
COMMONWEALTH OF AUSTRALIA

AUSTRALIAN DEFENCE STANDARD

# DEF(AUST) 5679

# THE PROCUREMENT OF COMPUTER-BASED SAFETY CRITICAL SYSTEMS

*

PUBLISHED UNDER AUTHORITY
OF DEPARTMENT OF DEFENCE

**DEF(AUST)5679**

---

**The following Government and industry organisations were consulted during the preparation of this Standard:**

**Electronic Warfare and Radar Systems Branch,**
**Defence Acquisition Organisation,**
**Australian Ordnance Council, Support Command Australia**

**Defence Safety Critical Systems and Software Working Group**

**Prepared by:** **Information Technology Division**
**Defence Science and Technology Organisation**
**PO Box 1500**
**SALISBURY SA 5108**

---

**Published by:**
Army Standardisation (AEA)
(Interim publisher)

**Sponsored by:**
Defence Science and Technology
Organisation

# AMENDMENT LIST

| AMENDMENT | | EFFECTED | |
|---|---|---|---|
| **NO** | **DATED** | **SIGNATURE** | **DATE** |
| | | | |

BLANK

**AUSTRALIAN DEFENCE STANDARD**

**DEF(AUST) 5679**

# PROCUREMENT OF COMPUTER-BASED SAFETY CRITICAL SYSTEMS

**STANDARD**

**AUGUST 1998**

---

**Approved by the Defence Standardisation Coordination Group.**

**Specific inquiries regarding the application of this Standard to Requests for Tender or contracts should be addressed to the Ordering Authority named in the Request for Tender, or to the Safety Authority named in the contract, as appropriate.**

---

# *TABLE OF CONTENTS*

# PART ONE

# BACKGROUND

BLANK

# 1 INTRODUCTION

## 1.1 Aim

1.1.1 This Standard presents requirements and guidance for the procurement of computer-based safety critical systems in Defence.

1.1.2 A *safety critical system* is one in which a failure could result in death or injury.

1.1.3 Computer-based components such as software, firmware and application-specific integrated circuits (ASICs) are playing an increasing role in systems being developed today, and are key components of a number of new capabilities currently being integrated into the Australian Defence Force (ADF). In these systems, safety may in part be dependent on the trustworthiness of such components.

1.1.4 The aim of this Standard is to describe procedures and practices to be carried out during system development that:

- provide assurance that the system will operate in its intended environment with an acceptable level of safety risk; and
- are appropriately rigorous relative to the perceived risk from the system.

1.1.5 Conformance to this Standard shall provide evidence that sufficiently rigorous development procedures and technical practices have been applied in trying to achieve a safe system.

## 1.2 Scope

1.2.1 The approach to be described in this Standard has evolved from that taken by the Australian Ordnance Council (AOC) Pillar Proceeding 223.93 [1] entitled "The Assessment of Munitions-Related Safety Critical Computing Systems". However, the scope of application of this Standard is computer-based safety critical systems generally - not just munition-related systems.

1.2.2 Software is increasingly being used to develop systems. It can provide an inexpensive and flexible means for introducing very powerful and complex features. This Standard is aimed at safety critical systems containing software, because it is these systems which present the greatest challenge to safety assurance. It is also aimed at systems that contain components that are developed with methods similar to software development methods. Such components include firmware, application-specific integrated circuits (ASICs), programmable logic controllers (PLCs), programmable gate arrays (PGAs) and other custom hardware.

1.2.3 This Standard applies to the Customer, Developer, Auditor and Evaluator of the system[1].

1.2.4 This Standard also applies to systems for which development began prior to the publication of this Standard, as well as to systems procured from overseas suppliers, and commercial off-the-shelf products (COTS). Such systems are called Non-Development Items (NDIs), and Section 11 gives specific requirements for their procurement.

1.2.5 For the purposes of this Standard, the following kinds of systems are to be considered safety critical:

[1] These terms are defined in Section 4.

- any munition-related system that controls or directly influences the prearming, arming, enabling, release, launch, firing, flight path or detonation of a munition system, including target identification, selection and designation;
- any system that controls or directly influences the movement of gun mounts, launchers, and other equipment, especially with respect to the aiming and firing safety of that equipment;
- any computer-based combat system;
- any system that controls or directly influences the movement of munitions and/or hazardous materials;
- any system that monitors the state of another system for safety purposes;
- any system that controls, regulates or contains potentially dangerous energy sources;
- any system used to compute safety critical data (including applications software that may not be connected to or directly control a safety critical hardware system, such as stress analysis programs);
- any system that collects, stores, manipulates, and reports or displays data that may be safety-critical in nature;
- any system that controls or partially controls the movement of a vehicle (e.g. ship, aircraft, land-based craft, radar-guided objects);
- any system that controls or partially controls potentially dangerous moving parts of equipment to which personnel or members of the public may come in close proximity.

1.2.6 For the purposes of this Standard, the following kinds of systems may be considered *not* to be safety critical:

- Software systems that do not control or interact with any physical devices, except the special purpose platform (and accessories) on which they are executed, and which do not process safety critical data. This includes software packages running on stand-alone or network computers, such as financial, project management, communications, software analysis packages etc.

1.2.7 Systems that are not included in the lists given in 1.2.5 and 1.2.6 must be considered safety critical until it has been proved otherwise (namely, by means of a Preliminary Hazard Analysis demonstrating that system behaviour cannot lead to any accidents).

## 1.3 Structure of this Standard

1.3.1 Figure 1 is a pictorial overview of the structure of this Standard, which groups the sections of the Standard according to the aspects of safety to which they relate.

1.3.2 **Part One** presents some background material. After this Introduction, Section 2 provides an introduction to the issues in safety management of the development and procurement of computer-based systems. These issues include duty of care, the need for a uniform approach to safety, and the special problems for safety that software and other computer-based components represent.

1.3.3 Section 3 contains the safety principles on which this Standard is built. These should be clearly understood and accepted by all parties to the system development process.

1.3.4 **Part Two** of this Standard covers aspects of the Safety Management Process. Sections 4 and 5 contain management requirements and guidance, including the roles of Safety Management Agents and an overview of the Safety Management Process.

1.3.5 Section 6 provides general requirements for System Development, while in Section 7 the overall requirements for Safety Analysis are presented.

1.3.6 Section 8 provides requirements for System Safety Reviews.

1.3.7 Section 9 describes the role of the System Evaluator. Installation and Maintenance issues are discussed in Section 10. Finally, the procurement of overseas-supplied systems and other non-development systems is discussed in Section 11.

1.3.8 Finally, in **Part Three,** we describe the detailed requirements for the structure of the Safety Case. Section 12 is an overview of the structure of the System Safety Case. The following Sections 13 to 17 are more technical in their content, and describe requirements and guidance for the development of the Safety Case.

1.3.9 ANNEX A contains a list of acronyms used.

1.3.10 ANNEX B contains a glossary of terms used.

1.3.11 ANNEX C contains a list of all deliverable documents required by this Standard.

1.3.12 ANNEX D contains a list of document references in this Standard.

1.3.13 ANNEX E is an index.

## 1.4 Requirements vs Guidance

1.4.1 This Standard contains both *requirements* and *guidance* for the development and evaluation of safety critical systems.

**1.4.2 Paragraphs containing requirements will be shown (like this one) in bold face. Compliance with these paragraphs is required in order to demonstrate that the demands of this Standard are met.**

1.4.3 Paragraphs that contain guidance will be shown in regular type face (like this one). Such paragraphs need not be specifically met in order to satisfy this Standard. However, they contain material that should be of assistance in meeting requirements.

## 1.5 Documents Required by this Standard

1.5.1 This Standard requires certain reports and documents to be produced during the production of the Safety Case and its Evaluation. A list of all these documents is given in Annex C.

1.5.2 These documents need not necessarily be distinct: two reports on Safety Case activities could, for example, be combined as part of a larger document, provided that sufficient detail is given to allow Evaluation to be carried out.

1.5.3 Some of the Safety Case documents may be identified with other documents produced during development (such as design documentation), as long as they contain the necessary information required by this Standard.

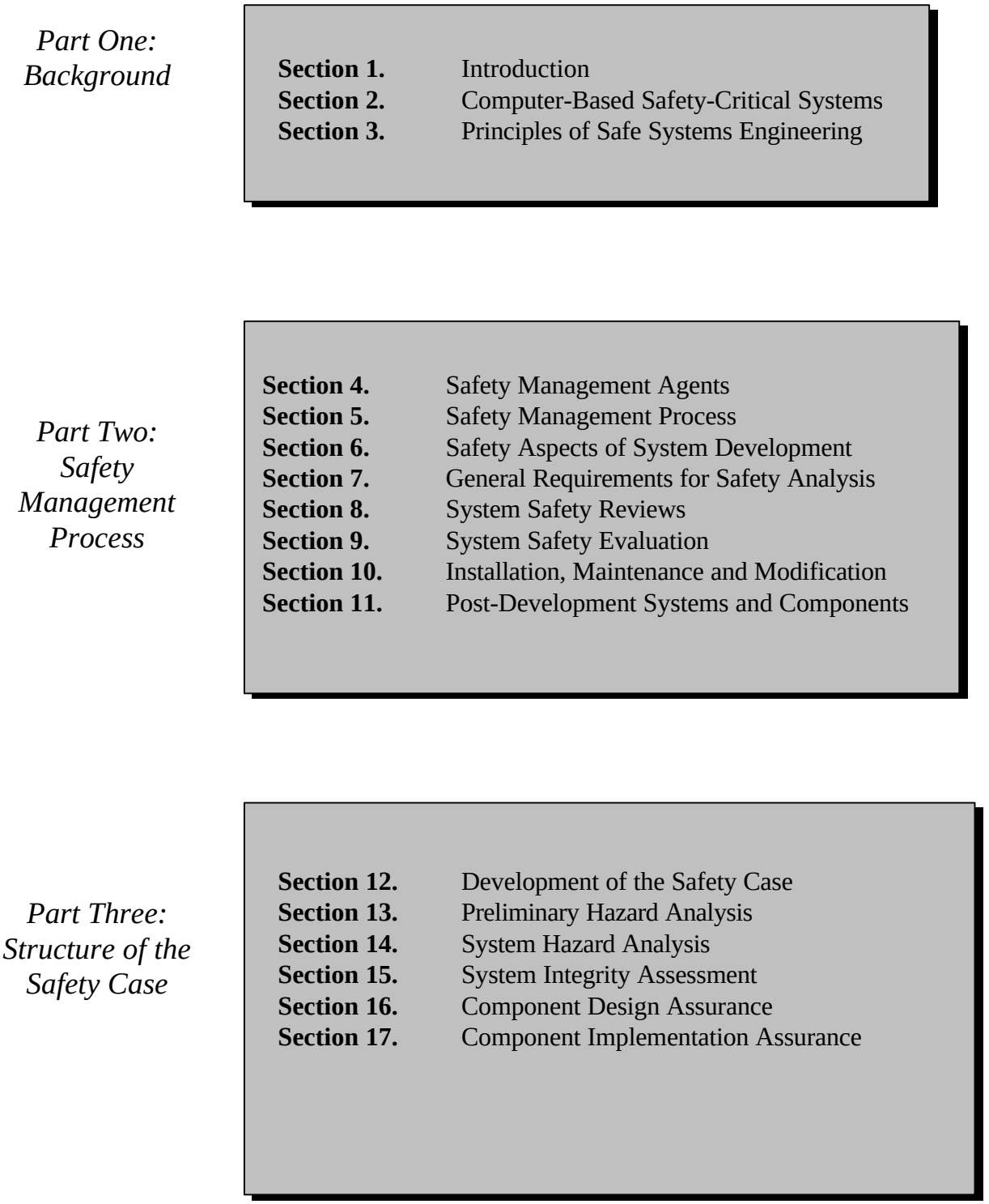


**Figure 1: Overview of Standard**

# 2 COMPUTER-BASED SAFETY CRITICAL SYSTEMS

## 2.1 Duty of Care

2.1.1 There is no totally reliable method of eliminating risk from the operation of safety critical systems. However, when systems are developed in ignorance of safety issues, the probability of deaths, or injuries, resulting from design flaws in the system increases. Both system developers and customers must be aware of the risks of using software and other computer-based components in safety critical systems being procured for the Australian Defence Organisation (ADO), and they must take appropriate measures to assure that such systems are appropriately safe. This Standard prescribes the activities that must be carried out during system development to provide a level of assurance commensurate with the perceived level of risk associated with system failure.

2.1.2 In Australia, the recent Occupational Health and Safety (OH & S) legislation [2] has focused attention on these issues. Those with the authority to develop, approve and accept into use safety critical systems have a duty of care arising from their legal obligation to take reasonable precautions to avoid reasonably foreseen and significant risk of danger to members of the public as well as their own employees. A breach of this duty could make them liable in the case of an accident.

## 2.2 A Uniform Approach to Safety

2.2.1 Recent years have seen a number of differing standards and guidelines established, particularly in the defence industry ([3], [5]), that apply to software in safety critical systems. The purpose of a standard is to act as a point of reference for approved practice, to be followed by software developers, and to enable a certifying body to determine if good practice has been used. It is important that such standards require a responsible attitude towards safety and specify the use of thorough and sound engineering and management methodologies.

2.2.2 Unfortunately, when it comes to standards, there is no international consensus as to how safety critical systems should be developed and evaluated. Moreover, there has not been a uniform approach to safety issues within the ADO.

2.2.3 The aim of this standard is to establish a uniform approach for the ADO towards procurement of computer-based safety critical systems that:

- integrates computer-based safety management with system-wide safety;
- specifies a level of rigour for system development and analysis that is appropriate to the critical nature of system components; and
- is not over prescriptive about development processes and methods to be used.

2.2.4 The principles contained in this Standard can be expected to be refined as experience is

2.2.5 gained and as other national and international standards appear.

## 2.3 Software and Safety Assurance

2.3.1 The implementation of system functions by software represents some unique risks to safety. Firstly, the flexibility of programming languages and the power of hardware computing elements such as current microprocessors means that a high level of complexity is easily introduced, thus making it harder to predict the behaviour of equipment under software control. Secondly, software appears

superficially easy and cheap to modify. Thirdly, the interaction of other elements of the system with the software is often poorly or incompletely understood.

2.3.2 The complex nature of software means that conventional engineering methods for system development are inadequate to establish the necessarily high levels of assurance that software will behave correctly with respect to its safety critical requirements. Special techniques are needed for the development and assessment of safety critical software.

2.3.3 At the very least, sound safety and software engineering practices must be used (including structured planning and design, in-depth analysis at all levels of system development, rigorous and extensive documentation, reviews and testing). In many cases, further assurance must be gained by the adoption of specific coding practices, the establishment of a safety test programme, and the use of formal mathematical system specifications and models. It must also be stressed that safety concerns must be considered at all stages of design and implementation.

2.3.4 The development of other computer-based components such as programmable hardware and ASICs also involves design processes of a similar complexity to that of software, and thus similarly high levels of rigour need to be applied during development.

2.3.5 Safety assurance for a given system is embodied in a Safety Case, which details and justifies all safety-related decisions, and provides the evidence that the system developed satisfies safety requirements.

# 3 PRINCIPLES OF SAFE SYSTEMS ENGINEERING

## 3.1 Introduction

3.1.1 Safety critical systems need to be developed and evaluated according to sound principles of safety management. This Standard takes the following as basic principles for the development of safe systems. These principles are categorised as *management* and *technical*.

## 3.2 Management Principles

3.2.1 *Systems shall be assumed to be safety critical unless demonstrated otherwise.*

The aim of this principle is to avert situations where a system is procured and used in service without its safety critical nature being realised and where it is implicitly assumed that a Standard such as this does not apply. The duty of care which Customers and Developers have requires that it be explicitly demonstrated by referring to 1.2.6, or by means of hazard analysis (see Section 7), that a system is not safety critical.

