work on this is required. Use of this method has highlighted other issues that we intend to investigate, mostly focussing on writing claims in a structured manner, because if they are not easy to model they can be misunderstood later in the design phase.

This approach has been a proof of concept, demonstrating that it is possible to use modelling as part of arming systems safety analysis. To aid the use of this technique in industry we require an approach to specify a safety theme, in a manner which supports this model checking. Additionally, we will need to investigate and resolve the issues of model size. A major benefit of this approach is that once a model has been developed it can be repeatedly and simply modified to perform sensitivity analysis, whereas manual analysis would need to be fully repeated.

Acknowledgment

This research has been sponsored by the EPSRC and AWE.

References

[1] International Nuclear Safety Advisory Group and International Atomic Energy Agency. *Defence in Depth in Nuclear Safety*. INSAG Series. International Atomic Energy Agency, 1996.

[2] Systems and software engineering system life cycle processes. *ISO/IEC 15288:2008(E) IEEE Std 15288-2008 (Revision of IEEE Std 15288-2004)*, pages 1–84, 31.

[3] M. R. Emes, A. Smith, and A. James. Left-shift vs the time value of money: Unravelling the business case for systems engineering. In *INCOSE Spring Conference*, 2007.

[4] J. M. Covan P. E. D'Antonio M. E. Ekman, P. W. Werner and E. Perry. A thematic approach to system safety. In *Process Safety Progress 17:3, American Institute of Chemical Engineers*, 1998.

[5] Ministry of Defence. JSP538 - Regulation of the Nuclear Weapon Programme, 2008.

[6] Ministry of Defence. JSP372 - Approving Authority Management Arrangements for the Trident Re-entry System, 2011.

[7] D.W. Plummer and W.H. Greenwood. The history of nuclear weapon safety devices. In *Conference: 34. AIAA/ASME/SAE/ASEE joint propulsion conference*, Cleveland, OH, Jul 1998.

[8] G. Elliott. US nuclear weapon safety and control. In *MIT Program in Science, Technology, and Society*, 2005.

[9] S. D. Spray and J.A. Cooper. Passive safety concepts applied to critical functions. In *Conference: Joint American Society of Mechanical Engineers (ASME)/Japan Society of Mechanical Engineers (JSME) pressure vessels and piping conference*, Honolulu, HI, Jul 1995.

[10] J. A. Cooper. System safety based on a coordinated principle-based theme. In *Conference: 16. International System Safety Conference*, Seattle, WA, Sept 1998.

[11] C. R. Johnson. Methodology for designing and analyzing high consequence arming systems. In J. M. Livingston, R. Barnes, D. Swallom, and W. Pottraz, editors, *Proceedings of the US Joint Weapons Systems Safety Conference 2009, Huntsville, Alabama*, pages 552–561, 2009. ISBN 9781617387142.

[12] D. Slipper, W. Ifill, G. Hunter, R. Green, R. Johnson, and A. A. McEwan. Towards tool support for design and safety analysis of high consequence arming systems using Matlab. In Ilia Bider, Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Stanislaw Wrycza, editors, *BMMDS/EMMSAD*, volume 113 of *Lecture Notes in Business Information Processing*, pages 393–405. Springer, 2012.

[13] D. Slipper, W. Ifill, G. Hunter, R. Green, R. Johnson, and A. A. McEwan. Matlab tool support for safety analysis of high consequence arming system designs. In *10th Annual Industrial Simulation Conference (ISC'2012)*, pages 49–52. EUROSIS, 2012.

[14] D. D. Carlson and T. R. Jones. Model-based safety assessments. In *Conference: Lockheed Martin systems engineering and software symposium*, New Orleans, LA, May 1998.

[15] M.A. Dvorack, T.R. Jones, D.D. Carlson, J.F. Wolcott, and G.A. Sanders. System safety assessments combining first principles and model based safety assessment methodologies. In *ESREL'98: European safety and reliability conference*, Trondheim, Norway, Jun 1998.

[16] M. Lind. Modeling safety barriers and defense in depth with mulitlevel flow modeling. *Proceedings of First International Symposium on Socially and Technically Symbiotic Systems*, 2012.

[17] G.J. Holzmnn. *The Spin Model Checker: Primer and Reference Model*. Addison-Wesley, 2004.

[18] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: A New Symbolic Model Checker. *International Journal on Software Tools for Technology Transfer*, 2:410–425, 2000.

[19] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[20] Formal Systems (Europe) Ltd. Failures-Divergence Refinement - FDR 2 User Manual, October 2010.

[21] P. Fenelon, J. A. McDermid, M. Nicolson, and D. J. Pumfrey. Towards integrated safety analysis and design. *SIGAPP Appl. Comput. Rev.*, 2:21–32, March 1994.

[22] A. W. Roscoe. *Understanding concurrent systems*. Springer-Verlag New York Inc, 2010.

[23] A. A. McEwan. A calculated implementation of a control system. In Ian R. East, David Duce, Mark Green, Jeremy M. R. Martin, and Peter H. Welch, editors, *Communicating Process Architectures 2004*, pages 265–280, sep 2004.

[24] A. Joshi and M. P. E. Heimdahl. Behavioral fault modeling for model-based safety analysis. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium*, HASE '07, pages 199–208, Washington, DC, USA, 2007. IEEE Computer Society.

[25] M. Bozzano, A. Villafiorita, O. kerlund, P. Bieber, C. Bougnol, E. Böde, M. Bretschneider, A. Cavallo, C. Castel, M. Cifaldi, A. Cimatti, A. Griffault, C. Kehren, B. Lawrence, A. Lüdtke, S. Metge, C. Papadopoulos, R. Passarello, T. Peikenkamp, P. Persson, C. Seguin, L. Trotta, L. Valacca, and G. Zacco. ESACS: an integrated methodology for design and safety analysis of complex systems. In *Proc. ESREL'2003*. Balkema publisher, 2003.

[26] W. Wu and T. Kelly. Failure modelling in software architecture design for safety. *ACM SIGSOFT Software Engineering Notes*, 30:1–7, 2005.

[27] M. Jones. Issues relating to the number of safety subsystems appropriate to a high consequence system. In *Proceedings of the International System Safety Conference 2005*, San Diego, USA, 2005.

B.3 Paper 3

A Framework for Specification of Arming System Safety Functions

D Slipper*, A A McEwan* and W Ifill †

**University of Leicester, UK, ds151, aam19@le.ac.uk †AWE Aldermaston, UK, wil.ifill@awe.co.uk*

Keywords: passive safety, specification, templates,

Abstract

Safety is the primary concern in the design process of high consequence arming systems. Claims form the argument about system safety, and need to be written as atomic, correct and unambiguous statements, which are easily verifiable. To support specification of claims in such a manner we contribute: a decomposition of the currently used claim types into lower level claims, a set of template claims which fit these types and finally we introduce a framework which details the relationships between these claim types. We analysed three industry projects using our approach, unearthing subtle errors. Key findings were that claims which did not fit into the defined categories described implementation detail about the safety functions, and that necessary claims could be missed. Analysts familiar with the domain may routinely leave out claims which seem 'obvious, however, this could have a detrimental impact later in the lifecycle if overlooked by designers.

1 Introduction

The primary concern in the design of high consequence arming systems (such as those used in nuclear weapons) is their safety. Failure to achieve an adequate level of safety can have a negative impact upon the environment, human life, political standing or even worldwide status and credibility. Near misses have been documented in the high consequence arming system domain in the past (in [1]), none of which resulted in nuclear yield. However, such incidents drove a revolution in weapon safety resulting in the introduction of passive safety-critical functions, as presented in [2]. These functions are achieved through the physical properties of one or more components of the system, designing them such that they behave in a predictable manner in both normal and abnormal environments (e.g. accident scenarios). In the UK necessary levels of system safety are defined in regulations Joint Service Publication (JSP) 538 [3]. Functions which contribute to system safety will exist outside of the arming section of the system, as discussed by Bardsley [4], however, the scope within this paper is purely upon the internal electro-mechanical elements of an arming system. The regulations require claims (or assertions as they are referred to in the literature [5]) about safety of this part of the system to be made in terms of physical *isolation* of components, *incompatibility* between components, and the *inoperability* of a single component. These are known as "the 3I's". A further concept of independence is also used to provide more than one protective layer to achieve defence-in-depth requirements. Collections of these claims are known as safety themes, as presented by Ekman in [5] and are defined early in the systems engineering process to abstractly specify how the safety-critical components should behave once implemented, and how this combination will achieve safety requirements when composed into a system. Later in the process potential components which will be selected to fulfil the specification laid out in the safety theme.

An individual claim should be atomic in the sense that it addresses a single low-level safety issue. Individual claims must be unambiguous and correct in the sense that they must be relevant to the safety of the system. Pragmatically, claims must be verifiable against a model of the 'system under test'. State of the art systems safety techniques and methodologies (such as that presented by Johnson in [6]) produce claims which do not always meet these three criteria of atomicity, correctness, and verifiability when defined during the systems design process. Identification of claims which do not meet these criteria is important. If such ambiguities or subtle issues still exist within approved documentation and are given to component designers these issues may not be identified or incorrect assumptions may be made. In such circumstances the cost of change could be significant, and if not identified the resultant system may fail to meet safety requirements or worse still, may pass when it should not. A safety theme needs to be specified correctly, provide a complete set of claims and be written such that it is useful throughout the entire life of the system. To address these problems, in this paper we decompose the 3I's principles into a number of lower level claims which are used in combination (discussed in Section 3). We present trees describing the relationships between the low level claims which form arguments of isolation and incompatibility in Section 4. Then in Section 5 we present results from three real industrial case studies, followed conclusions in Section 6.

2 Example system and safety theme

Figure 1 depicts the topology of an example arming system. This comprises of many common elements described in literature [1] to achieve adequate safety, such as Strong Links (SLs), Exclusion Region Barriers (ERBs), Lightning Arrestor Connectors (LACs), and also typical firing system elements

such as the Firing Unit (FU) and Detonator (DET). Weak links (WLs) are also referred to throughout this paper.

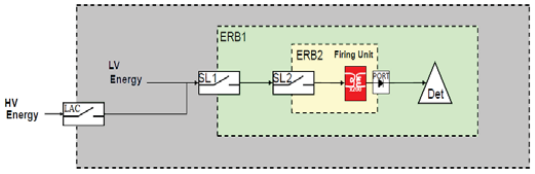


Figure 1: An example arming system topology

Claims describing the behaviour of the system in Figure 1 are listed in Table 1 and broken into two independent Safety Subsystems (SS) (SS1 and SS2 within the table), each SS is designed to provide complete and independent protection against detonation.

Claim	I Type	Text
SS1_1	Isolation	ERB1 shall isolate external energy from the Det, residual energy shall not be compatible with the Det.
SS1_2	Isolation	SL1 shall isolate external energy from the Det, residual energy shall not be compatible with the Det.
SS2_1	Isolation	ERB2 shall isolate external energy from the Det, residual energy shall not be compatible with the Det.
SS2_2	Isolation	SL2 shall isolate external energy from the Det, residual energy shall not be compatible with the Det.
SS2_3	Incompatibility	The Det is incompatible with Low Voltage energy.
SS2_4	Isolation	The LAC isolates HV from the internal region.
SS2_5	Isolation	The outer casing isolates HV from the internal region.
SS2_6	Isolation	PORT shall isolate any energy from entering the FU from its output, residual energy shall be incompatible with the FU.

Table 1: Claims for an example safety theme.

This example has been adapted from [6], adding a directional output port by the firing unit and defining that a LAC and outer casing are used to isolate external High Voltage (HV) energy. Low Voltage (LV) energy exists within the first Exclusion Region (ER). This example will be referred to throughout the paper and we will demonstrate how the claims in Table 1 can be decomposed into a number of atomic claims.

3 From the 3I's to a state machine

Throughout this section we will refer to a number of components, firstly any component which can potentially produce or propagate energy is described as something which can generate a *hazard*. Another term regularly used is *assured safety*, which means the system is expected to be safe. This is achieved through use of multiple components which will contribute to assured safety. In this section we will take the 3Is and the previous work upon them, and then describe how arming system safety concepts and functions can be viewed as a state machine.

3.1 Isolation

A system will be made up of components which generate energy (or hazards) and others which are vulnerable to such hazards. Other components are used to isolate these two compatible components. Johnson presented a breakdown of an isolation claim in [6], the claim consisted of several lower level claims which we will now discuss, referring to the example of ERB1 in Figure 1. ERB1 should stop LV energy from an outer region passing through to the inner region containing the Det. This is achieved by attenuating the LV energy, forming the first low level claim about the component. In reality it is difficult to completely isolate energy between two regions and there will always be some residual energy which can pass from one region to the other. Therefore a second low level claim is required, stating that any energy passing through ERB1 is required to be attenuated such that it is incompatible with the Det. These first two low level claims reflect claim SS1_1 listed in Table 1. Here, the assumption is made that ERB1 can perform this job perfectly.

A realistic component is imperfect and would not guarantee the required level of attenuation at all times. As such a component can be seen to have a number of logical states. These states are either 'safety assured', 'functional' or 'failed'. Failed states depict scenarios where safety can no longer be assured following the effect of a *hazard* above a safe threshold (e.g. temperature, electrical input etc.). We refer to an event which can cause a state change to an state with lower assurance of safety as a *vulnerability* of a component. In some environments hazards above this safe threshold may not be generated. In these situations a third low level claim can be made that the component which would generate the hazards is incompatible with the vulnerabilities of the isolating component. For example if the LV energy source (used in the example system in Figure 1) were potentially able to generate HV energy which could damage ERB1, however in normal environments it is known that only LV energy is generated, we can claim incompatibility between the components in normal environments.

