# Computer-Based Safety Critical Systems in Defence: Def (Aust) 5679

**Tony Cant**

Trusted Computer Systems Group
Information Networks Division
Defence Science and Technology Organisation
PO Box 1500, Edinburgh 5111, South Australia

Tony.Cant@dsto.defence.gov.au

## Abstract

The Australian Defence Standard Def (Aust) 5679, entitled "The Procurement of Safety-Critical Computer-Based Systems", was developed by the Australian Defence Science and Technology Organisation (DSTO), and published by the Land Engineering Agency in August 1998. It describes a systematic approach to the development of the Safety Case for a computer-based system. This paper gives a brief overview of Def (Aust) 5679, and describes plans for further technical revision.

*Keywords*: Safety-Critical System, Safety Case, High Assurance.

## 1 Introduction

Safety-Critical computer-based systems present unique challenges to providing assurance of system safety. All safety-related design and analysis, including high assurance activities, should form part of the Safety Case, which is the defence of the system as being safe for service.

The Australian Defence Science and Technology Organisation (DSTO) has developed a standard (Def (Aust) 5679) entitled "The Procurement of Computer-Based Safety-Critical Systems". Def(Aust) 5679 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.

Def (Aust) 5679 has been developed as a result of DSTO research and practical experience in Safety-Critical Systems over the last 15 years, as well as a careful study of existing military and civilian safety standards. The key requirements were formulated as follows:

1. It is essential to have a *system* approach to Safety Case development that incorporates techniques which are valid for hardware, operator, software and software-like components (such as custom hardware development).
2. System Design must be *inherently safe*: thus, issues raised during safety analysis must be allowed to impact system design if necessary.
3. The use of *integrity levels* allows the application of effort and rigour which is appropriate to the criticality of a component. A *practicable* yet *sound* approach is needed for the assessment of integrity levels for system components, whether these are implemented by means of software, hardware, operator etc.
4. Requirements that need to be highly trusted must be shown to be implemented to a high degree of integrity.
5. A well-defined set of appropriately rigorous steps must be applied to deliver *assurance* of safety once these integrity levels are assigned.

Existing standards fail to meet one or more of these requirements. For example, US Military Standard MIL-STD 882C (MIL-STD 882C) adopts a system-level approach to safety, and requires that software components be addressed in this context. However, in some cases the requirements of this standard may not be sufficient to provide assurance of safety for the most critical software components. The more recent version of this standard (MIL-STD 882D) has removed much of the guidance material, and gives system developers much more freedom (and commensurate responsibility) in safety management, in accordance with the aims of Acquisition Reform in the US. MIL-STD 882D is specifically designed for the US acquisition environment, and is not suitable as a stand-alone standard for use in Australia.

The UK Defence Standards (DEFSTAN 00-55; DEFSTAN 00-56) demand high assurance processes; however, it is not clear how the risk assessment techniques for the system-level approach of DEFSTAN 00-56 translate into specific integrity levels for software components that are to be developed in accordance with DEFSTAN 00-55. These standards are currently under review by the Defence Procurement Agency (DPA).

The International Electrotechnical Commission (IEC) 61508 standard (IEC 61508) is the most important civilian safety standard. It provides extensive guidance on assurance techniques, allowing system level integrity levels to be reduced for software and other kinds of components in the case that there are appropriate adjusting factors (protective measures). However, it does not provide sufficient definition of what these adjusting factors may be and it does not provide guidance for designing for safety.

Def (Aust) 5679 has been used to develop Safety Cases for a number of Defence projects in Australia. It was successfully applied to the development of the Electronics Unit (Series 2) for the Nulka Active Missile Decoy. It has been used for an exercise mine

development, and is currently being used on two Defence command support systems. Def (Aust) 5679 is also being used by New York City Transit (NYCT) as the basis for its safety certification process.

Starting in January, 1999, the Defence Materiel Organisation (DMO) contracted the Software Verification Research Centre (SVRC) to carry out a range of activities in support of the procurement of Safety-Critical Systems in Defence. The SVRC produced a number of technical studies that will be used to help in the further development of Def (Aust) 5679 (Hussey and Atchison, 1999; Smith and Atchison, 1999; Rae and Lindsay, 1999).

In this and the following sections, we shall describe the approach taken by Def (Aust) 5679 to Safety Case development, and discuss the plan for revision of the standard. For more details, the reader should consult the DSTO public web site for Def (Aust) 5679[1].