3.2.2 *Safety issues should be addressed early in the development lifecycle, and tracked throughout.*

This is of crucial importance. It is rarely possible for safety to be introduced as an afterthought. It is almost impossible to demonstrate the safety of a system for which safety requirements were not identified at the outset, and in which the design took no account of safety issues. Paying early attention to safety issues is the best way of ensuring that costly maintenance or re-engineering is not required later in the implementation and installation phases. Also, it has been shown that system faults usually result from deficiencies in requirements specifications [7]. Early consideration of safety issues may produce a design that entirely eliminates some hazards. For these reasons, safety management must be applied stringently from the earliest stages of development, and addressed consistently throughout system development.

3.2.3 *Safety assurance requires visibility of both the product and the process.*

Visibility means that the Developer's approach to system development, and all products of it, such as source code and documentation, must be made freely available to other parties involved in safety engineering and management. Visibility ensures that safety-related decisions are communicated effectively to others, and that their comments and criticisms can be fed back into the development process. The Developer's concern to protect proprietary information can be met by the other members of the Safety Management Group (see Section 4.8) entering into non-disclosure agreements, if necessary. However, the requirement for visibility is paramount if good safety management is to be applied.

3.2.4 *Safety is best achieved through a diversity of people, skills and techniques.*

Safety engineering requires a range of skills: sound engineering judgement; domain knowledge; knowledge of appropriate testing methods; mathematical expertise in software verification etc. Such skills do not usually reside in a single person, or perhaps even a single organisation. Also, assurance is increased by having others check specifications, designs, code, clarity of documents, etc. Diversity is very important; it is equally important that it be effectively managed.

3.2.5 *Safety demands the commitment and co-operation of all parties.*

The safety management process is very vulnerable. Successful safety engineering requires that all parties contribute fully, and do not withhold important information. It is vital that safety management meetings are run correctly, and that 'factional' issues do not affect safety deliberations.

3.2.6 *Safety assurance arguments require independent checking and review*.

This principle stresses the fact that true *independence* of checking and review can reveal problems overlooked by the team which wrote the specifications, carried out proofs, etc. Such problems can arise from hidden assumptions, misinterpretation of requirements, inconsistencies in arguments, etc. Most importantly, for the most critical functions, the technical justifications for safety (including formal specifications and proofs) must be checked and ratified by parties independent of those who performed them originally.

3.2.7 *Safety assurance must be transferable.*

Because safety management involves a number of agents, and uses people with a range of skills, it is essential to be able to convince others (perhaps without specialist knowledge) of one's conviction that the system is safe. In the event of a serious accident, it will be necessary to convince investigating bodies how best practice was applied, and explain how safety conclusions were reached. In particular, it is essential to show that design alternatives were explicitly considered, and that the eventual choice between these designs took account of safety issues.

3.2.8 *Safety must be demonstrated by means of an auditable and repeatable Safety Case.*

The Safety Case is the Developer's 'defence' that the system is safe. The detailed arguments for safety must be made in such a way so as to provide to the Evaluator and the Auditor a self-contained record of hazards associated with the system requirements and design, together with mitigations for each hazard.

3.3 Technical Principles

3.3.1 *Safety is best achieved by using tried and trusted techniques, where possible*.

The engineering of safe systems necessitates a balanced approach that combines vigilant safety management, modern systems and software engineering best practice, and extensive domain knowledge with a range of rigorous development and analysis.

The development process should use established industry practices, previous project experience, and the experience of the people involved. Extreme care must be taken for novel designs of systems, and the use of software to implement safety critical functions must be carefully considered. The use of any novel methods or technologies is discouraged, and any such use must be justified as providing sufficient assurance of safety.

3.3.2 *Safety is best achieved by means of an iterative, continuous and evolutionary process*.

This principle asserts the fact that system design can, and must, be influenced by safety concerns. A * design must be carried out with safety in mind; however, subsequent safety arguments (making up an initial 'safety case') might reveal that the design is flawed in some way from the safety point of view and needs to be modified. The system development cycle must make explicit the need for a number of safety reviews that can lead to changes in the design. The precise iterative process is explained later in this Standard.

3.3.3 *Safety assurance is best achieved if critical functions are as simple as possible*.

The design should be as simple as possible so as to make it easy to understand and to reduce the effort involved in assuring system safety. Software should also be kept simple, avoiding unnecessary or exotic features of any programming language, or the underlying hardware.

3.3.4 *Safety assurance is best achieved if critical functions are isolated from the rest of the system.*

The critical functions should be kept functionally and physically distinct from other system functions, wherever possible. This will simplify hazard and other analysis, and again reduce the effort involved in assuring system safety. More importantly, it is one of the most cost-effective known techniques for achieving designs that are inherently safe.

*3.3.5 Safety assurance requires, in the most critical cases, the use of formal methods.*

Testing can uncover errors, but is not sufficient to prove their absence. In the most critical cases, where high assurance is required, the system must be proved mathematically to satisfy its critical requirements. This requires formal specification and verification, of design and/or code, according to the criticality.

BLANK

# PART TWO

# THE SAFETY MANAGEMENT PROCESS

BLANK

# 4 SAFETY MANAGEMENT AGENTS

## 4.1 Introduction

4.1.1 The process of assuring the safety of a system throughout its lifecycle involves at least four principal agents: the Customer, the Developer, the Auditor and the Evaluator. It is also necessary to involve End Users. In many cases, a Certifier will need to accept the system as being safe and suitable for service.

## 4.2 Customer

4.2.1 The Customer is the organisation (usually a Project Office) within the ADO that is procuring the system.

**4.2.2 The Customer is responsible for procuring a system that is appropriately safe, and so must oversee the overall process of safety management. The Customer is also responsible for accurately specifying the operational requirements and environment for the system, as well as identifying End Users of the system. Operational requirements defined by the Customer may need to be revisited in the light of subsequent safety analysis.**

**4.2.3 The Customer is responsible for ensuring that the Developer is contractually bound to conform to this Standard.**

## 4.3 Developer

4.3.1 The Developer is the organisation (usually a commercial one) which is responsible for producing the system required by the Customer according to this Standard.

**4.3.2 The Developer is responsible for delivering a system for which sufficient assurance of safety has been achieved by means of a Safety Case.**

**4.3.3 Other organisations may be involved in system development and safety analysis. In such cases, the prime contractor will be regarded as the Developer for the purposes of this Standard. The prime contractor must ensure that sub-contractors meet applicable requirements in this Standard.**

**4.3.4 In some cases the Customer and Developer may be the same organisation (with contractors carrying out the development of components of the system). In such cases, the Customer must develop a Safety Case which will be evaluated at the time of purchase by any Defence organisation, and must define for each contract, the requirements of this Standard that need to be applied.**

**4.3.5 The Developer shall provide appropriately trained and knowledgeable (in safety management and procedures) personnel to carry out the development and safety analysis of the system.**

4.4 Auditor

4.4.1 The Auditor is a body, appointed by the Customer, that monitors the safety management process. The audit function will normally be carried out within the ADO, but this is not required by this Standard.

**4.4.2 The Auditor is responsible for ensuring that the procedural aspects of this Standard are followed. The Auditor shall ensure that independence is maintained, and (most importantly) act as an arbiter in the case of disputes among the other parties.**

**4.4.3 The Auditor must be independent of the Developer.**

4.5 Evaluator

4.5.1 The Evaluator is a body, appointed by the Customer with agreement from the Developer and the Auditor, that provides a thorough independent and objective review of the Safety Management Process and of the technical validity of the Safety Case.

**4.5.2 The Evaluator is responsible for checking the detail and validity of the Developer's arguments that the system will meet its safety critical requirements.**

**4.5.3 The Evaluator must be managerially and commercially independent of the Customer, Developer and Auditor.**

4.6 Certifier

4.6.1 The procurement process may also involve a Certifier. The Certifier is an organisation (such as the Australian Ordnance Council for munitions; the RAAF Directorate of Technical Airworthiness etc) that will assess safety and suitability for service of the System. The Certifier may impose specific requirements in addition to those of this Standard.

4.6.2 The roles of Certifier and Auditor may be fulfilled by the same individual or organisation.

4.7 End Users

4.7.1 The End Users of the system will typically be the intended operators, but may also include personnel that will maintain or install system equipment.

4.7.2 The End Users are a critical part of the safety engineering process, as they are able to provide additional insight to the use of the system, how they actually interact with it, and how they work around problems.

**4.7.3 All End Users must be appropriately trained and informed of safety aspects of the system.**

**4.7.4 End Users are responsible for supplying information about the Operational Context and for following operational, maintenance and installation procedures specified as a result of safety analysis.**

4.8 Safety Management Group

4.8.1 Before development begins, a *Safety Management Group* must be set up. This group comprises representatives of the Developer, Customer, Auditor and Certifier (if applicable). The function of the Safety Management Group is to provide continuing high-level management of safety issues.

**4.8.2 Through regular System Safety Reviews, the Safety Management Group must be satisfied that safety management and engineering processes have been correctly carried out, and that appropriate measures have been taken to address the risks to safety posed by the system.**

**4.8.3 The Safety Management Group shall meet regularly (more frequently during the initial stages of development). The Customer is responsible for chairing the meetings, and establishing and maintaining meeting schedules and minutes. Other parties may be invited from time to time, provided that the Auditor approves their presence.**

**4.8.4 At key checkpoints during system development, the Evaluator shall be invited to Safety Management Group meetings to hear presentations on the evolving Safety Case. The Auditor shall determine when this is appropriate.**

4.9 Summary

4.9.1 The way in which each of these agents is involved in the safety management process is summarised in Figure 2.

4.9.2 The Customer, developer and End Users are involved in defining the requirements of, and developing the system. To ensure safe design of the system, dialogue between the Customer and End Users, and the Developer must include the safety requirements of the system. In conjunction with system development, the Developer constructs the Safety Case. The Safety Case is reviewed by the Evaluator both at points during development and after delivery of the system. The Auditor checks that the whole development and safety process conforms to the Standard. The Certifier bases his assessment of safety and suitability for service on advice provided by the other agents, in particular the Evaluator and Auditor.

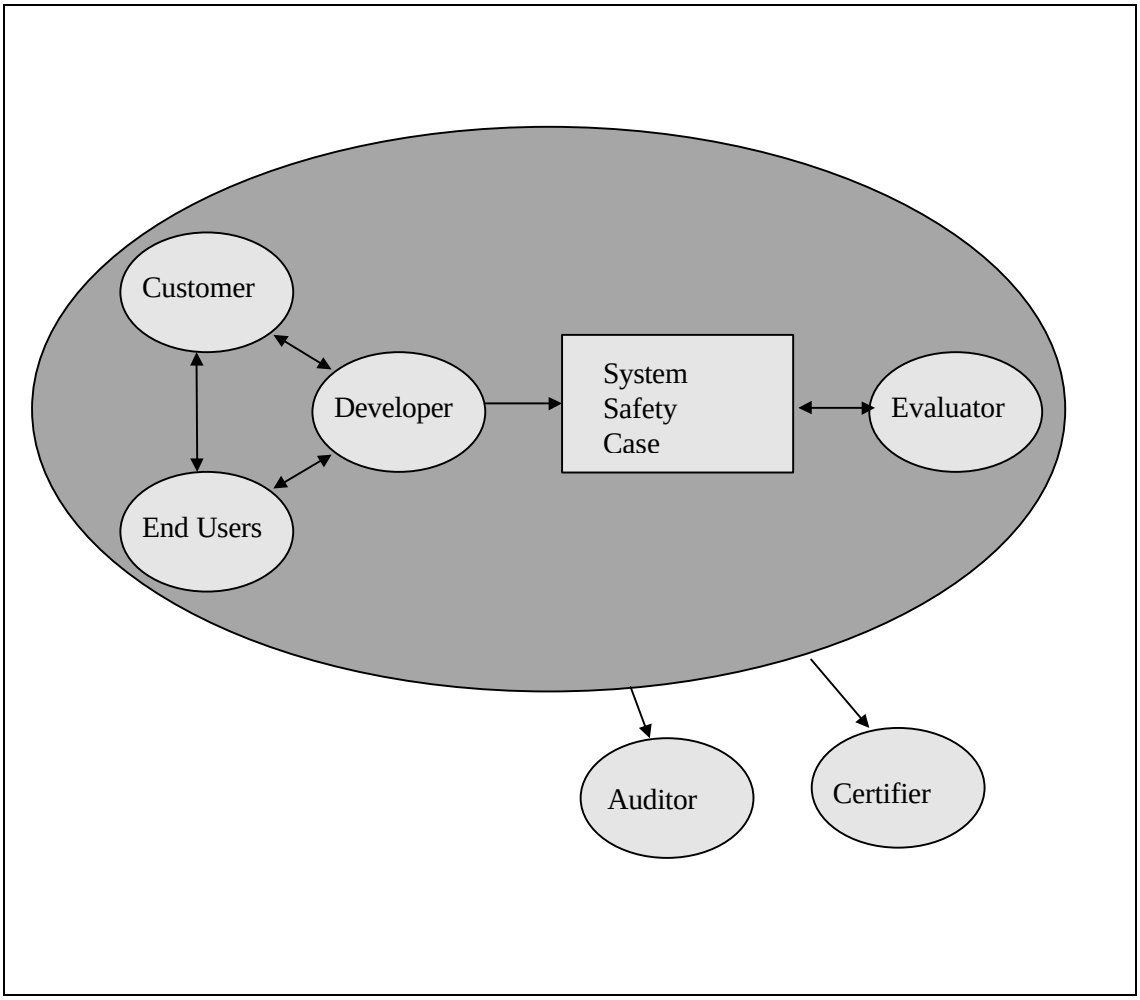


**Figure 2: Agents in the Safety Management Process**

# 5 THE SAFETY MANAGEMENT PROCESS

## 5.1 Introduction

5.1.1 Safety critical systems require a high level of assurance. All phases of the development of a system must be carried out with safety in mind. In addition, there are activities that are specifically related to safety issues. The system must not only be acceptably safe, it must be demonstrated to be acceptably safe. Many factors contribute to safety assurance and an approach to development that goes beyond ordinary quality assurance methods is required.

5.1.2 The process by which normal development phases are combined with safety-related activities to result in an assuredly safe system is called the *Safety Management Process*

5.1.3 The core of the Safety Management Process is the relationship between System Development and Safety Analysis. At key stages in development, the system as currently developed must be analysed to arrive at safety requirements and to show that any development decisions meet safety requirements. *This is an iterative process.* Results of safety analysis must be used as input to further development of the system. It may also mean that previous stages must be revisited: operational requirements may need to be changed, and previous design decisions modified. Provision must be made in all project planning for revisions of earlier system development phases.

5.1.4 The safety analysis carried out provides a major part of the construction of a Safety Case by the Developer.

5.1.5 The Safety Case shall be developed alongside the development of the system.

5.1.6 This section outlines the Safety Management Process to be followed. It is illustrated in Figure 3, which shows flow diagrams for two parallel processes: System Development and Safety Case Development. The solid arrows indicate the main chronological flow of these processes, and the dotted arrow indicates the information flow between the two processes. Further detailed descriptions of stages in the Safety Case are given in Part Three of this Standard.

## 5.2 System Development Phases

5.2.1 The System Definition phase incorporates all project activities that contribute to a well-defined description of the intended system and its requirements. Such activities include customer requirements capture and analysis, operational context definition, feasibility analysis, mission analysis and identifying the kinds of components of which the system may comprise.

5.2.2 The Preliminary Design phase incorporates all activities that lead to a high level system design.

5.2.3 The Design Development phase incorporate activities that lead to a more detailed design of each part of the system.

5.2.4 The Implementation phase incorporates all detailed design and implementation activities such as low-level circuit design, software coding, testing, preliminary production and fabrication of items, etc.

5.2.5 The Post-Development phase incorporates all phases of system production and operation that follows implementation, such as installation, maintenance, further production and fabrication, modification, etc.