This results in three lower level claim types: attenuation (performed by ERB1), incompatibility between the residual of the isolating component after attenuation and the protected component (i.e. the Det), and incompatibility of external hazards with the vulnerabilities of the isolating component (i.e. the environment and the LV energy source). We believe that there should be a further fourth low level claim which explicitly states that only the listed vulnerabilities can result in a change of state. Since arguing incompatibility is not beneficial where other vulnerabilities may be compatible. All of these lower level claims together form a realistic argument

of isolation, which begins to resemble a state machine form. Figure 2 shows the two states of ERB1 previously discussed and the vulnerabilities which would cause a transition from the 'isolating state' to the 'damaged state'. One may note that ERB1 does not have a functional state. A strong link, on the other hand, is required to open given authorised operation and would have a functional state, as also shown in Figure 2. Having multiple states requires low level claims to be made which explicitly state the conditions of each state change.

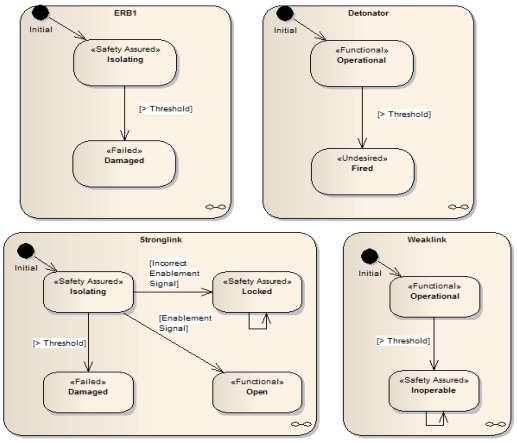


Figure 2: A state machine views of ERB1, the Det, a strong link and a weak link.

3.2 Incompatibility

Following identification of these low level claims which underpin an imperfect isolation claim, similarities can be reflected in the other principles from the 3I's. Figure 2 shows the state machine for a detonator which has an 'undesired' state (when considering the perspective of system safety). This is really a type of functional state, since it is required for operation, simultaneously it describes the undesired consequence of the system which safety devices are protecting against (therefore can still be viewed as a typical vulnerability of a component). It is acknowledged that unlike many safety-critical systems, arming systems have a conflict of interests between safety and reliability of the system. Claim SS2_3 from Table 1 provides an example of an incompatibility claim. This requires energy produced by the LV energy source to be unable to trigger a state change to the Det's undesired state. Since the Det has more than one state, the vulnerabilities which could cause such a state change should also explicitly stated.

When viewing the system as a state machine, two low level claims are used to form a claim of incompatibility. Firstly the usual incompatibility claim will be made between the hazards generated by a component and another's vulnerabilities. Secondly, the vulnerable component's ability to change state is explicitly defined. Incompatibility should be applicable for all vulnerabilities defined.

3.3 Inoperability

Inoperability is not used in the example in Section 2, however, similarly to the other I's it can also be viewed as a state change. For example Figure 2 shows a weak link component, which transitions from an operational state to a safety assured state. When the WL is in the safety assured state, it is claimed that the component is incompatible with any vulnerable component being protected. In this scenario the state change events will also be explicitly defined and should be compatible for any environment in which inoperability is required to occur. A common example is a capacitor weak link which is required to lose the ability to charge in high temperatures, therefore making it incompatible with a detonator. Since such a change results in an increase in assured safety, the change must be irreversible. This defines another type of low level claim limiting a state change. Alternatively inoperability could be achieved in the initial state of the component (e.g. a power source will not produce compatible energy until it is activated), and one would claim incompatibility between the state change stimulus and the environment.

3.4 Generalisation

What emerges from this is that a common set of low level claims that underpin the 3I's, which can be used if each component is viewed as a state machine. These are:

- Attenuation
- Incompatibility
- Limited ability to change state

Beyond the 3I's themselves there is also a common mention in literature about races between component failures in order to assure safety. The most common example is that a weak link is designed to change to a state where safety is assured before a strong link changes to one where there is no longer assurance of safety. Claims for race are used to specify the order in which such changes should occur. This introduces a final low level claim type:

- Race between two state changes

The relationship between the original, perfect 3I's style claims and these new lower level claims is shown in Table 2. What Table 2 alone does not tell us is how many of each type of claim we need or what flavour of it is necessary (e.g. the limitations upon the ability to change state can be used to define which events cause a state change or to claim that a reverse state change cannot occur). Table 3 summarises the different types of claim as a set of templates, which represent all realistic uses that would be commonly seen when developing a safety theme. Other templates and scenarios exist, however they have been omitted to avoid a specification which does not reflect reality (e.g. perfect isolation cannot be claimed, a state where safety is no longer assured must be defined). Claims specified with use of these templates will be atomic and unambiguous, whilst still able to be referred to in terms of the 3I's (as requested by regulators [4]).

Incompatibility is typically argued between a component which generates energy, and the detonator. What is not considered with use of these templates is that in reality, energy could be stepped up by other system components, since electrical isolation within the same ER is challenging to claim in abnormal environments.

3Is+Race\ low level claims	Attenuate	Incompatible	Limited ability to change state	State change order
Isolation	●	●	●	
Incompatibility		●	●	
Inoperability		●	●	
Race			●	●

Table 2: Relationship between the Is, race and the new claims

Low level claim	**#**	**Template text**
Attenuation	1	When (Element X) is in (State S) it shall attenuate outputs of (Element H)
	2	When (Element X) is in (State S) it shall attenuate outputs of (Element H) between (Thresholds T1 and T2)
Incompatibility	3	Output of (Element X) shall be incompatible with the vulnerabilities of (Element Y) when (Element Y) is in (State S)
	4	When (Element X) is in (State Sx) its output shall be incompatible with the vulnerabilities of (Element Y) when (Element Y) is in (State Sy)
Limited ability to change state	5	(Element X) shall only change from (State Sx1) to (State Sx2) given stimulus (V)
	6	Once (Element X) is in (State Sx2) it shall not change state again given any stimulus.
State change order	7	(Element X) shall change from (State Sx1) to (State Sx2) before (Element Y) shall change from (State Sy1) to (State Sy2)

Table 3: The exhaustive set of template claims

To resolve this, an additional '*after flowing via'* clause can be incorporated into the incompatibility claims. For example:
Output of (Element X) shall be incompatible with the vulnerabilities of (Element Y), after flowing via (Element Z)

This clause would not be heavily used, typical it would be used when making incompatibility claims about energy that could flow via the Firing Unit to the DET. However, it can be used to list a chain of components. Although Figure 2 shows the assumed initial conditions of the state machines, we acknowledge that pre-conditions for component states are not defined within this template claim set. In the next section we introduce two trees from a specification framework which shows how the mappings in Table 2 refer to the Template claims in Table 3, since at present no direct relationship exists.

4 Specification framework

The complete framework consists of four trees each of which shows the relationship between one of the 3I's (or race) and our low level claims. The trees use AND and OR constructs similar to a fault tree, showing choices between, or groups of, necessary low level claims which will contribute to a safety argument. In this section two of these diagrams are presented representing Isolation and Incompatibility since they are referred to in the example system in Section 2. Use of the isolation tree is demonstrated and the results are compared against those in Table 1.

4.1 Isolation tree

As an example, we may claim that an ERB isolates a given component of a system. That is to say the ERB is the safety component that isolates a known vulnerability from a known hazard by attenuating the hazard such that it is incompatible with the vulnerability. This can be seen in Figure 3, where the Isolation claim can be decomposed into a series of atomic claims about: attenuation, incompatibility and state change.

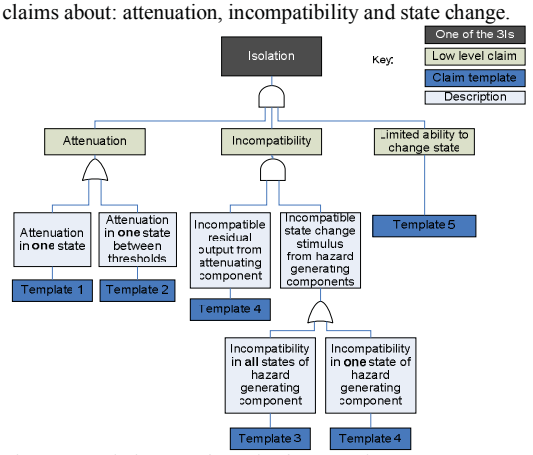


Figure 3: Isolation tree from the framework

4.2 Incompatibility tree

The incompatibility tree is made up of a low level claim of incompatibility and may also contain low level claims about the state changes of both: the component generating a hazard and the vulnerable component. The left hand side of the tree shows low level claims when incompatibility is required between the vulnerable component and the component generating the hazard, for all of the states of that component (e.g. a power source is incompatible regardless of activation or not). The right side of the tree shows the option where the

component generating the hazard is in one particular state (e.g. prior to a power source being activated). An additional low level claim is required to limit the state changes of the vulnerable component in all cases.

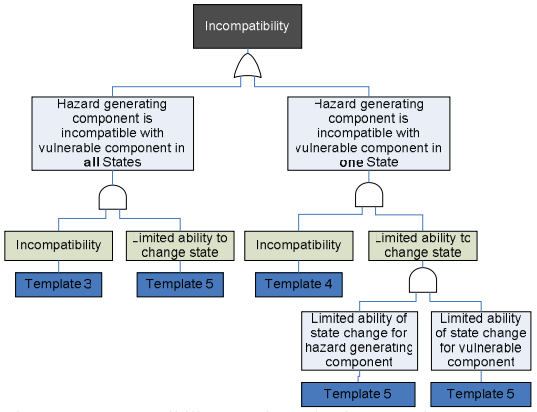


Figure 4: Incompatibility tree from the framework

4.3 Inoperability and Race trees

The inoperability tree is the largest of the trees, but similar to the incompatibility tree in that it is comprised of the same two low level claims (see Table 2), however more choices appear in the inoperability tree because incompatibility always require the vulnerable component to have more that one state, where more combinations of numbers of state are possible with inoperability. Inoperability can involve a change from a functional state to a safety assured state, or vice versa. The race tree is simpler and requires use of a low level claim for the race order, along with state change claims allowing one component to irreversibly change to a safety assured state before the second component leaves a safety assured state.

4.4 Example

Referring back to Table 1 and the claims from the example safety theme it is possible to use the framework to re-write the safety specification. Here we exemplify an isolation claim, previously SS1_1. Using the isolation tree we identify the template claims 1, 4, 3 and 5 were relevant. Using the templates resulted in the following claims being made to achieve the required isolation:

1. When ERB1 is in 'Isolating State' it shall attenuate outputs of the LV energy source
2. Output of the LV energy source shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the 'Isolating State'
3. When ERB1 is in the 'Isolating State' its output shall be incompatible with the vulnerabilities of the Det, when the Det is in the 'Operation State'
4. Det shall only change from the 'Operational State' to the 'Fired State' given Voltage >Vthreshold

5 Applications and results

The framework has been evaluated using safety themes from three industry projects which between them contained ten different safety subsystems. Safety themes from these projects were analysed by selecting the applicable 'I' which underpinned each claim. The appropriate tree from the framework was then used to collate a new list of claims about the system. Comparison was made between the claims in the original safety theme and the new claim list specified using the framework. Relations between the two sets of claims fell into the four categories listed in Table 4, as a relation from the old safety theme to the new list. Since the new low level claim types are designed to be atomic, no many-to-one relation existed (and existing claims were not repeated).

Relation	Explanation
One-to-one	The original claim was atomic
One-to-many	The original claim was not atomic and contained compound claims
No relation (old)	No associated claim was identified by using the framework
No relation (new)	A claim identified by using the framework did not exist previously

Table 4: Relations between new and old claims

Figure 5 shows the number of claims, both in the original safety subsystem and after applying the framework. The change in number of claims over the ten safety subsystems varied between increasing, decreasing and remaining the same. Use of similar template methods typically shows an increase in requirements as shown in [7].

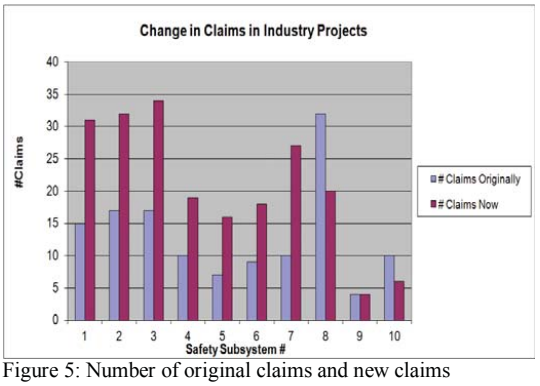


Figure 5: Number of original claims and new claims

If no relation exists from claims in the original safety theme to the new claim list, the original claim is not deemed as safety critical. Claims were documented which contained implementation specific detail about the explicit components used to achieve a particular safety function, and not an abstract specification of how this *should* be achieved. For

example a strong link requires a unique enablement signal to open (further discussed in [14]), an example of the unnecessary (for a safety theme) implementation specific detail was specification of precise detail about such an enablement signal.

Results demonstrated missing relations where claims identified using the framework were not in the original safety theme. In many cases this was due to missing how isolation arguments should be decomposed into isolation and incompatible residual output. Figure 6 shows the number of missed claims per safety subsystem that was analysed, both including and excluding the residual incompatibility claims. The results show that this is not the only scenario where claims were missed. Others that were missed referred to the incompatibility of a state change stimulus.

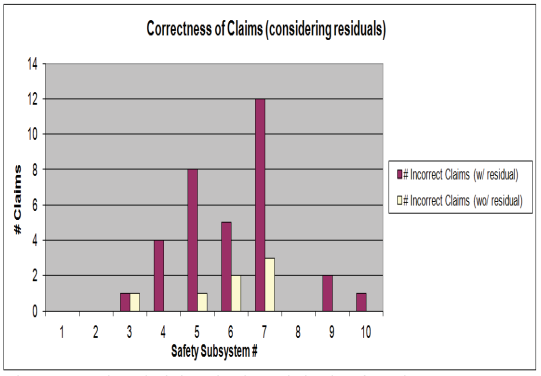


Figure 6: Missed claims in the original safety theme.

6 Discussion

Using the templates supports specification of a safety theme such that it can be read with less ambiguity. Similar requirements syntax techniques, such as Mavin's 'Easy Approach to Requirements Syntax' (EARS) [8] have previously been shown to remove ambiguities in claims and reduce compound claims to atomic ones. For our application this is beneficial because the specification can then be interpreted either by a human or by a computer. A significant technical advantage of our approach is that our claims can be converted into a machine readable form, and discharged automatically using a model-checking tool that exhaustively investigates every claim using generated test cases, the foundation of which is discussed in [7]. Use of the framework helps to ensure that necessary claims are not missed; however, we cannot allege that using the framework in isolation will result in a complete argument for safety being produced.

