



SAFETY-CRITICAL SOFTWARE STANDARDS - SURVEY AND SUMMARY *

0 EXECUTIVE SUMMARY

A literature search has been carried out for standards relating to the development of safety critical software. A total of approximately forty standards were found, including those concerned with the safety certification of programmable system as well as those covering software development.

Standards in five industry sectors were selected. The sectors are UK and US defence, civil aerospace, European rail transport and nuclear power. The applicable standards in each sector were summarised. Each summary uses the same headings so that the standards can be compared.

A notable standard covering safety critical software is draft IEC 1508, 'Functional Safety - Safety-Related Systems'. This is a generic standard, which is designed to be adapted to each industry sector. IEC 1508 not one of the standards selected for summary in this report, because of its complexity. However, the selected European railway standard, although based on an earlier draft of IEC 1508, is an example of an industry specific adaptation of IEC 1508.

* **This report is published by the Avionic Systems Standardisation Committee (ASSC) to advance the role of standardisation in avionics. The use of this is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom is the sole responsibility of the user.** Copies of this paper are obtainable from the ASSC Agency as an Avionic Systems Standardisation Committee publication.

Table of contents		Page no.
0	Executive summary	1
1	INTRODUCTION	4
	1.1. Background	4
	1.2. Scope	4
	1.3. Outline of Report	5
	1.4. Terminology	5
2	Standards for Safety Critical Software	5
	2.1. Search Results	5
	2.2. Standards Selected for Summary	6
	2.3. Summary Headings	7
3	UK DEFENCE: DEF STAN 00-55/56	8
	3.1. Scope	8
	3.2. Safety Life Cycle and Safety Management	9
	3.3. Safety Integrity Levels	10
	3.4. Software Development Life Cycle	11
	3.5. Software Verification and Validation	11
	3.6. Independence	13
4	Civil Aviation: RTCA DO-178B	13
	4.1. Scope	13
	4.2. Safety Life Cycle and Safety Management	14
	4.3. Safety Integrity Levels	14
	4.4. Software Development Life Cycle	15
	4.5. Software Validation and Verification	16
	4.6. Independence	17
5	US DEFENCE: MIL-STD-498/882C	18
	5.1. Scope	18
	5.2. Safety Life Cycle and Safety Management	20
	5.3. Safety Integrity Levels	21
	5.4. Software Development Life Cycle	22
	5.5. Software Validation and Verification	23
	5.6. Independence	24

Table of contents (cont)		Page no.
6	European Railway: prEN 50126/28	25
6.1.	Scope	25
6.2.	Safety Life Cycle and Safety Management	25
6.3.	Safety Integrity Levels	27
6.4.	Software Development Life Cycle	27
6.5.	Software Validation and Verification	28
6.6.	Independence	29
7	Nuclear POWER: IEC 880	29
7.1.	Scope	29
7.2.	Safety Life Cycle and Safety Management	30
7.3.	Safety Integrity Levels	30
7.4.	Software Development Life Cycle	30
7.5.	Software Validation and Verification	31
7.6.	Independence	32
ANNEX A	Standard relating to software and safety	A1
ANNEX B	IEEE Software standards	B1
ANNEX C	Details of Selected Standards	C1

1 INTRODUCTION

The objective of this report is to review the current situation with respect to the inter-relationship of standards addressing safety critical software. The report has been prepared for the Processing Working Group of the Avionics System Standardisation Committee.

There are presently a number of standardisation activities concerned with safety critical software, including Def Stan. 00-55/56 and RTCA DO-178B. The ASSC Progressing Working Group has expressed some confusion regarding the scope and inter-relation of each of these documents. To attempt to clarify the situation, ERA has carried out a search for standards relating to safety critical software. A short summary has been prepared for a selection of the standards found.

1.1. Background

The field of safety critical software has been subject to continuous development in recent years, both in terms of the scale of the critical software being developed, the design and verification techniques available and the required standards of safety evidence. One of the consequences of these developments has been a proliferation of standards, produced by different national or international agencies, in various industry sectors. Many of the standards have a draft status and are subject to frequent updates. This situation can represent a considerable overhead for both system developers and procuring organisations. This review of the current situation is proposed as a first step towards addressing wider questions; for example, whether a technical consensus is emerging which would allow the existing standards to be consolidated and where are remaining points of difference.

1.2. Scope

Standards for developing safety critical software have two aspects:

- 1) requirements for the process for developing highly reliable software, encompassing specification, design and verification
- 2) requirements for safety management, including the way that the possibility of software failure is allowed for in the system safety analysis.

These aspects can be covered in a single standard, as in the civil aviation guidelines RTCA DO-178B 'software considerations in airborne systems and equipment certification'. In other cases two or more standards combine to cover the required scope. The pair of draft standards from the MOD, Def-Stan 00-56 'Hazard analysis and safety classification of the computer and programmable electronic system elements of defence equipment' and Def-Stan 00-55 'The procurement of safety related software in defence equipment', are an example of this approach.

Since the draft Def-Stan 00-55/56 are of particular relevance to members of the ASSC, the scope of these standards is defined to be the scope of interest for this report.

1.3. Outline of Report

In chapter 2 the results of a search of safety critical software standards is given. From the complete list of standards, the most prominent standard or standards in five different industry sectors have been selected for further summary. The headings used in this summary are also presented in Chapter 2. Chapters 3 to 7 contain the summaries of the selected standard or standards in each sector.

1.4. Terminology

There is some variation in the terminology used in different standards. For example, MIL-STD-822C refers to 'mishap' and 'hazard risk assessment' where Def. Stan. 00-56 uses 'accident' and 'safety risk assessment'. The summaries of standards on Chapters 3 to 7 use the terminology of the standard being summarised. There is no attempt to rewrite the requirement of each standard in a uniform terminology. Elsewhere, Def. Stan. terminology is preferred. For example, the phrase 'safety integrity levels' which is used as the third section heading for each summary, is from Def. Stan. 00-56.

2 STANDARDS FOR SAFETY CRITICAL SOFTWARE

2.1. Search Results

Annex A contains a table of 38 standards with relevance to safety critical software development. The list is restricted to English language standards. The list is known not to be complete; for example, only a two national standard from continental Europe, the German VDE 801, and the French NFF 71-013 are included, and there are no standards from the Australian Standards organisation, since those found all had equivalents elsewhere in the table.

A number of software standards and guidance documents are issued by the IEEE. Only one of these relates specifically to safety, but since the others may also be of interest they are presented separately in Annex B.

The list includes standards and guidance documents from the following organisations:

Organisation	Sector
UK MOD	Defence
US DoD	Defence
RTCA/EUROCAE	Civil Aviation
Cenelec	Railways
IEC	Nuclear power, process industry

MISRA	UK Automotive
UK HSE	Process industry
American Nuclear Society	Nuclear power
VDI, Germany	Process industry
NASA	Space
ESA	Space
Electronic Industries Assoc	Defence, weapon systems
Canadian Standards Assoc	General
IEEE	General
Association Francaise de Normalisation	Railways
Underwriters Laboratory	General

2.2. Standards Selected for Summary

The following standards have been selected for further examination. The selection is based on the closest correspondence to Def-Stan 00-55/56 in five industry sectors. Annex C contains titles, date, contents list and other information for each selected standard.

Sector	Standards
UK Defence	draft Def-Stan 00-56 draft Def-Stan 00-55
Civil Aviation	RTCA DO-178B
US Defence	MIL-STD-498 MIL-STD-882C
European Railway	CENELEC prEN 50126, prEN 50128
European Nuclear Power	IEC 880

In the UK Defence sector, the latest available draft versions of Def-Stan 00-55/56 are considered, rather than the interim standards issued in April 1991. Although the draft versions do not have official standard status and may change before being adopted, it was considered to be of more relevance to summarise the latest versions, since these take account of industry reaction to the 1991 versions.

