Ministry of Defence

Defence Standard

00-55(PART 1)/Issue 2 1 August 1997

REQUIREMENTS FOR SAFETY RELATED SOFTWARE IN DEFENCE EQUIPMENT

PART 1: REQUIREMENTS

This Part 1 of Def Stan 00-55 supersedes INTERIM Def Stan 00-55/Issue 1 dated 5 April 1991

AMENDMENTS ISSUED SINCE PUBLICATION

AMD NO	DATE OF ISSUE	TEXT AFFECTED	SIGNATURE & DATE

Revision Note

Issue 2 of this Standard has been prepared to take account of comments received from users of Interim Issue 1 and to comply with current MOD policy.

Historical Record

Interim Issue 1	Initial Publication	5 April 1991

REQUIREMENTS FOR SAFETY RELATED SOFTWARE IN DEFENCE EQUIPMENT

PART 1: REQUIREMENTS

PREFACE

This Part 1 of Def Stan 00-55 Supersedes INTERIM Def Stan 00-55 (Part 1)

i This Part of the Standard describes the requirements for procedures and technical practices for the development of Safety Related Software (SRS). These procedures and practices are applicable to all MOD Authorities involved in procurement through specification, design, development and certification phases of SRS generation, and maintenance and modification. This Standard has been produced for the MOD under the authority of the CIS Procurement Board.

ii This Standard is one of a family of standards dealing with safety that is being developed or adopted by the MOD, taking into account international standardization activities and supporting research and development.

iii This Standard has been agreed by the authorities concerned with its use and is intended to be used whenever relevant in all future designs, contracts, orders etc and whenever practicable by amendment to those already in existence. If any difficulty arises which prevents application of this Defence Standard, the Directorate of Standardization shall be informed so that a remedy may be sought.

iv Any enquiries regarding this Standard in relation to an invitation to tender or a contract in which it is incorporated are to be addressed to the responsible technical or supervising authority named in the invitation to tender or contract.

v This Standard has been devised for the use of the Crown and its contractors in the execution of contracts for the Crown. The Crown hereby excludes all liability (other than liability for death or personal injury) whatsoever and howsoever arising (including, but without limitation, negligence on the part of the Crown its servants or agents) for any loss or damage however caused where this Standard is used for any other purpose.

CONTENTS PAGE

REQUIREMENTS FOR SAFETY RELATED SOFTWARE IN DEFENCE EQUIPMENT

PART 1: REQUIREMENTS

Section One. General

0 Introduction

0.1 Safety Related Software (SRS) is software that relates to a safety function or system and embraces all Safety Integrity Levels (SIL). Safety Critical Software (SCS) is software that relates to a safety critical function or system, ie software of the highest safety integrity level (S4), the failure of which could cause the highest risk to human life.

0.2 The safety integrity requirements of the SRS components in the equipment will have been determined by means of a preliminary hazard analysis and safety analysis in accordance with Def Stan 00-56. This Standard places particular emphasis on describing the procedures necessary for specification, design, coding, production and in-service maintenance and modification of SCS (safety integrity level S4). It also details the less rigorous approaches that are required to produce software of lower safety integrity levels (S1 to S3).

0.3 Assurance that the required safety integrity has been achieved is provided by the use of formal methods in conjunction with dynamic testing and static analysis. A Software Safety Case, which justifies the suitability of the software development process, tools and methods, is required for software of all integrity levels. This Software Safety Case contributes to and forms part of the equipment safety case required by Def Stan 00-56.

0.4 Guidance on the requirements in this Part of the Standard is contained in Part 2; *Part 1 should be read in conjunction with Part 2*. This Standard assumes a degree of familiarity with software engineering and the conduct of MOD projects.

1 Scope

1.1 This Standard specifies the requirements for all SRS used in Defence Equipment. It relates only to software and does not deal with safety of the whole system. Evidence of the safety principles applied during the development of the SRS contributes to the overall system safety case.

1.2 This Standard contains requirements for the tools and support software used to develop, test, certify and maintain SRS through all phases of the project life cycle.

2 **Warning**

This Defence Standard refers to a number of procedures, techniques, practices and tools which when followed or used correctly will reduce but not necessarily eliminate the probability that the software will contain errors. This Standard refers only to technical suitability and in no way absolves either the designer , the producer, the supplier or the user from statutory and all other legal obligations relating to health and safety at any stage.

3 Related Documents

3.1 The following documents and publications are referred to in this Standard.

ISO 9001	Quality Systems - Model for Quality Assurance in Design/Development, Production, Installation and Servicing.
ISO 9000 - 3	Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software.
Def Stan 00-22	The Identification and Marking of Programmable Items.
Def Stan 00-56	Safety Management Requirements for Defence Systems .
Def Stan 05-57	Configuration Management.
Def Stan 05-91	Quality System Requirements for Design/Development, Production, Installation and Servicing.
Def Stan 05-95	Quality System Requirements for the Design, Development, Supply and Maintenance of Software.
Def Stan 05-97	Requirements for Deliverable Quality Plans.
Int Def Stan 00 Software	Logistic Support Analysis - Application 60 (Part 3) to Aspects of Systems.
JSP 188	Requirements for the Documentation of Software in Military Operational Real Time Computer Systems.
RTCA/DO-178	Software Considerations in Airborne Systems and Equipment Certification.

3.2 Reference in this Part of the Standard to any related documents means, in any invitation to tender or contract, the edition and all amendments current at the date of such tender or contract unless a specific edition is indicated.

3.3 In the event of a conflict between the text of this Standard and the references cited herein, the text of this Standard shall take precedence.

3.4 Part 2 of this Standard lists documents that, although not directly referenced in this Part of the Standard, contain detailed information that is related to the scope of this Standard.

3.5 The related documents can be obtained from

DOCUMENT	SOURCE
International Standards (ISO)	BSI Customer Services BSI Standards 389 Chiswick High Road London W4 4AL
Defence Standards	Directorate of Standardization (Stan 1) Ministry of Defence Kentigern House 65 Brown Street GLASGOW G2 8EX
JSP 188	Ministry of Defence Mwrwg Road Llangennech LLANELLI SA14 8YP
RTCA DO 178	EUROCAE Attention Secretary-General 17 Rue Hamelin Paris Cedex 75783

4 Definitions

The definitions in annex A apply to this Standard.

Section Two. Safety Management

5 Safety Management Activities

The development of SRS requires the following safety management activities and documentation:

a) the production and maintenance of a Software Safety Plan (see clause **6**);

b) the production and maintenance of a Software Safety Case (see clause **7**);

c) the accumulation of evidence for the Software Safety Case in the Software Safety Records Log (see clause **9**);

d) safety analysis of the SRS (see clause **8**) and safety analysis of the SRS development process (see **7.4**);

e) the planning and implementation of software safety reviews (see clause **10**);

f) the planning and implementation of software safety audits (see clause **11**).