The safety themes we analysed were written by different authors and ranged in size and complexity, the results in Figure 6 show 7/10 safety subsystems had missing claims; these spanned all three safety themes analysed. Figure 5 presents results where the majority of cases show an increase in the number of claims after use of the framework. It could be argued that this is a negative result, since more claims must be accounted for by the design, and also be verified. However, a positive impact is that each claim is atomic and concise, and therefore easier to verify. Typically a safety theme will be specified using only an abstract system architecture and not define exact detail of the components being used. For example, Plummer [9] lists various models of SL, each of which performs the same function via different configurations of hardware or physical properties. For a safety theme, the allocation of requirements to a strong link is sufficient, detailed design information will be required later in the life cycle, at which point the risks are required to be `as low as reasonably practicable' (ALARP) as specified in [3]. Template documentation has been used to control the implementation of a safety theme through life, as part of the Pentagon /S/ process in [10]. These templates do not demonstrate a direct relationship back to the 3I's and therefore traceability to the very concepts underpinning a safety theme is not possible.

We have demonstrated that use of our templates and framework has supported specification of claims for three existing safety themes, showing that this approach is repeatable. The claims produced through this approach (as shown in the example in Section 4.4) are atomic and easily understandable as individual claims. A key finding of our approach is that it uncovered claims that should have been made about the systems under test which had previously been missed or omitted, thereby providing a more comprehensive and trustworthy argument about systems safety. From discussion with safety engineers some of these are well known concepts and may seem 'obvious' to them whilst documenting the safety theme, however these subtle mistakes could lead to larger consequences. We acknowledge that the results we have identified are specific to arming system safety functions and we cannot claim the applicability of this approach to wider system safety or security problems, investigation of its applicability will be part of our future work.

7 Conclusions

In this paper we introduced set of templates for writing safety claims, a framework of relationships between these different templates, and finally results from their application to three real industrial case studies. Results from the case studies show that our approach produced a more detailed, accurate, and cohesive safety argument for the components investigated. Our approach uncovered claims that should have been made about a system and had previously been missed or omitted, thereby providing a more comprehensive and trustworthy argument about systems safety.

Acknowledgements

This research has been sponsored by AWE and the EPSRC.

References

[1] D. W. Plummer, W. H. Greenwood. "The History of Nuclear Weapon Safety Devices", *AIAA /ASME /SAE/ASEE joint propulsion conference*, American Institute of Aeronautics and Astronautics (1998).

[2] S. D. Spray, J. A. Cooper. "Passive safety concepts applied to critical functions", *Joint American Society of Mechanical Engineers (ASME)/Japan Society of Mechanical Engineers (JSME) pressure vessels and piping conference* (1995).

[3] Ministry of Defence. "Joint Service Publication 538: regulation of the nuclear weapons programme", Issue 2.10 (2010).

[4] A. Bardsley. "Defining and assessing safety functions performed by people", *Cognition, Technology & Work*, *15*(1), 13–18. (2013).

[5] M. E. Ekman, P. W. Werner & P. E. D'Antonio. "A thematic approach to system safety", *Process Safety Progress*, *17*(3), 219–224. (1998).

[6] C. R. Johnson. "Methodology for Designing and Analyzing High Consequence Arming Systems", In *Proceedings of the US Joint Weapons Systems Safety Conference*, pp. 552–561, (2009).

[7] D. Slipper, A. A. McEwan & W. Ifill. "Modelling and Analysing Defence-in-Depth in Arming Systems", *IEEE International Conference on System Science and Engineering,* (2013).

[8] A. Mavin, P. Wilkinson, A. Harwood & M. Novak. "Easy Approach to Requirements Syntax (EARS)", *17th IEEE International Requirements Engineering Conference,*. (pp. 317–322). (2009).

[9] D. W. Plummer, W. H. Greenwood. "A primer on unique signal stronglinks", Sandia National Laboratories, (1993).

[10] P. E. D'Antonio, J. M. Covan & M. E. Ekman. "The Pentagon-S Process - A systematic approach for achieving high confidence in high-consequence products", *Integrated product and process design symposium*, (1997).

Appendix C

Application of the Approach

C.1 Overview of the Example Safety Theme

C.1.1 System Topology

In order to develop assertions of a safety theme using the approach presented in this thesis the system layout is required. This defines how the components within the system will be logically arranged. The states of each component are also necessary. This information has been captured diagrammatically by representing the components of the system and the topology in Figure C.1. Within each component is a state machine detailing its states, initial state and state transitions.

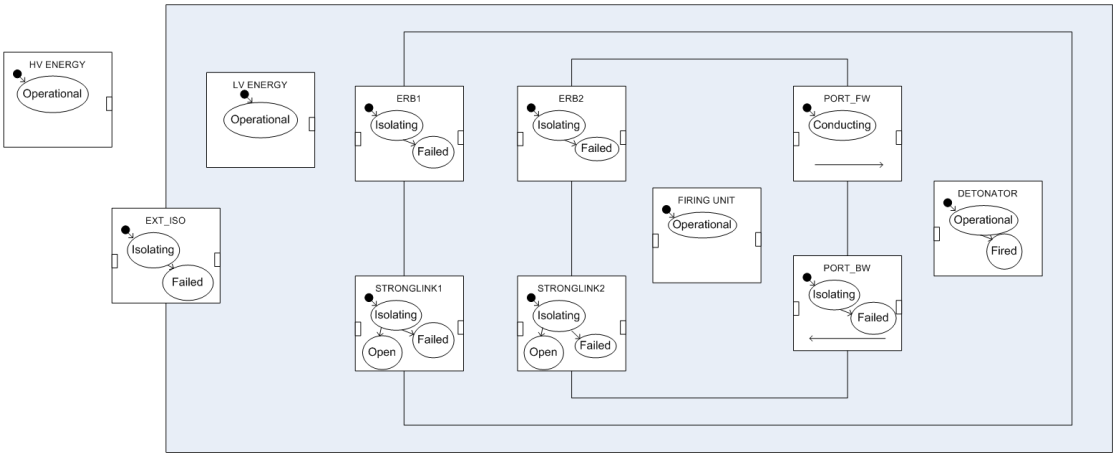


FIGURE C.1: Entire system topology for the example system.

C.1.2 Safety Subsystems

The safety theme is made up of multiple safety subsystems, each providing a complete line of defence against detonation. The topology of each safety subsystem can be viewed in isolation with components only used by other safety subsystems being ignored. This example system comprises of two safety subsystems, as shown in Figures C.2 and C.3.

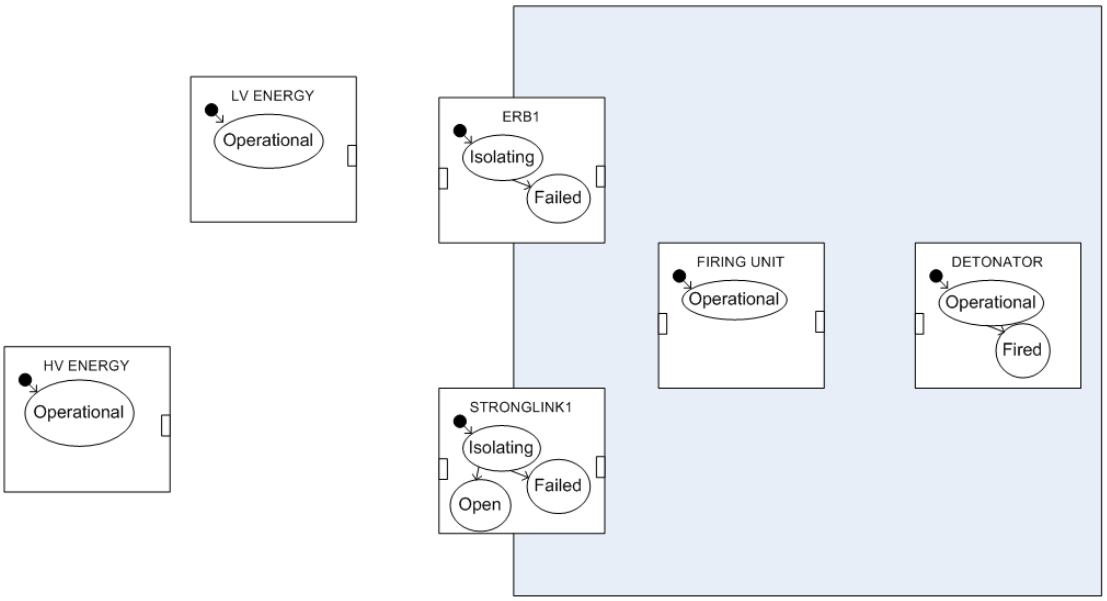


FIGURE C.2: Safety subsystem 1 topology.

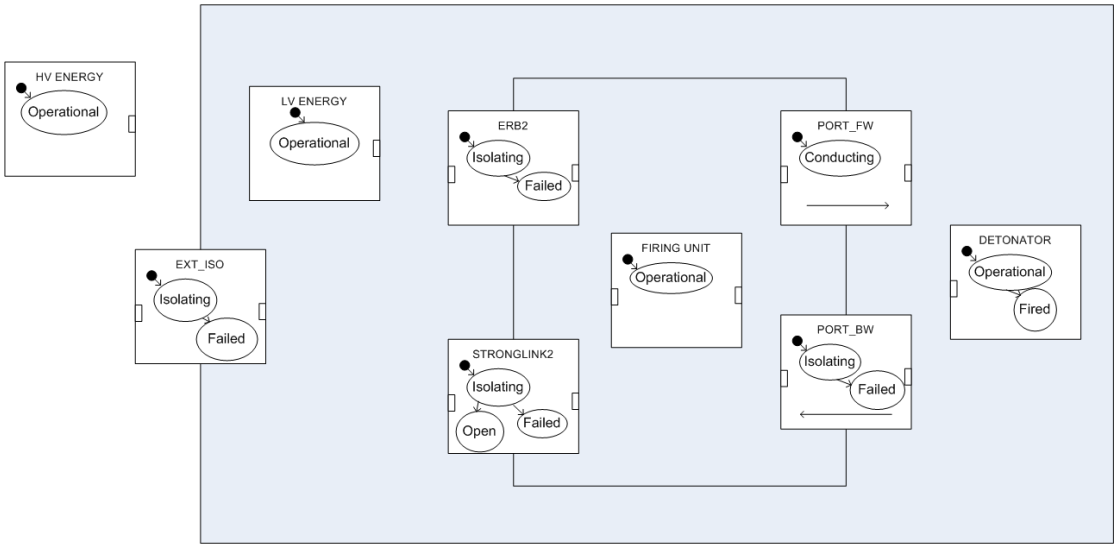


FIGURE C.3: Safety subsystem 2 topology.

Knowing which states each component has is required to identify the applicable patterns which can be used to derive the relevant assertions. In comparison to the original topology presented by Johnson in [74] the example here has an additional port included

between the FU and DET which passes through ERB2. This is seen as two port 'components' with different directions, the design mode use of which allows flow from the FU to reach the DET, however, in order to assure safety it must isolate in the opposite direction. This assertion is not initially noted by Johnson and has been identified from use of this approach, it has been incorporated into this example as a new assertion in safety subsystem 2.

C.1.3 Original Assertions

High level assertions written in terms of the 3I's are required as an input to the approach presented in this thesis. These may be as simple as 'ERB1 isolates'. However, may be more detailed and limit this behaviour to a given state of the component (or even reference lower level assertions such as attenuation). The top level assertions defined by Johnson are as follows:

1. ERB1 isolates
2. SL1 isolates
3. ERB2 isolates
4. SL2 isolates
5. EXT ISO isolates
6. Det is incompatible with LV
7. PORT BW isolates [1]

As noted throughout the thesis, some of these assertions are decomposed into lower level assertions by Johnson, however, only the top level assertions are required for use of this approach. The assertions for this example are logically grouped into the two safety subsystems. These are such that:

- Assertions 1 & 2 are in safety subsystem 1
- Assertions 3-7 are in safety subsystem 2

[1] This assertion was identified and included through use of the approach presented in this thesis

C.2 Identifying Applicable Patterns

In Table C.1 the assertions identified by Johnson are mapped to patterns presented within this thesis and the number of each template which will be used is listed. In some cases a template will be used multiple times due to the number of components within the same region or states that a component has.

TABLE C.1: Patterns used to derive template assertions for the example system.

Original Assertion ID	**Original assertion description**	**Patterns Used**	**Template Assertions**	**New ID**
1	ERB1 isolates	P1	T1	1a
			T1	1b
			T4	1c
			T4	1d
			T3	1e
			T3	1f
			T5	1g
			T6	1h
2	SL1 isolates	P1	T1	2a
			T1	2b
			T4	2c
			T4	2d
			T3	2e
			T3	2f
			T5	2g
			T6	2h
			T6	2i
3	ERB2 isolates	P1 P3	T1	3a
			T1	3b
			T4	3c
			T4	3d
			T3	3e
			T4	3f
			T5	3g
			T6	3h

4	SL2 isolates	P1 P3	T1 T1 T4 T4 T3 T4 T5 T6 T6	4a 4b 4c 4d 4e 4f 4g 4h 4i
5	EXT ISO isolates	P1	T1 T4 T3 T5 T6	5a 5b 5c 5d 5e
6	Det is incompatible with HV	P5	T3 T5 T6	6a 6b 6c
7	PORT BW isolates	P1 P3	T1 T1 T4 T4 T3 T4 T5 T6	7a 7b 7c 7d 7e 7f 7g 7h

C.3 New Assertions

The new assertions referenced in Table C.1 have been defined using the templates identified in the previous section. The full definition of each assertion is included as follows. Each assertion has been given a unique identifier. The following listing includes assertions from both safety subsystems of the example system.