## 2 Safety Management

The Safety Case is a comprehensive argument that the system is safe for service in its intended operational environment. It is the means by which assurance of safety is transferred to other parties, such as regulatory or certifying bodies.

In Def (Aust) 5679, 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.

The Customer is 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.

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

The Auditor is a body (appointed by the Customer) that monitors the safety management process. The Auditor must ensure that independence is maintained, and (most importantly) act as an arbiter in the case of disputes among the other parties.

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 technical validity of the Safety Case (a *product assurance* activity).

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 the Standard.

The End Users of the system will typically be the intended operators, but may also include personnel that will maintain or install system equipment. End Users are a critical part of the safety engineering process, and are responsible for supplying information about the Operational Context and for following operational, maintenance and installation procedures specified as a result of safety analysis.

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.

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.

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.

The Safety Case involves the following activities:

- Preliminary Hazard Analysis;
- System Hazard Analysis;
- System Integrity Assessment;
- Component Design Assurance; and
- Component Implementation Assurance.

In the following sections, we give a brief overview of the structure of these various stages in the Safety Case.

## 3 Preliminary Hazard Analysis

The aim of *Preliminary Hazard Analysis* (PHA) is to 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.

First of all, the system is described in terms of high level *System Functions*. These define the scope of the system, 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*.

An *accident* is an event resulting from the system interacting with its environment that could directly lead to death or injury[2]. The *severity* of an accident is a measure of the degree of its seriousness in terms of the number of

[1] http://www.dsto.defence.gov.au/isl/defaust5679.html

[2] Note that some standards (such as MIL ST 882C) widen safety to include hazards relating to system damage and environmental impact. However, Def (Aust) 5679 addresses only death or injury.

victims and the severity of injuries (see Table 1). *Accident sequences* are the chains of events that result in an accident.

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

Table 1: Accident Severities

A *System Hazard* is a system state that must occur, in combination with other events, for an accident to occur. For each System Hazard, there is a corresponding *System Safety Requirement* (SSR): namely that the System Hazard does *not* occur.

Some System Hazards are more dangerous than others (depending on the corresponding accident severity and likelihood); thus 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 implementation. The *Level of Trust* (LOT) of a System Safety Requirement is a measure of the level of confidence, or trust which one wishes to have that the system meets that System Safety Requirement.

Def(Aust) 5679 requires the use of seven Levels of Trust, labelled $T_0$ (the lowest) to $T_6$ (the highest)[3]. The assignment of a LOT to an SSR is done as follows:

- A *default* LOT is assigned to an SSR using Table 2.
- If no probabilities can be assigned to the accident sequences involving the corresponding System Hazard, the LOT must remain at the default value.
- If, for each relevant accident sequence, a probability can be calculated that represents the likelihood of the accident occurring given the System Hazard, then the LOT can be reduced in accordance with Table 2.

Note that it is not permitted to use this probability argument to reduce the LOTs excessively. This is a basic principle of *residual trust*[4]. For example, an accident of catastrophic severity must have a LOT of at least $T_4$. There are two main reasons for this. Firstly, there are difficulties in accurately calculating very small probabilities, and such calculations may involve unfounded or unverifiable assumptions, often unstated. Secondly, it is important that a reasonable level of safety analysis is carried out for severe hazards even if they are extremely unlikely to lead to an accident.

## 4 System Hazard Analysis

Once the Preliminary Hazard Analysis has been documented in the Safety Case and reviewed, *System Hazard Analysis* is carried out by the Developer.

| | 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 | $T_3$ | $T_3$ | $T_2$ | $T_1$ |

Table 2: Assignment of Levels of Trust

The LOT is a function of the severity of the resulting accidents and the likelihood that, given the corresponding System Hazard, the accidents will occur. The likelihood depends on the probability that the contributing external events will occur once the System Hazard has occurred.

In some cases (i.e. where the accident sequence includes random external events), it may be possible to ascribe a numerical probability to the likelihood of an accident occurring given that the System Hazard has been realised.

However, it is important to note that Def(Aust) 5679 does *not* allow such probabilities to be associated with the behaviour of software or, in general, to estimate the likelihood of design flaws that affect safety.

The aim of System Hazard Analysis is to describe Top-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 top-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.

[3] $T_0$ corresponds to the case where there are no safety implications at all.

[4] A similar principle applies to the reduction of Safety Integrity Levels (see later).

The *Top-Level System Design* (TLSD) consists of a decomposition of the system in terms of *components* that combine to carry out the system functions. The TLSD must describe how each component is intended to be implemented, i.e. whether a function of the component is carried out in software, hardware, or by operator procedure(s). It must also specify the interfaces between components.