6 Software Safety Plan

6.1 The Software Design Authority (see clause **14**) shall develop a Software Safety Plan in the earliest phases of the project prior to the development of the Software Specification, showing the detailed safety planning and control measures that will be employed.

6.2 The Software Safety Plan shall be updated at the commencement of each subsequent project phase.

6.3 The Software Safety Plan shall be agreed with the MOD PM.

6.4 Once the Software Safety Plan has been agreed, the MOD PM shall be notified of any changes prior to implementation.

7 Software Safety Case

7.1 General

7.1.1 The Software Design Authority shall produce a Software Safety Case as part of the equipment safety case defined in Def Stan 00-56.

7.1.2 The Software Safety Case shall present a well-organized and reasoned justification, based on objective evidence, that the software does or will satisfy the safety aspects of the Software Requirement.

7.2 Development of the Software Safety Case

7.2.1 The Software Safety Case shall be continually updated to provide a record of the progress of the implementation and evaluation of the safety requirements for the SRS.

7.2.2 The preliminary Software Safety Case shall be formally issued to the MOD PM after the review of the Software Requirement.

7.2.3 The preliminary Software Safety Case shall justify the way in which the Software Safety Plan is expected to deliver SRS that meets the safety requirements to the specified safety integrity level.

7.2.4 The interim Software Safety Case shall be formally issued to the MOD PM after production of the Software Specification.

7.2.5 The interim Software Safety Case shall refine the preliminary Software Safety Case in the light of the formalization and preliminary validation activities, and shall include the evidence that the safety requirements are met by the formal representation of the Software Specification.

7.2.6 The operational Software Safety Case shall be formally issued to the MOD PM prior to in-service use.

7.2.7 The operational Software Safety Case shall include the complete set of evidence that the safety requirements of the SRS have been met to the specified safety integrity level.

7.2.8 Later versions of the Software Safety Case shall document any deviations from the plans or from justifications contained in earlier versions.

7.3 Safety arguments

7.3.1 The Software Safety Case shall justify the achieved safety integrity level of the SRS by means of a safety analysis of the SRS development process (see **7.4**) supported by two or more diverse safety arguments.

7.3.2 The safety arguments shall include both:

a) Analytical arguments. These shall include:
   i) formal arguments used in preliminary validation to show that the formal specification complies with the safety requirements in the Software Requirement.
   ii) formal arguments that the object code satisfies the formal specification;
   iii) analysis of performance, timing and resource usage;

   iv) analysis of the effectiveness of fault detection, fault tolerance and fail safety features;

b) Arguments from testing. These shall include validation testing of the SRS.

7.3.3 All the safety arguments shall be analysed for common mode failures and sensitivity to variations in the evidence.

7.3.4 The main sources of uncertainty in the safety arguments shall be elaborated.

7.3.5 During development, a summary and appraisal of the results of the V&V activities that contribute to the safety arguments shall be included in the Software Safety Case, with appropriate references to the Software Safety Records Log (see clause **9**).

7.4 Justification of the SRS development process

7.4.1 A safety analysis of the SRS development process shall be carried out to demonstrate how the SRS development process will deliver SRS of the required safety integrity.

7.4.2 The result of the safety analysis of the SRS development process shall be included in the Software Safety Case in the form of a description and justification of the SRS development process.

7.4.3 Where previously developed software is incorporated into the SRS, the safety analysis of the SRS development process shall include analysis of the original development process and any additional activities undertaken to meet the requirements of this standard and the safety integrity requirements of the SRS (see clause **30**).

7.4.4 The safety analysis of the SRS development process shall determine the safety assurance requirements of the tools and manual processes used.

7.4.5 The Software Safety Case shall justify the selection of process data that are to be collected during the SRS development process to measure the performance of the SRS development process.

7.4.6 Acceptance criteria for the process data, described in the Software Safety Plan, should be justified against historical norms and the justification recorded in the Software Safety Case.

7.4.7 The process data described in the Software Safety Plan shall be collected during development and recorded in the Software Safety Records Log and compared with the acceptance criteria.

8 Safety Analysis

8.1 General

8.1.1 The Software Design Authority shall carry out safety analysis of the SRS throughout the project lifecycle as an integral part of requirements engineering, design, implementation, maintenance and reuse.

8.1.2 Safety analysis of the SRS shall:

a) identify potential failures of SRS that may cause new hazards or contribute to existing ones;

b) establish the safety integrity level of each component of the SRS;

c) ensure that safety features and derived safety requirements are correctly implemented, including the role of the SRS in fault detection and fault tolerance.

8.2 The results of the safety analysis of the SRS shall be documented in the Software Safety Records Log and referenced in the Software Safety Case.

8.3 The Software Design Authority shall make the results of items **8.1.2 (a)** and **8.1.2 (b**) available to the Design Authority for inclusion in the Hazard Log.

9 Software Safety Records Log

The Software Safety Records Log shall contain or reference all the data that supports the Software Safety Case as well as details of supporting analyses for the Software Safety Case.

10 Software Safety Reviews

10.1 Software safety reviews shall be carried out at planned points in the project as specified in the Software Safety Plan.

10.2 The Independent Safety Auditor and MOD PM shall be invited to attend the software safety reviews.

10.3 The software safety reviews shall consider the evidence of the SRS development with particular reference to the Software Safety Case, Software Safety Plan, the results of safety analysis of the SRS development process, the safety analysis of the SRS and the Software Safety Records Log.

10.4 The software safety reviews shall identify any additional or corrective actions that are required to be carried out to ensure the safety integrity of the SRS.

10.5 All changes to the Software Safety Case, the Software Safety Plan and the Software Safety Records Log shall be endorsed by a software safety review.

10.6 The results of the software safety reviews shall be recorded in the Software Safety Records Log.

11 Software Safety Audits

11.1 Software safety audits shall be carried out during the development of the SRS, in accordance with Def Stan 00-56.

11.2 The conduct and frequency of software safety audits shall be detailed in a Software Safety Audit Plan.

11.3 The results of the software safety audits shall be recorded in the Software Safety Audit Report.

Section Three. Roles and Responsibilities

12 General

12.1 The Design Authority shall demonstrate to the satisfaction of the MOD PM that all persons appointed by the Design Authority to the project have the seniority, authority, qualifications, training and experience appropriate for their assigned tasks.