In the US Defence sector, the situation is rather confused. following the recent Perry initiative to move towards commercial standards. MIL-STD-498 is a recent standard, replacing the familiar DOD-STD-2167A; it is understood that there is a possibility of a commercial standard based on MIL-STD-498. The other US Defence standard included

is MIL-STD-882 covering the safety program; however these two standards are not part of pair in the same way as Def-Stan 00-55/56.

An omission from the standards selected for summary is the draft IEC 1508 'Functional safety: safety related systems'. Although developed particularly in the process industry, this standard is proposed as a generic standard, which is to be customised for each industry sector IEC 1508 is the focus of considerable international effort to achieve a uniform standard for the safety of programmable systems, and it is widely believed that this standard will be key to future developments. The generic nature of the standard, together with its general length and complexity, makes it difficult to summarise in a short report. Instead, the European Railway standards prEN 50126/28 have been selected. These are based on earlier draft standards from the IEC SC65A Working Groups 9 & 10, who have developed IEC 1508, and may therefore be considered to be representative of a sector-specific customisation of IEC 1508.

It is recommended that further work be undertaken to produce a more detailed comparison of these standards and to include IEC 1508 in the set of standards considered.

2.3. Summary Headings

Each selected standard is summarised under the following standard headings. The headings most likely to be of interest have been chosen, but the headings do not cover every aspect of all the standards. For example, there is no consideration of configuration control, competence of personnel, or the requirements for using commercial off-the-shelf (COTS) products. It is noted that some of these issues are being addressed by ongoing studies in the UK and elsewhere.

2.3.1 Scope

Under this heading, a summary of the overall scope of the standard(s) is given.

2.3.2 Safety Life Cycle and Safety Management

Under this heading, the activities relating specifically to demonstration of safety are summarised. The activities may include setting safety requirements, planning a safety process, carrying out hazard analysis and producing a safety case.

2.3.3 Safety Integrity Levels

One of the results of the safety life cycle processes is to produce a measure of the degree of criticality of each component of the software. This is normally achieved by considering the consequences of software failure. Under this heading, we consider the number of integrity levels and the criteria for assigning software components to the levels. In some cases, the full requirements of the standards are applicable only to the most critical software, with a subset of requirements applying to less critical software.

2.3.4 Software Development Life Cycle

Under this heading, the activities relating to software development and programming are summarised. This includes any required or prohibited software development techniques. One point of difference between standards is the extent to which a particular software life cycle is mandated. This is of interest, since alternatives to the traditional top-down waterfall life cycle have become more prominent.

2.3.5 Software Validation and Verification

Under this heading, the activities relating to verification and validation are summarised, including software verification and testing techniques.

2.3.6 Independence

Under this heading, any requirements of the standard for certain activities to be carried out by a team which is independent from the main development team are summarised, taking account of the degree of independence required.

3 UK DEFENCE: DEF STAN 00-55/56

draft Def-Stan 00-56 Safety management requirements for defence systems containing programmable electronics.

draft Def-Stan 00-55 The procurement of safety related software in defence equipment.

A related standard which is not covered in this report is:

draft Def-Stan 00-58/1 A guideline for HAZOP studies on systems which include programmable electronic systems

3.1. Scope

Def Stan 00-56

This standard defines the safety programme requirements for systems containing programmable electronics. The requirements cover the safety programme, the management procedures, the required analysis techniques and the safety verification activities. The standard applies to all phases in the project life cycle from initiation to disposal. The required system safety analysis activities include determining the software integrity level.

Def Stan 00-55

This standard specifies requirements for the development of safety related software in defence equipment, covering specification, design, coding, test and integration. Safety management requirements include safety planning, safety analysis, safety reviews, safety audits and the production of a software safety case. These activities are restricted to

software, since Def-Stan 00-56 covers system safety. Roles and responsibilities are defined for safety and development activities. Requirements on the tools and techniques to be used for software development and testing are included.

3.2. Safety Life Cycle and Safety Management

The system safety life cycle required by 00-56 consists of the following steps:

- 1) establish the safety requirements
- 2) prepare a system safety plan, which describes both the management strategy and the technical tasks relating to safety
- 3) undertake system safety analysis
- 4) undertake risk estimation
- 5) undertake safety verification
- 6) prepare a safety case.

A system safety analysis must be carried out to determine whether the level of risk associated with a system is tolerable. The first phase, beginning at the earliest stage in the project life cycle, is the identification of hazards and accident scenarios. Def Stan 00-56 includes definitions for terms such as hazard and accident; for example, a hazard is a 'physical situation, often following from some initiating event, that can lead to an accident'. As the system design proceeds, the description of hazards and accident scenarios is refined by performing hazard analyses.

Risk estimation is the process of calculating a target probability for each accident scenario. This leads to the assignment of safety integrity levels, as described in Section 3.3. below.

A hazard log must be maintained as a record of the safety activities. In addition, the hazard log has a record of each hazard identified and its progress towards resolution. The safety case is a reasoned justification of the safety of the system, constructed from information contained in the hazard log.

The safety life cycle of Def Stan 00-55 fits within that of Def Stan 00-56. The principle activities are:

- 1) prepare a software safety plan
- 2) carry out software safety analysis
- 3) carry out software safety reviews
- 4) prepare a software safety case.

A record of these activities is maintained in the system hazard log. A software safety case is required, forming part of the system safety case described above. It is a reasoned justification that the software meets its safety requirements. It includes the following components:

- 1) a description and justification of the software development process
- 2) a safety argument based on analyses, including formal, i.e. mathematical, reasoning
- 3) a safety argument based on testing.

The justification of the development process must include:

- 1) an appraisal of the staff competence
- 2) an analysis of data describing the safety record of previous systems developed using the same, or similar, processes
- 3) a safety analysis, which examines the consequences of faults in the methods and software tools used in the development process.

3.3. Safety Integrity Levels

The safety integrity level is a measure of the required likelihood that a system achieves its safety requirements. There are four levels: the highest, S4, is allocated to the most critical systems, the lowest, S1 to the least critical.

Initially, a safety integrity is determined for each function of a system. Following the design, each component of the system, implemented by hardware or software, has the level of the most critical function it implements. Def Stan 00-56 includes requirements for the apportioning of safety integrity levels to components which combine to implement a critical function. Many of the requirements of the standards are moderated for less critical components of a system.

The following steps are involved in determining the safety integrity level:

- 1) assess the severity of each accident; there are four levels from catastrophic to negligible
- 2) determine the highest tolerable probability for each accident
- 3) determine the target probability for each hazard, taking account of the sequence of events leading from hazard to accident
- 4) determine the safety integrity level.

One effect of the rules for determining the integrity level from the target probability of a hazard is to limit the trust which can be placed in a single safety component. For example, an accident with catastrophic severity must have a probability no higher than the 'improbable' classification. However, even at the highest integrity level, a single

component cannot be allocated a failure probability lower than 'remote', with the result that at least two safety components are required to protect against a catastrophe. There are additional requirements for such components to be developed independently, to minimise the risk of a common design or implementation fault.

In Def. Stan. 00-55, the safety integrity level of the software determines the requirements; the full requirements of 00-55 apply only to the most critical software.

3.4. Software Development Life Cycle

The software development life cycle is described in the software development plan. The life cycle consist of a number of processes with defined products; the criteria for transition from one process to the next must also be defined.