## 5.3 Safety Case Development Phases

5.3.1 Preliminary Hazard Analysis consists of those activities that identify possible accidents, the potential severity of accidents, sequences of events that lead to accidents, and the System Hazards which identify how the system may contribute to an accident. Section 13 provides a detailed description of requirements for Preliminary Hazard Analysis.

5.3.2 System Hazard Analysis consists of those activities that determine the sequences of events that can cause the System Hazards identified in the Preliminary Hazard Analysis. Section 14 provides a detailed description of requirements for System Hazard Analysis.

5.3.3 System Integrity Assessment consists of those activities that determine the criticality of safety requirements for system Components, and assigns a Safety Integrity Level to them, indicating the level of rigour that must be applied to provide assurance that the requirements are met. Section 15 provides a detailed description of requirements for System Integrity Assessment.

5.3.4 Design Assurance consists of those activities that provide assurance that the system designs that are developed satisfy safety requirements. Section 16 provides a detailed description of requirements for Design Assurance.

5.3.5 Implementation Assurance consists of those activities that provide assurance that the system implementation satisfies safety requirements. Section 17 provides a detailed description of requirements for Implementation Assurance.

5.3.6 Final Review and Evaluation represents the review and evaluation activities carried out by the SMG and Evaluator to ensure that appropriate safety management and engineering processes have been carried out. Reviews and Evaluations are also carried out as part of each of the previous Safety Case Development phases. The Reviews and Evaluation may generate safety concerns requiring earlier system and safety case development stages to be revisited.

5.3.7 Any maintenance or modification of the system requires a further System Safety Review and Final Evaluation, possibly resulting in another iteration of the Safety Management Process, starting from the System Definition phase.

## 5.4 System Safety Management Plan

**5.4.1 Before system development begins, the Developer must propose and justify a System Safety Management Plan (SSMP) which details the approach to be adopted to ensure safety at all stages of the system lifecycle.**

5.4.2 An initial SSMP is proposed by the Developer before any system design takes place. The SSMP is augmented with more details when further development or analysis has been carried out.

**5.4.3 The contents of the SSMP must include the following:**

- **a *Safety Analysis Plan* (see Section 7), outlining the approach to the analysis of safety and how it will be integrated with the system development process as detailed in the *System Development Plan* (see Section 6);**

- **a description of how dependencies among sub-systems, components, and documentation will be tracked and kept consistent, and how safety analysis will be integrated with the configuration management process (see Section 6.6); and**
- **a *Safety Analysis and Review Schedule*, which details at what stages of development safety analysis, reviews and Evaluations will take place, and how this schedule will be integrated with the schedule for overall System Development.**

5.4.4 As System Development progresses, updates to the requirements or the SSMP may be necessary and shall be carried out promptly.

**5.4.5 The SSMP (and any subsequent modifications to it) must be accepted by the Safety Management Group as reflecting an appropriate approach to safety management.**

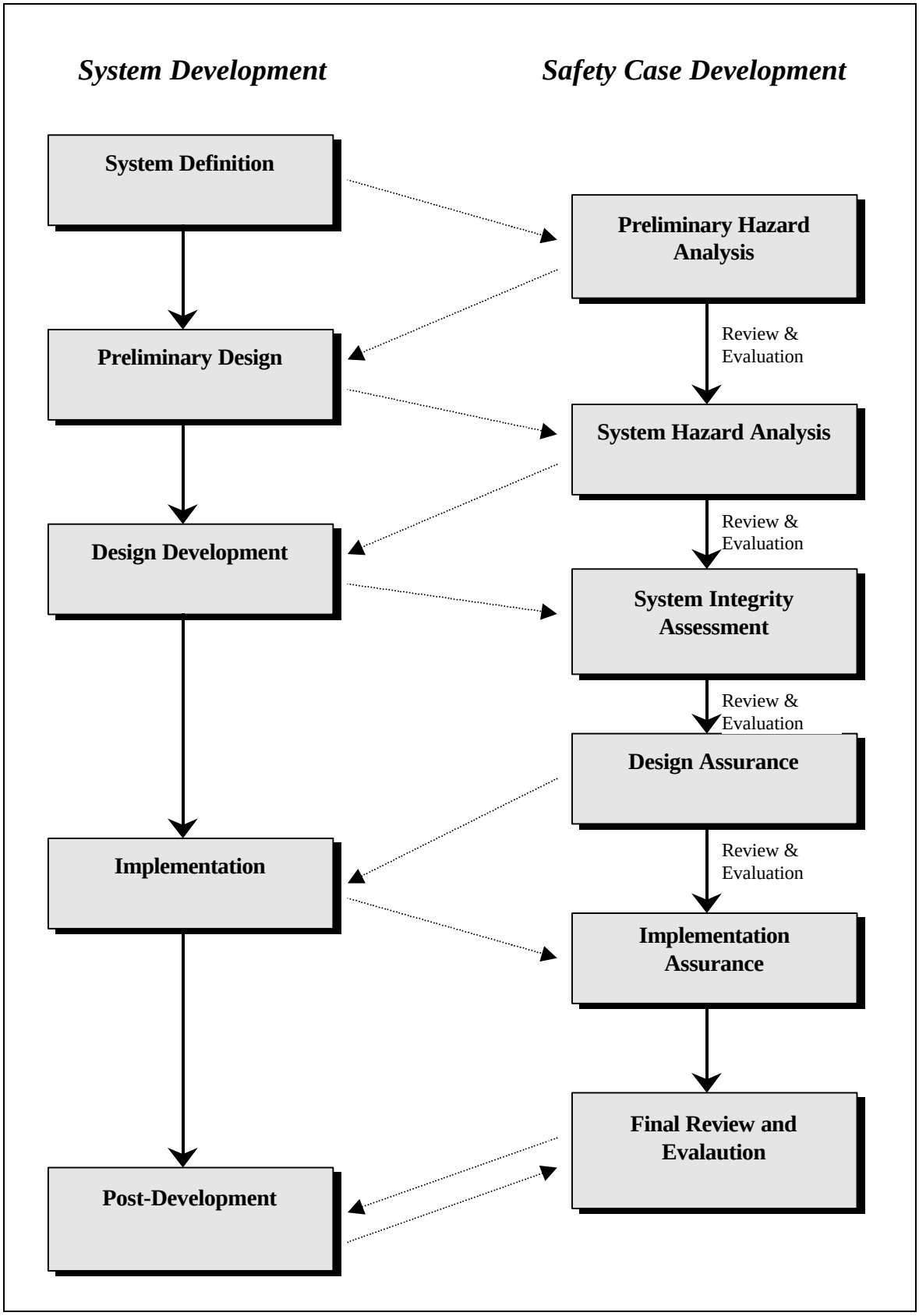


**Figure 3. The Integration of Safety Case Development with System Development.**

# 6 SAFETY ASPECTS OF SYSTEM DEVELOPMENT

## 6.1 Introduction

6.1.1 This section sets out general requirements and guidelines for the way in which system development is carried out. These requirements will ensure that the system and its development are amenable to safety analysis.

6.1.2 Developers are free to define and follow their own development processes, as long as the basic requirements of this Standard are met. This section identifies some basic development activities that must be carried out, and some basic documents that must be produced.

**6.1.3 The Developer is responsible for selecting a development process that accommodates and embraces the safety activities described in this Standard.**

**6.1.4 In many cases, a system is decomposed into several Subsystems that interact in a fairly simple and well-defined way. The development of these Subsystems is often carried out by separate sub-contractors. In this case, the Developer must ensure that system development and safety analysis is still carried out according to this Standard.**

## 6.2 System Development Plan

**6.2.1 Before development of the system begins, the Developer must provide a System Development Plan that describes the development process to be followed. This plan must specify what design methods and techniques will be used, and the Safety Management Group must approve the choices made.**

6.2.2 It is not intended that this Standard prescribe a particular method or language for system design or implementation.

**6.2.3 The Plan shall also detail which staff will be involved in System and Software development with details of their training and experience relevant to the project.**

6.2.4 System (or subsystem) development contracts may specify conformance to other standards that provide further guidance on design for safety.

**6.2.5 The System Development Plan must detail the process to be used for developing software, including design methodologies, programming languages and processors to be used.**

## 6.3 Customer Requirements and Operational Context

**6.3.1 The Customer is responsible for delivering to the Developer an initial set of requirements for the system.**

**6.3.2 The Customer must also provide details of the environments in which the system is planned to operate. This is called the *Operational Context*.**

**6.3.3 At project start-up, the Developer must communicate with the Customer and End Users to refine the scope and high-level requirements of the system.**

**6.3.4 The Developer and Customer (in consultation with End Users) must identify likely sources of hazards, e.g. toxicity, high voltages, flammability, explosive conditions, etc.**

**6.3.5 The Customer is responsible for specifying contractually other standards to be followed by the Developer in conjunction with this Standard. In the case of conflicts between the requirements of any other such standard and this one, this Standard shall take precedence.**

6.3.6 Other standards may include:

- manufacturing process and materials standards;
- industry-specific standards such as aircraft standards;
- safety-related design guidelines, such as STANAG 4404 [14];
- process standards, such as ISO 9001[16]; and
- software development standards such as IEEE Standard 498[13].

**6.3.7 If the requirements or Operational Context change, checks must be made to ensure that all further analysis is still valid. If it is not, the analysis needs to be repeated, and the Safety Case updated accordingly.**

6.3.8 The formulation of requirements and Operational Context form the basis of all further development and analysis.

## 6.4 Design Requirements

### 6.4.1 General

**6.4.1.1 A structured design method that adopts, where possible, widely used and well-understood design paradigms, shall be used for all custom hardware and software Components.**

6.4.1.2 The method and notation used will depend on the nature of the Component: for example, circuit diagrams for hardware and data flow diagrams for software. Some design languages such as state machines with transition diagrams are applicable to a wide range of types of Components.

**6.4.1.3 The design method shall be appropriate to the function of the Component. It must also allow the assurance of safety for the design in accordance with the relevant Safety Integrity Levels for Component Safety Requirements (see Section 16)**

### 6.4.2 Hardware Development

6.4.2.1 There are three types of hardware which may be contained in a system:

- computing hardware including microprocessors on which software is executed;
- custom hardware, including application-specific integrated circuits (ASICs), programmable logic controllers (PLCs), and programmable gate arrays (PGAs) that are developed as part of the development of the system; and
- other non-computing hardware.

**6.4.2.2 For non-custom hardware, it is required that reliable, thoroughly tested and robust equipment is used. Such equipment must be analysed with respect to safety, including the potential interaction between computing systems.**

**6.4.2.3 The Customer shall specify contractually any safety standards for non-custom hardware to be followed by the Developer in conjunction with this Standard. In the case of**

**conflicts between the requirements of any other such standard and this one, this Standard shall take precedence.**

**6.4.2.4 The design of custom hardware Components must use a structured architecture approach in which elements with defined interfaces are combined.**

**6.4.2.5 The detailed design of custom hardware Components must be expressed using a well-known hardware description language.**

6.4.2.6 The following hardware description languages are recommended: VHDL [9], Verilog [10], Ella [12].

**6.4.2.7 The design of custom hardware Components must be supported by the use of a mature, reliable, computer-aided design (CAD) tool.**

6.4.3 Software Development

6.4.3.1 It should be recognised that implementation of system functions by software means that it is easy to introduce unmanageable complexity and hinder predictability of Component behaviour. Software does not have to obey any physical laws and its behaviour is not continuous, so failure of software Components is not predictable.

**6.4.3.2 For functions implemented in software, it must be explicitly justified in the Safety Case that the use of software is sufficiently beneficial, and that other design options were considered.**

6.4.3.3 It is recommended that contracts for the production of software Components specify conformance to a software standard such as IEEE Standard 498, entitled “Software Development and Documentation” [13].

**6.4.3.4 Sound software engineering and structured programming must be practised both in software design and coding.**

6.4.3.5 Sound software engineering and structured programming principles include:

- abstraction that exposes essential properties of data structures, modules and functions;
- modularity that groups together data and functions into appropriately sized blocks, and uses information hiding;
- localisation that ensures that modules comprise of logically related sources;
- uniformity in the naming and structure of sub-Components;
- documentation for all phases that is thorough, accurate and well-structured;
- well commented code; and
- verifiability.

**6.4.3.6 Safety critical software shall be as simple as possible in structure and in execution. Where deviations from well-accepted design and coding principles occur, the Developer shall provide justification as part of the Safety Case.**

6.4.3.7 The aim is to minimise sources of unpredictability in operation, and maximise the opportunity for independent verification of its correctness by the use of tools. Other coding practice constraints may have to be applied as a consequence of the hazard analysis and consideration of the hardware environment in which the software operates.

**6.4.3.8 The programming language chosen must allow the assurance of safety in accordance with the relevant Safety Integrity Levels for critical requirements of Components containing software, as specified in Section 17. The choice of language, and its implications for safety, must be clearly addressed in the Safety Case.**

6.4.3.9 In general, this Standard requires that a mature high level language, conforming to accepted modern software engineering principles, shall be used for critical functions implemented in software. Recommended languages include those that have a well-defined syntax and semantics (i.e. meaning), are block-structured, strongly typed, have elegant means of code modularization, include exception-handling mechanisms and that guard against wild jumps in control flow, running out of memory and overwrites.

6.4.3.10 Mature, industrial-strength, validated compilers should be used for the development of safety critical software.

6.4.3.11 Clearly, in requiring the use of a high-level language, a good deal of trust must be placed in the compiler used. Note that, for the most critical functions implemented in software, a 'safe' subset of a high-level language must be used. Such subsets contain only code constructs which have a well-defined formal semantics, and which lend themselves to code verification. In special cases, the Developer may propose the use of assembly languages. This Standard does not rule out their use for critical code; however, the code must be small (a few hundred lines of code at the most), with a well-defined function. Moreover, the code must be developed in accordance with the coding practices outlined above. The use of large assembly language programs that cannot be easily analysed is not permitted for safety critical code.

**6.4.3.12 Software developed shall be subject to thorough testing.**

**6.4.4** Operator Interface Design

**6.4.4.1 The operators shall be considered part of the system.**

6.4.4.2 The way in which human operators interact with other Components of the system is very important to overall safety. The design involved in operator Components consists of defining standard and emergency procedures that those operators must follow in interacting with the system and the training they must undergo before using it.

6.4.4.3 In addition to the core activities of design, attention should be paid to the human-computer interface and the procedures and training required of personnel that operate, install or maintain parts of the system, with special focus on the safety-related functions of the system.

**6.4.4.4 Human factors analysis, especially the potential for operator induced error, shall be carried out as part of safety analysis and design for displays.**

6.4.4.5 Particular attention shall be paid to the way in which Component errors and failures are indicated and the timing requirements for operator actions. The possibility for shortcuts in operator procedures is undesirable.

**6.4.4.6 The incorporation of shortcuts such as battle-shorts must be justified in the Safety Case and approved by the Safety Management Group.**

6.4.4.7 Prototype Operator Interfaces should be developed at an early stage, and used by several End Users. Feedback from this procedure should be used in further design and safety analysis.

6.5 Documentation

6.5.1 ANNEX C contains a list of documents required by this Standard.

**6.5.2 Any other system and development documentation that the Auditor deems relevant shall be provided on request to the Safety Management Group and the Evaluator.**

**6.5.3 In all cases, documents shall be accurate, consistent, up-to-date, and suitable for their intended readers. All documents shall be clearly marked with a date and/or version number so that it is possible to identify the latest revision.**

**6.5.4 Documentation must indicate the functions and portions of Components that are safety critical.**

**6.5.5 Safety considerations must be taken into account at every stage of design and implementation, and these considerations shall be documented in appropriate places.**

6.6 Configuration Management

6.6.1 Effective Configuration Management is an essential element in assuring the safety of the system being procured.

**6.6.2 The scope of the configuration management shall include at least all documentation required by this Standard, and all Components that have been identified as safety critical.**

**6.6.3 All safety critical configuration items shall be clearly identified as safety critical.**

6.6.4 The Configuration Management system should be supported by the use of an automated Configuration Management tool.

**6.6.5 For software, the Configuration Management system shall provide revision control for all configuration items from the time of their creation and shall provide the ability to recreate the complete software and its development environment. Configuration Management shall not be applied retrospectively.**

6.6.6 The Configuration Management system should be applied during all phases of development in such a way as to support subsequent maintenance of the system.