12.2 The justification for the key appointments made shall be recorded in the Software Safety Plan (see clause **6**).

13 Design Authority

13.1 The Design Authority shall have responsibility for all aspects of the design and production of SRS and shall be responsible for all safety management.

13.2 The Design Authority shall appoint a Software Design Authority (see clause **14**).

13.3 The Design Authority shall ensure that the requirements of the Software Safety Plan are met by its contractors, subcontractors and suppliers.

14 Software Design Authority

14.1 The Software Design Authority shall appoint a Software Project Manager (see clause **15**).

14.2 The Software Design Authority shall appoint a Design Team (see clause **16**).

14.3 The Software Design Authority shall appoint a V&V Team (see clause **17**).

14.4 The Software Design Authority shall appoint a Software Project Safety Engineer (see clause **19**).

14.5 The Software Design Authority shall ensure that the Independent Safety Auditor has unrestricted access to the people, SRS development process and documentation.

14.6 The Software Design Authority shall certify that the SRS, its development and its documentation comply with the Software Requirement (see clause **33**) and this Standard by signing the SRS Certificate of Design (see clause **38**).

15 Software Project Manager

15.1 The Software Project Manager shall be responsible for discharging the Software Design Authority's responsibilities in accordance with the requirements of this Standard.

15.2 The Software Project Manager shall be the principal point of contact with the Software Design Authority throughout the SRS development.

15.3 The Software Project Manager shall be responsible for the definition and application of adequate and appropriate resources to the SRS development in accordance with the Software Development Plan (see clause **22**).

15.4 The Software Project Manager shall be responsible for production of documentation in accordance with **annex B**.

16 Design Team

16.1 The Design Team leader shall be named in the Software Safety Plan.

16.2 The Design Team shall specify, design and code the SRS.

16.3 The Design Team shall work in accordance with the Code of Design Practice (see clause **27**).

17 V&V Team

17.1 The V&V Team leader shall be named in the Software Safety Plan.

17.2 The V&V Team shall be independent of the Design Team.

17.3 The V&V Team shall produce the Software Verification and Validation Plan (see clause **24**).

17.4 The V&V Team shall carry out or review all testing of the SRS.

17.5 The V&V Team shall check the correctness of formal arguments.

18 Independent Safety Auditor

18.1 The Independent Safety Auditor shall be appointed in accordance with Def Stan 00-56.

18.2 The Independent Safety Auditor , either as an individual or a team, shall be named in the Software Safety Plan.

18.3 The Independent Safety Auditor shall produce a Software Safety Audit Plan (see clause **11**) and shall update the Software Safety Audit Plan as necessary during the project.

18.4 The Independent Safety Auditor shall conduct specific audits to ensure conformance to this Standard and to any project-specific standards, guidelines or codes of practice specified in the Software Safety Plan.

18.5 The Independent Safety Auditor shall produce a Software Safety Audit Report (see clause **11**) detailing the results of the audit activities.

18.6 The Independent Safety Auditor shall endorse the SRS Certificate of Design.

19 Software Project Safety Engineer

19.1 The Software Project Safety Engineer shall be responsible for ensuring that the safety management of the SRS development, as recorded in the Software Safety Plan, is carried out.

19.2 The Software Project Safety Engineer shall attend the software safety reviews (see clause **9**) and shall be responsible for ensuring that any actions arising, related to software safety, are carried out.

19.3 The Software Project Safety Engineer shall endorse the SRS Certificate of Design (see clause **38**)

Section Four. Planning Process

20 Quality Assurance

20.1 All SRS development shall be performed within a quality system that is certificated both to the requirements of ISO 9001 and to the application of ISO 9000-3 guidance.

20.2 A Software Quality Plan shall be produced by the Software Design Authority prior to the commencement of the SRS development process (see **section five**).

20.3 The Software Quality Plan shall detail the planned compliance to this Standard.

21 Documentation

21.1 The Software Design Authority shall produce a set of documentation for the SRS that covers the entire SRS development process.

21.2 Any divergence from the documentation set detailed in annex B shall be agreed between the Software Design Authority, the Design Authority and the MOD PM.

21.3 All documentation shall conform to a contractually agreed interpretation of JSP 188 or to an alternative documentation standard agreed by the MOD PM.

21.4 The documentation shall be sufficiently comprehensive for the entire SRS to be re-established.

21.5 The documentation shall identify all:

a) the deliverable SRS;

b) other deliverable software that supports the use of the SRS;

c) software produced as part of the SRS development;

d) other relevant software, including the software tools used and the host development system.

21.6 All documentation shall be prepared at the appropriate time during the definition, design, development and production phases.

21.7 Documentation shall be processed in accordance with Def Stan 05-91.

22 Development Planning

22.1 The development of the SRS shall be planned by the Software Design Authority prior to the commencement of the SRS development process, in accordance with Def Stan 05-91 and Def Stan 05-95. Development planning shall involve defining the overall process by which the delivered software is produced from the Software Requirement (see clause **33**).

22.2 The development planning shall take into account the probability of changes to the software Requirement and a development process shall be defined which takes this probability into account.

22.3 Development planning shall include the identification of the metrics that will be recorded during development and shall define how those metrics will be used to provide supporting evidence of the suitability of the SRS development process,.

22.4 The output from development planning shall be represented in a Software Development Plan.

23 Project Risk

23.1 As part of development planning, risk identification and risk analysis shall be undertaken to demonstrate that the SRS development can be undertaken with acceptable risk to the success of the project. This analysis shall be undertaken at the start of the SRS development process, as part of the defined risk management process.

23.2 The project risk analysis and the procedures for performing the risk management related to SRS development, shall be included in the Software Risk Management Plan.

23.3 The risk management process shall require the risk identification and risk analysis to be reviewed at milestone points throughout the SRS development.

24 Verification and Validation Planning

24.1 The conduct of Verification and Validation (V&V) activities shall be planned by the Software Design Authority prior to the commencement of the SRS development process.

24.2 The V&V planning shall include:

a) the definition of the responsibilities of the V&V Team;

b) identification of the activities performed by the V&V Team;

c) details of the means by which the appropriate degree of independence from the Design Team will be achieved.

24.3 The output from the V&V planning shall be represented in a Software Verification and Validation Plan.

24.4 As part of the V&V planning, acceptance criteria for each SRS item shall be defined in the Software Verification and Validation Plan.

25 Configuration Management

25.1 The Software Design Authority shall employ an effective configuration management system for all phases of the SRS lifecycle.

25.2 The configuration management system shall conform to the requirements of Def Stan 05-57, except that Engineering Changes (Amendments) shall not be used for SRS.