The input to the software development process is the software requirements specification. The design authority verifies that the requirements are consistent and un-ambiguous.

3.4.1 Software Specification

The first development process is software specification. The specification must always make use of formal methods for safety functions and all functions must be formally specified unless the restriction to a subset of the functionality has been agreed by the MOD's project manager. Formal methods are 'mathematically based techniques' such as Z or VDM. Each software requirement must be traced to a corresponding item in the software specification.

3.4.2 Software Design

The software design is developed from the software specification, using one or more levels of abstraction.

3.4.3 Software Coding

The software is coded from the lowest level design specification. Use of a subset of a high level, strongly typed, block structured language with predictable execution is preferred. Assembler may only be used in exceptional circumstances. The compilation system should have an appropriate validation certificate and a level of assurance corresponding to the requirement determined by the safety analysis of the development process.

3.5. Software Verification and Validation

Def Stan 00-55 has requirements for the verification and validation required at each stage of the software development lifecycle. Review activities are also required, for example to ensure the completeness of the formal verification.

3.5.1 Verification of the Software Specification

Preliminary validation ensures that the specification satisfies the software requirements. A combination of formal argument and executable prototyping are used. Formal

argument is used to validate the safety properties and safety functions of the specification. The executable prototype is produced from the specification, with formal arguments demonstrating the mapping from specification to prototype. The safety functions are exercised by testing the prototype, following a test specification produced by the V & V team.

The formal specification is syntax and type checked using an appropriate tool. Proof obligations for the internal consistency of the formal specification are discharged using either formal proof or rigorous argument.

3.5.2 Verification of the Software Design

Formal arguments are used to discharge proof obligations showing the consistency of the design with the specification of the previous level of design. Non-formal parts of the designed are verified for accuracy against the preceding specification and for completeness and consistency by review. The requirements for checking the consistency of the design are similar to those for the software specification.

3.5.3 Code Verification

Static analysis is performed on all the source code, checking conformance to the required language subset, satisfaction of control, data and information flow requirements and checking worse case conditions for non-functional requirements such as timing. Proof obligations are constructed and are discharged by formal argument to ensure that the code correctly implements the specification.

Object code verification, ensuring that the source is correctly implemented by the compiler on the target hardware, is required unless the compiler has been formally specified and verified. The verification uses either formal argument to show that the object code implements the source code, or testing, with coverage of every instruction. If testing is used to verify the object code, some additional analyses are required.

3.5.4 Unit and Integration Tests

Individual units of code and partially integrated units are tested using a number of test coverage criteria including all branches, all extreme values of variables and a range of number of iterations of each loop. Integration testing is required to exercise all interfaces. System tests are required for functional and non-functional requirements. System tests are traced to the software specification and requirements to show that all requirements are satisfied.

3.5.5 Validation Tests

Validation tests are derived from the software requirements and are not disclosed to the design team prior to testing. The selection of tests must be justified, but all safety functions must be tested. The testing is carried out on the target hardware in condition approaching as nearly as possible to actual use.

3.6. Independence

Def Stan 00-56 requires an independent safety auditor (ISA) is appointed whenever preliminary hazard analysis has revealed risks in classes A or B. The ISA must be acceptable to the MOD's project manager; he must be free of 'possible conflicts of interest' and 'sufficiently commercially and managerially independent' from the design function. Responsibilities of the ISA include auditing the project for compliance with the standard, checking the adequacy of the safety requirements specification, auditing the hazard analysis and risk estimation activities, carrying out sample system analyses and auditing the safety verification test programme. The ISA prepares audit report and endorses the statement of risk classification.

Verification and validation activities required by Def Stan 00-55 are planned by a team which is independent, up to senior management level, from the design team. The V & V team either carry out or review all testing and check the correctness of the formal arguments.

Def. Stan. 00-55 requires an independent safety auditor (ISA) who is 'commercially and managerially' independent from the design authority to be named in the software safety plan and allowed direct communication with the MOD's project manager. The ISA audits all aspects of work relating to safety and produces an audit report. The ISA endorses the certificate of design for the safety related software.

4 CIVIL AVIATION: RTCA DO-178B

RTCA/DO-178B Software considerations in airborne systems and equipment certification.

4.1. Scope

DO-178B is a guidance document containing recommendations produced by industry consensus. It does not have official standard status. The guidance covers 'aspects of airworthiness certification that pertain to the production of software for airborne systems'. Software life cycle processes covered include planning, development, verification, configuration management, quality assurance and liaison with certification authorities. The requirements for software life cycle data to support certification are given.

The system life cycle is described in order to show its relation to the software life cycle, but system life cycle processes such as system safety assessment are not covered and operational aspects of the software life cycle are not included. The allocation of systems functions to software and the severity of failures of system functions are products of the system life cycle. This information is made available to the determination of software levels described in Section 4.3 below.

The division of responsibilities between the applicant for certification and their suppliers is not covered, nor are personnel qualifications. These aspects of certification are determined by each certification authority.

4.2. Safety Life Cycle and Safety Management

DO-178B does not identify a specific safety life cycle. However, relevant requirements include production of the following documents:

- 1) Plan for Software Aspects of Certification
- 2) Software Quality Assurance Plan
- 3) Software Accomplishment Summary.

The plan for software aspects of certification is used by the certification authority to approve the proposed software development life cycle. The plan includes an overview of the system and its software, the basis for certification including the software level, the proposed life cycle and the life cycle data to be produced, and additional information, such as the proposed use of multiple-version dissimilar software or user-modifiable software.

The software accomplishment summary is the primary data item showing that an applicant for certification has complied with the plan for software aspects of certification. The summary includes a statement of the software size and resource requirements, a description the actual life cycle including deviations from the plan, references to the life cycle data and a statement of compliance.

Quality assurance assesses the software development processes to ensure that they comply with the processes specified during the software planning process and that the criteria for transition between development phases are satisfied. Audits are carried out to ensure that plans exist and are consistent with the standard, that deviations from plans are detected and that the software development process takes account of inputs from the system safety assessment.

Prior to delivery, a software conformity review is required to gain assurance that all stages of the software development process have been completed and that all the software life cycle data has been generated.

DO-178B also contains requirements for liaison by an applicant with the certification authority throughout the life cycle. An overview of the certification process is included for information.

4.3. Safety Integrity Levels

Five software levels are defined, based on the severity of the system failure condition which could be caused by a software fault. The following categorisation of failures is used, derived from FAA and JAA regulations:

Catastrophic failure prevents continued safe flight and landing

Hazardous/ Severe-Major large reduction in safety margins; crew unlikely to cope with conditions; potentially fatal injuries

Major	reduction in aircraft's capability or crew's ability to cope
Minor	increase in crew workload but conditions within their capabilities; slight reduction in aircraft safety margins
No Effect	no effect on aircraft's operational capability, nor increase in crew workload

The software levels are defined as follows, depending on the category of the failure which anomalous software behaviour could cause or contribute to:

- A catastrophic system failure
- B hazardous/severe-major system failure
- C major system failure
- D minor system failure
- E no effect; further requirements of DO-178B do not apply to software confirmed at this level

The software level determines the effort required to meet certification requirements. In particular, at the lower software levels, the software verification requirements are reduced, less data derived from the software lifecycle is required for certification and there are fewer requirements for independence.

4.4. Software Development Life Cycle

DO-178B does not prescribe a preferred software life cycle. Instead, the requirements apply to processes common to most life cycle and the interactions between them. The life cycle must be defined as a number of processes, with transition criteria which determine when a process may be entered. Each process consists of a defined sequence of activities, required inputs and resulting outputs. Life cycles for incremental development of software can be accommodated.