# 7 GENERAL REQUIREMENTS FOR SAFETY ANALYSIS

## 7.1 Introduction

7.1.1 Safety Analysis consists of those activities which:

- identify requirements on the behaviour of the system in order that its operation is safe; and
- provide assurance that such safety requirements are met by the system.

7.1.2 Safety Analysis activities include

- hazard analysis at various levels of design and implementation detail;
- safety testing;
- simulation to check that safety properties are not violated;
- analysis of component interfaces such as the operator interface;
- static analysis of software, such as control and data flow analysis;
- formal methods, such as the mathematical specification and proof of safety requirements; and
- general design and documentation reviews.

## 7.2 Management Aspects

**7.2.1 Before any Safety Analysis is carried out, the Safety Management Group shall discuss safety aspects of the system and of its operational environment.**

7.2.2 In order that the analysis is as complete as possible, it is important that the people involved in Safety Analysis have a diversity of relevant skills and knowledge. End Users are a group frequently neglected during procurement, but can provide special insight into the operational environment, human factors aspects, and assistance in identifying potential accident sequences.

**7.2.3 The Customer shall determine the extent of End User involvement desirable in the Safety Management Group, and this level of involvement by End Users shall be maintained from the outset of the safety management process.**

7.2.4 This section sets out general requirements and guidelines for safety analysis activities that are carried out during the development of a safety critical system. More specific requirements and guidelines are given in Sections 13-17.

## 7.3 Definitions

7.3.1 An *accident* is an event resulting from the system interacting with its environment that could directly lead to death or injury.

7.3.2 The *severity* of an accident is a measure of the degree of its seriousness in terms of the potential number of victims and the severity of injuries.

7.3.3 *Accident sequences* are the chains of events that result in an accident.

7.3.4 A *hazard* is a system event or state that must occur, in combination with other events, for an accident to occur.

7.3.5 *Hazard Analysis* is an activity by which sequences of events that can lead to hazards or accidents are identified, and the chance of such a sequence occurring is estimated.

7.3.6 A *safety requirement* is a requirement on the behaviour of the system or one of its Components that must be satisfied so that a given hazard does not occur.

7.4 Design for Safety

7.4.1 Safety considerations must be taken into account in formulating system designs.

**7.4.2 In particular, design shall be guided by previous safety analysis.**

7.4.3 Design that uses the results of safety analysis may aim to do any of the following things (listed in decreasing order of effectiveness):[1]

- eliminate the possibility of a hazard, which is called Hazard Elimination;
- decrease the chance of hazards occurring, which is called Hazard Reduction.
- decrease the chance that an accident happens, given that System Hazards have occurred, which is called Hazard Detection and Control; or
- decrease the severity of an accident once it has happened, which is called Damage Limitation.

7.4.4 Hazard Elimination may involve elimination of unnecessary interdependencies between system functions and Components, the substitution of safer Components for less safe ones, and the elimination of the possibility of some operator errors.

7.4.5 Hazard Reduction can be achieved by introduction of safeguards such as barriers, interlocks and redundancy. We also use the terms *mitigation* and *protective measures* to mean Hazard Reduction techniques.

7.4.6 Hazard Detection and Control may include the introduction of warning lights, messages and horns, emergency shutdown procedures, or operator intervention mechanisms.

7.4.7 Damage Limitation techniques include the establishment of operational procedures which minimise exposure to harm if an accident occurs, for example, protective clothing, physical isolation of system, evacuation of unnecessary personnel, as well as accident emergency procedures and equipment (e.g. fire-fighting drills and equipment).

**7.4.8 System documentation relating to Procedures and Training must clearly describe the operator's role in contributing to System Safety.**

7.4.9 Note that System Hazard Analysis should address such issues as operator error, and subsequent system safety assurance must take account of them.

**7.4.10 The accident sequences identified in the Preliminary and System Hazard Analyses shall be used to design the operator procedures and training.**

**7.4.11 Instances where human intervention contributes to the mitigation of threats to safety arising from system behaviour must be clearly described in the Safety Case.**

7.5 Safety Requirements

7.5.1 Some requirements are more critical than others, in the sense that failure to satisfy them can have more dangerous consequences. The effort in analysing the safety of the system shall be directed

[1] See Reference [7] for more details.

towards the most critical requirements. Methods for judging the criticality of requirements form part of what is called System Integrity Assessment, and are described in Section 15.

## 7.6 Hazard Log

7.6.1 The Hazard Log provides a documentation trail for hazards from initial identification through to resolution. Hence, it becomes an extremely important aspect of a comprehensive safety programme.

**7.6.2 The Developer must implement a Hazard Log to record hazards, critical functions and safety requirements throughout system development and safety analysis. For each item, all references to the item in any safety analysis and design documentation must be recorded.**

**7.6.3 As the Hazard Log is highly dependent on many other documents, it must be subject to System Configuration Management. The Developer must ensure that each item is uniquely identified, and that the Log is up-to-date and consistent with all other documentation.**

## 7.7 Safety Testing

7.7.1 Safety Testing is the activity of experimentation that attempts to reveal errors in the design of a system with respect to a set of safety requirements.

7.7.2 Thorough and extensive tests, simulations and trials that do no reveal any violation of safety requirements are an important source of evidence for the Safety Case.

7.7.3 Particular forms of safety testing include *unit*, *integration* and *system tests*.

**7.7.4 Unit and integration tests shall be conducted on individual units, on partially integrated units, and on Components when development is completed.**

**7.7.5 System tests shall be conducted before installation, as described in Section 10.**

7.7.6 It is vital that the test data used for safety testing represents a wide and accurate coverage of possible system states and inputs.

7.7.7 Safety Test data should be generated from the safety requirements, not from any knowledge of the system design and implementation.

**7.7.8 The Safety Test team shall include persons that were not closely involved in development of the system.**

## 7.8 Safety Proofs

7.8.1 In addition to testing, this Standard requires *proof* that safety requirements are met. Such proofs may take the form of informal arguments. In more critical cases, rigorous arguments or *formal methods* should be used.

7.8.2 This Standard defines formal methods to be the modelling and prediction of the behaviour of a system within a mathematical formalism. In many cases, formal methods are supported by automated tools. Such tools provide increased repeatability of analysis, increased soundness and extra assurance.

It is essential that the modelling is accurate. This includes aspects such as the semantics of the mathematical language, as well as accurate modelling of the environment etc.

7.8.3 In this Standard, the terms *formal proof* and *rigorous proof* are used in the following mathematical sense. A formal proof (usually assisted by some form of reasoning or verification tool) is one in which rules of reasoning are used to derive theorems from axioms. In some well-defined cases, tools may carry out fully automatic proofs. A rigorous proof is less formal, in that some aspects of the full proof may be lacking or that certain steps are informally expressed, but is nevertheless mathematically convincing. Rigorous proof may make use of formal reasoning but usually in some restricted way, for example by simplifying the problem or by providing lemmas. A rigorous proof may be partly tool-assisted or be done entirely "by hand".

7.8.4 The application of formal methods to the analysis of safety requires mathematical skills, and, in particular, an appreciation of the nature of proof. These skills are difficult and time-consuming to acquire and are in short supply. Training in the effective use of verification tools is usually a lengthy process. To increase the assurance of safety, it is crucial that judicious use of tools is made. These factors must be carefully considered by the Developer in order to be able to scope the amount of effort required.

7.8.5 This Standard asserts that experimentation, and testing in particular, is not sufficient to prove safety, while proof alone is not sufficient to demonstrate safety. They are complementary activities; both are required to assure system safety.

7.8.6 It is also noted that the use of formal methods can only assure safety with respect to those safety requirements that have been identified. Thorough hazard analysis must be carried out at all levels of system definition and development in order to identify, as far as possible, a correct and complete set of safety requirements.

# 8 SYSTEM SAFETY REVIEWS

## 8.1 Aim

8.1.1 One of the purposes of System Safety Reviews is to ensure that safety concerns are addressed at a sufficiently early stage during the development of the Safety Case. Safety concerns discovered at later stages of development can have significant impact on schedules and costs. Such reviews are thus of crucial importance. Subsequent system development steps must only be carried out when such safety concerns have been addressed.

## 8.2 Documentation and Schedule

**8.2.1 The Safety Analysis Plan shall include a schedule for System Safety Reviews.**

**8.2.2 Following each System Safety Review, the Developer shall provide a System Safety Review Report. This Report shall also contain a statement from the Auditor indicating satisfaction (or otherwise) that development is being carried out according to the SSMP and this Standard.**

## 8.3 Description

8.3.1 A System Safety Review consists of a formal review by the Safety Management Group of progress on the development of the Safety Case.

**8.3.2 System Safety Reviews will cover all safety analysis activities carried out by the Developer, as well as any Evaluation Reports that have been provided (see Section 9).**

**8.3.3 The Evaluator shall attend System Safety Reviews.**

**8.3.4 All safety concerns raised during Safety Reviews shall be addressed in revised System Requirements or Design, if necessary, and subsequent safety analysis updated or repeated, with such decisions and actions documented. Safety concerns raised during Safety Reviews shall be resolved before the development process is taken any further.**

**8.3.5 On completion of a System Safety Review, the Customer shall write a report detailing any remaining safety concerns. This report shall also contain a statement from the Auditor indicating satisfaction (or otherwise) that development is being carried out according to the SSMP and this Standard.**

**8.3.6 Reviews shall also address whether all personnel proposed to be involved in the development process, including management activities, have the training, technical knowledge, skills, experience and qualifications appropriate to their responsibilities and have been briefed on the potential hazards inherent in the system(s) they are developing.**

**8.3.7 The System Safety Review Report shall also contain any further actions that the Safety Management Group deems to be necessary to carry out to ensure a safe system.**

# 9 SYSTEM SAFETY EVALUATION

## 9.1 Aim

9.1.1 The aim of System Safety Evaluation is to increase the assurance of safety by providing a thorough independent and objective review of the Safety Management Process and of the technical validity of the Safety Case.

## 9.2 Documentation

**9.2.1 The Safety Analysis Plan shall include a schedule for Evaluations.**

**9.2.2 The Evaluator shall provide an Evaluation Plan, which sets out the approach to be taken to Evaluation, including details of personnel to be involved. This is to accord with the SSMP, and shall be submitted to the Safety Management Group for endorsement before any Evaluation proceeds.**

**9.2.3 A Safety Evaluation Report shall be submitted on completion of each Evaluation. This Report shall include specific recommendations and safety concerns, and be presented to a meeting of the Safety Management Group.**

**9.2.4 The Evaluation Report shall contain an Executive Summary, in which the specific safety concerns shall be listed, and recommendations made.**

**9.2.5 The Safety Evaluation Report shall contain a description of the evaluation activities carried out, and the findings of such activities. Special attention shall be paid to the compliance or otherwise of the development with the relevant requirements (i.e. paragraphs shown in bold face) of this Standard.**

9.2.6 Such recommendations can involve *any* aspect of the safety management process; they may be procedural or technical in nature.

9.2.7 Recommendations may be qualified (if necessary) as follows:

- *essential recommendations* (these must be followed to achieve required level of safety assurance);
- *desirable recommendations* (these must be followed unless the Developer can provide convincing arguments to the contrary); or
- *optional recommendations* (these can be taken up at the Developer's or Customer's discretion)

## 9.3 Description

**9.3.1 The Evaluator shall systematically examine the technical validity of the Safety Case, by reviewing the Safety Analysis activities carried out by the Developer.**

**9.3.2 Specifically, the Evaluator shall carry out the following:**

- **review the Developer's approach to Safety Management;**
- **review all safety-related documentation;**

- **check the validity of all hazard analyses carried out for the system;**
- **check that the Levels of Trust and Safety Integrity Levels that have been assigned to safety requirements are appropriate;**
- **review the selection of tools used during safety analysis, and check that they have been effectively applied;**
- **review formal specifications for consistency, structure and appropriateness;**
- **assess the adequacy of Component Implementation Assurance;**
- **check the adequacy of safety testing; and**
- **check design and implementation verification arguments.**

**9.3.3 The Evaluator must carry out a sufficient level of review to be satisfied that the Developer has conformed to this Standard.**

**9.3.4 When the System Safety Case is complete, a final Evaluation shall be carried out to determine whether or not the required safety assurance has been achieved.**

9.4 Evaluator Requirements

**9.4.1 The Evaluator shall be commercially and managerially independent of all the members of the Safety Management Group, and must not take part in System Development activities. Such independence must be demonstrated.**

**9.4.2 If the Evaluating organisation includes personnel who have been so involved, they shall be strictly excluded from the Evaluation process.**

**9.4.3 The Evaluator shall, by agreement with the Auditor, attend Safety Management Group meetings where necessary, to hear the Developer's presentations of stages in the Safety Case.**

9.5 Access to Documentation

**9.5.1 Access shall be given to *all* system documentation, design information and source code relevant for the assessment of safety. The Evaluator shall also be permitted access to the Developer's site to see at first hand how safety-related development and analysis activities are carried out.**

9.5.2 If necessary, non-disclosure agreements can be entered into with the Developer to protect proprietary information.

9.6 Resolution of Disputes

**9.6.1 If the Safety Case is deemed to be inadequate, then either or both of the following must be carried out:**

- **safety analysis should be extended, corrected or repeated, and the Supporting Documents updated as necessary; or**
- **the design and/or implementation must be changed and a new Safety Case presented.**

**9.6.2 In the event of fundamental disagreement on safety issues involving the Customer, Developer or Evaluator, the Auditor acts as an arbitrator, and shall have the final decision on how such issues shall be resolved.**

# 10 INSTALLATION, MAINTENANCE AND MODIFICATION

## 10.1 Installation

**10.1.1 An Installation Plan shall describe procedures to be followed during installation of the system to ensure its safe operation.**

**10.1.2 Installation of the system shall be considered in hazard analysis and the results used to define the installation procedures.**

**10.1.3 Once a system has been installed, system tests shall be carried out to demonstrate the correct implementation of all functional and non-functional requirements. The system tests shall be traceable to safety requirements.**

## 10.2 Maintenance

**10.2.1 The Maintenance of safety critical systems shall be carried out in such a way as to ensure that System Safety Requirements are not violated by any changes made to system software.**

**10.2.2 The Safety Case shall be modified, if necessary, to reflect any changes made during maintenance.**

**10.2.3 Personnel carrying out maintenance shall be fully aware of the safety aspects of the system, and must fully document any changes made along with arguments to demonstrate that safety has not been compromised.**

**10.2.4 Where interlocks, etc. must be overridden to perform tests or maintenance, they shall be designed such that they cannot be inadvertently overridden, or left in the overridden state once the system is restored to operational use. The override of the interlocks for maintenance shall not be controlled by a computing system.**

**10.2.5 The system design shall prevent unauthorised or inadvertent access to or modification of the software (source or assembly) and object code. This includes preventing self-modification of the code.**

**10.2.6 Machine code patches of software are prohibited. All software changes shall be made at the source code level.**

**10.2.7 If the hardware on which software is executed is changed, then the software shall be re-tested to ensure that its performance profile is still within the acceptable limits.**

## 10.3 Modification

**10.3.1 Major system changes, such as major new releases of software, re-engineering of parts of the system etc, shall be analysed from the point of view of their impact on safety. All major system changes shall be made in accordance with the development and analysis requirements of this Standard.**

10.3.2 Such changes will require the on-going involvement of the Safety Management Group or the formation of a new Safety Management Group.