25.3 The Software Design Authority shall establish a configuration management system containing configuration items that comprise:

a) a library of all SRS components;

b) all internal instances of the SRS;

c) all tools and support software used in the SRS development;

d) all support, development and production hardware related to the SRS;

e) all documentation in the defined documentation set (see **annex B**).

f) the system for which the software was designed.

25.4 The configuration management system shall provide revision control for all configuration items from the time of their creation.

25.5 The configuration management system shall provide the identification and definitions to recreate the complete SRS and its development environment.

25.6 The configuration management system shall include the use of an automated configuration management tool which satisfies the criteria set out under clause **29**.

25.7 Each configuration item shall be uniquely identified, from its time of creation onwards.

25.8 The safety integrity level of each configuration item shall be identified.

25.9 The configuration management system shall include arrangements for disaster protection and protection from subversion and sabotage of the configured items.

25.10 Only authorized personnel shall create, modify or extract configuration items.

25.11 The configuration management system shall be described in a Software Configuration Management Plan.

25.12 The configuration of the completed SRS shall be defined in the Software Configuration Record

26 Selection of Methods

26.1 Complementary methods shall be selected that maximize the probability that the SRS development process will produce safe software.

26.2 Required methods. The methods used in the SRS development process shall include all of the following:

a) formal methods of software specification and design;

b) structured design methods;

c) static analysis (comprising subset analysis, metrics analysis, control flow analysis, data use analysis, information flow analysis, semantic analysis/path function analysis and safety properties analysis);

d) dynamic testing.

27 Code of Design Practice

27.1 A Code of Design Practice shall be produced by the Software Design Authority prior to the commencement of the SRS development process.

27.2 The Code of Design Practice shall define the methods selected for the SRS development, and the procedures for the use of the methods and their supporting tools.

28 Selection of Language

28.1 High-level language requirements. The SRS shall, except where permitted in **28.5**, be programmed in an implementation language which is a high-level language, or pre-defined subset of a high-level language. This implementation language shall have the following characteristics:

a) strongly typed;

b) block structured;

c) formally-defined syntax;

d) predictable program execution.

28.2 Justification for the choice of implementation language shall be provided in the Software Safety Case.

28.3 Coding practices for the implementation language shall be provided in the Code of Design Practice.

28.4 If the implementation language is a subset of a high-level language, the definition of the subset and the means by which the subset will be enforced shall be provided in the Code of Design Practice.

28.5 Use of assembler

28.5.1 The only permissible exception to the production of the SRS in a high-level language shall be the use of assembler language for small sections of code in the following circumstances:

a) Sections of the software where close interaction with hardware is required that cannot easily be accommodated with a high-level language.

b) Sections of the software where performance constraints cannot be met by a high-level language.

c) Very small applications where the use of a high-level language, compilation system and more powerful processor would increase, rather than reduce, safety risk.

28.5.2 Each use of assembler within the SRS shall be agreed with the Design Authority, the Independent Safety Auditor and the MOD PM.

28.5.3 Each use of assembler within the SRS shall be justified in the Software Safety Case.

28.5.4 Rules for programming in the particular assembler language shall be detailed in the Code of Design Practice.

28.6 Compilation systems

28.6.1 Compilation systems shall meet the requirements of clause **29**.

28.6.2 Compilation systems shall meet both of the following requirements:

a) The version used shall be validated with an approved international or national validation certificate.

b) They shall be verified to the extent indicated from the safety analysis of the SRS (see clause **8**) when used in accordance with the Code of Design Practice. As a minimum they shall have been developed within a recognized quality system conforming to ISO 9001 and ISO 9000-3 or equivalent or shall have a demonstrable track record in correctly compiling the defined language subset.

28.6.3 The required safety assurance of the compilation system shall be determined through the conduct of safety analysis of the SRS development process (see **7.4**) which specifically addresses the effectiveness of the planned verification activities in showing conformance of object code to source code (see also **36.6**).

28.6.4 The compilation system or its settings (including compilation options) shall not be changed during the SRS development process without assessing the implications for object code verification.

29 Selection of Tools

29.1 All tools used in the SRS development shall have sufficient safety assurance to ensure that they do not jeopardise the safety integrity of the SRS.

29.2 The safety assurance requirements for the tools shall be deduced from the safety analysis of the SRS development process (see **7.4**). This analysis shall define the safety assurance required of each tool with respect to its use in the project.

29.3 Process hazards shall be identified where the level of safety assurance required is not met by the process and tools and where the limitations of particular tools introduce a specific technical flaw into the process.

29.4 Appropriate safeguards shall be put into place as defence against the identified process hazards such that the complete development process achieves the required safety assurance.

29.5 Tool selection criteria

29.5.1 Each tool shall be evaluated to determine conformance with the required safety assurance requirements for the tool's proposed use in the development of the SRS.

29.5.2 In addition to conformance to safety assurance requirements, tool selection criteria should include usability, interoperability, stability, commercial availability, maintenance support and familiarity to the Design Team.

30 Use of Previously Developed Software

30.1 Where previously developed software is to be used in a new or modified system it shall not adversely affect the safety of the new system. Safety analysis of the SRS (see clause **8**) shall be conducted to demonstrate that there are no adverse safety implications resulting from the use of the software in the new system.

30.2 Unreachable code.

Unreachable code should only be allowed to remain in the final application where it can be shown that the risks of leaving it in are less than the risks of modifying the code to remove it.

30.3 Previously developed software that is to be used in the final delivered equipment should conform to the requirements of this Standard.

30.4 Previously developed software not developed to the requirements of this Standard

30.4.1 Reverse engineering and V&V activities shall be carried out on any previously developed software that is required to be used in the final application but that has not been produced to the requirements of this Standard.

30.4.2 The extent of the reverse engineering and V&V activities shall take into account the rigour of the previously developed software's development process, the extent and functionality of the previously developed software, and its in-service history.

30.4.3 The degree of reverse engineering and V&V activities and subsequent suitability for use of the software shall be assessed through safety analysis of the software development and reverse engineering process.

30.5 In-service history

In-service history shall only be taken into account as evidence of the integrity of the previously developed software where reliable data exists relating to in-service usage and failure rates.

30.6 All changes to previously developed software made as part of its incorporation in the SRS shall be to the requirements of this Standard.

30.7 All previously developed software incorporated in the SRS shall be identified in the Software Safety Case.

30.8 Justification that the previously developed software will not adversely affect the safety of the final application shall be provided in the Software Safety Case.

30.9 All previously developed software incorporated in the final application shall be provided with documentation equivalent to the rest of the SRS as defined in this Standard (see **annex B**).