Assertions from safety subsystem 1:

1a) T1 - When ERB1 is in the Isolating State it shall attenuate outputs of LV energy

1b) T1 - When ERB1 is in the Isolating State it shall attenuate outputs of HV energy

1c) T4 - When ERB1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

1d) T4 - When ERB1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

1e) T3 - Output of LV Energy shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the Isolating State

1f) T3 - Output of HV Energy shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the Isolating State

1g) T5 - ERB1 shall initially begin in the Isolating State

1h) T6 - ERB1 shall only change from the Isolating State to the Failed State given stimulus >HV

2a) T1 - When SL1 is in the Isolating State it shall attenuate outputs of LV energy

2b) T1 - When SL1 is in the Isolating State it shall attenuate outputs of HV energy

2c) T4 - When SL1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

2d) T4 - When SL1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

2e) T3 - Output of LV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State

2f) T3 - Output of HV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State

2g) T5 - SL1 shall initially begin in the Isolating State

2h) T6 - SL1 shall only change from the Isolating State to the Failed State given stimulus >HV

2i) T6 - SL1 shall only change from the Isolating State to the Open State given stimulus of UQS1

Assertions from safety subsystem 2:

3a) T1 - When ERB2 is in the Isolating State it shall attenuate outputs of LV energy

3b) T1 - When ERB2 is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)

3c) T4 - When ERB2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

3d) T4 - When ERB2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

3e) T3 - Output of LV Energy shall be incompatible with the vulnerabilities of ERB2 when ERB2 is in the Isolating State

3f) T4 - When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of ERB2 when ERB2 is in the Isolating State

3g) T5 - ERB2 shall initially begin in the Isolating State

3h) T6 - ERB2 shall only change from the Isolating State to the Failed State given stimulus >HV

4a) T1 - When SL2 is in the Isolating State it shall attenuate outputs of LV energy

4b) T1 - When SL2 is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)

4c) T4 - When SL2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

4d) T4 - When SL2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

4e) T3 - Output of LV Energy shall be incompatible with the vulnerabilities of SL2 when SL2 is in the Isolating State

4f) T4 - When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of SL2 when SL2 is in the Isolating State

4g) T5 - SL2 shall initially begin in the Isolating State

4h) T6 - SL2 shall only change from the Isolating State to the Failed State given stimulus >HV

4i) T6 - SL2 shall only change from the Isolating State to the Open State given stimulus of UQS2

5a) T1 - When EXT_ISO is in the Isolating State it shall attenuate outputs of HV energy

5b) T4 - When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

5c) T3 - Output of HV Energy shall be incompatible with the vulnerabilities of EXT_ISO when EXT_ISO is in the Isolating State

5d) T5 - EXT_ISO shall initially begin in the Isolating State

5e) T6 - EXT_ISO shall only change from the Isolating State to the Failed State given stimulus of >HV

6a) T3 - Output of LV Energy shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

6b) T5 - The Detonator shall initially begin in the Operational State

6c) T6 - The Detonator shall only change from the Operational State to the Fired State given stimulus of HV

7a) T1 - When PORT_BW is in the Isolating State it shall attenuate outputs of LV energy

7b) T1 - When PORT_BW is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)

7c) T4 - When PORT_BW is in the Isolating State its output after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

7d) T4 - When PORT_BW is in the Isolating State its output after flowing via the Firing Unit and PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

7e) T3 - Output of LV Energy shall be incompatible with the vulnerabilities of PORT_BW when PORT_BW is in the Isolating State

7f) T4 - When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of PORT_BW when PORT_BW is in the Isolating State

7g) T5 - The PORT_BW shall initially begin in the Isolating State

7h) T6 - The PORT_BW shall only change from the Isolating State to the Failed State given stimulus of >HV

C.4 Component Specifications

It is possible to identify which assertions are relevant for each component of the system. Through manually scanning the requirements and grouping them the following listings include the requirements relevant for each component.

HV Energy

- When ERB1 is in the Isolating State it shall attenuate outputs of HV energy
- Output of HV Energy shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the Isolating State
- When SL1 is in the Isolating State it shall attenuate outputs of HV energy
- Output of HV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State
- When EXT_ISO is in the Isolating State it shall attenuate outputs of HV energy
- Output of HV Energy shall be incompatible with the vulnerabilities of EXT_ISO when EXT_ISO is in the Isolating State

EXT ISO

- When ERB2 is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of ERB2 when ERB2 is in the Isolating State

- When SL2 is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of SL2 when SL2 is in the Isolating State
- When EXT_ISO is in the Isolating State it shall attenuate outputs of HV energy
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- Output of HV Energy shall be incompatible with the vulnerabilities of EXT_ISO when EXT_ISO is in the Isolating State
- EXT_ISO shall initially begin in the Isolating State
- EXT_ISO shall only change from the Isolating State to the Failed State given stimulus of >HV
- When PORT_BW is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of PORT_BW when PORT_BW is in the Isolating State

LV Energy

- When ERB1 is in the Isolating State it shall attenuate outputs of LV energy
- Output of LV Energy shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the Isolating State
- When SL1 is in the Isolating State it shall attenuate outputs of LV energy
- Output of LV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State
- When ERB2 is in the Isolating State it shall attenuate outputs of LV energy
- Output of LV Energy shall be incompatible with the vulnerabilities of ERB2 when ERB2 is in the Isolating State
- When SL2 is in the Isolating State it shall attenuate outputs of LV energy
- Output of LV Energy shall be incompatible with the vulnerabilities of SL2 when SL2 is in the Isolating State

- Output of LV Energy shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When PORT_BW is in the Isolating State it shall attenuate outputs of LV energy
- Output of LV Energy shall be incompatible with the vulnerabilities of PORT_BW when PORT_BW is in the Isolating State

ERB1

- When ERB1 is in the Isolating State it shall attenuate outputs of LV energy
- When ERB1 is in the Isolating State it shall attenuate outputs of HV energy
- When ERB1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- Output of LV Energy shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the Isolating State
- Output of HV Energy shall be incompatible with the vulnerabilities of ERB1 when ERB1 is in the Isolating State
- ERB1 shall initially begin in the Isolating State
- ERB1 shall only change from the Isolating State to the Failed State given stimulus >HV

SB1

- When SL1 is in the Isolating State it shall attenuate outputs of LV energy
- When SL1 is in the Isolating State it shall attenuate outputs of HV energy
- When SL1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator, when the Detonator is in the Operational State

- Output of LV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State
- Output of HV Energy shall be incompatible with the vulnerabilities of SL1 when SL1 is in the Isolating State
- SL1 shall initially begin in the Isolating State
- SL1 shall only change from the Isolating State to the Failed State given stimulus >HV
- SL1 shall only change from the Isolating State to the Open State given stimulus of UQS1

ERB2

- When ERB2 is in the Isolating State it shall attenuate outputs of LV energy
- When ERB2 is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)
- When ERB2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- Output of LV Energy shall be incompatible with the vulnerabilities of ERB2 when ERB2 is in the Isolating State
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of ERB2 when ERB2 is in the Isolating State
- ERB2 shall initially begin in the Isolating State
- ERB2 shall only change from the Isolating State to the Failed State given stimulus >HV

SB2

- When SL2 is in the Isolating State it shall attenuate outputs of LV energy

- When SL2 is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)
- When SL2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- Output of LV Energy shall be incompatible with the vulnerabilities of SL2 when SL2 is in the Isolating State
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of SL2 when SL2 is in the Isolating State
- SL2 shall initially begin in the Isolating State
- SL2 shall only change from the Isolating State to the Failed State given stimulus >HV
- SL2 shall only change from the Isolating State to the Open State given stimulus of UQS2

PORT FW

- When ERB2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

PORT BW

- When PORT_BW is in the Isolating State it shall attenuate outputs of LV energy
- When PORT_BW is in the Isolating State it shall attenuate outputs of EXT_ISO (when it is in the Isolating State)
- When PORT_BW is in the Isolating State its output after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When PORT_BW is in the Isolating State its output after flowing via the Firing Unit and PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- Output of LV Energy shall be incompatible with the vulnerabilities of PORT_BW when PORT_BW is in the Isolating State
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of PORT_BW when PORT_BW is in the Isolating State
- The PORT_BW shall initially begin in the Isolating State
- The PORT_BW shall only change from the Isolating State to the Failed State given stimulus of >HV

Firing Unit

- When ERB1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

- When PORT_BW is in the Isolating State its output after flowing via the Firing Unit and PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

Detonator

- When ERB1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL1 is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL1 is in the Isolating State its output, after flowing via the Firing Unit, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When ERB2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL2 is in the Isolating State its output, after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When SL2 is in the Isolating State its output, after flowing via the Firing Unit and the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When EXT_ISO is in the Isolating State its output shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- Output of LV Energy shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- The Detonator shall initially begin in the Operational State

- The Detonator shall only change from the Operational State to the Fired State given stimulus of HV
- When PORT_BW is in the Isolating State its output after flowing via the PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State
- When PORT_BW is in the Isolating State its output after flowing via the Firing Unit and PORT_FW, shall be incompatible with the vulnerabilities of the Detonator when the Detonator is in the Operational State

In the following section a model of the components used in safety subsystem 1 of the example are presented. Section C.4 shows model checks against the components of this safety subsystem and of a model of the topology of safety subsystem 1 along with a model check against this topology. Only one safety subsystem has been demonstrated to reduce the size of the model in this appendix.

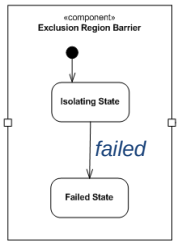
C.5 CSP Model of Components

```
--=============================================================
--Definitions
--=============================================================
1 datatype EnergyTypes = veryhighv | highv | lowv | none | residual | uqs1 | uqs2
2 channel input, output : EnergyTypes
3 IOEvents ={| input, output |}

--Transition events
4 channel failed, unlock, fire
5 TransitionEvents = {failed, unlock, fire}
6 AllEvents = union(TransitionEvents, IOEvents)

7 ERB1FailEvents   = {failed}
8 SL1FailEvents    = {failed}
9 SL1OpenEvents    = {unlock}
10 DETFireEvents   = {fire}

--=============================================================
--Rule Sets
--=============================================================

11 hv_energy_1        = {     ( none,          residual)}
12 hv_energy_2        = {     ( none,          lowv)    }
13 hv_energy_3        = {     ( none,          highv)   }

14 HV_ENERGYRules  = Union( {hv_energy_1,
15                            hv_energy_2,
16                            hv_energy_3 } )
--=============================================================

17 lv_energy_1        = {     ( none, residual)}
18 lv_energy_2        = {     ( none, lowv)    }

19 LV_ENERGYRules  = Union( {lv_energy_1,
20                            lv_energy_2 } )

--=============================================================

21 erb1_1             = {     ( residual,      none)    }
22 erb1_2             = {     ( lowv,          none)    }
23 erb1_3             = {     ( highv,         residual)}
24 erb1_4             = {     ( veryhighv,     failed)  }
25 erb1_5             = {     ( uqs1,          none)    }
26 erb1_6             = {     ( uqs2,          none)    }

27 ERB1Rules  = Union( {erb1_1,
28                       erb1_2,
29                       erb1_3,
30                       erb1_4,
31                       erb1_5,
32                       erb1_6 } )
33 erb1_failed_1        = {     ( residual,      residual) }
34 erb1_failed_2        = {     ( residual,      none)     }
35 erb1_failed_3        = {     ( lowv,          lowv)     }
```

Defines energy types used in the model and that energy can flow in or out with all using these types

Defines all input.**energytype** and output.**energytype** events

Defines state change events and defines a set which contains all of them, also defines a set of all events



Defines which events are applicable for which state changes, e.g. the *failed* event depicts a transition from the Isolating State to the Failed State. This is a channel so it can be used for model checks later on

What follows up to line 168 are rule sets for each component in the system. The first two components are energy sources, therefore have **none** as an input to generate energy. Sections of comments separate each component.