**10.3.3 A Safety Case shall be constructed for the modified system.**

10.3.4 However, if the scope of the modifications is limited to a small part of the system, the new Safety Case will typically be able to use much of the original one.

**10.3.5 If the original Developer is involved with the system change, then the Developer shall be responsible for constructing the new Safety Case.**

**10.3.6 If the original Developer is not involved with the system change, then the Customer shall be responsible for overseeing the construction of the new Safety Case. In this case, the prime contractor involved in the system changes shall be responsible for delivering the necessary Supporting Documents.**

**10.3.7 For major system changes, the Auditor shall have the power to decide if a further Evaluation is necessary.**

# 11 POST-DEVELOPMENT SYSTEMS AND COMPONENTS

## 11.1 Introduction

11.1.1 This Standard focuses on safety management during system development. However, it is acknowledged that it is often necessary for ADF capabilities to be procured after primary development has taken place. Such systems are called *Non-Development Items (NDIs*).

11.1.2 NDIs include the following:

- *systems that are procured from overseas suppliers;*
- *commercial off-the-shelf (COTS) products; and*
- *systems for which development began prior to the publication of this Standard.*

11.1.3 The procurement of NDIs falls into two categories:

- *items which are Components of a larger system under development (Non-Development Components); or*
- *self-contained, complete systems (Non-Development Systems*).

11.1.4 The requirements for these two categories are set out in the remainder of this section.

## 11.2 Non-Development Components

**11.2.1 Components to which there is no access to source code or design documentation, and which have not been developed according to any other safety standard, shall not be assigned a SIL beyond $S_0$.**

**11.2.2 Components that have been developed in accordance with another safety standard may be assigned a SIL of up to $S_2$, without further analysis of the Component, provided that the Auditor approves. In this case, the Developer or supplier shall provide evidence that the CSRs for that Component are satisfied by the specifications of the Component as issued from the supplier.**

**11.2.3 For NDI Components to be assigned a SIL of $S_3$ or higher, access by the Customer and Auditor to design documentation and source code is necessary, and full design and implementation assurance satisfying Sections 16 and 17 is necessary.**

## 11.3 Non-Development Systems

**11.3.1 A safety standard, for example in use overseas, shall have been used for the safety management and development of any non-development system.**

**11.3.2 For the procurement of Non-Development Systems, a Safety Management Group shall still be formed, and an Evaluator must be appointed.**

**11.3.3 The Auditor shall assess the Safety Management Process used.**

**11.3.4 A Safety Case shall still be made by the Developer, Supplier, or the Customer, in accordance with this Standard as far as is possible. The Safety Case should address the intent of each of the Supporting Documents detailed in Sections 13-17.**

11.3.5 The Safety Case shall be presented to the Evaluator, who shall assess the system in accordance with this Standard as far as possible.

**11.3.6 The Evaluator shall assess the technical arguments for system safety, and has the discretion to find a favourable result even if full details of the system are not available (such as source code).**

11.3.7 However, the Evaluator must be satisfied that there is sufficient evidence to be assured of safety, and the reasons for the assessment shall be fully documented. The Evaluator may recommend that further System Safety Assurance be carried out before procurement can proceed.

11.4 Use of Other Safety Standards

11.4.1 Systems procured overseas may have been developed in accordance with one or more of the following international safety standards:

- *UK Def Stan 00-55 & 00-56 [3,4]*
- *MIL-STD-882C [5]*
- *RTCA DO-178B [11]*
- *STANAG 4452 [15]*
- *IEC 61508 [17]*

11.4.2 In many cases, documentation related to the development of NDI Components and systems to these standards may form the basis of the Safety Case.

11.4.3 As guidance, Table 1 provides details of documents and activities of other standards that can be applicable to use for Safety Case documentation. However, in order to satisfy the requirements of this Standard it will usually be necessary to augment such documentation.

## Table 1: Use of Other Safety Standards for Safety Case

| SAFETY CASE DOCUMENT | DOCUMENT | REFERENCE |
|---|---|---|
| *Preliminary Hazard Analysis Report* | UK Def Stan 00-56 | 7.2 (Hazard Identification) |
| | MIL-STD-882C | Task 201 (Preliminary Hazard List)<br>Task 202 (Preliminary Hazard Analysis)<br>Task 207 (Health Hazard Assessment) |
| | IEC 1508 | Part 1 |
| | RTCA DO-178 | Section 2.0 (System Aspects Relating to Software Development) |
| | STANAG 4452 | Analysis Task 1 (Requirements Hazard Analysis) |
| *System Hazard Analysis Report* | UK Def Stan 00-56 | 7.3 (Safety Criteria Definition)<br>7.4 (Risk Estimation) |
| | MIL-STD-882C | Task 203 (Safety Requirements/Criteria Analysis)<br>Task 204 ((Subsystem Hazard Analysis)<br>Task 205 (System Hazard Analysis)<br>Task 206 (Operating and Support Hazard Analysis) |
| | IEC 1508 | Part 2. |
| | STANAG 4452 | Analysis Task 2 (Design Hazards Analysis)<br>Analysis Task 4 (Interface Hazards Analysis)<br>Analysis Task 5 (User Interface Analysis) |
| *System Integrity Assessment Report* | UK Def Stan 00-56 | 7.4 (Risk Estimation)<br>7.1 (Safety Compliance Assessment) |
| | MIL-STD-882C | Task 301 (Safety Assessment)<br>Task 402 (Safety Compliance Assessment) |
| | IEC 1508 | Part 2. |
| *Component Design Assurance Report* | UK Def Stan 00-55 | Section 4 (Planning Process)<br>Section 5 (SRS Development Process) |
| | MIL-STD-882C | Task 401 (Safety Verification) |
| | RTCA DO-178 | Section 6.0 (Software Verification Process) |
| | IEC 1508 | Part 2. |
| *Component Implementation Assurance Report* | UK Def Stan 00-55 | Section 4 (Planning Process)<br>Section 5 (SRS Development Process) |
| | MIL-STD-882C | Task 401 (Safety Verification) |
| | RTCA DO-178 | Section 6.0 (Software Verification Process) |
| | IEC 1508 | Parts 2 & 3. |
| | STANAG 4452 | Analysis Task 3 (Code Integrity Analysis)<br>Analysis Task 8 (Safety Test Program) |

BLANK

# PART THREE

# STRUCTURE OF THE SAFETY CASE

BLANK

# 12 THE SAFETY CASE

## 12.1 Introduction

12.1.1 The aim of the System Safety Case is to show how all of the stages of the Safety Engineering and Management Process have resulted in the assurance of a safe system.

12.1.2 The Safety Case is the Developer's 'defence' of the system as being safe for service, and is to be thoroughly examined by the other members of the Safety Management Group. The Evaluator also reviews the Safety Case at well-defined stages during System Development.

## 12.2 Structure of the Safety Case

12.2.1 The Safety Case is generated in an incremental manner as system development progresses. The Safety Case has two Components: High-Level Arguments and Supporting Documents. These will be discussed in detail. The High Level Arguments are of paramount importance. They provide a brief but self-contained overview of the process followed, and the arguments for safety. These arguments can be presented using techniques such as goal structuring [8].

12.2.2 The Supporting Documents are produced as a result of the safety analyses carried out at various stages of development. There are specific activities that must be documented in the Safety Case.

**12.2.3 Documentation of the following safety analysis activities form the Supporting Documents of the Safety Case and shall be provided by the Developer:**

- **Preliminary Hazard Analysis (Section 13);**
- **System Hazard Analysis (Section 14);**
- **System Integrity Assessment (Section 15);**
- **Component Design Assurance (Section 16); and**
- **Component Implementation Assurance (Section 17).**

**12.2.4 The Safety Analysis Plan shall include a schedule specifying when Safety Case Supporting Documents will be produced with respect to system development milestones.**

**12.2.5 The Safety Analysis Plan shall also outline a High Level Argument describing how the Supporting Documents will provide evidence of acceptable safety assurance.**

**12.2.6 On completion and documentation of each of the above safety analysis activities, the Safety Case must be submitted to the Safety Management Group for a System Safety Review, and to the Evaluator for an Interim Evaluation.**

**12.2.7 Both the Supporting Documents and the High Level Argument shall be subject to system Configuration Management.**

12.2.8 On completion of system development and the Supporting Documents, a full High Level Argument is constructed. The complete Safety Case represents a full justification for safety and is passed to the Evaluator for the Final Evaluation (see Section 9).

12.2.9 In the following Sections 13 to 17, a detailed description will be given of the structure of the Safety Case.

# 13 PRELIMINARY HAZARD ANALYSIS

## 13.1 Aims

13.1.1 The aim of Preliminary Hazard Analysis is to define the system boundary and identify all possible accidents, and their severities, that may be caused by a combination of the states of the system, environmental conditions and external events.

13.1.2 Preliminary Hazard Analysis is carried out by the Developer once the requirements have been specified in sufficient detail to define system functions and boundary.

## 13.2 Documentation

13.2.1 The Preliminary Hazard Analysis Report is the first Supporting Document of the System Safety Case. It applies to the entire system.

13.2.2 The Preliminary Hazard Analysis Report consists of:

1. A System Description and Function Definition;
2. An Accident List with severities;
3. An Accident Sequences List with estimates of probabilities (if applicable);
4. A System Hazard List;
5. A System Safety Requirement List; and
6. An Assignment of Levels of Trust to System Safety Requirements.

## 13.3 System Description and Function Definition

**13.3.1 As the first stage of Preliminary Hazard Analysis, the system shall be described in terms of the high level functions it is intended to perform. These are formulated using the System Functional Requirements Specification.**

13.3.2 System Functions define the scope of the system, which includes the operators of the system. In other words, they draw the boundary defining what can be changed or designed by a Developer (including operator procedures). Factors that are beyond the control of the system are called *external*.

13.3.3 This Standard does not restrict the language or methodology that may be employed in developing and documenting the System Functional Requirements Specification. However, it is suggested that appropriate Systems Engineering methods and standards be invoked to assist with this phase.

## 13.4 Accidents and Accident Severities

13.4.1 As defined in Section 7.2, an accident is an external event that could directly lead to death or injury. Preliminary Hazard Analysis is concerned with identifying all accidents that could conceivably arise from a combination of system and external conditions. The severity of an accident is a measure of the degree of its seriousness in terms of the potential number of victims and the severity of the injuries.

**13.4.2 A detailed list of possible accidents shall be provided. In particular, the Preliminary Hazard Analysis Report must provide an estimate of the *severity* of each accident: this is a measure of the degree of seriousness of the accident.**

13.4.3 Accident severity is obtained by assessing the likely consequences once the accident has occurred, in terms of the number of victims and the severity of injuries.

13.4.4 There are four accident severities, defined in Table 2.

**Table 2: Description of Accident Severities**

| ACCIDENT SEVERITY | DEFINITION |
|---|---|
| Catastrophic | Multiple Loss of Life |
| Fatal | Loss of Life |
| Severe | Severe Injury |
| Minor | Minor Injury |

13.5 Accident Sequences

13.5.1 As defined in Section 7.2, accident sequences are the chains of events, including system states or events that can result in an accident.

13.5.2 Accident sequences can be depicted by diagrams such as Figure 4. In general, there may be a number of possible sequences of events leading to a given accident; these can be enumerated by any suitably precise means (such as the use of event trees).

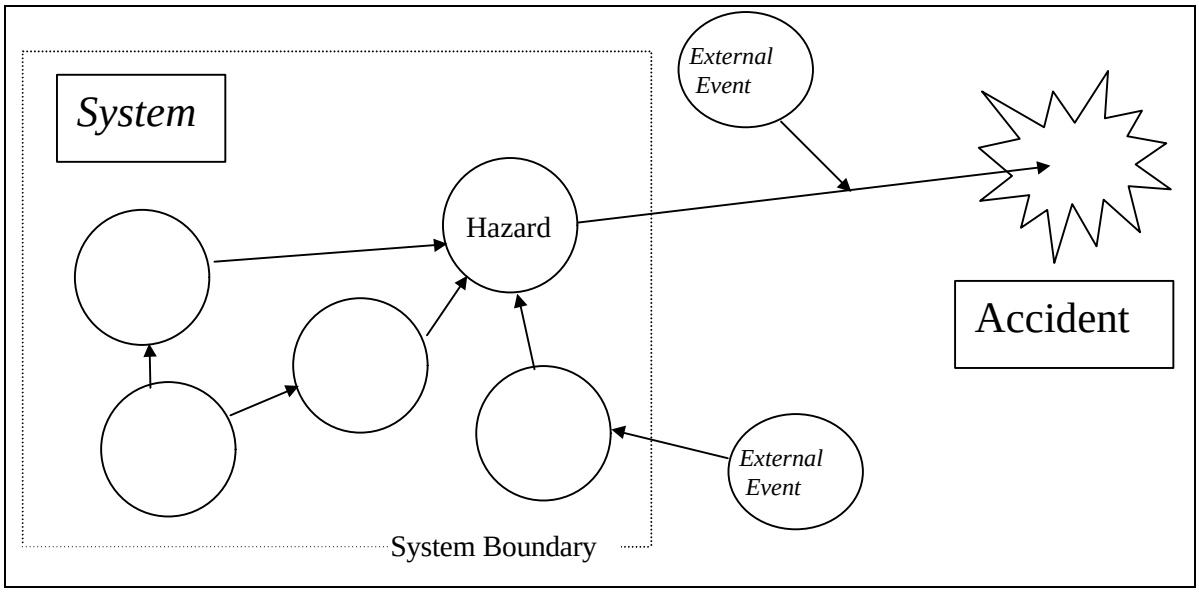


**Figure 4: Accident Sequences**

13.5.3 In a limited number of cases (i.e. where the accident sequence includes random external events such as environmental factors, the behaviour and condition of hardware etc), it may be possible to ascribe a numerical probability to the chance of an accident occurring *given that a System Hazard has been realised*. Where possible, such estimates and their justification (e.g. statistical evidence) shall be provided as part of the Preliminary Hazard Analysis Report. They are to be used to assign Levels of Trust to System Safety Requirements.

**13.5.4 For the computer-based systems covered by this Standard, probabilities shall not be assigned to the chance of the System Hazard itself being realised. In particular, such probabilities shall not be associated with the behaviour of software.**

13.5.5 Possible methods for protecting the system from certain external factors, and for decreasing the chance of an accident, should be identified and documented at this stage, so that system requirements and design can incorporate them.

13.6 System Hazards

13.6.1 System Hazards are top-level states or events of the system from which an accident could conceivably result arising from a further chain of events external to the system. Thus, System Hazards can be regarded as being at the system boundary, and are a potentially dangerous means for the system to play a role in accidents.

**13.6.2 System Hazards shall be identified through a process that considers the sources of possible hazards such as hardware failure, unintended behaviour of software, incorrect timing, user interfaces, the effect of environmental factors and subsystem intercommunication.**

13.6.3 Hazard Analysis should draw upon relevant insight and previously reported personal and corporate experience.

13.6.4 By its nature, the identification of System Hazards is an iterative activity requiring domain expertise and engineering judgement and knowledge. It is important that the relevant System Hazards are identified before any detailed system design is carried out. Because of this, it is of great value to invest considerable effort into creating the System Hazard list, and it is important that these hazards be thoroughly checked, reviewed and updated as necessary.

**13.6.5 If the analysis demonstrates that the behaviour of the system cannot contribute to an accident, this fact shall be documented in the Preliminary Hazard Analysis report. The system shall then be deemed not to be safety critical and no further requirements in this Standard need be satisfied.**

13.7 System Safety Requirements

13.7.1 A System Hazard that is identified describes a system state or event that is undesirable because of its potential to cause external harm. For each System Hazard, there is a corresponding *System Safety Requirement* (SSR), namely that the System Hazard does *not* occur.