31 Use of Diverse Software

31.1 Software diversity techniques may be used for additional confidence in safety. Where the diverse software shares common components, for example the Software Requirement, there shall be no relaxation in the requirements of this Standard relating to the shared components.

31.2 Where software diversity is used, the diverse components of the software will effectively be at a lower level of safety integrity compared to a non-diverse implementation and the rules of apportionment of integrity levels defined in Def Stan 00-56 shall apply.

31.3 The reduction of the safety integrity level through diversity shall be demonstrated by safety analysis of the SRS development process and justified in the Software Safety Case.

Section Five. SRS Development Process

32 Development Principles

32.1 The lifecycle for the SRS development

32.1.1 The lifecycle that is followed for the SRS development shall be documented in the Software Development Plan (see clause **22**).

32.1.2 The documentation shall provide a decomposition of the SRS development process into its constituent development processes, the tasks within these development processes and their relationships in terms of their inputs, outputs and scheduling.

32.2 Production of the Software Specification, Software Design and code

32.2.1 The Software Design Authority shall provide a progressive refinement of what the SRS should do and how this is achieved which shall be documented in the Software Specification, Software Design and code.

32.2.2 The Software Specification and Software Design shall conform to the following:

a) All functional and all non-functional requirements and constraints shall be explicitly detailed.

b) All safety functions and safety properties shall be explicitly identified.

c) All safety functions and all safety properties shall be included in the formal representation of the Software Specification and Software Design.

d) The formal representation of the Software Specification and Software Design shall represent the entire set of functional requirements and shall not rely on any informal techniques, unless formal representation of a subset of the functional requirements has been agreed between the Software Design Authority, the Design Authority, the Independent Safety Auditor and the MOD PM.

e) If formal representation of a subset of the requirements has been agreed the justification for the subset shall be documented in the Software Safety Case (see clause **7**).

f) If formal representation of a subset of the requirements has been agreed the subset shall be documented in the Software Development Plan.

g) If safety functions or safety properties rely on the characteristics or properties of data which form part of the SRS, the characteristics and properties of the data shall be formally specified and designed.

h) Any informal parts of the Software Specification and Software Design shall not conflict with the formal representation.

32.2.2 (Cont)

i) English commentary shall be provided for all parts of the formal representation of the Software Specification and Software Design.

j) The layout of the documents shall allow the English commentary to be viewed in parallel with the formal representation of the Software Specification and Software Design.

32.2.3 The development of the Software Specification shall be recorded in the Specification Record and the development of the Software Design and code in the Design Record. These provide a record of how the SRS was produced, including investigation of alternative approaches and the verification performed at each stage.

32.2.4 The testing of the compiled code shall be recorded in the Test Record.

32.2.5 To facilitate maintainability of the Software Specification, Software Design and code and to ensure that the development is understandable by people outside the Design Team:

a) assumptions shall be explicitly recorded;

b) the reasoning behind decisions shall be recorded.

32.2.6 Prototype designs and code shall not form part of the SRS.

32.2.7 Prototype designs and code shall be clearly identified as such.

32.3 Traceability

32.3.1 The Software Specification, Software Design and code shall be so structured that each item (function, data or constraint) is uniquely referenced and traced to the corresponding item of preceding documents and to the Software Requirement.

32.3.2 Formal arguments and dynamic tests shall be given traceability to the requirement or part of the design which is being verified.

32.3.3 Traceability shall be maintained throughout the project.

32.3.4 Requirements which evolve during the course of the SRS development process (derived requirements) shall be fed back to the Design Authority for incorporation into the Software Requirement in accordance with the Code of Design Practice (see clause **27**).

32.4 Verification of the Software Specification, Software Design and code

32.4.1 The formal representation of the Software Specification and Software Design shall be syntax and type checked using an appropriate tool.

32.4.2 The results of the above checks shall be recorded in the Specification Record or Design Record as appropriate.

32.4.3 Non-formal parts of the Software Specification and Software Design, including the English commentary, shall be verified to ensure that they are accurate, complete and self-consistent, and that they are consistent with the formal parts and with the Software Specification.

32.4.4 The results of the above verification shall be recorded in the Specification Record or Design Record as appropriate.

32.4.5 Proof obligations shall be constructed that will enable:

a) verification of the internal consistency of the formal parts of each development process output;

b) verification of the formal parts of each development process output against the outputs of the previous development process.

32.4.6 The proof obligations shall be discharged using formal arguments:

a) Formal proof or rigorous argument shall be used in accordance with the Code of Design Practice.

b) Rigorous arguments shall only be used in place of formal proof where the Software Design Authority can justify the reduction in confidence compared with the risk of failure to the satisfaction of the Design Authority, the Independent Safety Auditor and the MOD PM.

c) The degree of detail in rigorous arguments shall be in accordance with the Code of Design Practice.

32.5 Review

32.5.1 Review of formal verification

The proof obligations and the formal arguments that discharge them shall be reviewed by the V&V Team to ensure that:

a) all necessary proof obligations have been constructed;

b) the formal arguments discharge all proof obligations;

c) all formal proofs are complete and correct;

d) all rigorous arguments are complete and correct;

e) rigorous arguments have been used instead of formal proof only where agreed and justified.

32.5.2 Review of Software Specification, Software Design and code

The output from each development process shall be subject to review to the satisfaction of the V&V Team which shall address:

a) the correctness, completeness, consistency and absence of ambiguity of the documentation;

b) the consistency of the style and the conformance to the Code of Design Practice;

c) the mapping of the outputs of the development process to the inputs to the development process, ensuring the traceability of requirements through the process

d) whether the verification meets the requirements of this Standard, is consistent with the Software Verification and Validation Plan (see clause **24**) and is justifiable for the required safety integrity;

e) whether parts of the Software Specification, Software Design or code which have been corrected or amended after commencement of verification have been reverified in accordance with this Standard and the Software Verification and Validation Plan and whether the reverification performed is justifiable for the required safety integrity;

f) whether the experience of using the development and verification processes and tools is consistent with the assumptions made during the safety analysis of the SRS development process (see **7.4**).

32.5.3 Review of anomaly correction

Whenever an anomaly is discovered, correction or justification of the anomaly shall be in accordance with the defined change control system (see clause **27**).

32.5.4 Before corrections are made a review shall consider:

a) whether the anomaly constitutes an error in the SRS;

b) the reasons for the anomaly;

c) the reasons for any earlier failure to detect the anomaly;

d) the impact of any proposed correction;

e) the effect, including on maintainability, of not correcting the anomaly;

f) the possibility of other similar anomalies;

g) the adequacy of the verification processes;

h) the additional verification required after correction of the anomaly.