Defines an input.**highv** event

Defines an output.**residual** event

Creates a single set defining all of the rules for a state

This set of rules define behaviour in a failed state for ERB1

```
36 erb1_failed_4        = {     ( lowv,          residual) }
37 erb1_failed_5        = {     ( lowv,          none)     }
38 erb1_failed_6        = {     ( highv,         highv)    }
39 erb1_failed_7        = {     ( highv,         lowv)     }
40 erb1_failed_8        = {     ( highv,         residual) }
41 erb1_failed_9        = {     ( highv,         none)     }
42 erb1_failed_10       = {     ( veryhighv,     veryhighv)}
43 erb1_failed_11       = {     ( veryhighv,     highv)    }
44 erb1_failed_12       = {     ( veryhighv,     lowv)     }
45 erb1_failed_13       = {     ( veryhighv,     residual) }
46 erb1_failed_14       = {     ( veryhighv,     none)     }
47 erb1_failed_15       = {     ( uqs1,          uqs1)     }
48 erb1_failed_16       = {     ( uqs2,          uqs2)     }
49 ERB1FailedRules      = Union( {erb1_failed_1,
50                                 erb1_failed_2,
51                                 erb1_failed_3,
52                                 erb1_failed_4,
53                                 erb1_failed_5,
54                                 erb1_failed_6,
55                                 erb1_failed_7,
56                                 erb1_failed_8,
57                                 erb1_failed_9,
58                                 erb1_failed_10,
59                                 erb1_failed_11,
60                                 erb1_failed_12,
61                                 erb1_failed_13,
62                                 erb1_failed_14,
63                                 erb1_failed_15,
64                                 erb1_failed_16 }
65                                 erb1_failed_17 }
--================================================================
66 sl1_1          = {     ( residual,      none)     }
67 sl1_2          = {     ( lowv,          none)     }
68 sl1_3          = {     ( highv,         residual)}
69 sl1_4          = {     ( veryhighv,     failed)   }
70 sl1_5          = {     ( uqs1,          unlock)   }
71 sl1_6          = {     ( uqs1,          none)     }

72 SL1Rules      = Union( {sl1_1,
73                          sl1_2,
74                          sl1_3,
75                          sl1_4,
76                          sl1_5,
77                          sl1_6 } )

78 sl1_failed_1 = {     ( residual,      residual) }
79 sl1_failed_2 = {     ( residual,      none)     }
80 sl1_failed_3 = {     ( lowv,          lowv)     }
81 sl1_failed_4 = {     ( lowv,          residual) }
82 sl1_failed_5 = {     ( lowv,          none)     }
83 sl1_failed_6 = {     ( highv,         highv)    }
84 sl1_failed_7 = {     ( highv,         lowv)     }
85 sl1_failed_8 = {     ( highv,         residual) }
86 sl1_failed_9 = {     ( highv,         none)     }
87 sl1_failed_10        = {     ( veryhighv,     veryhighv)}
88 sl1_failed_11        = {     ( veryhighv,     highv)    }
89 sl1_failed_12        = {     ( veryhighv,     lowv)     }
```

```
90 sl1_failed_13        = {     ( veryhighv,    residual) }
91 sl1_failed_14        = {     ( veryhighv,    none)    }
92 sl1_failed_15        = {     ( uqs1,         uqs1)    }
93 sl1_failed_16        = {     ( uqs2,         uqs2)    }
94 SL1FailedRules       = Union( {sl1_failed_1,
95                          sl1_failed_2,
96                          sl1_failed_3,
97                          sl1_failed_4,
98                          sl1_failed_5,
99                          sl1_failed_6,
100                         sl1_failed_7,
101                         sl1_failed_8,
102                         sl1_failed_9,
103                         sl1_failed_10,
104                         sl1_failed_11,
105                         sl1_failed_12,
106                         sl1_failed_13,
107                         sl1_failed_14,
108                         sl1_failed_15,
109                         sl1_failed_16 } )

110 sl1_open_1          = {     ( residual,     residual) }
111 sl1_open_2          = {     ( residual,     none)    }
112 sl1_open_3          = {     ( lowv,         lowv)    }
113 sl1_open_4          = {     ( lowv,         residual) }
114 sl1_open_5          = {     ( lowv,         none)    }
115 sl1_open_6          = {     ( highv,        highv)   }
116 sl1_open_7          = {     ( highv,        lowv)    }
117 sl1_open_8          = {     ( highv,        residual) }
118 sl1_open_9          = {     ( highv,        none)    }
119 sl1_open_10         = {     ( veryhighv,    veryhighv)}
120 sl1_open_11         = {     ( veryhighv,    highv)   }
121 sl1_open_12         = {     ( veryhighv,    lowv)    }
122 sl1_open_13         = {     ( veryhighv,    residual) }
123 sl1_open_14         = {     ( veryhighv,    none)    }
124 sl1_open_15         = {     ( uqs1,         uqs1)    }
125 sl1_open_16         = {     ( uqs2,         uqs2)    }
126 SL1OpenRules        = Union( {sl1_open_1,
127                          sl1_open_2,
128                          sl1_open_3,
129                          sl1_open_4,
130                          sl1_open_5,
131                          sl1_open_6,
132                          sl1_open_7,
133                          sl1_open_8,
134                          sl1_open_9,
135                          sl1_open_10,
136                          sl1_open_11,
137                          sl1_open_12,
138                          sl1_open_13,
139                          sl1_open_14,
140                          sl1_open_15,
141                          sl1_open_16 } )
--=================================================================
142 fu_1        = {     ( residual,     residual)}
143 fu_2        = {     ( lowv,         highv)   }
144 fu_3        = {     ( highv,        veryhighv)}
145 fu_4        = {     ( veryhighv,    veryhighv)}
```

```
146 fu_4        = {     ( veryhighv,    veryhighv)}
147 fu_5        = {     ( uqs1,         uqs1)    }
148 fu_6        = {     ( uqs2,         uqs2)    }

149 FURules    = Union( {fu_1,
150                       fu_2,
151                       fu_3,
152                       fu_4,
153                       fu_5,
154                       fu_6 } )
--=================================================================
155 det_1      = {     ( residual,     none)    }
156 det_2      = {     ( lowv,         none)    }
157 det_3      = {     ( highv,        fire)    }
158 det_4      = {     ( veryhighv,    none)    }
159 det_5      = {     ( uqs1,         none)    }
160 det_6      = {     ( uqs2,         none)    }

161 DETRules  = Union( {det_1,
162                       det_2,
163                       det_3,
164                       det_4,
165                       det_5,
166                       det_6 } )
--=================================================================
167 DETFiredRules              = {}
--=================================================================

--GENERIC PROCESSES

168 GENERIC(Set1) =
169   let
170     EX(CurrentSet) =
171      ([] x : EnergyTypes @
172            ([] y : ( { b | (A,b) <- CurrentSet, member(x, {A}) } ) @
173                  if   x==none
174                  then output.y -> EX(CurrentSet)
175                  else input.x -> output.y -> EX(CurrentSet) ))
176   within
177     EX(Set1)
--=================================================================
178 GENERIC2(Set1, Set2, StateChangeLocal, StateChangeGlobal, perm1) =
179  let
180   EX(CurrentSet) =
181    ([] x : EnergyTypes @
182            ([] y : ( { b | (A,b) <- CurrentSet, member(x, {A}) } ) @
183                        if   x==none
184                        then output.y -> EX(CurrentSet)
185                        else input.x -> if member(y, StateChangeLocal)
186                                        then y -> (EX(Set2)
187                                          [] not perm1 & EX(Set1))
188                                        else output.y -> EX(CurrentSet)))
189  within
190   EX(Set1)
191   []
192   ([] x : StateChangeGlobal @ x -> (EX(Set2)
193                                     [] not perm1 & EX(Set1)))
```

This is the event which is to be avoided at the safety subsystem level. Represents the detonator firing.

No behaviour defined when in the Det Fired state as this is the final event of the chain we are interested in.

Processes used to create models with rule sets as an input. These are used relative to the number of states of the components in the system.

Takes and input rule set of behaviour in its one state

Selects each input within the EnergyTypes set and determines if it is applicable as an input. Where selected they are offered as an external choice []

If the input is none, then the component is an energy source and should only provide an output. Otherwise the input -> output events are defined. The process then repeats to provides an infinite trace

As the above process to model a component with 2 states. Takes input of the behaviour in states 1 and 2, *local* events which cause a state change in terms of inputs and then global events triggered by the environment. None of these *global* events are used in the example, but have been for industry applications. The *perm1* input is used to model components where template T7 is used. Again, not in this example.

Defines that a global event e.g. high temperature can change the state to behave like set 2.

As above, but local changes can result in a state change to behaviour defined in set 2 if the component should not be held permanently in the current state.

```
194 GENERIC3(Set1, Set2, Set3, StateChangeLocal1, StateChangeGlobal1, perm1,
        StateChangeLocal2, StateChangeGlobal2, perm2) =
195  let
196    EX(CurrentSet) =
197      ([] x : EnergyTypes @
198               ([] y : ( { b | (A,b) <- CurrentSet, member(x, {A}) } ) @
199                        if   x==none
200                        then output.y -> EX(CurrentSet)
201                        else input.x -> if member(y, StateChangeLocal1)
202                                        then y -> (EX(Set2) [] not perm1 & EX(Set1))
203                                        else if member(y, StateChangeLocal2)
204                                       then y -> (EX(Set3) [] not perm2 & EX(Set1))
205                                       else output.y -> EX(CurrentSet)))
206  within
207    EX(Set1)
208    []
209    ([] x : StateChangeGlobal1 @ x -> ((EX(Set2) [] not perm1 & EX(Set1))))
210    []
211    ([] x : StateChangeGlobal2 @ x -> (EX(Set3) [] not perm2 & EX(Set1)))
--=================================================================
212 GENERIC_SC(Set1, CauseSet1) =
213     let
214        EX(CurrentSet) =
215          ([] x : EnergyTypes @
216             ([] y : ( { b | (A,b) <- CurrentSet, member(x, {A}) } ) @
217                    if   x==none
218                   then output.y -> (STOP [] EX(CurrentSet))
219                    else input.x -> if member(y, CauseSet1)
220                                    then y -> STOP
221                                    else output.y -> (STOP [] EX(CurrentSet)) ))
222     within
223        EX(Set1)
--=================================================================
--Instantiations
224 HV_ENERGY = GENERIC(HV_ENERGYRules)
225 LV_ENERGY = GENERIC(LV_ENERGYRules)
226 ERB1= GENERIC2(ERB1Rules, ERB1FailedRules, ERB1FailEvents, {}, false
227 SL1 = GENERIC3(SL1Rules, SL1FailedRules, SL1OpenRules, SL1FailEvents, {}, false,
SL1OpenEvents, {}, false)
228 FU   = GENERIC(FURules)
229 DET = GENERIC2(DETRules, DETFiredRules, DETFireEvents, {}, false)
--=================================================================
230 NONE(Set1) = [] x : inter(Set1, {none})     @ output.x -> STOP
231 RES(Set1)  = [] x : inter(Set1, {residual}) @ output.x -> STOP
232 LV(Set1)   = [] x : inter(Set1, {lowv})     @ output.x -> STOP
233 HV(Set1)   = [] x : inter(Set1, {highv})    @ output.x -> STOP
234 VHV(Set1)  = [] x : inter(Set1, {veryhighv})@ output.x -> STOP

235 LTres = {none}
236 LTlv   = {none, residual}
237 LThv  = {none, residual, lowv}
238 LTvhv = {none, residual, lowv, highv}
239 OUTPUT_GT_NONE = [] x : EnergyTypes @ output.x -> STOP
240 OUTPUT_GT_RES  = [] x : diff(EnergyTypes, LTres) @ output.x -> STOP
241 OUTPUT_GT_LV   = [] x : diff(EnergyTypes, LTlv) @ output.x -> STOP
242 OUTPUT_GT_HV   = [] x : diff(EnergyTypes, LThv) @ output.x -> STOP
243 OUTPUT_GT_VHV  = [] x : diff(EnergyTypes, LTvhv) @ output.x -> STOP
```

As with the previous process, but is used to define a component with three states, therefore requiring three sets of behaviour, two sets of local and global state change events and two permanent change flags. This approach could be improved for scalability.

Process which is used for model checks, this represents a component which is allowed to perform a state change given an input, but never shows how it would behave in the state it transitions to. Hence the process will STOP.

Defines processes to represent the behaviour of each component in the safety subsystem. These are instantiated using the "GENERIC" processes by defining the component's behaviour in each state

These processes are used during model checking to extract each energy type from a rule set (if it exists) and output it.

Specification processes used to state output any energy type over a given type (e.g. OUTPUT_GT_HV produces everything **veryhighv**, as it is greater than HV).

C.6 Component and Safety Subsystem Model Checks

Selects which of the energy types is output by the LV Energy component and produces a list of hazard.

```
--Assertion: 1a)
--Template1 ERB1 in ISO state attenuates LV
244 LV_ENERGYHazards = inter(EnergyTypes,{b | (A,b) <- LV_ENERGYRules})
245 ERB1_ISO = (ERB1 [| ERB1FailEvents |] STOP)
```

ERB1_ISO is a process which represents ERB1 when it is only behaving as its "Isolating State". This is achieved by suppressing the state change event to its failed state by synchronising with STOP upon that event – this means the event cannot occur as STOP will not perform this event.

If the first component in this test (LV Energy) produces a hazard of a given energy type, this is provided by the NONE,RES, LV etc. process and synchronised with ERB1 in its Isolating State. The specifications used test whether the same level of energy or above are produced by ERB1 in the isolating state. It is expected that in at least one case it will attenuate and therefore fail.

```
--Expected: At least one must fail
246 assert OUTPUT_GT_NONE [T= NONE(LV_ENERGYHazards) [ output <-> input ] ERB1_ISO
247 assert OUTPUT_GT_RES  [T= RES(LV_ENERGYHazards)  [ output <-> input ] ERB1_ISO
248 assert OUTPUT_GT_LV   [T= LV(LV_ENERGYHazards)   [ output <-> input ] ERB1_ISO
249 assert OUTPUT_GT_HV   [T= HV(LV_ENERGYHazards)   [ output <-> input ] ERB1_ISO
250 assert OUTPUT_GT_VHV  [T= VHV(LV_ENERGYHazards)  [ output <-> input ] ERB1_ISO
--------------------------------------------------------------------------------
--Assertion: 1b)
--Template1 ERB1 in ISO state attenuates HV
```

As above but repeated for HV Energy

```
251 HV_ENERGYHazards = inter(EnergyTypes,{b | (A,b) <- HV_ENERGYRules})
--Expected: At least one must fail
252 assert OUTPUT_GT_NONE [T= NONE(HV_ENERGYHazards) [ output <-> input ] ERB1_ISO
253 assert OUTPUT_GT_RES  [T= RES(HV_ENERGYHazards)  [ output <-> input ] ERB1_ISO
254 assert OUTPUT_GT_LV   [T= LV(HV_ENERGYHazards)   [ output <-> input ] ERB1_ISO
255 assert OUTPUT_GT_HV   [T= HV(HV_ENERGYHazards)   [ output <-> input ] ERB1_ISO
256 assert OUTPUT_GT_VHV  [T= VHV(HV_ENERGYHazards)  [ output <-> input ] ERB1_ISO
--------------------------------------------------------------------------------
--Assertion: 1c)
--Template 4 - ERB1 ISO incompatible with DET (local only)
--Allow anything but the state change vulnerability events
257 NotDetEvents = diff(AllEvents, DETFireEvents)
```

NotDetEvents is a set containing all events but the one which represents the DET firing (i.e. **fire**), DET_INCOMP_SPEC is a process that allows any event, in any order except **fire**. The model check tests whether outputs of ERB1 in the Isolating State cause this event.

```
258 DET_INCOMP_SPEC = [] x : NotDetEvents @ x -> DET_INCOMP_SPEC
--Expected: True
259 assert DET_INCOMP_SPEC [T= (ERB1_ISO [ output <-> input ] DET)
--------------------------------------------------------------------------------
--Assertion: 1d)
--Template 4 - ERB1 ISO via FU incompatible with DET (local only)
```