13.7.2 Depending on the corresponding accident severity and probability, some System Hazards are more dangerous than others; therefore some SSRs are more important than others. The development of a safety critical system must aim to give an appropriate level of assurance that the SSRs are satisfied by the delivered implementation. In this Standard, a measure of the importance of a System Safety Requirement is given by a *Level of Trust*.

13.8 Safety Critical Levels of Trust

13.8.1 This section describes the assignment of Levels of Trust to SSRs, using the accident sequences, including, if applicable, probabilities associated with external events.

13.8.2 The *Level of Trust* (LOT) of a SSR is a measure of the level of confidence, or *trust* which one wishes to have that the system meets that SSR.

13.8.3 As explained in Section 13.7, each SSR corresponds to a System Hazard which, in turn, may occur in several accident sequences. The LOT is a function of the severity of the resulting accidents and the chance that the accidents will occur, given the occurrence of the corresponding System Hazard and any contributing external events.

13.8.4 This Standard requires the use of seven Levels of Trust, labelled $T_0$, $T_1$ ,...,$T_6$. The lowest level ($T_0$) represents the situation where the system function has no safety critical implications (i.e. it cannot result in any System Hazards). The other levels range in the required level of trust from $T_1$ (the lowest) to $T_6$ (the highest). Table 3 provides a description of Levels of Trust.

## Table 3: Description of Levels of Trust

| LEVELS OF TRUST | CONFIDENCE REQUIRED THAT THE SYSTEM MEETS THE SYSTEM SAFETY REQUIREMENT |
|---|---|
| $T_6$ | Utmost |
| $T_5$ | Extremely High |
| $T_4$ | Very High |
| $T_3$ | High |
| $T_2$ | Moderate |
| $T_1$ | Low |
| $T_0$ | None |

**13.8.5 The assignment of a LOT to an SSR shall satisfy the following conditions.**

- **A *default* LOT is assigned to an SSR using Table 4. The accident severity refers to the most severe accident to which the corresponding System Hazard could lead.**
- **If no probabilities can be assigned to the accident sequences involving the corresponding System Hazard, then the LOT must remain at the default value for that severity.**
- **If, for each relevant accident sequence, a probability can be calculated that represents the chance of the accident occurring given that the System Hazard has occurred, then the LOT assigned can be lower than the default value according to Table 4.**

## Table 4: Levels of Trust Assignment

| | DEFAULT LEVEL | ACCIDENT PROBABILITY | | |
|---|---|---|---|---|
| ACCIDENT SEVERITY | | $> 10^{-2}$ | $10^{-2}$ - $10^{-4}$ | $< 10^{-4}$ |
| Catastrophic | $T_6$ | $T_6$ | $T_5$ | $T_4$ |
| Fatal | $T_5$ | $T_5$ | $T_4$ | $T_3$ |
| Severe | $T_4$ | $T_4$ | $T_3$ | $T_2$ |
| Minor | $T_3$ | $T_3$ | $T_2$ | $T_1$ |

# 14 SYSTEM HAZARD ANALYSIS

## 14.1 Aim

14.1.1 The aim of System Hazard Analysis is to describe Component-Level system design and the proposed implementation of the system in sufficient detail to be able to determine the criticality (in terms of safety) of the Components of the Component-Level system design. It also describes the sequences of events which can cause the System Hazards (identified in the Preliminary Hazard Analysis) to be realised from unexpected behaviour in System Components.

## 14.2 Documentation

14.2.1 System Hazard Analysis carried out during development will occur at several levels of detail according to the extent of Component design completed. The System Hazard Analysis Report is an extensive summary of these several levels of analysis.

**14.2.2 The System Hazard Analysis Report shall consist of:**

1. **A Component-Level System Design Document consisting of:**
   - **A List of Components with proposed implementation;**
   - **A Component Architecture Description; and**
   - **A List of Safety-Related Design Decisions.**

2. **A List of Component Level Hazards and the traces to System Level Hazards; and**

3. **A List of Component Safety Requirements.**

## 14.3 Component-Level System Design

**14.3.1 Once the Preliminary Hazard Analysis has been documented in the Safety Case and reviewed, a high-level description of the system shall be produced by the Developer. This description is called the *Component-Level System Design (CLSD*).**

14.3.2 The CLSD consists of a decomposition of the system in terms of Components that combine to carry out the system functions.

14.3.3 The CLSD is not required to correspond to the highest level of design produced in system development. It may be the case that the CLSD corresponds to a level of detail that emerged after several stages of design.

14.3.4 Note that although the CLSD is required as an input to the safety process, and may have been influenced by that process, it is not a separate document of the safety process, but rather is a design document that has been identified for the safety process.

**14.3.5 The CLSD shall contain an overall architecture of the intended system, and define how the Components of the architecture combine to achieve the system functions. It shall define how each Component is intended to be implemented, that is, whether a function of the Component is carried out in software, hardware, or by operator procedure.**

14.3.6 The results of safety analysis carried out thus far should be carefully considered in the implementation proposals.

14.3.7 The CLSD shall also specify the data interface, information and control flow between Components, using a widely used and well-understood design paradigm.

**14.3.8 The Developer shall carry out a structured approach to the CLSD, using formal modelling if appropriate.**

**14.3.9** Techniques such as state-machine models of designs can be of great benefit in understanding the system, and are also useful in subsequent development.

**14.3.10 The design shall localise safety critical functions, and isolate them, as far as is possible, functionally and physically from other system functions. The safety critical functions shall be made readily recognisable and kept as small and as simple as feasible.**

14.3.11 Components should be as high level as possible. The reason is that protective measures are most effective if they are included in a design at a high level.

14.3.12 There may be several hardware or software components, but components should be loosely coupled. For software components, this means they are separately compilable, and have no shared variables. This would mean they are more likely to consist of, say, 1000 lines of code rather than 10 lines.

14.3.13 The Components described here may not necessarily correspond to *functional* components. For example, protective measures such as interlocks or checksum guards may not be top-level items with respect to the functionality of the overall system, but because they carry out well-defined safety functions, they may appear as components in the CLSD. Other safety functions that are more closely coupled to software may not help to reduce the SIL of a component, but they can be very useful in verifying that a component satisfies its safety requirements (See Sections 16 and 17).

**14.3.14 All safety considerations shall be documented and the CLSD shall detail all safety related design decisions, including:**

- **the justification for software implementation;**
- **decisions that are intended to increase safety;**
- **decisions that are intended to increase amenability to safety assurance;**
- **decisions that have been made to meet constraints other than safety (e.g. reliability, operational, equipment etc.) which make safety a concern; and**
- **evidence that significant design alternatives have been considered, and that a safety-based assessment was made.**

**14.3.15 The CLSD shall document the relationship between these design decisions and the results of the Preliminary Hazard Analysis.**

14.4 Component-Level Hazards

14.4.1 System Hazard Analysis involves a trace of System Hazards in terms of possible Component-Level Hazards that could result in System Hazards.

**14.4.2 The decomposition of System Hazards shall be carried out using a systematic approach, as detailed in the Safety Analysis Plan of the SSMP. Whatever methods are used, the analysis**

**shall not neglect the interfaces between Components, to the external environment, and the user interface. Computing systems in related safety critical automatic test equipment and built in test equipment also shall not be neglected.**

14.4.3 One popular and recommended method is *fault-tree analysis*, but other approaches, such as *cause-consequence analysis*, are acceptable[1]. To gain a wide coverage of possible sources of hazards, a combination of approaches is highly recommended.

14.4.4 A fault tree has at its root one of the top-level hazards that has been identified. From this node there are one or more branches to more specific events that, in combination with other events or system states, cause the hazard to occur. The tree consists of several levels of such events, the leaves being basic events (software, hardware or external) that cannot, or need not, be decomposed further. Lower level events can be combined either through AND nodes (when *all* events occur to cause the higher level event to occur) or OR nodes (when any *one* of the events can cause the higher level event to occur). There are also other symbols that can be added to a fault tree to provide further information about the conditions under which hazards may occur.

14.4.5 The resulting design level hazards are used to formulate the Component Safety Requirements which are requirements on the behaviour of each safety critical Component of the design. These Safety Requirements are the basis for subsequent safety analysis of the system.

14.5 Component Safety Requirements

14.5.1 The CSRs are of crucial significance, and it is essential that they capture the safety requirements for the component. Component Safety Requirements (CSRs) serve to focus the System Safety Assurance effort on those parts of the system where it is most needed. However, the remainder of the system cannot be ignored in the overall safety assessment of the system. Subsequent safety analysis must demonstrate that the other parts of the system do not affect the safety of the system.

14.5.2 Once Component Safety Requirements have been documented they can serve as top-level safety specifications for system Components. These specifications need not say much about the functionality of the Component, for the completeness of functionality is usually a separate requirement to that of safety[2]. However for some systems there may be significant overlap between these requirements.

**14.5.3 To ensure that the set of Component Safety Requirements is as complete as possible , the CSRs shall also be obtained by decomposing the SSRs, by way of the Component-Level Design, into requirements for individual Components.**

14.5.4 It is also the case that CSRs may be derived from other sources such as safety requirements specified by the Customer, and safety requirements stated in design standards such as STANAG 4404 [14].

**14.5.5 The process of checking that the CSR captures the safety requirements shall be carried out by the Developer, in consultation with other appropriate parties such as End Users. Checks shall be made to ensure that:**

- **the list of safety requirements is *complete,* as omissions will lead to safety critical components being ignored;**

[1] See References [6] and [7] for a description of fault-tree analysis and other hazard analysis methods.
[2] In other words, *safety-critical* aspects are distinguished from *mission-critical* ones.

- **the requirements are *consistent,* as conflicting requirements cannot be satisfied simultaneously;**
- **the requirements are *relevant,* as irrelevant criteria result in inefficient use of resources; and**
- **the requirements are *unambiguous* and therefore less likely to lead to misunderstandings.**

**14.5.6 In the case that there are conflicting safety requirements, the Safety Management Group shall propose a resolution of the conflicts. The proposal shall be approved by the Auditor, in consultation with the Evaluator.**

14.5.7 It is possible that CSRs may be in conflict with other Component requirements (e.g. reliability). Appropriate trade-off studies should be initiated and overseen by the Customer to ensure a balance between safety and other desirable system traits.

## 15 SYSTEM INTEGRITY ASSESSMENT

15.1 Aim

15.1.1 The aim of System Integrity Assessment is to assign a Safety Integrity Level (SIL) to each CSR that indicates the level of rigour that must be applied to provide assurance that the CSR is met. The level of rigour is commensurate with the level of trust to be placed on the SSR from which it is derived.

15.2 Documentation

**15.2.1 A System Integrity Assessment Report shall be produced, which documents the SIL assigned to each CSR. The System Integrity Assessment Report must contain a justification of each SIL assignment with reference to Sections 15.4 and 15.5.**

15.3 Description

15.3.1 System Integrity Assessment uses the results of Preliminary and System Hazard Analysis to assign SILs to CSRs. This section describes the approach to be adopted in System Integrity Assessment.

15.3.2 SILs indicate the degree of rigour that is appropriate for system modelling and verification activities.

15.3.3 Figure 5 illustrates the process by which SILs are assigned to CSRs.

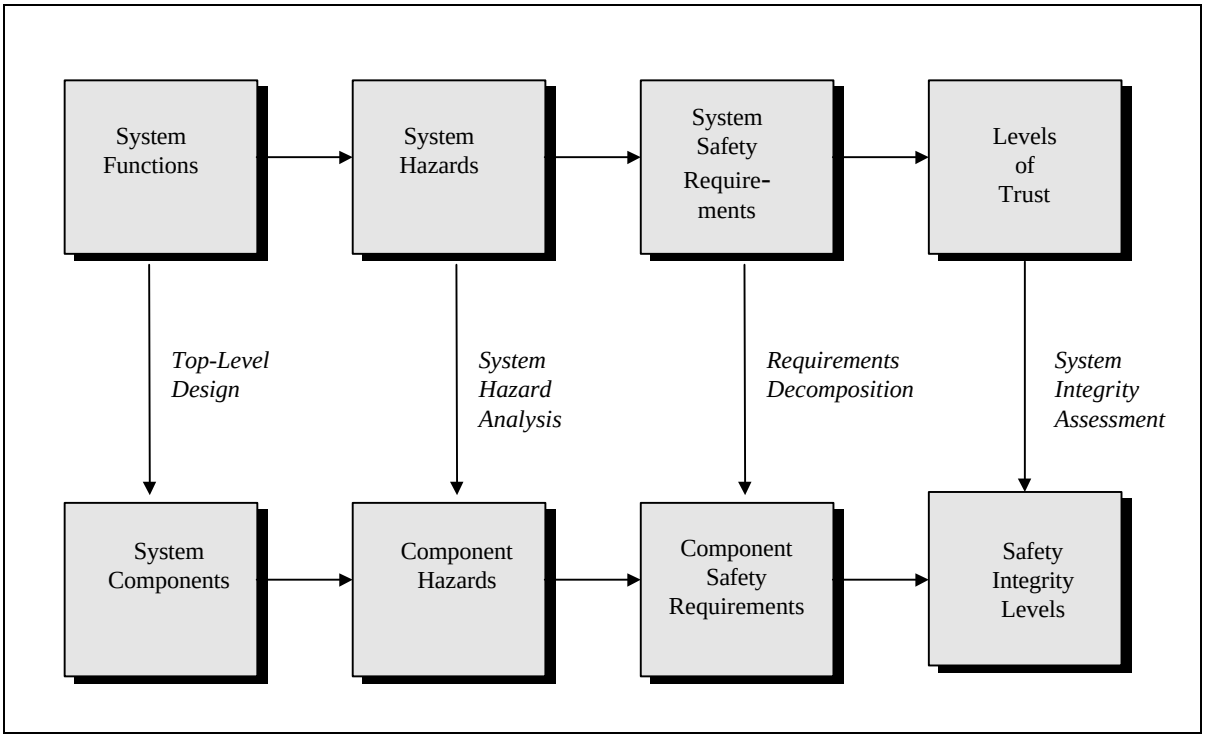


**Figure 5: Levels Of Trust and Safety Integrity Levels**

15.3.4 At an early stage of development, the required system functions are identified and a Preliminary Hazard Analysis performed to identify System Hazards, from which are derived System Safety Requirements (SSRs).

**15.3.5 The System Safety Requirements shall be assigned LOTs according to the chance of external events causing an accident to occur once System Hazards have been realised.**

15.3.6 Next, a design for implementing system functions by system Components is formulated. Using this design, System Hazard Analysis identifies Component-level hazards and thus Component Safety Requirements. The System Safety Requirements can also be decomposed to identify Component Safety Requirements. Component Safety Requirements are assigned Safety Integrity Levels according to the LOT of the parent SSR and the role that hazards at the Component Level have in the System Hazards. Exactly how these SILs are assigned will now be described.

15.4 Safety Integrity Levels

15.4.1 The SIL of a CSR prescribes the system development and safety analysis techniques that need to be applied during development to help ensure that a given system Component will achieve the required level of assurance.

15.4.2 Corresponding to the seven LOTs, there are seven SILs, labelled $S_0$, $S_1$, ..., $S_6$. $S_6$ is the most demanding, requiring mathematically formal specification and development. $S_0$ is the least demanding, but still requires sound software engineering practices. The SILs are actually defined by means of their development and analysis techniques appropriate for a given level. This will be discussed further in Section 16, which provides the design attributes for each SIL, and Section 17 which provides the implementation attributes for Component SILs.

15.4.3 It is important that design takes into account the results of the Preliminary Hazard Analysis. In particular, the design should include preventative, protective and recovery measures that decrease the chance of System Hazards occurring, giving priority to those hazards that may lead to the most severe accidents.