Three categories of process are distinguished:

- 1) the software planning process
- 2) the software development process
- 3) integral processes; these are performed concurrently with other life cycle processes and include software verification (see Section 4.5), configuration management and software quality assurance.

Guidance on the selection of the software development environment includes a preference for the use of qualified tools; however, qualification in this context is not defined.

Specific guidance is given for the selection and use of the software language and the compiler, including the following:

- 1) optimisation of object code is permitted provided that the structural test coverage requirements can be met
- 2) means to detect and test any object code which is not directly related to source code, such as an array bounds check, may be required
- 3) verification of object code may need to be repeated if a new compiler version is introduced.

The software development processes are requirements, design, coding and integration. Objectives and activities are specified for each of these processes; for example, two objectives of the requirement process are:

- 1) the system requirements allocated to software should be analysed for ambiguities, inconsistencies and incompleteness
- 2) each system requirement allocated to software should be traceable to one or more software requirements.

Equipment designed to include several configurations by using deactivated code or data which is not used, is permitted. Use of software patches is also permitted in restricted circumstances. Guidance emphasises the adoption of a planned approach, evidence for correct functionality and configuration management.

4.5. Software Validation and Verification

The software verification process assesses the products of the development progress using review, analysis and testing. The objectives of the verification process is to verify that the system requirements allocated to software have been developed, through software requirements, software architecture and low-level requirements and source code into executable object code satisfying the requirements.

Review and analyses are applied to the results of the software development process and to the results of the verification process itself, including high and low level requirements, source code, test cases, test procedures and test results. The objectives of the reviews vary for the different products; for review of requirements the objectives include compliance with the requirements at the level above, accuracy, compatibility with the target hardware and verifiability.

Testing has two objectives, which are to demonstrate that:

- 1) the software satisfies its requirements
- 2) error which could lead to hazardous failures have been removed.

These objectives are satisfied by basing test cases primarily on requirements, including normal range test cases and robustness test cases, which aim to establish conditions revealing potential errors. Requirements coverage analysis ensures that all requirements have been tested and code structural coverage analysis determines whether all software structures have been exercised. If some software structures cannot be tested by requirements based test cases, additional testing is carried out. A justification for this must be given.

Software can be tested at three levels of integration: testing integrated software on target hardware is preferred, but testing at lower levels of integration may also be required to achieve the software coverage. The level of structural coverage required depends on the software level:

Coverage	Level	Code Level
modified condition/decision	A	object code
decision	A, B	source code
statement	A, B, C	source code
data and control coupling	A, B, C	source code

Object code coverage is only required for level A software and only when the compiler produces object code, such as array bound check, which does not directly correspond to a source level statement.

4.6. Independence

DO-178B requires particular process objectives to be 'satisfied with independence'. However there is no definition of independence; the proposed means to satisfy this requirement must be specified in the appropriate plan, such as the software verification plan. The objectives for which independence is required depend on the software level. At the most critical level, some objectives in the following sections of DO-178B must be satisfied with independence:

- 1) verification of software requirements (3 out of 7 objectives)
- 2) verification of software design (6 out of 13 objectives)
- 3) verification of software coding and integration (3 out of 7 objectives)
- 4) testing of integrated software (2 out of 5 objectives)
- 5) software verification (all objectives)

In addition, all software assurance objectives must be satisfied with independence at all software levels

5 US DEFENCE: MIL-STD-498/882C

MIL-STD-498	Software development and documentation
MIL-STD-882C	System safety program requirements

These US Military standards have been selected as the nearest available equivalents to Def-Stan 00-56/55; however the equivalence is only approximate. The total scope covered by these standards is rather wider than that of Def-Stan 00-56/55, since MIL-STD-882C applies to safety-related systems of all sorts, rather than just programmable electronic systems and MIL-STD-498 applies to all sorts of software development, rather than just safety-related software. Additionally, the US standards are not part of a pair as the Def-Stan are; for this reason the following summary distinguishes between the two standards.

5.1. Scope MIL-STD-498

This standard is intended to establish uniform requirements for software development and documentation. Software development covers all activities resulting in software products, including new development, modification, reuse, re-engineering and maintenance. The standard is intended to be tailored for any application, by selecting the subset of the requirements and data items applicable.

The standard describes the activities involved in software development, from project planning through to software deployment, including configuration management, quality assurance, problem reporting and technical and management reviews. The standard also specifies the documentation of these activities as Data Item Descriptions (DIDs).

Additional guidance is provided on:

- 1) the interpretation of the standard for reusable software products
- 2) classification of problems submitted to the corrective action system
- 3) the evaluation of software products
- 4) possible indicators to be used by management for program monitoring
- 5) the joint management reviews which may be held during a software development project.

MIL-STD-882C

This standard applies to all DOD systems and facilities. The requirements for a safety program apply to all activities in a system life cycle, from research, through design and test to maintenance and disposal. The standard provides requirements for developing and implementing a system safety program, in order to identify hazards and prevent mishaps. A number of data item descriptions, which are published separately, are referenced.

The standard consists of a number of tasks and is tailored to each application by selecting the applicable tasks. The range of possible tailorings of the standard is illustrated by a statement in the guidance of Appendix A that, for programs of low safety risk, the only requirements might be for the system safety assessment task to be carried out. The tasks are grouped into the following sections.

Task Section 100 Program Management and Control

The tasks in this section cover establishing and planning a safety program. Other topics are management of subcontractors, safety program reviews and audits, progress reporting, contractor support for safety working groups, and use of a hazard log to track hazards from identification to resolution. Some of these areas are described in Section 5.2 below.

Task Section 200 Design and Integration

The tasks in this section cover the safety analysis required at different stages of the system development life cycle. These are described in Section 5.2 below. There are also tasks for hazard analysis of operational and support activities and for hazards to health.

Task Section 300 Design Evaluation

This section includes a task for a system safety assessment, described in Section 5.2 below. Another task requires safety reviews to accompany all forms of engineering change proposals. The final task is concerned with the application of safety principles during system testing and evaluation, for example covering matters of safety on ranges.

Task Section 400 Compliance and Verification

The first task in this section is for safety verification, which describes the requirements for verification of the safety properties of a system. The second task is for safety compliance assessment, which draws on the safety assessments of other tasks and demonstrates compliance with appropriate safety standards. Two further tasks are concerned with hazards from explosive ordnance.

5.2. Safety Life Cycle and Safety Management

MIL-STD-882C

The majority of MIL-STD-882C tasks can be considered to be part of the safety life cycle. Only the major requirements are described here.

A safety program is established by planning safety tasks, providing qualified people to accomplish these tasks, establishing their authority and setting up communications with safety and functional aspects of a program. Communication with organisations with specific safety expertise is also required. The safety program must have milestones and these should be related to major program milestones. The planning process results in a system safety program plan, which details the safety tasks to be performed.

A large program may involve several contractors, who must also have system safety programs; in this case an integrated system safety program plan is prepared to integrate these safety programs, to describe the flow down of system safety requirements to subcontractors and to specify the safety analyses delivered by the subcontractors to the integrator.

The safety program must be supported by reviews and audits, and by presentations to safety certification authorities. Only general requirements are stated in this area, with additional details to be specified in a statement of work. A hazard log must be maintained and a procedure developed to document the progress of all hazards through the safety process, providing an audit trail of hazard resolutions.