32.5.5 The conclusions of the review, and the actions to be taken as a result shall be recorded and justified in the Specification Record, Design Record or Test Record as appropriate.

33 Software Requirement

33.1 General

The Software Requirement forms the input to the specification and development processes. Before proceeding with the specification of the SRS, the Software Design Authority shall conduct verification to ensure that:

a) the Software Requirement is self-consistent and unambiguous;

b) there is sufficient information to enable the Software Design Authority to produce the Software Safety Case.

33.2 The verification of the Software Requirement shall ensure that a safety integrity level has been assigned to the SRS which is consistent with the described role of the SRS.

33.3 Any requirements that are incomplete, inconsistent or require further interpretation shall be resolved in accordance with the procedures detailed in the Code of Design Practice.

34 Specification Process

34.1 The Software Design Authority shall produce a Software Specification based on the Software Requirement, in accordance with the development principles (see **32.2**).

34.2 Verification of the Software Specification

The Software Specification shall be verified in accordance with the development principles (see **32.4**).

34.3 Preliminary validation

34.3.1 Preliminary validation shall be conducted to show that the Software Specification embodies the requirements of the Software Requirement.

34.3.2 Preliminary validation shall be conducted using a combination of formal arguments and executable prototyping.

34.3.3 The safety properties and safety functions shall be:

a) validated by formal arguments;

b) included in the functionality of one or more executable prototype;

c) demonstrated by the tests performed on the executable prototype(s).

34.3.4 The approach used for preliminary validation shall be documented and justified in the Software Development Plan.

34.3.5 The Design Authority and users of the SRS in the final application shall be involved in the preliminary validation to the extent that is necessary to ensure that the specified software will perform safely and according to operational need.

34.3.6 The procedures in the Code of Design Practice
for resolving any problems with the Software Requirement and the Software Specification which are found during preliminary validation shall be followed.

34.3.7 The results of the preliminary validation shall be recorded in the Specification Record.

34.3.8 The Software Design Authority shall define and justify in the Software Development Plan the extent to which preliminary validation will be repeated if changes are made to the Software Requirement or to the Software Specification during the development.

34.4 Preliminary validation via formal arguments

34.4.1 Formal arguments shall be constructed to show that the formal specification embodies all the safety features described in the Software Requirement.

34.4.2 The formal arguments shall be recorded in the Specification Record.

34.5 Preliminary validation via executable prototype

34.5.1 Where executable prototyping is used, the executable prototypes shall be developed from the formal specification and from parts of the informal representation which are not covered by the formal specification.

34.5.2 Formal arguments shall be produced which demonstrate the mapping between the formal specification and the executable prototypes.

3**4.5.3** The mapping of non-formal parts of the Software Specification to the executable prototype(s) shall be documented.

34.5.4 The limitations of the executable prototype(s) shall be documented in the Specification Record.

3**4.5.5** The purpose, scope, and functionality covered for each executable prototype shall be documented and justified in the Specification Record.

34.5.6 Tests for the executable prototypes shall be designed to:

a) exercise all safety functions of the SRS;

b) demonstrate to the Design Authority that the SRS will perform according to operational need;

c) comply with criteria which are defined and justified in the Code of Design Practice.

d) document the purpose of the test, the test inputs and the expected results;

e) be produced before the tests are run;

f) be produced by the V&V Team.

34.5.7 The development and verification of the executable prototypes as well as the Executable Prototype Test Specification and test results shall be documented in the Specification Record.

34.5.8 The Executable Prototype Test Specification shall be preserved for use during validation testing (see **37.4**).

34.5.9 The existence of the executable prototypes shall not constrain the design and development of the SRS.

34.6 Checking and review of the specification process

34.6.1 The Software Specification shall be checked and reviewed in accordance with the development principles (see **32.5**).

34.6.2 The review shall include all formal arguments including those produced during preliminary validation of the Software Specification. The review shall check that the formal arguments are sufficient, correct and complete.

34.6.3 If executable prototypes have been produced, the review shall address:

a) the proportion of the requirements covered by the prototypes and whether this is justifiable for the required safety integrity level;

b) the scope and extent of the testing of the executable prototypes;

c) the results of the testing of the executable prototypes.

35 Design Process

35.1 The Software Design shall be developed from the Software Specification in accordance with the development principles (see **32.2**) using the methods and procedures defined in the Code of Design Practice.

35.2 Design normally progresses via levels of abstraction. At each level of abstraction the Software Design shall include all the information required for subsequent levels of design, for coding and for integration steps.

35.3 The Software Design shall be:

a) in accordance with good design practice embodied in the Code of Design Practice;

b) compliant with all limits specified in the Code of Design Practice for the overall size and complexity of the SRS and for the size and complexity of its constituent parts;

c) such that justification can be provided which shows that it meets its specification in terms of both functionality and performance and that it does nothing else;

d) consistent with all non-functional requirements of the Software Requirement, including fault-tolerance, size, capacity, accuracy, maintainability, reliability, usability and configuration.

35.4 The means by which non-functional requirements are implemented shall be:

a) established at the highest level of design;

b) consistent throughout the design process;

c) recorded in the Code of Design Practice.

35.5 Verification of the Software Design

35.5.1 The Software Design shall be verified in accordance with the development principles (see **32.4**).

35.5.2 Performance modelling by analysis and by execution of models shall be undertaken to demonstrate that the Software Design will result in a software product which meets the specified targets for timing, accuracy, sizing and capacity.

35.6 Checking and review of the design process

35.6.1 Every subsystem at every level of design shall be reviewed in accordance with the development principles (see **32.5**).

35.6.2 Where there is more than one subsystem, the overall Software Design shall also be reviewed.

35.6.3 Each review shall consider whether the Software Design provides an implementation of the Software Specification that that satisfies all requirements and constraints.

35.6.4 The performance modelling (see **35.5.2**) shall be reviewed by the V&V Team.

36 Coding Process

36.1 Coding standards

36.1.1 The code shall be written in accordance with coding standards

36.1.2 The coding standards shall define rules that lead to clear source code that is analysable by formal arguments and static analysis.

36.2 The implementation language and coding standards used shall conform to the Code of Design Practice.

36.3 Justification shall be provided to show that the code is not affected by any known errors in the compilation system or target hardware.

36.4 If the run-time system or operating system upon which the SRS is based is not written especially for SRS, it shall conform to the requirements of clause **30**.

36.5 Static analysis and formal verification

36.5.1 Static analysis in accordance with **26.2** shall be performed on the whole of the source code to verify that the source code is well formed and free of anomalies that would affect the safety of the system.