As above but tests energy from the ERB passing through the FU and then to the DET.

```
--Expected: True
260 assert DET_INCOMP_SPEC [T= ((ERB1_ISO [ output <-> input ] FU) [ output <-> input ] DET)
--------------------------------------------------------------------------------
--Assertion: 1e)
--Template 3 -  LV incompatible with ERB1 (local only)
```

As with the previous checks, except checking whether LV Energy is incompatible with the vulnerabilities of ERB1 i.e. any events which cause it to enter the failed state.

```
261 ERB1_INCOMP_SPEC = [] x : diff(AllEvents, ERB1FailEvents) @ x -> ERB1_INCOMP_SPEC
--Expected: True
262 assert ERB1_INCOMP_SPEC [T= LV_ENERGY [ output <-> input ] ERB1
--------------------------------------------------------------------------------
--Assertion: 1f)
--Template 3 -  HV incompatible with ERB1 (local only)

--Expected: True
263 assert ERB1_INCOMP_SPEC [T= HV_ENERGY [ output <-> input ] ERB1
```

```
--------------------------------------------------------------------------------
--Assertion: 1g)
--Template 5 -  ERB1 initially in Isolating State

264 ERB1_ISO_SC = GENERIC_SC(ERB1Rules, ERB1FailEvents)
265 ERB1_UNDER_TEST = ERB1 [| ERB1FailEvents |] STOP
266 ERB1_INITIAL_SPEC = ERB1_ISO_SC \ ERB1FailEvents

--Expected: Both tests true
267 assert ERB1_INITIAL_SPEC [T= ERB1_UNDER_TEST
268 assert ERB1_UNDER_TEST [T= ERB1_INITIAL_SPEC
--------------------------------------------------------------------------------
--Assertion: 1h)
--Template 6 -  ERB1 only changes to Failed State given veryhighv
269 ERB1_SC_SPEC = [] x : diff(AllEvents, ERB1FailEvents) @ x -> ERB1_SC_SPEC

--Expected: True
270 assert ERB1_SC_SPEC [T= ERB1_ISO_SC [| {input.veryhighv} |] STOP
--------------------------------------------------------------------------------
--Assertion: 2a)
--Template1 SL1 in ISO state attenuates LV
271 SL1_ISO = (SL1 [| union(SL1FailEvents, SL1OpenEvents) |] STOP)
--Expected: At least one must fail
272 assert OUTPUT_GT_NONE [T= NONE(LV_ENERGYHazards) [ output <-> input ] SL1_ISO
273 assert OUTPUT_GT_RES  [T= RES(LV_ENERGYHazards)  [ output <-> input ] SL1_ISO
274 assert OUTPUT_GT_LV   [T= LV(LV_ENERGYHazards)   [ output <-> input ] SL1_ISO
275 assert OUTPUT_GT_HV   [T= HV(LV_ENERGYHazards)   [ output <-> input ] SL1_ISO
276 assert OUTPUT_GT_VHV  [T= VHV(LV_ENERGYHazards)  [ output <-> input ] SL1_ISO
--------------------------------------------------------------------------------
--Assertion: 2b)
--Template1 SL1 in ISO state attenuates HV

--Expected: At least one must fail
277 assert OUTPUT_GT_NONE [T= NONE(HV_ENERGYHazards) [ output <-> input ] SL1_ISO
278 assert OUTPUT_GT_RES  [T= RES(HV_ENERGYHazards)  [ output <-> input ] SL1_ISO
279 assert OUTPUT_GT_LV   [T= LV(HV_ENERGYHazards)   [ output <-> input ] SL1_ISO
280 assert OUTPUT_GT_HV   [T= HV(HV_ENERGYHazards)   [ output <-> input ] SL1_ISO
281 assert OUTPUT_GT_VHV  [T= VHV(HV_ENERGYHazards)  [ output <-> input ] SL1_ISO
--------------------------------------------------------------------------------
--Assertion: 2c)
--Template 4 - SL1 ISO incompatible with DET (local only)
--Expected: True
282 assert DET_INCOMP_SPEC [T= SL1_ISO [ output <-> input ] DET
--------------------------------------------------------------------------------
--Assertion: 2d)
--Template 4 - SL1 ISO via FU incompatible with DET (local only)
--Expected: True
283 assert DET_INCOMP_SPEC [T= ((SL1_ISO [ output <-> input ] FU) [ output <-> input ] DET)
--------------------------------------------------------------------------------
--Assertion: 2e)
--Template 3 -  LV incompatible with SL1 (local only)
284 SL1_INCOMP_SPEC = [] x : diff(AllEvents, union(SL1FailEvents, SL1OpenEvents)) @ x ->
SL1_INCOMP_SPEC
--Expected: True
285 assert SL1_INCOMP_SPEC [T= LV_ENERGY [ output <-> input ] SL1
```

Uses the GENERIC_SC process to create a model of the ERB which can only behave in one state with a transition event, not behave as its second state.

Suppresses state changes for the full component model

Hides the state change event so it does not show in a trace

Ensures no state change transition can be seen and that the component behaves like its initial state should

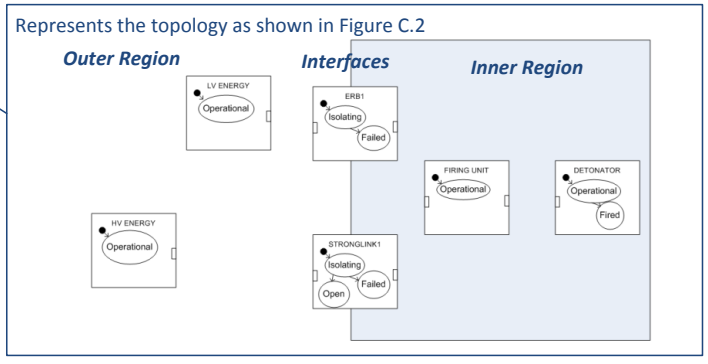
Tests that the state change event of ERB1 cannot be seen if the event input.veryhighv is suppressed, therefore checking if it is the only cause of state change.

The following model checks repeat the same format as the previous ones for the component SL1

```
--Assertion: 2f)
--Template 3 -  HV incompatible with SL1 (local only)

--Expected: True
286 assert SL1_INCOMP_SPEC [T= HV_ENERGY [ output <-> input ] SL1
--------------------------------------------------------------------------------
--Assertion: 2g)
--Template 5 -  SL1 initially in Isolating State

287 SL1_ISO_SC = GENERIC_SC(SL1Rules, union(SL1FailEvents, SL1OpenEvents))
288 SL1_UNDER_TEST = SL1 [| union(SL1FailEvents, SL1OpenEvents) |] STOP
289 SL1_INITIAL_SPEC = SL1_ISO_SC \ union(SL1FailEvents, SL1OpenEvents)

--Expected: Both tests true
290 assert SL1_INITIAL_SPEC [T= SL1_UNDER_TEST
291 assert SL1_UNDER_TEST [T= SL1_INITIAL_SPEC
--------------------------------------------------------------------------------
--Assertion: 2h)
--Template 6 -  SL1 only changes to Failed State given veryhighv
292 SL1_SC_SPEC1 = [] x : diff(AllEvents, SL1FailEvents) @ x -> SL1_SC_SPEC1

--Expected: True
293 assert SL1_SC_SPEC1 [T= SL1_ISO_SC [| {input.veryhighv} |] STOP
--------------------------------------------------------------------------------
--Assertion: 2i)
--Template 6 -  SL1 only changes to Open State given uqs1
294 SL1_SC_SPEC2 = [] x : diff(AllEvents, SL1OpenEvents) @ x -> SL1_SC_SPEC2

--Expected: True
295 assert SL1_SC_SPEC2 [T= SL1_ISO_SC [| {input.uqs1} |] STOP

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

--Model check on entire system - normal component behaviour
296 NO_DET_SPEC = []  x : diff(AllEvents, {fire}) @ x -> NO_DET_SPEC

--Safety subsystem 1 model
297 OUTER_REGION = LV_ENERGY [] HV_ENERGY
298 INTERFACES   = ERB1 [] SL1
299 INNER_REGION = FU [ output <-> input ] DET
300                []
301                DET
302 SAFETYSUBSYSTEM1 = (OUTER_REGION [ output <-> input ] INTERFACES) [ output <->
input ] INNER_REGION

--Model check on safety subsystem 1
303 assert NO_DET_SPEC [T= SAFETYSUBSYSTEM1
```

Provides a specification for the safety subsystem where any event can occur with the exception of that which represents detonation

Produces processes which represent different regions of the safety subsystem under test



Test that the safety subsystem provides complete protection

Appendix D

Letter for Validation

AWE Aldermaston

Reading

Berks

RG7 4PR

Date: 11/12/2014

F.A.O the examiners,

I provide you with this letter to confirm the contribution of the work Dan Slipper has presented in his thesis presented in fulfilment of the requirements of the degree Doctor of Engineering.

Dan has been embedded within the "Surety Analysis" team at AWE during his EngD studies, in which time he worked closely with the team who were involved in authoring safety themes and developing new practices.

Dan used three case studies within the company to apply the approach he has developed over the course of the EngD programme. He has been unable to present the results from these case studies in their full detail within his thesis or publications due to the nature of the projects. I can confirm that I was involved in reviewing the results of these case studies, as were the original authors of each safety theme.

As presented, Dan used his approach to identify: assertions which were not originally included in the analysed safety themes, requirements which were implementation specific detail and not assertions, and finally that in some cases the original assertions were not presented as simple, verifiable statements.

For one of these case studies Dan also created a system model upon which he applied model checking techniques to verify the assertions of one safety theme. Although his modelling approach had size limitations, which were encountered when model checking a number of safety subsystems and trying to determine independence between them. However, his approach was successfully applied to verify that individual assertions were met by models of the individual components and their composition into individual safety subsystems (the scope of which his template assertions and patterns define).

This ability to be able to model safety themes is particularly useful in being able to verify safety assertions early in the system lifecycle due to the time and costs of such a project. We expect that the template assertions that Dan has presented will be verifiable during testing of the physically implemented components which support the safety theme.

Dan's work on templates had an impact on our approach to safety theme development. Our safety themes are formulated to have defence in depth. In particular we strive to offer multiple independent Lines of Defence (LOD) arguments against any insult that would otherwise lead to harm. To avoid common mode failures each system LOD is based on a different independent concept, such as isolation, incompatibility or inoperability, which we term the 3Is. Dan's work demonstrated that a clear understanding of the underlying axioms of the 3Is is necessary by his application of template patterns. Before Dan's template work we worked with an assumed common understanding of the underlying structures to the 3Is. This shared understanding of the structure varied slightly from practitioner to practitioner and this could lead to rework later in the development cycle. We now are developing a clearer understanding of how the 3Is are applied across a range of projects and we are carefully documenting this understanding for internal and external consumption. Dan's template approach provided us with a process that produces repeatable results. We are in the process of applying it to a research development project.

Since Dan's work we have run a number of research projects to optimise his model checking approach. New releases of the FDR model checker have enable us to utilise new compiler optimisation and make Dan's safety system model more tractable.

During his placement at AWE Dan has been actively involved in applying the Systems Engineering approach both within his research and had involvement in improvement of this within the company by supporting a Systems Engineering Community of Practice. Dan represented the company by presenting his work to several visiting bodies and represented AWE at conference.

Regards,

Dr. Wilson Ifill

Dan's Industrial Supervisor

Richard Johnson

Dan' Surety Analysis Team Project Placement Technical Lead.

Bibliography

[1] J.-R. Abrial, M. K. Lee, D. Neilson, P. Scharbach, and I. H. Sørensen. The B-method. In *VDM'91 Formal Software Development Methods*, pages 398–405. Springer, 1991. http://dx.doi.org/10.1007/BFb0020001.

[2] S. Adolph, P. Bramble, A. Cockburn, and A. Pols. *Patterns for Effective Use Cases*. Agile Software Development Series. Addison-Wesley, 2003. ISBN 9780201721843. URL http://www.pearsoned.co.uk/bookshop/detail.asp?item=228544.

[3] I. Alexander and R. A. Stevens. *Writing Better Requirements*. Addison-Wesley, 2002. ISBN 9780321131638. URL http://www.pearsoned.co.uk/bookshop/detail.asp?item=100000000017657.

[4] P. E. D. Antonio, J. M. Covan, and M. E. Ekman. The pentagon-S process - a systematic approach for achieving high confidence in high-consequence products. In *Integrated product and process design symposium*, Fort Worth, TX (United States), Oct. 1997. Office of Scientific and Technical Information. URL http://www.osti.gov/scitech/servlets/purl/532656.

[5] P. E. D. Antonio, J. A. Cooper, S. D. Spray, M. Caldwell, and J. M. Covan. Potential disadvantages of microtechnology for future high consequence safety applications. In *Government Microcircuit Applications Conference*, Monterey, CA (United States), Dec. 1998. Office of Scientific and Technical Information. URL http://www.osti.gov/scitech/biblio/2814.

[6] P. Armstrong, M. Goldsmith, G. Lowe, J. Ouaknine, H. Palikareva, A. W. Roscoe, and J. Worrell. Recent developments in FDR. In P. Madhusudan and S. Seshia, editors, *Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 699–704. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31423-0. doi:10.1007/978-3-642-31424-7_52. URL http://dx.doi.org/10.1007/978-3-642-31424-7_52.

[7] Atlantic Systems Guild Ltd. Volere requirements specification template - extracts and samples from the template, 2014 accessed online 10/10/2014. URL `http://www.volere.co.uk/template.htm`.

[8] Atomic Weapons Establishment. Welcome to the atomic weapons establishment, 2014. URL `http://www.awe.co.uk/`.

[9] M. Aubury, I. Page, G. Randall, J. Saul, and R. Watts. Handel-C language reference guide. Technical report, Computing Laboratory, Oxford University, UK, 1996.

[10] A. T. Bahill and S. J. Henderson. Requirements development, verification, and validation exhibited in famous failures. *Systems Engineering*, 8(1):1–14, March 2005. ISSN 1098-1241. URL `http://10.1002/sys.v8:1`.