**15.4.4 The SIL that is assigned to a CSR shall depend on the protective measures in the design for reducing the chance of the corresponding hazard.**

**15.4.5 If no specific measures have been taken, then the SIL of the CSR shall be the default level, which equals the LOT Level of the SSR from which it was derived.**

15.4.6 It is highly undesirable to assign a SIL greater than $S_4$ to a CSR, especially for software or custom hardware Components. Therefore, protective measures should be implemented wherever it is seen to reduce the chance of a hazard, and therefore the SIL required.

15.4.7 If protective measures have been taken to reduce the chance of a System Hazard, then the Developer may assign an appropriate SIL to the corresponding CSR. In general, it will not be possible to provide numerical probabilities; instead a *qualitative* argument must be given.

15.4.8 Although protective measures are an important method for containing the consequences of failures within the system, there is a limit to their use. A certain amount of trust may be placed in protective measures and the corresponding arguments in the Safety Case as to how the assurance of safety will be increased by these measures. However, the indiscriminate use of arguments for safety that rely on protective measures can result in components being assigned an integrity level far below their

corresponding contribution to possible hazards. The amount of trust that is gained by the use of such measures is not offset by the trust that is lost by excessive reduction of SILs. This is because the assurance gained by activities carried out at higher SILs (such as design modelling, verification etc) outweigh the assurance gained by the use of protective measures and hazard mitigation arguments.

**15.4.9 The SIL of a CSR shall be no less than *two* levels lower than the LOT Level of the SSR from which it was derived.**

15.4.10 For example, if the LOT of an SSR is $T_5$ then the possible SILs for the CSRs derived from it are $S_3$, $S_4$, $S_5$.

15.4.11 To take into account human error, the SIL assigned to a Component to be implemented by an operator procedure should not normally be higher than $S_3$. However, systems exist in which operators do function at a very high SIL (such as platform commanders). In such cases, the SILs assigned to operators must be explicitly justified in the Safety Case, and approved by the Auditor and Evaluator. More details of operator SILs are given in 17.7.

**15.4.12 Software that can be modified by a user after installation shall be assigned a SIL of $S_0$.**

**15.4.13 The assignment of a SIL to a CSR must be justified by a detailed analysis of the protective measures put in place.**

15.5 Component Independence

15.5.1 One way of increasing safety is to design the system so that several *independent* Components must violate certain CSRs in order that a System Hazard is realised. One Component is *independent* of another if its operation cannot be changed, misdirected, delayed or inhibited by the other Component.

15.5.2 The notion of Component independence has several dimensions. These include:

- physical isolation (for software components this means that each Component runs on a separate processor);
- diversity of implementation, for example, one Component may be implemented in software, another implemented by hardware or operator procedure;
- data independence (for example, the input data for the Components is not to be generated by the same mechanism); and
- control independence, meaning that one Component cannot affect the control flow of another Component.

15.5.3 Independence of Components is one way of justifying the SILs of CSRs. If two independent Components must both fail to meet respective Component Safety Requirements for the system to fail to meet a SSR then the relevant Component Safety Requirements may be assigned a lower SIL.

**15.5.4 If a SIL assignment depends on the independence of components, evidence of the independence shall be documented. The documented evidence shall state how independence is achieved and how independence is used as a protective measure.**

15.5.5 Independence can be a protective measure in a number of ways, including functional redundancy, data redundancy, and software control measures, as described in the remainder of this section

15.6 Redundancy

15.6.1 A particular kind of independence is *functional redundancy*. This means the use of several Components to carry out the same (critical) function. If these Components are independent, then they must all fail for the function to fail.

15.6.2 Redundancy is most successful for protection against random errors such as hardware failure or against faults that are external to the function.

15.6.3 *Data redundancy*, which includes the use of checksums, parity bits, message acknowledgement etc. may also provide increased assurance.

15.6.4 Sometimes, redundancy is used to guard against faults caused by design flaws. Such techniques are called *design redundancy*. A particular kind of design redundancy is software redundancy in which multiple versions of code are written and a voting or other mechanism is used to decide the result. However, it is generally very difficult to achieve an acceptable level of independence for design redundancy to decrease the chance of a hazard occurring [7].

**15.6.5 Design redundancy alone shall not provide sufficient justification for assigning a lower than default SIL to a CSR.**

15.7 Safety Kernels

15.7.1 A Safety Kernel is an independent software unit that monitors the state of the system or a Component to prevent the system from entering an unsafe state and provides transitions to known safe states.

15.7.2 Safety Kernels are another example of the use of independence. The use of Safety Kernels may lower the integrity levels of the Components that it monitors if System Hazard analysis shows that incorrect behaviour of such Components is less hazardous if a Kernel is present.

15.7.3 Software Control Categories

15.7.4 Several standards (e.g. [1], [5]) make use of the notion of Software Control Categories to evaluate the chance of a System Hazard. Software Control Categories measure the amount of control that software Components have with respect to identified System Hazards.

15.7.5 Software control is characterised by the kind of functions the software must perform: whether it only provides information, whether it suggests recommended actions, and whether it carries out actions. There is a spectrum of classes that represent the level of software control, ranging from full software decision and action to software that generates information for making decisions but does not make any decision nor take any action itself.

15.7.6 The safest level of software control depends on the type of system being developed. In general, functions that require operator decision, confirmation or action at critical stages in the control loop are considered safer, but in systems where timing is critical, or where operators may be unable to check the information, this may not be feasible.

15.7.7 If the control of a software Component or sub-Component is sufficiently and appropriately constrained, the SILs of the corresponding CSRs may be lowered by one level.

**15.7.8 Software Control Category arguments alone shall not provide sufficient justification for assigning a SIL two levels below the default SIL for the CSR.**

# 16 COMPONENT DESIGN ASSURANCE

## 16.1 Aim

16.1.1 The aim of Component Design Assurance is to analyse Component Design and the process of Component Development, and provide sufficient assurance that, if the Component is implemented as designed, the Component Safety Requirements are met.

## 16.2 Documentation

**16.2.1 Before development and safety assurance of Components is carried out, a Component Design Assurance Plan shall be submitted to the Safety Management Group, covering the Component Design and Safety Assurance activities.**

16.2.2 Such a plan should include

1. A list of CSRs (resulting from System Hazard Analysis);
2. A list specifying the SIL for each CSR in the list, as determined by System Integrity Assessment;
3. A plan and schedule for the development of each Component; and
4. A plan and schedule for carrying out design and implementation assurance for each Component.

**16.2.3 A Component Design Assurance Report shall be submitted to the Safety Management Group according to the schedule specified in the Component Design Assurance Plan and must contain the following:**

1. **A brief report on the process used to develop each Component;**
2. **A design model or each Component; and**
3. **A report on design assurance carried out for each Component.**

**16.2.4 The Component Design Assurance Report shall provide evidence that the requirements of Section 6 have been met during design development of Components.**

## 16.3 Description

16.3.1 Component Design Assurance is the process of Component Design and analysis so as to reduce to a minimum the hazards which can arise from faults caused by design flaws, and the provision of sufficient evidence that the system, if implemented as designed, is safe for use.

16.3.2 Assurance of system safety is established through the application of specific development and analysis activities for each System Component according to the SILs of associated CSRs.

**16.3.3 The activities and the degree of formality for design assurance for each** SIL **shall be carried out in accordance** with Table 5**.**

**16.3.4 Any proposed variation from the activities specified** in Table 5 **shall be justified by the Developer, agreed by the other members of the Safety Management Group, and formally recorded. Moreover, the Evaluator shall concur with the proposed variation. All subsequent requests for deviations from the original agreement shall be treated in the same way.**

## Table 5: Safety Integrity Design Attributes

| | $S_1$ | $S_2$,$S_3$ | $S_4$ | $S_5$,$S_6$ |
|---|---|---|---|---|
| Specification of Component Safety Requirements | Informal | Semi-Formal | Formal | Formal |
| System Component Model | Informal | Semi-Formal | Formal | Formal |
| Design Verification | Informal | Rigorous | Rigorous | Formal |

**16.3.5 The Design of a Component shall be shown to have taken into account the System Hazard Analysis and System Integrity Assessment.**

**16.3.6 The Component Design Assurance Report shall provide evidence that the verification of the Component Model with respect to CSRs has been carried out with the appropriate degree of rigour, and must provide enough detail so that the Evaluator may assess the verification and degree of rigour.**

16.4 Safety Requirements Specification

16.4.1 System Hazard Analysis identifies Component Safety Requirements that a Component must satisfy in order that System Hazards are not realised. These safety requirements will generally be expressed as English language assertions about the Component's interface with other Components. Safety requirements expressed in this way are called *informal*.

16.4.2 For CSRs that have been assigned a low SIL such informal safety requirements are sufficient. However, at higher levels of safety integrity a more precise formulation is needed.

**16.4.3 For the highest SILs, Component Safety Requirements must be expressed in a formal specification language.**

16.4.4 There are a number of languages suitable for the formal specification of systems. Most prominent among these are Z and VDM. Editing and proof support tools are available for these languages. The choice of language must be made with some care, and must be acceptable to the Auditor and Evaluator.

16.4.5 For intermediate safety integrity levels, some form of semi-formal specification, which typically combines structured natural language with diagrams and mathematical notation, is needed.

16.5 Component Models

**16.5.1 In order to reason about the behaviour of a Component and whether it meets its CSRs, a model of the Component Design shall be formulated.**

**16.5.2 There may be several CSRs with different SILs for a Component. For the highest of the SILs of the CSRs for a Component, the language used to formulate the model of the Component shall be at the level of formality specified in** Table 5**.**

16.6 Design Verification

**16.6.1 Evidence in the form of proof (formal, rigorous or informal) shall be given that each system Component meets its CSRs, if implemented as specified by its model. The degree of formality of the proofs is specified in** Table 5 **according to the SIL of the CSR.**

16.6.2 This activity is called *Design Verification,* and is used to provide assurance of safety of the design.

# 17 COMPONENT IMPLEMENTATION ASSURANCE

## 17.1 Aim

17.1.1 The aim of Component Implementation Assurance is to analyse Component Implementation and the process of Component Implementation, and provide sufficient assurance that the Component Safety Requirements are met.

## 17.2 Documentation

**17.2.1 The Component Implementation Assurance Report shall be submitted to the Safety Management Group according to the schedule specified in the Safety Analysis Plan and must include the following:**

1. **A brief report on the implementation process for each Component;**
2. **A report on the safety testing carried out for each Component; and**
3. **A report on the implementation assurance carried out for each Component.**

## 17.3 Description

17.3.1 Component Implementation Assurance is the process of system development and analysis so as to reduce to a minimum the hazards which can arise from faults caused by flaws in detailed design and implementation, and the provision of sufficient evidence that the system is safe for use.

17.3.2 Assurance of system safety is established through the application of specific development and analysis activities for each System Component, according to its Integrity Level.

**17.3.3 The implementation of a Component shall be shown to have taken into account the System Hazard Analysis and System Integrity Assessment.**

17.3.4 This section gives general requirements for implementation assurance that are applicable to all kinds of Components. These general requirements are given in Section 17.4.

17.3.5 In addition to the general requirements, this section specifies requirements that are particular to software, custom hardware and operator interface aspects of Components.

**17.3.6 Any proposed variation from the activities specified in this section shall be justified by the Developer, agreed by the other members of the Safety Management Group, and formally recorded. Moreover, the Evaluator must concur with the proposed variation. All subsequent requests for deviations from the original agreement must be treated in the same way.**

## 17.4 General Requirements for Implementation Assurance.

17.4.1 This section gives requirements for implementation assurance for all Components.

17.4.2 Safety Testing

**17.4.2.1 Safety Testing shall be carried out for all Components.**

17.4.2.2 Safety testing is experimentation conducted with the intention of demonstrating that the Component does not fail in respect of one or more of its Component Safety Requirements.

**17.4.2.3 A safety test that exposes a violation of a CSR identifies that the system is unsafe with respect to that CSR for that test case. Corrective action and re-testing shall be carried out.**

17.4.2.4 A safety test that does not expose a violation of a CSR demonstrates that the system is safe within the limits of that test case.

**17.4.2.5 The Developer shall define the safety tests to be conducted in the Safety Test Plan.**

**17.4.2.6 The Developer shall provide a reasoned justification of how the safety tests, in conjunction with safety arguments, provide assurance of safety. It shall also include traceability of tests to requirements.**

**17.4.2.7 The Developer shall also provide the results of all safety tests conducted.**

**17.4.2.8 In order to enable tests to be re-run, all aspects of safety testing shall be conducted under Configuration Management.**

**17.4.2.9 Any simulators, models and emulators used during safety testing shall also be validated to ensure that they provide accurate representations of the real system.**

17.5 Specific Requirements for Software Implementation

17.5.1 The specific implementation assurance requirements for software contained in Components are given in this section, and involve:

- restricted use of programming languages;
- software safety testing;
- flow analysis; and
- code verification.

17.5.2 Specific requirements for software attributable to each SIL are shown in Table 6. It is emphasised that sound software engineering practices must be applied at every SIL.

**17.5.3 The requirements for the programming language and flow analysis cannot be applied to single CSRs but rather to Components themselves. In this case the level applied shall be the maximum of the CSR levels associated with a Component.**

## Table 6: Safety Integrity Level Attributes for Software

| | $S_1$ , $S_2$ | $S_3$ , $S_4$ | $S_5$ | $S_6$ |
|---|---|---|---|---|
| Programming Language | High Level Language | Safe Subset of High Level Language | Safe Subset of High Level Language | Safe Subset of High Level Language |
| Flow Analysis | Optional | Yes | Yes | Yes |
| Specification of Implementation | Informal | Semi-Formal | Formal | Formal |
| Code Verification | Informal | Rigorous | Formal | Formal Object Code |

17.5.3.1 Language and Structure of Code

**17.5.3.1.1 The development of software code shall incorporate defensive programming techniques.**

17.5.3.1.2 Examples of such techniques are:

- the use of default and failsafe values for variables;
- limited use of data structures that require dynamic memory allocation;
- limited use of non-deterministic features of the programming language;
- limited use of recursion, tasking and interrupts;
- value range checks;
- complete coverage in conditional control structures; and
- overflow checks.

17.5.3.2 *Flow Analysis*

17.5.3.2.1 Flow analysis involves the study of aspects of code such as:

- *control flow*, which analyses the control structure of the software, indicating any unreachable code or multiple entry and exit points of loops;
- *information flow*, which analyses the dependencies of output variables on input variables, constants and conditional statements, may count the number of paths through the code and identifies redundant code;
- *data use*, which analyses the use of program variables and constants, indicating potential error conditions such as variables that are used before being initialised or never used.

**17.5.3.2.2 Flow analysis shall be supported by static analysis tools, and the results of applying these tools shall be included in the Implementation Verification Report. The report shall provide evidence that any potential problems uncovered by flow analysis have been investigated and the anomalies rectified accordingly.**

*17.5.3.3 Software Safety Testing*

**17.5.3.3.1 Software tests shall demonstrate that the software is executable and that it produces the expected results for the specified test cases.**

**17.5.3.3.2 Software testing shall attempt to demonstrate the following properties of the software:**

- **that unintended infinite loops are not possible (for example, showing that watchdog timers are exercised);**
- **that the functions of the software perform as intended within the system;**
- **the correctness of all interfaces to the software;**
- **that the software performs as intended under transition and boundary conditions and that special cases, including transition through zero, are satisfied; and**
- **the adequacy of performance under stress conditions, such as maximum load or memory conditions.**

**17.5.3.3.3 The criteria for test coverage of unit test shall include:**

17.5.3.3.4 Standard Unit Testing (S1 only)

- **all input variables shall be set to minimum and maximum values as well as an intermediate value; and**
- **all booleans shall be executed at least oncewith true and false values;**

17.5.3.3.5 Medium Unit Testing (S2 and S3)

- **all statements and all branches shall be executed, including zero and non-zero numbers of iteration of loop (where semantically possible).**
- **all feasible combinations of predicates shall be executed; and**
- **all variables of enumerated type shall be set to each possible value.**

17.5.3.3.6 High Unit Testing (S4 , S5 and S6)

- **all loops shall be executed 0, 1, an intermediate and maximum times, where this is semantically possible.**

17.5.3.3.7 Basic Integration Testing (S1 only)

**17.5.3.3.7.1 Basic integration testing shall cover the full-range of realistic operating conditions. Software tests shall be conducted using final and unamended object code on the target microprocessor unless the Developer provides a written justification to the satisfaction of the Auditor.**

17.5.3.3.8 Medium Integration Testing (S2 and S3)

**17.5.3.3.8.1 It is required that for CSRs of these levels, specific tests are carried out to test the CSRs.**

17.5.3.3.9 High Integration Testing (S4 , S5 and S6)

**17.5.3.3.10 Testing shall demonstrate that:**

- **all functions in the requirements specifications have been executed;**
- **all inputs and outputs have been executed with their minimum and maximum values;**

- **all booleans have been exercised with true and false values; and**
- **all testable non-functional requirements, including timing and capacity have been demonstrated.**

*17.5.3.4 Code Verification*

17.5.3.4.1 For the most critical Components implemented in software, it is necessary to prove that the code meets its Component Safety Requirements. This must be done by means of either rigorous or formal mathematical proof that every execution of the program will satisfy the given requirements. This activity is termed *Code Verification*.