36.5.2 Proof obligations shall be:

a) constructed to verify that the code is a correct refinement of the Software Design and does nothing that is not specified;

b) discharged by means of formal argument.

36.5.3 The requirements of this Standard for static analysis and formal verification shall include any runtime system software, unless this is covered under clause **30**.

36.5.4 If any data used by the SRS are required to have particular characteristics or properties, and the correct behaviour of the SRS relies on these characteristics or properties, the actual data to be used by the SRS shall be shown to have these characteristics or properties.

36.5.5 The static analysis and formal verification shall either be performed by the V&V Team or reviewed by them.

36.5.6 The results of the static analysis and formal verification shall be recorded in the Design Record.

36.6 Object code verification

36.6.1 Justification based on deterministic arguments, on testing or on a combination of both, shall be provided to show that the source code will be correctly implemented on the target processor.

36.6.2 If deterministic justification is used to provide object code verification, it shall consist of:

a) evidence that the compiler has been formally specified and formally verified, and that the mapping between the source and object code produced by the compiler is a mathematically correct transformation;
or

b) static analysis performed as in **36.5** on the object code to verify that the object code implements the source code;
or

c) construction of proof obligations to demonstrate that the object code is a correct implementation of the source code, and discharge of the proof obligations using formal arguments.

36.6.3 If justification of object code correctness is based upon testing:

a) Evidence of low error rates produced by the compiler shall be presented in accordance with **28.6**.

b) The testing shall use the final version of the object code.

c) Tests which are designed to show full coverage of the final version of the object code shall exercise the source code as extensively as those required in **37.2** and shall achieve coverage of every instruction used at object code level.

d) A mapping shall be generated between the control flow of the source code and the control flow of the object code.

36.6.3 (Cont)

e) The source code shall be scrutinized for any potentially dangerous syntax, ie language syntax that is known to be difficult to compile, or that is not generally used, including low level hardware interaction and complex syntax.

f) If the source code contains any potentially dangerous syntax the object code shall be checked for faults.

36.6.4 The object code verification shall be documented in either the Design Record or the Test Record.

36.7 Checking and review of the coding process

36.7.1 The code, the proof obligations and formal arguments shall be checked and reviewed in accordance with the development principles (see **32.5**) by the V&V Team.

36.7.2 The static analysis of the source code shall reviewed by the V&V Team.

36.7.3 All anomalies shall be reviewed in accordance with **32.5.3** to ensure that they are not due to an error.

36.7.4 The source code shall be reviewed in accordance with the development principles (see **32.5**).

36.7.5 Using the results of the formal verification and static analysis, the review shall establish that the source code correctly refines and implements the formal design and conforms to the Code of Design Practice.

36.7.6 The reviews shall be recorded in the Design Record.

36.7.7 Object code verification shall be reviewed to ensure that the source code is correctly implemented on the target processor.

36.7.8 The object code verification review shall consider the justification for the safety integrity of the compilation system balanced against the amount of object code verification performed.

36.7.9 The review of the object code verification shall be recorded with the object code verification.

37 Testing and Integration

37.1 Principles of testing

37.1.1 The testing shall:

a) exercise all properties and functions which have not been demonstrated by formal verification;

37.1.1 (Cont)

b) demonstrate that the formally verified functions of the SRS perform as desired on the target hardware and that the formally verified properties hold;

c) demonstrate that dynamic and performance requirements are satisfied;

d) demonstrate that the functions and properties of the SRS are as specified even when the SRS is tested in stress conditions, including at maximum capacity;

e) provide a diverse argument of the Software Safety Case (see clause **7**).

37.1.2 Before any tests are executed:

a) The scope of the testing shall be documented in the Software Verification and Validation Plan (see clause **24**).

b) The scope of the testing shall be justified in the Software Safety Plan (see clause **6**).

c) Coverage metrics and criteria for test design for each phase of testing shall be defined in the Software Verification and Validation Plan.

d) The justification for the selection or omission of coverage metrics and the selection of targets for each metric shall be provided in the Software Verification and Validation Plan.

37.1.3 For each test, the purpose of the test, the design elements or requirements being tested, all inputs and all expected outputs shall be documented in the Test Specification (see **annex B**).

37.1.4 Where the testing is designed and conducted by the Design Team, the Test Specification shall be reviewed by the V&V Team.

37.1.5 Where the testing is designed and conducted by the Design Team the testing process shall be audited by the V&V Team.

37.1.6 The tests shall be derived from the Software Specification and Software Design.

37.1.7 Tests shall be conducted under configuration control. The test environment, including all tools, stubs, harnesses, test data and any additional software necessary to run the test shall be recorded and stored in a manner which allows the test to be re-run.

37.1.8 Testing which is used as an argument for the correctness of the dynamic properties of the SRS code, or for the correct behaviour of the SRS code on the target processor, shall be

a) conducted on the target processor or, if this is impractical, on an emulator of the target hardware;

b) performed on the final version of the object code (which shall not include instrumentation or other amendments for testing purposes).

37.1.9 If an emulator of the target hardware is used, justification for the correctness of the emulator shall be provided.

37.1.10 Justification shall be provided in the Software Verification and Validation Plan to show that test tools, harnesses or stubs do not affect the object code, or the behaviour and properties that are being tested.

37.1.11 The test results, including the actual outputs shall be documented in the Test Record.

37.1.12 Justification shall be provided in the Test Record for any discrepancies between the actual and expected outputs.

37.1.13 If failures are discovered which result in corrections to the Software Specification, Software Design or code, the corrections shall be reviewed in accordance with **32.5.3** and all tests using the affected code shall be repeated.

37.2 Integration shall follow the strategy defined and justified in the Software Verification and Validation Plan.

37.3 Unit, integration and system testing

37.3.1 Unit and integration tests shall be conducted on individual units and on partially integrated units to demonstrate that the software is executable and that it produces the expected results for the specified test cases.

37.3.2 The criteria for code test coverage of unit tests shall include:

a) all source code statements and all source code branches;

b) all source code variables set to minimum and maximum values as well as an intermediate value;

c) all source code booleans executed with true and false values;

d)all feasible combinations of source code predicates executed;

e) all source code variables of enumerated type set to each possible value;

f) all source code loops executed 0, 1, an intermediate number and maximum times, where this is semantically feasible;

g) special cases, for example source code variables and source code expressions which can take or approach the value zero.