The next stage is to decompose System Hazards into possible *Component-Level Hazards*. This is done by hazard analysis techniques such as fault trees (Leveson, 1995) For each Component-Level Hazard, there is a *Component Safety Requirement* (CSR), namely that the Component-Level Hazard does *not* occur. Component Safety Requirements are requirements on the behaviour of each safety critical component of the design. The CSRs serve to focus the System Safety Assurance effort on those parts of the system where it is most needed.

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 likelihood of System Hazards, giving priority to those hazards that may lead to the most severe accidents.

## 5 System Integrity Assessment

There are two aspects (*trust* and *assurance*) that are implicitly combined in the traditional concept of a Safety Integrity Level. One is the level of trust desired that a given SSR is satisfied, and the other is the level of assurance: i.e. the level of rigour or detail of analysis carried out to show that the implementing component meets its CSR.

Def(Aust) 5679 makes this explicit, by assigning Levels of Trust (LOTs) to SSRs, and Safety Integrity Levels (SILs) to CSRs. By making these concepts distinct, discrepancies between LOTs and SILs are highlighted, showing where the analysis alone is insufficient to provide the trust desired. In such cases there is a need for supporting arguments (e.g. mitigating factors).

The aim of *System Integrity Assessment* is to assign a *Safety Integrity Level* 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 must be related to the level of trust to be placed on the SSR from which it is derived.

Figure 1 illustrates the key steps of the Safety Case development up to the assignment of Safety Integrity Levels. Component Safety Requirements (identified in System Hazard Analysis) are assigned SILs according to:

- the LOT of the parent SSR (assigned as part of PHA); and
- the role that hazards at the Component Level have in the System Hazards.

These SILs are assigned as follows. 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 development practices. The SILs are actually defined by means of their development and analysis techniques appropriate for a given level. This will be discussed later in this Section.

The SIL that is assigned to a CSR depends on the protective measures in the design for reducing the likelihood of the corresponding hazard. If no specific measures have been taken, then the SIL of the CSR is the default level, which corresponds to the LOT Level of the SSR from which it was derived.

If protective measures have been taken to reduce the likelihood of a System Hazard, then the Developer can assign a SIL to the corresponding CSR that is appropriate to the likelihood of the hazard. In general, it will not be possible to provide numerical probabilities to these likelihoods; instead a convincing qualitative argument must be given.

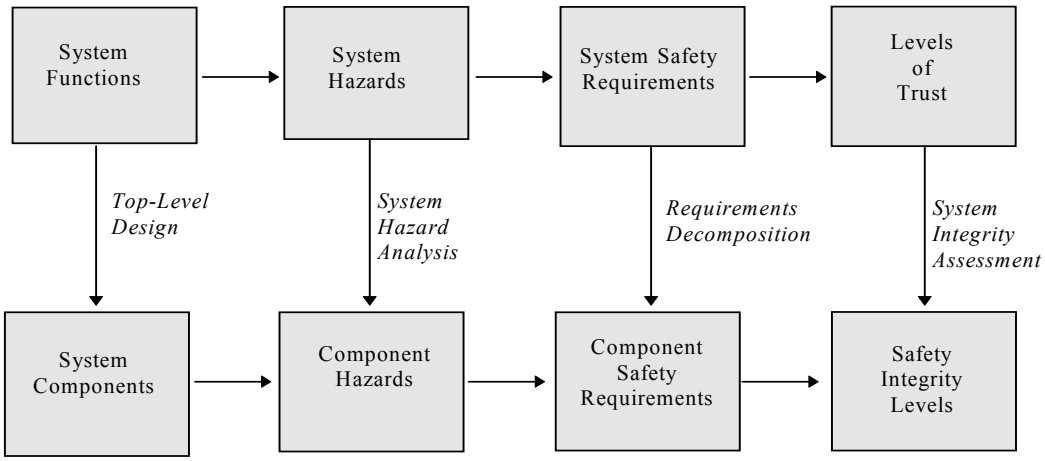


Figure 1 - Levels of Trust and Safety Integrity Levels

Clearly, if a CSR has extensive protective measures (such as independent redundancy of the data, or the use of mechanical interlocks) then it may not need the same degree of rigour in analysis as another CSR, whose implementation does not (or cannot) provide protective measures. Protective measures encourage the development of inherently safer designs, and the reduction of SILs ultimately leads to lower costs.