**17.5.3.4.2 Code verification proofs shall be documented in the Component Implementation Assurance Report and shall be given in sufficient detail to convince (to the degree appropriate to the SIL) the Evaluator that they are correct.**

**17.5.3.4.3 All formal verification shall be carried out with support from formal proof tools. The tools used shall be approved by the Auditor.**

17.5.3.4.4 The Evaluator should also be consulted on the choice of tools, as they will need to access them for System Safety Evaluation.

17.6 Requirements for Custom Hardware Implementation

17.6.1 The specific implementation assurance requirements for custom hardware contained in Components are given in this section, and involve:

- design aspects;
- input/output interface considerations;
- hardware testing; and
- hardware verification.

*17.6.1.1 Design Aspects*

**17.6.1.1.1 Where Read-Only-Memories are used, positive measures shall be taken to ensure that the data cannot be corrupted or destroyed.**

*17.6.1.2 Input/Output Interfaces*

**17.6.1.2.1 The interface to input/output ports (e.g. sensors) shall be designed so that failure of the input/output devices does not adversely affect the safe operation of the Component.**

*17.6.1.3 Testing*

17.6.1.3.1 Software simulation should be used to test the functional behaviour and performance of custom hardware Component Designs.

**17.6.1.3.2 Once a device has been fabricated, it shall be tested using a set of test vectors, which should be generated using a test pattern generation tool.**

**17.6.1.3.3 Analyses shall also be performed to ensure that interfacing devices are capable of providing valid data within the required time frame for CPU access.**

**17.6.1.3.4 The final device shall be tested in an environment that is similar to the Operational Context.**

17.6.1.4 *Hardware Verification*

17.6.1.4.1 For the most critical Components implemented in custom hardware, it is necessary to prove that the detailed hardware design meets its Component Safety Requirements, including any timing requirements that are critical to safety. This activity is termed *Hardware Verification*.

**17.6.1.4.2 Hardware verification proofs shall be documented in the Component Implementation Assurance Report and shall be given in sufficient detail to convince (to the degree appropriate to the SIL) the Evaluator that they are correct.**

**17.6.1.4.3 All formal hardware verification shall be carried out with support from formal proof tools. The tools used shall be approved by the Auditor.**

**17.6.1.4.4 The Evaluator shall also be consulted on the choice of tools, as they will require access to them for System Safety Evaluation.**

**17.6.1.4.5 Specific requirements for custom hardware verification attributable to each SIL are shown in** Table 7**.**

## Table 7: Safety Integrity Level Attributes for Custom Hardware Units

| | $S_1$ , $S_2$ | $S_3$ , $S_4$ | $S_5$ | $S_6$ |
|---|---|---|---|---|
| Hardware Verification | Informal | Rigorous | Formal | Formal |

**17.6.2 Static timing analysis shall be carried out and reported in the Component Implementation Assurance Report, and the design should provide adequate margins on critical timing parameters.**

17.7 Requirements for Operators and Operator Interfaces

17.7.1 The specific implementation assurance requirements for the operator interface and procedures contained in Components are given in this section.

**17.7.2 Components shall be designed so that the operator may cancel any procedure with a single action, with the system reverting to a defined safe cancelled state.**

**17.7.3 At least two separate operator actions shall be required to initiate a potentially hazardous sequence of events.**

**17.7.4 These actions shall be designed so that inadvertent initiation of such events is not likely.**

17.7.5 When a potentially hazardous sequence of events has been initiated, the operator(s) should be alerted, both visually and aurally.

**17.7.6 Operators should be qualified and trained to a level commensurate with their assigned SIL as 'Components' of the System.**

**17.7.7 Specific requirements for operators attributable to each SIL are shown in** Table 8**.**

## Table 8: Safety Integrity Level Attributes for Operator Components

| | $S_1$ , $S_2$ | $S_3$ , $S_4$ | $S_5$ , $S_6$ |
|---|---|---|---|
| Operator Skill Level | Basic | Extensive | Advanced |
| Operator Training | Basic Instruction & Training Manuals | Intermediate Instruction & Training Manuals | Advanced Courses & Formally Stated Procedures |

17.8 Requirements for Other Implementation Technologies

17.8.1 System Developers will often wish to implement Components using technologies other than those described above (for example, the use of mechanical components, fuses and explosives).

17.8.2 Appropriate standards that address safety issues for such technologies should be invoked by the Developer, in consultation with other members of the Safety Management Group.

**17.8.3 In the case of conflicts between the requirements of any other such standard and this Standard, this Standard shall take precedence.**

**17.8.4 The use of such technologies must meet the general requirements of this Standard (in particular, the requirements of Section 7.**

**17.8.5 The Developer must propose and justify tables of SIL attributes that will apply to components developed using such technologies. These tables must be acceptable to the System Evaluator and Auditor.**

# ANNEX A ACRONYMS

ADO - Australian Defence Organisation

ASIC - Application-Specific Integrated Circuit

CAD - Computer-Aided Design

CASE - Computer-Aided Software Engineering

COTS - Commercial Off-The-Shelf

CSR - Component Safety Requirements

LOT - Levels of Trust

NDI - Non-Developmental Items

SIL - Safety Integrity Level

SSMP - System Safety Management Plan

SSR - System Safety Requirement

CLSD - Component-Level System Design

VHDL - Very high speed integrated circuits Hardware Description Language

# ANNEX B GLOSSARY

**Accident.** A well-defined external event which could directly lead to death or injury.

**Accident Sequence**. A chain of (external) events from which a given System Hazard, if realised, could result in an accident.

**Accident Severity**. A measure of the degree of seriousness of an accident.

**Auditor**. An independent individual or team who monitors the safety management process, making sure that procedural aspects of the standard are followed, that independence is maintained, and acting as an arbiter in the case of disputes among the other parties.

**Code Verification**. Proof that software code meets safety requirements.

**Component**. A unit identified in the Component-Level System Design that interacts with other Components to carry out system functions. A component may be implemented by means of hardware, software or an operator.

**Component Safety Requirement**. A requirement that must be met by a system Component in order that System Hazards are not realised.

**Control Flow Analysis**. Analysis of the control structure of the software, indicating any unreachable code or multiple entry and exit points of loops.

**Customer**. The organisation within the ADO that is procuring the system in the light of a clear operational need.

**Data Flow Analysis**. Analysis of dependencies of output variables on input variables, constants and conditional statements. It may involve path analysis and the identification of redundant code.

**Data Use Analysis**. Analysis of the use of program variables and constants, indicating potential error conditions such as variables that are read before being written or written and never read.

**Design Assurance**. The process of Component Design and analysis so as to reduce to a minimum the hazards which can arise from faults caused by design flaws, and the provision of sufficient evidence that the system, if implemented as designed, is safe for use.

**Design Verification**. Proof (formal, rigorous or informal) that each system Component meets its safety requirements, if implemented as specified by its detailed design.

**Developer**. The organisation (usually a commercial one) which is responsible for producing the system required by the Customer.

**End Users**. The intended operators of the system, but may also include personnel that will maintain or install system equipment.

**Evaluator**. An independent individual or team who carries out a review of the Safety Case, checking the detail and validity of the System Developer's arguments that the system will meet its critical requirements.

**Formal Methods**. The modelling and prediction of the behaviour of a system within a mathematical formalism.

**Formal Proof**. A mathematical proof in which rules of reasoning are used to derive theorems from axioms.

**Formal Specification**. The use of a well-defined, logically sound mathematical notation to express the requirements of a system or Components.

**Independence.** One Component is independent of another if its operation cannot be changed, misdirected, delayed or inhibited by the other Component.

**Informal Specification**. The use of natural language to express the requirements of a system or Components.

**Preliminary Hazard Analysis**. The activity in which top-level safety issues are identified, including: a description of possible accident scenarios; corresponding accident severities; a detailed list of System Hazards; a list of accident sequences; and the assignment of Confidence Levels to System Safety Requirements.

**Level of Trust**. The Level of Trust of a System Safety Requirement is a measure of the level of confidence, or trust that one wishes to have that the system meets that System Safety Requirement.

**Rigorous Proof**. A mathematically convincing proof that is not formally complete and in which some steps are informally expressed. It may make use of formal reasoning but usually in some restricted way, for example by simplifying the problem or by providing lemmas.

**Safety Critical System**. A system in which failure to function as expected in a safe manner could result in death or serious injury.

**Safety Integrity Level**. A measure of the level of assurance which one wishes to have that a Component meets a Component Safety Requirement. It indicates the degree of rigour that is appropriate for system development and analysis activities.

**Semi-Formal Specification** The combination of structured natural language, diagrams and mathematical notation to express the requirements of a system or Components.

**System Function**. High-level tasks that the system is required to perform.

**System Safety Requirements**. Requirement that must be met by the system in order that System Hazards are not realised.

**System Hazard**. System states from which an accident could conceivably result arising from a further chain of events external to the system.

**System Hazard Analysis**. The activity of describing the sequences of events that can cause the System Hazards to be realised from unexpected behaviour in the system Components.

**Verification**. Mathematical proof (formal or rigorous) that a system Component satisfies safety requirements.

# ANNEX C DOCUMENT LIST

This Annex lists all the documents that are required by this Standard.

| **Document** | **Responsibility** | **Page References** |
|---|---|---|
| System Safety Management Plan | Developer | 20, 23, 43 |
| Preliminary Hazard Analysis Report | Developer | 44 |
| System Hazard Analysis Report | Developer | 48 |
| System Integrity Assessment Report | Developer | 52 |
| Design Assurance Report | Developer | 57 |
| Implementation Assurance Report | Developer | 60 |
| Hazard Log | Developer | 30 |
| System Safety Review Reports | Developer | 32 |
| Evaluation Plan | Evaluator | 33 |
| Evaluation Reports | Evaluator | 33 |

All relevant documents must be updated in a timely fashion, when changes to design etc. have occurred.

# ANNEX D REFERENCES


1. Australian Ordnance Council, Department of Defence, Canberra. *Pillar Proceeding 223.93: Assessment of Munition Related Safety Critical Computing Systems*, August 1993.
2. Occupational Healthy and Safety (Commonwealth Employment) Act, 1991.
3. UK Ministry of Defence. *Defence Standard 00-55: Requirements for Safety Related Software in Defence Equipment*, Issue 2, 1997.
4. UK Ministry of Defence. *Defence Standard 00-56, Safety Management Requirements for Defence Systems,* Issue 2, 1996.
5. US Department of Defense. *MIL-STD-882C: Military Standard: System Safety Program Requirements*, Washington DC 20301, 19 January 1993.
6. Systems and Reliability Research Office, US Nuclear Regulatory Commission, *Fault Tree Handbook*, January 1981.
7. Nancy G. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, 1995.
8. S. P. Wilson, S. Hick, and P. M. Kirkham. Department of Computer Science, University of York, Heslington York YO1 5DD. *SAM User Guide*, 1995.
9. IEEE Std 1076. *VHDL Language Reference Manual*, 1993.
10. IEEE Std 1364. *The Verilog Hardware Description Language*, 1995.
11. RTCA Inc. 1140 Connecticut Avenue, N. W., Suite 1020, Washington, D.C. 20036. DO-178/ED-12: *Software Considerations in Airborne Systems and Equipment Certification*, 1992.
12. H. Barringer, G. Gough, T. Longshaw, B. Monohan, M. Peim, A. Williams. Department of Computer Science, University of Manchester. Technical Report UMCS-92-4-6: *A Semantics and Verification Framework for Ella*, 1992.
13. IEEE Std 498. *Software Life Cycle Processes – Software Development* Acquirer-Supplier Agreement. Trial Use Standard for Information Technology, 1995.
14. NATO, Standardization Agreement (STANAG) 4404: *Safety Design Requirements and Guidelines for Munition Related Safety Critical Computing Systems*, first draft, 1996.
15. NATO, Standardization Agreement (STANAG) 4452: *Safety Assessment Requirements for Munition Related Computing Systems*, first draft, 1996.
16. International Standards Organisation, ISO 9001: *Quality Systems - Model for Quality Assurance in Design, Development, Production, Installation and Servicing*.
17. International Electrotechnical Commission, IEC 61508, *Functional Safety of electrical/electronic/programmable electronic safety-related systems*, draft 61508-2 Ed 1.0. 1998

# ANNEX E INDEX

BLANK

**DOCUMENT IMPROVEMENT PROPOSAL**

DEF(AUST)5679 – THE PROCUREMENT OF COMPUTER-BASED SAFETY CRITICAL SYSTEMS

The purpose of this form is to solicit comments which will assist in maintaining the above document as both practical and realistic. When completed, the form and any additional papers should be forwarded to:

Assistant Program Coordinator [TEMPORARY COORDINATOR]
Army Standardisation
Army Technology & Engineering Agency
Private Bag 12
P.O. Ascot Vale
VIC 3032

**NB** Comments submitted do not constitute or imply authorisation to waive any requirement of the document or to amend contractual requirements.

**1. Has any part of this Standard created problems or required interpretation in use? State paragraph no(s) and any rewording suggested.**

**2. Has any new technology rendered any process obsolete? Suggestions supported by examples are welcome where the new process/hardware has proved satisfactory.**

**3. Comments on any requirements considered to be too rigid/too expensive.**

**4. Remarks (attach any relevant data which may be of use in improving this document).**

BLANK

# REQUESTS FOR DOCUMENTS

Requests for copies of this Standard, or certain of the listed Applicable Documents, may be directed to the appropriate source listed below:

**DEPARTMENT OF DEFENCE (Navy Office)**
Director General Naval Engineering Services
Department of Defence (Navy Office)
Campbell Park Offices (CP1-4-16)
CANBERRA ACT 2600
Attention: NELTIS Specification Centre & Library
Telephone: (02) 6266 2906
Facsimile: (02) 6266 4994

**DEPARTMENT OF DEFENCE (Army Office)**
The Commander
Army Engineering Agency
Raleigh Road
MARIBYRNONG VIC 3032
Postal Address: Private Bag No 12
PO ASCOT VALE VIC 3032
Attention: Equipment Information Officer
Telephone: (03) 9319 5802
Facsimile: (03) 9319 5800

**DEPARTMENT OF DEFENCE (Air Force Office)**
Specifications and Standards (S&S1)
RMF Publications Unit
RAAF Williams
LAVERTON VIC 3027
Telephone: (03) 9256 4179
Facsimile: (03) 9256 4178

**OTHER USERS**
Refer Army Office at address above

**PUBLISHED UNDER AUTHORITY OF DEPARTMENT OF DEFENCE**

Copies of DEF(AUST) Documents are obtainable from sources shown inside back cover.