[11] A. S. Benjamin. Risk assessment methodologies for nuclear weapons compared to risk assessment methodologies for nuclear reactors. In *Joint American Society of Mechanical Engineers (ASME)/Japan Society of Mechanical Engineers (JSME) pressure vessels and piping conference*, Honolulu, HI (United States), July 1995. Office of Scientific and Technical Information. URL `http://www.osti.gov/scitech/biblio/46581`.

[12] R. R. Bennett and D. A. Summers. A comparison of commercial/industry and nuclear weapons safety concepts. In *National system safety conference*, Albuquerque, NM (United States), Aug. 1996. Office of Scientific and Technical Information. URL `http://www.osti.gov/scitech/biblio/266630`.

[13] R. L. Bierbaum and D. L. Wright. Reliability assessment methodology for 1-shot systems. In *Reliability and Maintainability Symposium*, pages 536–541. IEEE, 2002. URL `http://dx.doi.org/10.1109/RAMS.2002.981699`.

[14] A. Blackstone. *Sociological Inquiry Principles: Qualitative and Quantitative Methods*. 2012. URL `http://2012books.lardbucket.org/`.

[15] J. Boulanger. *Formal Methods: Industrial Use from Model to the Code*. Wiley, 2013. ISBN 9781118614389. URL `http://dx.doi.org/10.1002/9781118561898`.

[16] J. P. Bowen and M. G. Hinchey. Ten commandments revisited: a ten-year perspective on the industrial application of formal methods. In *Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, pages 8–16. ACM, 2005. URL `http://dx.doi.org/10.1145/1081180.1081183`.

[17] M. Broy. Towards a formal foundation of the specification and description language sdl. *Formal Aspects of Computing*, 3(1):21–57, 1991. ISSN 0934-5043. doi:10.1007/BF01211434. URL `http://dx.doi.org/10.1007/BF01211434`.

[18] S. Burge. The systems engineering tool box - pugh matrix, 2009. URL http://www.burgehugheswalsh.co.uk/uploaded/documents/Pugh-Matrix-v1.1.pdf. Burge-Hughes-Walsh.

[19] M. Caldwell and P. D'Antonio. A study of using electronics for nuclear weapon detonation safety. In *34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Joint Propulsion Conferences. American Institute of Aeronautics and Astronautics, July 1998. doi:doi:10.2514/6.1998-3465. URL http://dx.doi.org/10.2514/6.1998-3465http://arc.aiaa.org/doi/abs/10.2514/6.1998-3465.

[20] D. D. Carlson and T. R. Jones. Model-based safety assessments. In *Lockheed Martin systems engineering and software symposium*, New Orleans, LA (United States), May 1998. Office of Scientific and Technical Information. URL http://www.osti.gov/scitech/biblio/587666.

[21] R. Cloutier. Toward the application of patterns to systems engineering. In *Proceedings of the 2005 Conference on Systems Engineering Research*, Hoboken, NJ, (United States), Mar. 2005. Stevens Institute of Technology. URL http://www.researchgate.net/publication/229016765_Toward_the_Application_of_Patterns_to_Systems_Engineering/file/50463519df92590b00.pdf.

[22] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone. The concept of reference architectures. *Systems Engineering*, 13(1):14–27, 2010. ISSN 1520-6858. doi:10.1002/sys.20129. URL http://dx.doi.org/10.1002/sys.20129.

[23] R. J. Cloutier and D. Verma. Applying the concept of patterns to systems architecture. *Systems Engineering*, 10(2):138–154, 2007. ISSN 1520-6858. doi:10.1002/sys.20066. URL http://dx.doi.org/10.1002/sys.20066.

[24] J. A. Cooper. Mathematical aspects of unique signal assessment. Office of Scientific and Technical Information, 2001. URL http://www.osti.gov/scitech/biblio/800955.

[25] J. M. Covan and J. A. Cooper. Predictable Safety in the Control of High Consequence Systems. In *The 3rd IEEE International Symposium on High-Assurance Systems Engineering*, HASE '98, pages 200–204, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-9221-9. URL http://dl.acm.org/citation.cfm?id=645432.652393.

[26] J. R. Davis. *Integrated safety, reliability, and diagnostics of high assurance, high consequence systems.* PhD thesis, Vanderbilt University, Nashville, Tennessee, May 2000.

[27] C. Denger, D. M. Berry, and E. Kamsties. Higher quality requirements specifications through natural language patterns. In *Proceedings. IEEE International Conference on Software: Science, Technology and Engineering, 2003. SwSTE'03.*, pages 80–90. IEEE, Nov. 2003. doi:10.1109/SWSTE.2003.1245428.

[28] R. Denning. Applied R&M manual for defence systems (GR-77), Part C - techniques, chapter 29 fault / success tree analysis, 2009, accessed online 03/11/2014. URL `http://www.sars.org.uk/old-site-archive/BOK/Applied%20R&M%20Manual%20for%20Defence%20Systems%20(GR-77)/p3c29.pdf`.

[29] Department of Defence. *The Nuclear Matters Handbook, Expanded Edition.* Federation of American Scientists. URL `http://fas.org/man/eprint/NMHB2011.pdf`.

[30] Department of Defence. Narrative summaries of accidents involving U.S. nuclear weapons. Technical report, 1981. URL `http://www.dod.mil/pubs/foi/operation_and_plans/NuclearChemicalBiologicalMatters/21.pdf`.

[31] Department of Defense. MIL-P-1629 - Procedures for performing a failure mode effect and critical analysis. Nov. 1949. URL `http://www.assistdocs.com/search/document_details.cfm?ident_number=86479`.

[32] Department of Defense. MIL-STD 882E Standard Practice for System Safety. *US Department of Defense*, 2012.

[33] J. Dick and J. Llorens. Using statement-level templates to improve the quality of requirements. In *24th International Conference on Software & Systems Engineering and their Applications*, Paris, Oct 2012. French Association for Systems Engineering and INCOSE.

[34] J. Donnell, A P. The Black Swan and nuclear weapon safety. In *International System Safety Society Conference*, Vancouver, BC, Canada, 2008.

[35] M. A. Dvorack, T. R. Jones, D. D. Carlson, J. F. Wolcott, and G. A. Sanders. System safety assessments combining first principles and model based safety assessment methodologies. In *ESREL'98: European safety and reliability conference*, Trondheim, Norway, June 1998. URL `http://www.osti.gov/scitech/biblio/645487`.

[36] K. M. Eisenhardt. Building theories from case study research. *Academy of management review*, 14(4):532–550, 1989. doi:10.5465/AMR.1989.4308385.

[37] M. E. Ekman, P. W. Werner, J. M. Covan, and P. E. D'Antonio. A thematic approach to system safety. *Process Safety Progress*, 17(3):219–224, 1998. ISSN 1066-8527. doi:10.1002/prs.680170312.

[38] G. Elliott. US Nuclear Weapon Safety and Control. Technical report, MIT, 2005. URL http://web.mit.edu/gelliott/Public/sts.072/paper.pdf.

[39] Engineering and Physical Sciences Research Council. The EPSRC industrial doctorate centre scheme good practice guidance. http://www.epsrc.ac.uk/newsevents/pubs/the-epsrc-industrial-doctorate-centre-scheme-good-practice-guidance/, May 2011.

[40] N. Evans and W. Ifill. Hardware verification and beyond: using B at AWE. In *Proceedings of the 7th international conference on Formal Specification and Development in B*, B'07, pages 260–261, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-68760-2, 978-3-540-68760-3. doi:10.1007/11955757_24. URL http://dx.doi.org/10.1007/11955757_24.

[41] S. Fetter and F. Von Hippel. The hazard from plutonium dispersal by nuclear-warhead accidents. In *Science & Global Security*, volume 2, pages 21–41. Taylor & Francis, 1990. doi:10.1080/08929889008426345.

[42] D. Firesmith. Specifying good requirements, 2003 (accessed September 5, 2014). URL http://www.jot.fm/issues/issue_2003_07/column7/.

[43] D. Firesmith. Using V models for testing. *Carnegie Mellon University Software Engineering Institute Blog*, November 2013. URL http://blog.sei.cmu.edu/post.cfm/using-v-models-testing-315.

[44] Formal Systems Ltd. (Europe). Failures-Divergence Refinement - FDR 2 User Manual, Oct. 2010.

[45] D. Fowler and R. Pierce. Safety engineering – a perspective on systems engineering. In *Proceedings of the Twentieth Safety-Critical Systems Symposium*, pages 115–136, Bristol, UK, February 2012. Springer London. ISBN 978-1-4471-2493-1. URL http://dx.doi.org/10.1007/978-1-4471-2494-8_10.

[46] K. Fowler. *Mission-Critical and Safety-Critical Systems Handbook: Design and Development for Embedded Applications*. Elsevier Science, 2009. ISBN 9780080942551. URL http://dx.doi.org/10.1016/B978-0-7506-8567-2.00010-X.

[47] S. E. Fowler. Safety and arming device design principles, 1999. URL http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA363924. Defense Technical Information Center.

[48] S. Friedenthal and A. M. Steiner, editors. *Appendix A - SysML Reference Guide*. The MK/OMG Press. Morgan Kaufmann, Boston, second edition edition, 2012. ISBN 978-0-12-385206-9. URL http://dx.doi.org/10.1016/B978-0-12-385206-9.15001-X.

[49] S. Friedenthal, A. Moore, and R. Steiner. OMG systems modeling language (OMG SysML) tutorial. In *INCOSE International Symposium*, 2006. URL http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf.

[50] D. Gaševic, D. Djuric, and V. Devedžic. *Model driven engineering and ontology development*, volume 2. Springer, 2009. ISBN 978-3-540-32182-8. http://dx.doi.org/10.1007/3-540-32182-9_1.

[51] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno. A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, 18(1):25–41, 2013. ISSN 0947-3602. URL http://dx.doi.org/10.1007/s00766-011-0134-z.

[52] T. Gruber. Ontology. In *Encyclopedia of Database Systems*, pages 1963–1965. Springer, 2009. URL http://dx.doi.org/10.1007/978-0-387-39940-9_1318.

[53] R. A. Guidotti and P. Masset. Thermally activated ("thermal") battery technology: Part I: An overview. *Journal of Power Sources*, 161(2):1443–1449, 2006. ISSN 0378-7753. doi:http://dx.doi.org/10.1016/j.jpowsour.2006.06.013. URL http://www.sciencedirect.com/science/article/pii/S0378775306011396.

[54] A. Hall. Seven myths of formal methods. *Software, IEEE*, 7(5):11–19, Sept 1990. ISSN 0740-7459. doi:10.1109/52.57887.

[55] M. Hammersley and P. Atkinson. *Ethnography: Principles in practice*. Routledge, 2007.

[56] N. R. Hansen. Nuclear Safety Themes for Earth Penetrating Weapons. Albuquerque, NM, Apr. 1993. Sandia National Laboratories. URL http://www.nukestrat.com/us/afn/97-14h_SNL040193.pdf.

[57] L. Harms-Ringdahl. On the modelling and characterisation of safety functions. In *Safety and Reliability—Proceedings of ESREL '99, The Tenth European Conference on Safety and Reliability*, pages 1459–1462. Taylor & Francis, 1999.

[58] J. C. Helton, J. D. Johnson, and W. L. Oberkampf. Probability of loss of assured safety in temperature dependent systems with multiple weak and strong links. volume 91, pages 320–348. Elsevier, 2006. URL http://dx.doi.org/doi:10.1016/j.ress.2006.09.005.

[59] J. C. Helton, J. D. Johnson, and W. L. Oberkampf. Verification of the calculation of probability of loss of assured safety in temperature-dependent systems with multiple weak and strong links. volume 92, pages 1363–1373. 2007. URL `http://dx.doi.org/doi:10.1016/j.ress.2006.09.005`.

[60] J. C. Helton, J. D. Johnson, and W. L. Oberkampf. Effect of delayed link failure on probability of loss of assured safety in temperature-dependent systems with multiple weak and strong links. volume 94, pages 294–310. elsevier, 2009. URL `http://dx.doi.org/doi:10.1016/j.ress.2008.03.007`.

[61] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985. ISBN 0-13-153271-5.

[62] E. Hollnagel. Risk + barriers = safety? volume 46, pages 221 – 229. Elsevier, 2008. URL `http://dx.doi.org/doi:10.1016/j.ssci.2007.06.028`.

[63] House of Commons. Parlimentary Question Regarding Trident Missiles, 2008. URL `http://www.publications.parliament.uk/pa/cm200708/cmhansrd/cm080714/text/80714w0036.htm`.

[64] M. E. C. Hull, K. Jackson, and J. Dick. *Requirements Engineering, Second Edition*. Springer, 2005. ISBN 978-1-85233-879-4.

[65] IBM. IBM rational DOORS, 2014, accessed online 30/10/2014. URL `http://www-03.ibm.com/software/products/en/ratidoor`.

[66] IEC. 61508 functional safety of electrical/electronic/programmable electronic safety-related systems. *International Electrotechnical Commission*, 1998.

[67] IEEE. *IEEE Recommended Practice for Software Requirements Specifications - IEEE Std 830-1998*. Oct 1998. doi:10.1109/IEEESTD.1998.88286.

[68] IEEE. A guide to the project management body of knowledge. *IEEE Std 1490-2011*, pages 1–508, Nov 2011. doi:10.1109/IEEESTD.2011.6086685.

[69] INCOSE. What is systems engineering?, Oct. 2014, accessed online 14/10/2014. URL `http://www.incose.org/practice/whatissystemseng.aspx`.

[70] International Electrotechnical Commission. Hazard and operability studies (HAZOP studies)–application guide. [IEC 61882]. *Geneva: International Electrotechnical Commission*, 2001.

[71] International Nuclear Safety Advisory Group. *Defence in Depth in Nuclear Safety*. INSAG Series. International Atomic Energy Agency, 1996. ISBN 9789201025968. URL `http://www-pub.iaea.org/MTCD/publications/PDF/Pub1013e_web.pdf`.

[72] International Organization for Standardization. ISO 15288 - Systems and software engineering – system life cycle processes. 2008. URL `http://www.iso.org/iso/catalogue_detail?csnumber=43564`.