The justification of a SIL for a CSR is based on the effectiveness of these protective measures. The *independence* of one component from another is one way of arguing the effectiveness of a protective measure. One component is *independent* of another if its operation cannot be changed, misdirected, delayed or inhibited by the other component. 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 can be assigned a lower SIL. The notion of independence can include physical isolation (e.g. separate microprocessors), diversity of implementation (e.g. hardware interlocks), data independence (e.g. multiple sensor input), and control independence (e.g. the use of human operators).

Although protective measures are an important method for containing the consequences of failures within the system, there is a limit to their use. Assurance that CSRs are met by the system must still be relative to the required level of trust for the corresponding SSRs. Consequently, the SIL of a CSR cannot be more than two levels lower than the LOT Level of the SSR from which it was derived – this is the second example of the principle of *residual trust*.

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

Table 3: Safety Integrity Design Attributes

## 6 Component Design Assurance

*Component Design Assurance* is the process of component design and analysis so as to reduce to a minimum the hazards that can arise from faults caused by design flaws. The 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. The activities and the degree of formality for design assurance for each SIL is shown in Table 3.

For CSRs that have been assigned a low SIL informal (English language) safety requirements are sufficient. However, at higher levels of safety integrity a more precise formulation is needed. For the highest SILs, Component Safety Requirements must be expressed in a formal specification language. For intermediate safety integrity levels, some form of semi-formal specification, which typically combines structured natural language with diagrams and mathematical notation, is needed.

In order to reason about the behaviour of a component and whether it meets its CSRs, a *model* of the component

| | $S_1$ , $S_2$ | $S_3$ , $S_4$ | $S_5$ | $S_6$ |
|---|---|---|---|---|
| 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 |

Table 4: Safety Integrity Level Attributes for Software

design must be formulated. The language used to formulate the model of the component must be at the level of formality specified in Table 3.

To provide assurance of safety, evidence in the form of proof (formal, rigorous or informal) must be given that each system component meets its CSRs, if implemented as specified by its model. This is called *Design Verification*. The degree of formality of a design verification proof is specified in Table 3 according to the SIL of the CSR.

For the highest level of integrity ($S_5$ and $S_6$), formal statements of the requirement and the component model are required, together with a proof of their correspondence. This highly stringent requirement is viewed as essential in the design phase, as it is here that fundamental misunderstanding of design or requirements are most likely to lead to problems.

## 7 Component Implementation Assurance

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

The specific implementation assurance requirements for software contained in components involve:

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

Safety Testing is experimentation that attempts to reveal errors in the design of a system with respect to a set of safety requirements. Thorough and extensive tests, simulations and trials which do not reveal any violation of safety requirements are an important source of evidence for the Safety Case. Safety Test data should be generated from the safety requirements, not from any knowledge of the system design and implementation. Safety testing can be invasive (and hence costly) by nature; thus, it is important to seek ways to minimise the amount of such testing while still delivering the required assurance.

Requirements for software attributes that vary for each SIL are shown in Table 4 There are similar requirements for hardware and operator components. It is emphasised that sound software engineering practices must be applied at every SIL.

Flow analysis involves the study of aspects of code such as: *control flow*, *information flow*, and *data use*. Flow analysis must be supported by static analysis tools.

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*. All formal verification must be carried out with support from formal proof tools.

For software the assurance activities for $S_5$ require formal specification and verification of the code, while for $S_6$ this is required for object code. These are costly and undesirable activities; Def (Aust) 5679 strongly discourages the use of $S_5$ and $S_6$ software. In practice, the hazard control and protective measures present in a good design should ensure that a system does not need $S_5$ or $S_6$ software.

## 8 Non-Development Items

Def (Aust) 5679 focuses on safety management during system development. However, it is often necessary for Defence systems to be procured *after* primary development has taken place. Such systems are called Non-Development Items (NDIs). NDIs pose perhaps the greatest challenge to system assurance, and are difficult to deal with in a consistent way. Such systems include:

- systems procured from overseas suppliers;
- commercial off-the-shelf (COTS) products; and
- systems for which development began prior to the publication of Def (Aust) 5679.

A basic principle is that non-development components for which there is no access to source code or design documentation, and which have not been developed according to any other safety standard, cannot be assigned a SIL beyond $S_0$.

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, subject to evidence that the CSRs for that Component are satisfied by the specifications of the Component as issued from the supplier.

For 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 must be carried out.

In the case of non-development systems, a recognised safety standard must have been used for safety management and development. A Safety Case still needs to be made by the Developer, Supplier, or the Customer, in accordance with Def (Aust) 5679 as far as is possible, and assessed by an Evaluator. The Safety Case should address the intent of each of the Supporting Documents that normally make up a Safety Case.