The standard emphasises the importance of early identification of hazards so that safety can be achieved by design; hazards identified in system concept stage are investigated further as the design proceeds. The stages of analysis are:

- 1) preliminary hazard list; potential hazard areas are identified following the system concept definition
- 2) preliminary hazard analysis; a proposed design or function is evaluated for hazard severity and probability and for possible operational constraints
- 3) safety requirements/criteria analysis (SRCA); this is described below
- 4) subsystem hazard analysis; the compliance of each subsystem with its safety requirements is verified and any additional hazards identified
- 5) system hazard analysis; the compliance of the system with its safety requirements is verified and any additional hazards identified.

The techniques to be used for hazard analysis are not specified, but must be approved by the acquirer. The standard does describe techniques for hazard risk assessment, used to classify the acceptability of different hazards; see Section 5.3 below. The standard

implies that hazard risk assessment should follow each phase of hazard analysis, but this is not made explicit.

The SRCA task involves the following stages:

- 1) the safety requirements are determined drawing on the preliminary hazard analysis
- 2) applicable safety design requirements and guidelines from federal, military, national and industry regulations or standards are incorporated
- 3) safety critical software components and safety critical interfaces between hardware and software are identified; however, the method to be used is not described
- 4) a preliminary hazard risk assessment on the safety critical software is carried out, to give a ranking of the criticality of the software
- 5) safety related test requirements are developed.

The final part of the safety life cycle is the safety assessment. This must identify all safety features of the hardware, software or system design and all the hazards which may be present, including the procedural controls which are required. The assessment summarises the safety criteria used and the results of all analyses and tests. A signed statement is required to the effect that all identified hazards have been controlled to reduce risks to the contractually specified levels.

5.3. Safety Integrity Levels

MIL-STD-882C

The standard includes suggested hazard severity categories and hazard probability levels. Hazard severity ranges from catastrophic to negligible in four categories, while hazard probability ranges from frequent to improbable in five levels. Hazard severity and probability together contribute to hazard risk. Example hazard risk assessment matrices are given in the guidance part of the standard.

The guidance also describes a techniques for hazard risk assessment of hazards from safety critical software. Since the probability of software failure cannot be assessed, the concept of 'software control category' replaces probability for software hazard risk assessment. The control categories are:

- I the software exercises autonomous control over hazardous hardware and software failure leads directly to the hazard
- II the software exercises control over hazardous hardware, but its failure is partially mitigated by independent safety systems, or the software displays

information which may require immediate action from an operator to prevent a hazard occurring

- III the software issues commands to hazardous hardware but control requires operator intervention and each hazard is protected by redundant safety measures, or the software generates safety critical information but redundant safety measures protect against hazards
- IV the software does not control safety critical hardware and does not provide safety critical information.

An example software hazard criticality matrix is given, classifying software in five levels. The highest risk is for software in control categories I and II combined with catastrophic hazards or control category I combined with critical hazards. It is suggested that the testing and analysis resource applied to the software should depend on the software risk, but no detailed requirements are given.

5.4. Software Development Life Cycle

MIL-STD-498

The standard describes the activities required for software development. However, it does not specify a fixed order for the activities, nor is delivery of documentation required for all activities performed. Both these features support a more flexible approach to the software development life cycle: both incremental and evolutionary development and concurrent engineering of different system components can be compliant with the standard. Guidance on the possible life cycles, called strategies, and the application of the standard to each strategy is provided in an appendix.

The following software development activities are required:

- 1) project planning and oversight
- 2) establishing a software development environment
- 3) system requirements analysis
- 4) system design
- 5) software requirements analysis
- 6) software design
- 7) software implementation and unit testing
- 8) unit integration and testing.

Further activities relating to verification are described in Section 5.5 below. In addition, the standard describes the following activities, which are integral to all life cycle phases:

- 1) software configuration management
- 2) software product evaluation
- 3) software quality assurance
- 4) corrective action
- 5) joint technical and management reviews.

5.5. Software Validation and Verification

MIL-STD-498

The following activities which relate to software V & V are described:

- 1) unit integration and testing
- 2) CSCI (computer software configuration item) qualification testing
- 3) CSCI/HWCI (hardware configuration item) integration testing
- 4) system qualification testing.

As a representative example, the requirements for CSCI qualification testing are described in more detail. The testing must be carried out with independence (see Section 5.6 below). The testing must be carried out on the target computer hardware, or an equivalent. A software test description should be prepared, containing:

- 1) system identification and overview
- 2) a unique identifier for each test case
- 3) the procedure to prepare the hardware and software for each test case
- 4) the requirement addressed
- 5) the test inputs
- 6) the expected test results and the criteria for acceptance
- 7) the test procedure, as a sequence of steps required to carry out the test
- 8) requirements traceability, showing the test case which addresses each requirement.

The tests are performed, first as a dry-run, then witnessed by the acquirer. If necessary, the software is revised and retested. The results are recorded in a software test report, which includes:

- 1) system identification and overview
- 2) overview of the test results, including an assessment of the software tested, consideration of the impact of the test environment on the test results and recommended improvement to software under test
- 3) summary of the result of each test case as pass, fail or deviation from test procedure
- 4) details of failed tests and tests in which the actual procedure deviated from the specified test procedure
- 5) a log of the testing, including dates, times, location, personnel and hardware and software configuration.

MIL-STD-882C

MIL-STD-882C includes a task for safety verification. This requires the safety requirements to be verified by analysis, inspection, demonstration or test, for the system and its components, including safety critical software. Use of induced or simulated failures should be considered to demonstrate safety under failure conditions. Safety testing may be combined with other testing activities.

5.6. Independence

MIL-STD-498

The requirement for independence in development activities is given in the following table. Independence implies that different personnel must be used; however, it does prevent these original team members relaying their expertise to the independent team.

Activity	Independent from
CSCI qualification testing	Detail design or implementation
System qualification testing	Detailed design or implementation
Software product evaluation	Product development
Quality assurance	Performed activity subject to quality assurance

6 EUROPEAN RAILWAY: PREN 50126/28

prEN 50126 Railway applications - The specification and demonstration of dependability, reliability, availability, maintainability and safety (RAMS).

prEN 50128 Railway applications - Software for railway control and protection systems.

An additional standard in the series, which is not described here is:

prEN 50129 Railway applications - Safety-related electronic railway control and protection systems.

6.1. Scope

prEN 50126

In this standard, dependability is defined in terms of reliability, availability, maintainability and safety (RAMS). The standard defines a process for managing dependability, for specifying requirements for RAMS and for demonstrating that these requirements have been met. However, specific targets for RAMS are not defined. The standard is applicable to all levels in the hierarchy of systems and subsystems, including systems containing software, and at all phases in the life cycle of an application.

In the summary below, only requirements relating to safety are covered.

prEN 50128

In this standard procedures and technical requirements for the development of software in railway control and protection systems are specified. Both safety-related software and software which is not safety related are covered. In this context, software includes application software, operating systems, support tools and firmware. The use of commercially available software is covered. The standard also specifies requirements for systems configured by application data.

Commercial issues and retrospective application of the standard are excluded from the scope.

6.2. Safety Life Cycle and Safety Management

The dependability management process of prEN 50126 is based on the system life cycle. The process requires:

- 1) definition of safety requirements
- 2) assessment of threats to dependability, including safety hazards
- 3) planning dependability tasks

- 4) implementing dependability tasks
- 5) demonstrating compliance.

The standard describes a system life cycle. For each phase of the life cycle, the objectives, requirements, inputs and deliverables of the RAMS tasks are defined. Although the standard is based on a particular life cycle, alternative life cycles may comply; justification for the use on an alternative life cycle must be given. The standard acknowledges that safety can be in conflict with other RAMS requirements; however, the resolution of such conflicts is the responsible of the applicable Railway Authority.