[73] C. Johnson. Use of visual propositional calculus to derive safety critical functions. In *System Safety Conference incorporating the Cyber Security Conference 2013, 8th IET International*, pages 1–6, Oct 2013. URL `http://dx.doi.org/10.1049/cp.2013.1703`.

[74] C. R. Johnson. Methodology for Designing and Analyzing High Consequence Arming Systems. In J. M. Livingston, R. Barnes, D. Swallom, and W. Pottraz, editors, *Proceedings of the US Joint Weapons Systems Safety Conference 2009, Huntsville, Alabama*, pages 552–561. The International System Safety Society, 2009. ISBN 9781617387142.

[75] M. Jones. The CBRN threat and lessons to be learned from high consequence safety. In *Proceedings of the 21st International Safety Conference*, Ottawa, 2003. The International System Safety Society.

[76] M. Jones. Issues relating to the Number of Safety Subsystems Appropriate to a High Consequence System. In *In proceedings of the 23rd International System Safety Conference*, San Diego, CA, 2005. The International System Safety Society.

[77] R. Kalawsky, Y. Tian, D. Joannou, I. Sanduka, and M. Masin. Incorporating architecture patterns in a sos optimization framework. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1726–1731, Oct 2013. URL `http://dx.doi.org/10.1109/SMC.2013.297`.

[78] J. E. Kasser and D. K. Hitchins. Unifying the different systems engineering processes. In *Proceedings of the 8th Conference on Systems Engineering Research*, Hoboken, NJ, (United States), March 2010. Stevens Institute of Technology.

[79] T. Kelly. A systematic approach to safety case management. In *Proceedings of SAE 2004 World Congress*, Detroit, MI, 2004. Society of Automotive Engineers. URL `http://dx.doi.org/doi:10.4271/2004-01-1779`.

[80] T. Kelly, I. Bate, J. McDermid, and A. Burns. Building a preliminary safety case: An example from aerospace. Sydney, Australia, Oct. 1997. Australian Computer Society.

[81] R. E. Kidder. *Report to Congress: Assessment of the Safety of US Nuclear Weapons and Related Nuclear Test Requirements.* Lawrence Livermore National Laboratory, 1991.

[82] M. Kossmann and M. Odeh. Ontology-driven requirements engineering–a case study of OntoREM in the aerospace context. In *Proceedings of the INCOSE Systems Engineering Conference*. INCOSE, 2010. URL `http://eprints.uwe.ac.uk/13172/`.

[83] M. Kossmann, R. Wong, M. Odeh, and A. Gillies. Ontology-driven requirements engineering: Building the ontorem meta model. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6, April 2008. doi:10.1109/ICTTA.2008.4530315.

[84] G. Lami. QuARS: A tool for analyzing requirements. Technical report, Carnegie Mellon Software Engineering Institute, 2005. URL `http://www.dtic.mil/dtic/tr/fulltext/u2/a441307.pdf`.

[85] A. Lamsweerde. Formal specification: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 147–159. ACM, 2000. URL `http://doi.acm.org/10.1145/336512.336546`.

[86] C. Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 3 edition, 2004. ISBN 978-0131489066.

[87] S. Li, J. Li, B. Suo, and L. Y. Xiao. Safety design of high consequence systems based on first principles. In *Advances in Safety, Reliability and Risk Management*, pages 1218–1222. CRC Press, Aug. 2011. ISBN 978-0-415-68379-1. URL `http://dx.doi.org/10.1201/b11433-172`.

[88] M. Luisa, F. Mariangela, and I. Pierluigi. Market research for requirements analysis using linguistic tools. volume 9, pages 40–56. Springer-Verlag New York, Inc., Secaucus, NJ, USA, Feb. 2004. URL `http://dx.doi.org/10.1007/s00766-003-0179-8`.

[89] R. Maguire. *Safety Cases and Safety Reports: Meaning, Motivation, and Management*. Ashgate Publishing, 2006. ISBN 978-0-7546-4649-5.

[90] A. Mavin and P. Wilkinson. Big Ears (The Return of "Easy Approach to Requirements Engineering"). In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 277–282, 2010. doi:10.1109/RE.2010.39.

[91] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak. Easy Approach to Requirements Syntax (EARS). In *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, pages 317–322, 2009. doi:10.1109/RE.2009.9.

[92] A. A. McEwan and J. C. P. Woodcock. A Verified Implementation of a Control System. In *The Military and Aerospace Programmable Logic Device*, 2005. URL `http://klabs.org/mapld05/papers/185_mcewan_paper.pdf`.

[93] J. Medalia. Nuclear Warheads: The Reliable Replacement Warhead Program and the Life Extension Program. Federation of American Scientists, 2007. URL `http://fas.org/sgp/crs/nuke/RL33748.pdf`.

[94] G. Meszaros and J. Doble. Pattern languages of program design 3. chapter A Pattern Language for Pattern Writing, pages 529–574. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. ISBN 0-201-31011-2. URL `http://dl.acm.org/citation.cfm?id=273448.273487`.

[95] G. H. Miller, P. S. Brown, and C. T. Alonso. Report to Congress on stockpile reliability, weapon remanufacture, and the role of nuclear testing. Technical report, Lawrence Livermore National Lab., CA (USA), 1987.

[96] Ministry of Defence. JSP538 - Regulation of the Nuclear Weapon Programme, 2008.

[97] Ministry of Defence. JSP372 -Approving Authority Management Arrangements for the Trident Re-entry System, 2011.

[98] T. Murray. On the limits of refinement-testing for model-checking CSP. *Formal Aspects of Computing*, 25(2):219–256, 2013. ISSN 0934-5043. URL `http://dx.doi.org/10.1007/s00165-011-0183-6`.

[99] R. J. Nikolic. Extending the Life of an Aging Weapon. Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2012.

[100] Nuclear Movements and Nuclear Accident Response Group. Operational Safety Case for Transport of Nuclear Weapons Executive Summary, 2005. URL `http://www.newscientist.com/data/images/ns/av/mg19125594.300.pdf`. Defence Logistics Organisation.

[101] K. O'Nions, R. Pitman, and C. Marsh. Science of nuclear warheads. volume 415, pages 853–857. Nature Publishing Group, 2002. URL `http://dx.doi.org/10.1038/415853a`.

[102] N. Plat and P. G. Larsen. An overview of the iso/vdm-sl standard. *ACM SIGPLAN Notices*, 27(8):76–82, Aug. 1992. ISSN 0362-1340. URL `http://doi.acm.org/10.1145/142137.142153`.

[103] D. W. Plummer and W. H. Greenwood. A primer on unique signal stronglinks. Technical report, Albuquerque, NM, 1993.

[104] D. W. Plummer and W. H. Greenwood. The History of Nuclear Weapon Safety Devices. In *AIAA/ASME/SAE/ASEE joint propulsion conference*, Cleveland, OH, July 1998. URL `http://dx.doi.org/10.2514/6.1998-3464`.

[105] M. J. Pont. *Patterns for time-triggered embedded systems: building reliable applications with the 8051 family of microcontrollers*. ACM Press/Addison-Wesley Publishing Co., 2001.

[106] S. Pugh. *Total design: integrated methods for successful product engineering*. Addison-Wesley Wokingham, 1991.

[107] A. Pyster, D. Olwell, N. Hutchison, and S. Enck. Guide to the Systems Engineering Body of Knowledge (SEBoK), 2013. URL `http://www.sebokwiki.org/`.

[108] G. T. Randall. High Consequence System Surety process description. Office of Scientific and Technical Information, Sept. 1995. URL `http://dx.doi.org/10.2172/106598`.

[109] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. In *Automata, Languages and Programming*, pages 314–323. Springer, 1986.

[110] L. Rising. Patterns: a way to reuse expertise. *Communications Magazine, IEEE*, 37(4):34–36, Apr 1999. ISSN 0163-6804. doi:10.1109/35.755446.

[111] S. Robertson and J. Robertson. *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley, 2012. ISBN 9780321815743.

[112] A. W. Roscoe and C. A. R. Hoare. The laws of occam programming. volume 60, pages 177–229. Elsevier, 1988. URL `http://dx.doi.org/10.1016/0304-3975(88)90049-7`.

[113] A. W. Roscoe, C. A. Hoare, and R. Bird. *The theory and practice of concurrency*, volume 169. Prentice Hall Englewood Cliffs, 1998.

[114] L. Rosenberg. Generating high quality requirements. In *AIAA Space 2001 Conference and Exposition*, SPACE Conferences & Exposition. American Institute of Aeronautics and Astronautics, Aug. 2001. URL `http://dx.doi.org/10.2514/6.2001-4524`.

[115] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language reference manual*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1999. ISBN 0-201-30998-X.

[116] T. L. Saaty. How to make a decision: the analytic hierarchy process. In *European Journal of Operational Research*, volume 48, pages 9–26. Elsevier, 1990. URL http://dx.doi.org/10.1016/0377-2217(90)90057-I.

[117] G. A. Sanders. Less than severe worst case accidents. In *National System Safety Conference*, Albuquerque, NM (United States), Aug 1996. Office of Scientific and Technical Information. URL http://www.osti.gov/scitech/servlets/purl/270731.

[118] S. Schneider. *Modelling security properties with CSP (Technical report)*. University of London, Royal Holloway, Department of Computer Science, 1996. URL http://www.computing.surrey.ac.uk/personal/st/S.Schneider/papers/secprop.pdf.

[119] S. Schneider, A. Cavalcanti, H. Treharne, and J. Woodcock. A layered behavioural model of platelets. In *Engineering of Complex Computer Systems, 2006. ICECCS 2006. 11th IEEE International Conference on*. IEEE, 2006. URL http://dx.doi.org/10.1109/ICECCS.2006.1690359.

[120] M. Shibaoka, H. Kaiya, and M. Saeki. GOORE : Goal-Oriented and Ontology Driven Requirements Elicitation Method. In J.-L. Hainaut, E. Rundensteiner, M. Kirchberg, M. Bertolotto, M. Brochhausen, Y.-P. Chen, S.-S. Cherfi, M. Doerr, H. Han, S. Hartmann, J. Parsons, G. Poels, C. Rolland, J. Trujillo, E. Yu, and E. Zimányie, editors, *Advances in Conceptual Modeling – Foundations and Applications*, volume 4802 of *Lecture Notes in Computer Science*, pages 225–234. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-76291-1. URL http://dx.doi.org/10.1007/978-3-540-76292-8_28.

[121] K. Siegemund, E. J. Thomas, Y. Zhao, J. Pan, and U. Assmann. Towards ontology-driven requirements engineering. In *10th International Semantic Web Conference (ISWC)*, Bonn, Germany, Oct. 2011. URL http://iswc2011.semanticweb.org/fileadmin/iswc/papers/workshops/swese/4.pdf.

[122] D. Slipper and A. A. McEwan. A Systems Re-engineering Case Study: Programming Robots with occam and Handel-C. In P. H. Welch, A. T. Sampson, J. B. Pedersen, J. Kerridge, F. R. M. Barnes, and J. F. Broenink, editors, *Communicating Process Architectures 2011*, pages 317–327. IOS Press, 2011. URL http://www.wotug.org/papers/CPA-2011/SlipperMcEwan11/SlipperMcEwan11.pdf.

[123] M. S. Soares and J. Vrancken. Model-driven user requirements specification using SysML. In *Journal of Software*, volume 3. Academy Publisher, 2008. URL http://dx.doi.org/doi:10.4304/jsw.3.6.57-68.

[124] I. Sommerville. *Software Engineering*. International Computer Science Series. Pearson/Addison-Wesley, 2011. ISBN 9780137053469.

[125] I. Sommerville and P. Sawyer. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1 edition, 1997. ISBN 978-0471974444.

[126] J. M. Spivey. An introduction to Z and formal specifications. In *Software Engineering Journal*, volume 4, pages 40–50. IET, 1989. URL `http://dx.doi.org/10.1049/sej.1989.0006`.

[127] S. Spray and J. Cooper. Passive safety concepts applied to critical functions. In *Joint American Society of Mechanical Engineers (ASME)/Japan Society of Mechanical Engineers (JSME) pressure vessels and piping conference*, Honolulu, HI (United States), July 1995. Office of Scientific and Technical Information. URL `http://www.osti.gov/scitech/biblio/87057`.

[128] S. D. Spray and J. A. Cooper. The unique signal concept for detonation safety in nuclear weapons. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1993. URL `http://dx.doi.org/10.2172/10177801`.

[129] S. D. Spray and J. A. Cooper. An examination of the consequences in high consequence operations. In *Proceedings of the International conference on probabilistic safety assessment and management*, Crete (Greece), 1996. URL `http://www.osti.gov/scitech/biblio/242627`.

[130] S. D. Spray and J. A. Cooper. Structured Design and Assessment of Safety Systems Based on Principles. In *Probabilistic safety assessment and management, PSAM 4*, pages 1597–1602. Springer, 1998. ISBN 3540762620.

[131] N. N. Taleb. *The Black Swan: The Impact of the Highly Improbable*. Penguin Books Limited, 2008. ISBN 9780141906201.

[132] S. F. Tjong, N. Hallam, and M. Hartley. Improving the Quality of Natural Language Requirements Specifications through Natural Language Requirements Patterns. In *The Sixth IEEE International Conference on Computer and Information Technology*, page 199. IEEE, 2006. URL `http://dx.doi.org/10.1109/CIT.2006.103`.

[133] A. D. Toro, B. B. Jiménez, A. R. Cortés, and M. T. Bonilla. A requirements elicitation approach based in templates and patterns. In *Ibero-American Workshop on Requirements Engineering*, pages 17–29, Buenos Aires, Argentina, 1999.

[134] R. Traeger and E. Ehrman. The Lightning Arrestor Connector. *Parts, Hybrids, and Packaging, IEEE Transactions on*, 12(2):89–94, 1976. ISSN 0361-1000. doi:10.1109/TPHP.1976.1135121.

[135] J. Woodcock, A. Cavalcanti, J. Fitzgerald, P. Larsen, A. Miyazawa, and S. Perry. Features of CML: a formal modelling language for systems of systems. In *System of Systems Engineering (SoSE), 2012 7th International Conference on*, pages 1–6. IEEE, 2012.