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.

## 9 Technical Issues & Plan for Revision

High assurance standards such as Def (Aust) 5679 need to be constantly evaluated in the light of project experience and further research. Techniques such as formal specification and verification of design or code are subject to active research by the formal methods community. New methods and tools are constantly being developed; such advances will inevitably affect the cost and feasibility of the kinds of safety analysis required by Def (Aust) 5679.

As part of the DefSafe project, the SVRC has carried out a detailed study of the technical issues affecting the construction of a sound safety case (Hussey and Atchison, 1999; Smith and Atchison, 1999; Rae and Lindsay, 1999; Lermer and Lindsay, 2001). (Technical issues are distinguished from management issues which focus on personnel and processes used to perform the safety activities).

Technical areas identified as requiring attention include the use of non-development items (NDIs), the role of operators, treatment of random physical failures and Safety Integrity Levels.

The SVRC suggested four potential strategies for generating NDI assurance:

- Transfer of assurance from other development processes, provided these are of equivalent rigour to Def(Aust) 5679.
- Isolation of NDIs using appropriate design (e.g. wrappers and redundancy).
- Post-development safety case construction. This is the approach favoured by most standards, including Def(Aust) 5679, but can be prohibitively expensive.
- Using empirical evidence of operational use.

The proposed approach to treatment of operator issues within Def(Aust) 5679 is as follows:

- model the role of the operators within total system design and the operator interface as separate components within the system design;
- specify task-based models of operator procedures and operator interfaces as the "component design";
- identify the likely contributions of operators and operator interfaces to system hazards based on knowledge of operator tasks and human error mechanisms;
- identify safety requirements to reduce the likelihood of operator and operator interface induced hazards;
- verify that safety requirements are satisfied by analysing task models of operator procedures and operator interfaces;
- provide assurance that procedures will be performed reliably by consideration of Human Performance Factors which influence the likelihood of human error within the work environment.

In its existing form, Def (Aust) 5679 contains no instructions on the issue of physical reliability. Whilst the scope of the standard includes both software and hardware aspects of computer systems, detailed requirements are only provided for addressing systematic failures caused by design errors. Treatment of hardware, including analysis of physical reliability, is both possible and desirable within the existing framework of the standard.

Recent research has focused on the issue of Safety Integrity Levels (Lermer and Lindsay, 2001). SILs are widely used in safety standards; however, the concept is not well-understood and varies between different standards. Consequently, it is difficult to make a direct comparison of SILs in different standards because of their different approaches. SILs have been useful as an auxiliary concept, but it may be that an evidence-based approach would be a better one in the longer term (McDermid, 2001).

Def (Aust) 5679 also needs to be improved to ensure that it can be verified that the paragraphs noted as "Requirements" have been met.

## 10 References


DEF (AUST) 5679 (1998): The Procurement of Computer-Based Safety Critical Systems. *Land Engineering Agency, Department of Defence*.

DEFSTAN 00-55 (1995): Defence Standard 00-55: Requirements for Safety Related Software in Defence Equipment. *UK Ministry of Defence*.

DEFSTAN 00-56 (1995): Safety Management Requirements for Defence Systems. *UK Ministry of Defence*.

HUSSEY, A. and ATCHISON, B. (1999): Report on Operator Issues. SVRC Report CA38809-311.

IEC 61508 (1995): International Electrotechnical Commission. Functional Safety: Safety-Related Systems. Draft International Standard IEC 1508, June 1995.

LERMER, K. and LINDSAY, P. (2001): Setting and Achieving Safety Integrity Levels. SVRC Report CA38809-351.

LEVESON, N. (1995): Safeware: System Safety and Computers. Addison-Wesley, 1995.

McDERMID, J. (2001): Software Safety: Where's the Evidence? In *Proceedings of the 6th Australian Workshop on Safety-Critical Systems and Software*, Brisbane. Conferences in Research and Practice in Information Technology, Vol. 3 P Lindsay, Ed.

MIL-STD-882C (1993): System Safety Program Requirements. *US Department of Defense.*

MIL-STD-882D (2001): Standard Practice for System Safety. *US Department of Defense.*

RAE, A. and LINDSAY, P. (1999): Treatment of Physical Reliability in Def (Aust) 5679, SVRC Report CA38809-331.

SMITH, G. and ATCHISON, B. (1999): Treatment of Non-Development Items in Def(Aust) 5679. SVRC Report CA38809-321.