The following table summaries the principal life cycle phases, up to system acceptance, and the related safety tasks.

Phase	Safety Task
Concept	Review existing safety performance Review safety policy
System Definition	Establish safety plan Define criteria for tolerability of risk Perform preliminary hazard analysis
Risk Analysis	Perform system hazard & safety risk analysis Set up hazard log Perform risk assessment
System Requirements	Specify system safety requirements Define safety acceptance criteria Define safety related functional requirements Establish safety management
Design and Implementation	Implement safety plan, by test, review, analysis Use hazard log Prepare generic safety case
Manufacture and Installation	Implement safety plan, by test, review, analysis Use hazard log Establish installation programme
Validation and Commissioning	Establish commissioning programme Prepare specific safety case
Acceptance	Assess safety case

6.3. Safety Integrity Levels

The standard prEN 50126 defines safety integrity level (SIL) as one of four possible levels specifying the integrity requirement of a safety requirement. A level 4 function has the highest integrity and one of level 1, the lowest. The safety integrity of each requirement is derived by risk assessment. Although the risk assessment process is described in the standard, the rules required to determine the SIL from the risk classification are not given. The standard states that the safety integrity correlates with probability of failure. Further details are determined in each application by the relevant Railway Authority.

The SIL of each subsystem is determined by allocating each system safety requirement to one of the subsystems; the subsystem has the highest SIL of any of the requirements allocated to the subsystem. The standard does give detailed requirements for determining the SIL of a subsystem when the overall safety requirement is implemented by two or more subsystems, for example in a redundant configuration. If a subsystem contains software, the SIL of the software is the same as the SIL of the subsystem.

The safety integrity level of a function determines the system architecture, methods and techniques which must be used to be confident that a system satisfies its integrity requirement. The software development and verification techniques appropriate to each SIL are given in prEN 50128. In prEN 50128, safety integrity level zero is also used, so that the requirements for software with no safety-related requirements can be given.

6.4. Software Development Life Cycle

The standard prEN 50128 requires a software development life cycle to be selected and described in the software quality assurance plan. There is no prescribed software development life cycle. Instead, a number of phases of the development life cycle are identified and the requirements of the standard are given for each phase. For example, the documents which must be produced in each phase are specified. General requirements are for each phase to be divided into elementary tasks, with defined input, output and activity and for the tasks in a phase to be defined fully before the phase begins. Verification is carried out on the outputs of the software development process; the verification activities and reports are described in a software quality assurance plan.

The life cycle phases are:

- Software Planning
- Software Requirements
- Software Design
- Software Module Design
- Coding
- Module Testing
- Software Integration
- Software/Hardware Integration
- Software Validation

Software Assessment
Software Maintenance

Detailed requirements are given for each phase of the life cycle, although the clauses of the standard may cover more than one of the phases identified above. For example, clause 11 'Software Design and Development' covers detailed design and coding. Detailed requirements for design and coding given in this clause include:

- 1) the software size and complexity must be kept to a minimum
- 2) the software design must be based on decomposition into modules
- 3) the programming language chosen must facilitate the identification of errors
- 4) coding standards must specify good programming practices and a standard form for comments.

More detailed requirements are given in tables relating the desirability of applicable techniques and measures to the software SIL. In the table for design and development, 19 techniques are rated, ranging from formal methods, which are highly recommended for levels 3 and 4, to use of structured programming, which is highly recommended at all safety-related levels.

6.5. Software Validation and Verification

Clause 12 of prEN 50128 covers all aspects of verification including:

- 1) software requirements verification
- 2) software architecture and design verification
- 3) software module verification
- 4) source code verification
- 5) software integration testing.

Clause 14 covers validation, which is analysis and testing of the integrated system to ensure compliance with the software requirements, including safety requirements. Software verification activity must follow a software verification plan, which documents:

- 1) the verification criteria, techniques and tools to be used
- 2) the evaluation of verification results
- 3) the role and responsibility of personnel carrying out the tests
- 4) the degree of test coverage required.

Testing must be fully documented and the results must be auditable against the requirements of the software verification plan. The verification and testing techniques

required are given in the following table, where 'R' stands for recommended and 'HR' stands for highly recommended:

Technique / Measure	Integrity Level			
	1	2	3	4
Formal proof	R	R	HR	HR
Probabilistic testing	R	R	HR	HR
Static analysis	HR	HR	HR	HR
Dynamic analysis and testing	HR	HR	HR	HR
Metrics	R	R	R	R
Traceability matrix	R	R	HR	HR

There are additional requirements given in notes. For example, for SIL 3 and 4, the 'approved' combinations of techniques are either formal proof with dynamic analysis and testing, or static analysis with dynamic analysis and testing. Other classifications used in other tables are 'M' for mandatory and 'NR' for not recommended.

6.6. Independence

The standard prEN 50128 requires verification and validation activities to be carried out with independence, depending on the safety integrity level. For example, at SIL 4, the requirements are:

- 1) the roles of project manager, designer and verifier must be taken by different people
- 2) the role of validator must be taken by a person who has none of the other roles and who does not report to the project manager
- 3) the role of assessor must be taken by a person from another company.

7 NUCLEAR POWER: IEC 880

IEC 880 Software for computers in safety systems of nuclear power stations.

7.1. Scope

The standard is applicable to highly reliable software required for computers used in the safety systems of nuclear power plants for safety functions. For the classification of such

functions, IEC 643 is referenced. Requirements are provided for all stages of software generation, including design, development, qualification and operation. Requirements for the documentation of each stage of the software generation are also given.

7.2. Safety Life Cycle and Safety Management

The standard does not describe a specific safety life cycle, nor are there activities specific to safety management. Related requirements include:

- 1) consideration of risk and safety features in the software requirements, including a requirement for self-supervision by the software
- 2) restrictions on the design and implementation of the software
- 3) requirements for software verification and validation
- 4) quality assurance requirements, including a software quality assurance plan.

7.3. Safety Integrity Levels

There is no reference to the safety integrity level concept in the standard.

7.4. Software Development Life Cycle

General requirements concerning the life cycle include:

- 1) the project is divided into phases and each phase of the software life cycle into elementary tasks
- 2) the products of each phase are checked; the phase ends with a critical review
- 3) each phase generates appropriate document; a minimum list of documents is included
- 4) the results of all verification and review activities are recorded.

A software development life cycle is described as part of the system development life cycle. Before the software development, the requirements to be satisfied by the safety system are specified. The software development life cycle consists of the following phases:

- 1) software requirements specification; this is carried out in parallel with the hardware requirements specification and the integration requirements specification activities of the system development life cycle
- 2) software design
- 3) software coding.

Verification is carried out at the transition from one phase to another. Following the software development, system integration and computer system testing are carried out.

A prioritised table of techniques and design guidelines is included. The required practices include:

- 1) top-down development
- 2) verification of intermediate products
- 3) if a change is required, it should be made at the earliest possible life cycle stage
- 4) the program should be decomposed into modules, each handling a specific system operation
- 5) the use of operating systems should be restricted to the simplest possible; none is preferred
- 6) under peak load, the execution time should be short compared to the safe maximum
- 7) use of interrupts is restricted, but permitted if necessary, provided the usage is fully documented
- 8) the output must default to safe values on the occurrence of errors
- 9) code changes by the plant operator must be prevented
- 10) runtime error checks should be implemented, including flow of control checks, parameter checks and array bounds checks
- 11) unstructured control flow features, such as a branch into a loop, must be avoided; the maximum iterations of a loop should be determined by a constant

7.5. Software Validation and Verification

The products of each phase in the software life cycle must be verified before the next phase begins, in accordance with a software verification plan.

Design verification is carried out by review, addressing:

- 1) the completeness of the design with respect to the requirements
- 2) the clarity, testability and maintainability of the decomposition into modules and its impact on performance
- 3) satisfaction of quality requirements.

Module testing of all software must be carried prior to integration of hardware and software, in order to show the presence of intended functions and the absence of any unintended functions. A test specification describes the test environment, the test procedure and the acceptance criteria. A software test report presents the results of the tests.

Integrated system verification is an activity of the system integration phase. Verification is described in the system integration plan. Test cases should be selected to exercise all module interfaces, in a system which is complete as far as possible.

Computer system validation is the final V & V activity in the system development life cycle. The computer system is verified against the safety requirements, following a formal validation plan. Static and dynamic simulation of input signals is required, covering normal operation, anticipated operation occurrences and accident conditions. Each safety function should be tested.

7.6. Independence

Verification is carried by an independent person or team. The management of the verification team must be separate from that of the development team. All communications between verifiers and developers must be in writing.

Prepared by:

**D W Marsh
Software and Systems Integrity Department
ERA Technology**

ANNEX A Standards Relating to Software and Safety

Number	Date	Status	Title
Def Stan 00-31/Iss 1	Jul 87	Interim	The development of safety critical software for airborne systems
Int Def Stan 00-55	Apr 91	Interim	The procurement of safety critical software in defence equipment
Int Def Stan 00-56	Apr 91	Interim	Hazard analysis and safety classification of the computer and programmable electronic system elements of defence equipment
Def Stan 00-55	Jul 95	Draft	The procurement of safety related software in defence equipment
Def Stan 00-56	Aug 95	Draft	Safety management requirements for defence systems containing programmable electronics
JSP 188	Mar 95	Published	Joint MOD/Industry computing policy for military operational systems
JAC Paper 1231	Sep 93	Draft	Def Stan 00-970: Chapter 738/728 software; Chapter 739/729 safety critical software
DOD-STD-2167A	Feb 88	Superseded	Defence system software development
DOD-STD-2168	Apr 85	Superseded	Software quality evaluation
MIL-HDBK-287	Aug 89	Superseded	A tailoring guide for DOD-STD-2167A
MIL-HDBK-347			Mission-critical computer resources software support
Mil-Std-1803/Not 1	Apr 94	Standard	Software development integrity program (Restricted)
MIL-STD-882C	Jan 93	Standard	System safety program requirements
MIL-STD-498	Dec 94	Standard	Software Development and Documentation
MIL-STD-499A/1	Feb 95	Standard	Engineering Management
Cenelec prEN 50126	Nov 95	Draft	Railway applications - The specification and demonstration of dependability, reliability, availability, maintainability and safety (RAMS)

Number	Date	Status	Title
Cenelec prEN 50128	Nov 95	Draft	Railway applications - Software for railway control and protection systems
Cenelec prEN 50129			Railway applications - Safety-related electronic railway control and protection systems
IEC 880	1986	Standard	Software for computers in the safety systems of nuclear power stations
IEC 987			Programmed digital computers important to safety for nuclear power stations
IEC 1508	Jun 95	Draft	Functional safety: safety related systems
HSE PES Guidelines	1987	Published	Programmable electronic systems in safety-related applications: 1. An introductory guide. 2. General technical guidelines.
MISRA Guidelines	Nov 94	Published	Development Guidelines for Vehicle Based Software
RTCA/DO-178B	Dec 92	Published	Software Considerations in Airborne Systems and Equipment Certification
ESA PSS-01-02 Iss 2	Apr 91	Published	Software product assurance requirements for ESA space systems
ESA PSS-05-0 Iss 2	Feb 91	Published	ESA software engineering standards, Issue 2
NASA NSS 1740.13	Jun 94	Published	Software safety standard
IEEE 1228	1993	Standard	Standard for Software Safety Plans
DIN VDE 801 A1	Oct 94	Standard	Principles for computers in safety-related systems; Amendment A1
NES 620 Iss 4	Jun 91	Standard	Requirement for software for use with digital processors
NFF 71-013	Dec 90	Standard	Railway fixed equipment and rolling stock. Data Processing. Software dependability. Adapted methods for software safety analysis
AAMI MDS	Jul 91	Published	Developing safe, effective, and reliable medical software

Number	Date	Status	Title
American Nuclear Society ANS 7.4.3.2	1990	Standard	Application criteria for programmable digital computer systems in safety systems of nuclear power generating stations
EIA SEB6 ver. A	1990	Published	System safety engineering in software development
CSA Q396.1.1	1989	Standard	Quality assurance program for the development of software used in critical applications
CSA Q396.1.2	1989	Standard	Quality assurance program for previously developed software used in critical applications
UL 1998	Jan 94	Published	Safety-related software

Annex B**IEEE Software Standards**

Number	Date	Title
730	1989	Standard for Software Quality Assurance Plans
730.1	1995	Guide for Software Quality Assurance Plans
828	1990	Standard for Software Configuration Management Plans
829	1983/91	Standard for Software Test Documentation
830	1993	Recommended Practice for Software Requirement Specification
982.1	1988	Standard Dictionary of Measures to Produce Reliable Software
982.2	1988	Guide for the Use of the Standard Dictionary of Measures to Produce Reliable Software
990	1987/92	Recommended Practice for Ada as a Program Design
1002	1987/92	Standard Taxonomy for Software Engineering Standards
1008	1987/92	Standard for Software Unit Testing
1012	1986/92	Standard for Software Verification and Validation
1016	1987/93	Recommended Practice for Software Design Descriptions
1016.1	1993	Guide to Software Design Descriptions
1028	1988/93	Standard for software reviews and audits
1042	1987/93	Guide to software configuration management
1058	1987/93	Standard for software project management plans
1059	1993	Guide for software verification and validation

Number	Date	Title
1061	1992	Standard for a software quality metrics methodology
1074	1995	Standard for developing software life cycle processes
1074.1	1995	Guide for developing software life cycle processes
1219	1992	Standard for software maintenance
1220	1995	Trial-use standard for the application and management of the systems engineering process
1228	1993	Standard for software safety plans
1298	1992	Software quality management systems, Part 1: Requirements
1498	1995	Trial-use standard for information technology - software life cycle processes - software development: acquirer-supplier agreement

ANNEX C DETAILS OF SELECTED STANDARDS

Title:	Safety Management Requirements for Defence Systems Containing Programmable Electronics	Number:	Def. Stan. 00-56
Date:	9 August 1995	Status:	Draft for comment
Produced by:	MOD, UK	Notes:	Previous version Interim Def. Stan 00-56/Issue1 5 April 1991
Related Standards:	Def. Stan. 00-55		
Contents:		Pages:	39 + 78

Part 1	Requirements
Part 2	General Application Guidance
0	Introduction
1	Scope and applicability
2	Related Documents
3	Definitions
4	General Principles
5	Management and Associated Documentation
6	Safety Requirements
7	System Safety Analysis
8	Data Management
9	Test Programme
10	Work Programme
A	Definitions

Title:	The Procurement of Safety Related Software in Defence Equipment	Number:	Def.Stan. 00-55
Date:	July 1995	Status:	Draft for comment
Produced by:	MOD, UK	Notes:	Previous version Interim Def. Stan. 00-55/Issue 1 5 April 1991
Related Standards:	Def. Stan. 00-56		
Contents:		Pages:	Part 1 pg 41 Part 2 pg 69 Annex pg 27
Part 1	Requirements		
Part 2	Guidance		
Section One	General		
Section Two	Safety Management		
Section Three	Roles and Responsibilities		
Section Four	Planning Process		
Section Five	Software Development Process		
Section Six	Certification and In-service Use		
Annex A	Definition of Terms		
Annex B	Documentation		
Annex C	Safety Critical Software Certificate		
Annex D	Software of Lower Integrity Levels		

Title: Software Considerations in Airborne Systems and Equipment Certification
Date: December 1, 1992
Produced by: RTCA, SC-167
Number: RTCA/DO-178B
Status: Published

Contents: **Pages:** 79

1	Introduction
2	System Aspects Relating to Software Development
3	Software Life Cycle
4	Software Planning Process
5	Software Development Process
6	Software Verification Process
7	Software Configuration Management Plans
8	Software quality Assurance Process
9	Certification Liaison Process
10	Overview of Aircraft and engine Certification
11	Software Life Cycle Data
12	Additional Considerations
A	Process Objectives and Outputs by Software Level
B	Acronyms and Glossary of Terms

Title: Software Safety Program requirements
Date: January 1993
Produced by: US DoD

Number: MIL-STD-882C
Status: Standard

Contents:

Pages: Standard pg 16
 Tasks pg 52
 Appendices pg 40

1	Scope
2	Applicable Documents
3	Acronyms and Definitions
4	General Requirements
5	Detailed Requirements
6	Notes
Task 101	System Safety Program
Task 102	System Safety Program Plan
Task 103	Integration/Management of Associate contractors, Subcontractors and Architect and Engineering Firms
Task 104	System Safety Program Reviews/Audits
Task 105	System Safety Group/System Safety Working Group Support
Task 106	Hazard Tracking and Risk Resolution
Task 107	System Safety Progress Summary
Task 201	Preliminary Hazard List
Task 202	Preliminary Hazard Analysis
Task 203	Safety Requirements/Criteria Analysis
Task 204	Subsystem Hazard Analysis
Task 205	System Hazard Analysis
Task 206	Operating and Support Hazard Analysis
Task 207	Health Hazard Assessment
Task 301	Safety Assessment
Task 302	Test and Evaluation Safety

Task 303	Safety Review of Engineering Change Proposals, Specification Change Notices, Software Problem Reports, and Requests for Deviation/Waiver
Task 401	Safety Verification
Task 402	Safety Compliance Assessment
Task 403	Explosive Hazard Classification and Characteristics Data
Task 404	Explosive Ordnance Disposal Data
Appendices	
A	Guidance for implementation of system safety program requirements
para 10	General
para 20	Referenced documents
para 30	System Safety Requirements
para 40	Task Selection
para 50	Rationale and guidance for Task Selections
B	System safety program requirements related to lifecycle phases
C	Supplementary requirements
D	Data Requirements for Mil-Std-882C

Title: Software development and documentation
Date: 5 December 1994
Produced by: US DoD

Number: MIL-STD-498
Status: Standard

Related Standards:

Contents:

Pages: Standard pg 61
 DIDs pg 129

1	Scope
2	Referenced Documents
3	Definitions
4	General Requirements
5	Detailed Requirements
5.1	Project planning and oversight
5.2	Establishing a software development environment
5.3	System requirements analysis
5.4	System design
5.5	Software requirements analysis
5.6	Software design
5.7	Software implementation and unit testing
5.8	Unit integration and testing
5.9	CSCI qualification testing
5.10	CSCI/HWCI integration and testing
5.11	System qualification testing
5.12	Preparing for software use
5.13	Preparing for software transition
5.14	Software Configuration
5.15	Software product evaluation
5.16	Software quality assurance
5.17	Corrective action
5.18	Joint Technical and management review
5.19	Other activities
6	Notes

Appendix

A	List of acronyms
B	Interpreting MIL-STD-498 for incorporation of reusable software products
C	Category and priority classification for problem reporting
D	Software product evaluations
E	Candidate joint management reviews
F	Candidate Management indicators
G	Guidance on program strategies, tailoring, and build planning
H	Guidance on ordering deliverables
I	Conversion guide from DOD-STD-2167A and DOD-STD-7935A

Title:	Railway Applications - The Specification and Demonstration of Dependability, Reliability, Availability, Maintainability and Safety (RAMS)	Number:	Draft prEN 50126
Date:	November 1995	Status:	Draft European Standard
Produced by:	CENELEC		
Related Standards:	prEN 50128, prEN 50129		
Contents:		Pages:	
1	Forward		
2	Introduction		
3	Scope		
4	Normative References		
5	Definitions, Symbols and Abbreviations		
6	Railway Dependability		
7	Management of Railway Dependability		
A	Bibliography (Informative)		
B	Translation of Definitions (Informative)		

Title:	Railway Applications - Software for Railway Control and Protection Systems	Number:	Draft prEN 50128
Date:	November 1995	Status:	Draft European Standard
Produced by:	CENELEC	Notes:	
Related Standards:	prEN 50126, prEN 50129		
Contents:		Pages:	72 (Normative) 60 (Informative)
1	Introduction		
2	Scope		
3	References		
4	Definitions		
5	Objectives and Conformance		
6	Software Integrity Levels		
7	Personnel and Responsibilities		
8	Lifecycle Issues and Documentation		
9	Software Requirements Specification		
10	Software Architecture		
11	Software Design and Development		
12	Software Verification and Testing		
13	Software/Hardware Integration		
14	Software Validation		
15	Software Assessment		
16	Software Quality Assurance		
17	Software Maintenance		
18	Systems Configured by Application Data		
A	Tables (Normative)		
B	Bibliography (Informative)		

Title: Software for Computers in the Safety systems of nuclear power stations
Date: 1986
Produced by: IEC
Number: IEC 880
Status: Standard

Contents: **Pages:** 133

Forward

Preface

Introduction

- Clause 1 Scope and object
- 2 Terms and definitions
- 3 Project structure
- 4 Software requirements
- 5 Development (design and coding) of safety system software
- 6 Verification
- 7 Hardware/software integration
- 8 Computer system validation
- 9 Maintenance and modification
- 10 Operation

Appendices

- A System development lifecycle and details of software requirements
- B Detailed recommendations for the development (design and coding) of safety related software
- C outline for the software performance specification
- D Language, translator, linkage editor, etc
- E Software testing
- F List of documents needed

Title: Functional safety - safety-related systems
Date: June 1995
Produced by: IEC, Working groups 9 & 10, SC 65A
Number: EC 1508
Status: Draft

Contents: **Pages:** (1) 57 (2) 59 (3) 52
(4) 19 (5) 42 (6) 54
(7) 64

- Part 1: General Requirements
- Part 2: Requirements for electrical/electronic/programmable electronic systems
- Part 3: Software requirements
- Part 4: Definitions and Abbreviations of Terms
- Part 5: Guidelines on the application of Part 1
- Part 6: Guidelines on the application of Parts 2 and 3
- Part 7: Bibliography of techniques

Contents of Part 3:

- 1 Scope
- 2 Normative references
- 3 Definitions and abbreviations
- 4 Conformance to this international standard
- 5 Competence of persons
- 6 Safety management
- 7 Software safety lifecycle requirements
- 8 Functional safety assessment
- 8.2 Requirements
- 9 Documentation
- 10 Guidance on deriving application specific standards
- Annex A Guidelines to the Selection of Techniques and Measures
- Annex B Detailed Tables
- Annex C Application of Parts 2 and 3