37.3.3 Integration tests shall as a minimum demonstrate the correctness of all interfaces.

37.3.4 System tests shall be traced to the Software Specification and the Software Requirement. The traceability shall demonstrate that:

a) all functions in the Software Requirement and the Software Specification have been executed;

b) all numerical inputs and all outputs have been set to their minimum, maximum and an intermediate value;

c) all booleans, inputs and outputs, have been set to both true and false values;

d) all non-numerical outputs, including error messages, have been tested;

e) all non-numerical inputs have been tested;

f) all testable non-functional requirements, including timing and capacity have been tested.

37.3.5 System tests shall include `stress tests' which exercise the SRS and its data interfaces at the extremes of its performance requirements.

37.3.6 System tests shall include tests designed to demonstrate that the software interfaces correctly with the hardware.

37.4 Validation tests

37.4.1 Validation testing shall be performed to demonstrate that the SRS operates in a safe and reliable manner under all conceivable operating conditions.

37.4.2 Validation testing shall be performed to demonstrate that the SRS performs according to the Software Requirement.

37.4.3 Validation testing shall be performed in accordance with the testing principles (see **37.1**).

37.4.4 Justification shall be provided in the Software Verification and Validation Plan for the number of tests conducted with respect to the number required to generate statistical evidence for the reliability and safety of the SRS.

37.4.5 Validation tests shall be specified, designed and conducted by the V&V Team.

37.4.6 The actual tests to be used shall not be disclosed to the Design Team until the commencement of the testing process.

37.4.7 The validation tests shall be designed from the Software Requirement including any derived requirements, and from interface specifications of the associated hardware.

37.4.8 Validation tests developed from the Executable Prototype Test Specification shall be executed on the SRS.

37.4.9 Validation tests shall exercise all safety functions.

37.4.10 Validation tests shall be included to demonstrate the non-functional aspects of the software, including timing, accuracy, stability and error handling.

37.4.11 Justification shall be provided for the selection of the tests.

37.4.12 Validation tests shall be included which consist of real or simulated data which is representative of real operating conditions.

37.4.13 Validation tests shall be executed on the target processor, in real-time, using the real run-time system and the final version of the object code.

37.4.14 Justification shall be provided in the Software Safety Plan for any differences between the test system used and the final system of which the SRS will be a part.

37.4.15 If any failures of the SRS are discovered during validation testing:

a) the cause of the error, the reasons why the error had not been detected earlier and the potential for similar errors shall be investigated and recorded in the Test Record;

b) changes shall be made to the Code of Design Practice if necessary;

c) the scope of the validation testing shall be reviewed, with a view to extending the testing.

37.4.16 The validation test results, including the actual outputs shall be documented in the Test Record.

37.4.17 Justification shall be provided in the Test Record for any discrepancies between the actual and expected outputs.

Section Six. Certification and In-service Use

38 Certification

38.1 Certification is a pre-condition for the delivery of SRS to the MOD. The Software Design Authority shall submit an SRS Certificate of Design to the Design Authority prior to delivery of the SRS to the Design Authority.

38.2 The SRS Certificate of Design shall be an unambiguous, clear and binding statement by accountable signatories from the Software Design Authority, that the SRS meets its Software Requirement (including safety requirements) and conforms to the requirements of this Standard.

38.3 Certification shall be supported by the evidence in the Software Safety Case (see clause **7**), the Software Safety Records Log (see clause **9**) and supporting documentation.

38.4 The SRS Certificate of Design shall be countersigned by the Independent Safety Auditor to indicate that the auditing has shown that the work has been carried out in accordance with this Standard, as defined in the Software Safety Plan (see clause **6**), to the satisfaction of the Independent Safety Auditor.

39 Acceptance

39.1 Acceptance testing shall be carried out prior to acceptance of the SRS by the MOD PM.

39.2 The acceptance tests shall be performed against an Acceptance Test Specification (see **annex B**) that shall be agreed between the Software Design Authority, the Design Authority and the MOD PM.

39.3 The acceptance test results should be recorded in the Acceptance Test Report (see **annex B**).

39.4 Acceptance shall be based upon the content of the Software Safety Case, the signing of the SRS Certificate of Design and satisfactory acceptance testing.

40 Replication

40.1 Binary images (including firmware) incorporating SRS shall be produced in such a way as to preserve the safety integrity of the SRS.

40.2 The replication shall be under configuration control. The version and configuration of firmware shall be uniquely identified in accordance with Def Stan 00-22.

40.3 The binary images (including firmware) shall be checked by comparison with the master copy of the image held under configuration control.

40.4 The replication process shall be subject to safety analysis as part of the safety analysis of the SRS development process (see **7.4**). The replication process and associated checks shall be defined such that the safety analysis shows that their safety assurance is appropriate for the level of integrity of the SRS.

41 User Instruction

41.1 User Manuals covering the SRS shall be produced, in accordance with recognised standards or guidelines on good user documentation.

41.2 User instruction shall be defined, for promulgation to the end user, that includes all information of which the user should be aware in order to operate the equipment safely in both normal and abnormal conditions.

42 In-service

42.1 The in-service Software Design Authority shall be responsible for the SRS during the in-service phase.

42.2 The in-service Software Design Authority shall establish appropriate procedures for recording in the Software Safety Records Log all reported anomalies in SRS that are discovered during in-service use, whether or not the anomalies result in hazardous situations or accidents.

42.3 Each reported in-service anomaly shall be reviewed and assessed in a timely manner to determine whether there are any safety implications of the anomaly.

42.4 Whenever an anomaly is reported that could prejudice the safety of the system, the in-service Software Design Authority shall ensure that the in-service Design Authority is aware of the problem.

42.5 Whenever an anomaly is reported that could prejudice the safety of the system, the in-service Software Design Authority shall ensure that immediate precautions are taken to prevent an unsafe situation from occurring.

42.6 Each anomaly report shall as far as practicable include the circumstances surrounding the discovery of the anomaly, the likely consequences of the anomaly and any corrective action taken.

42.7 A method of change control shall be defined, and documented in the Software Maintenance Plan, such that:

a) Impact analysis of any proposed change shall be conducted to assess any effect on safety before the change is implemented.

b) The impact analysis shall involve safety analysis to determine the extent of required assurance activities for unchanged parts of the SRS.

c) The results of the impact analysis shall be reported in the Software Safety Case.

d) All changes to the SRS shall be made to the requirements of this Standard, with the software modification process being at least equivalent in formality and rigour to the original SRS development process.

42.7 (Cont)

e) All unrepeated assurance activities relating to unchanged code shall be reviewed to ensure that the activities are unaffected by any changes and that the safety integrity of the software is not compromised by not repeating the activity.

f) Justification for the extent of the assurance activities shall be provided in the Software Safety Case.

g) Following modification, the SRS shall undergo certification and acceptance activities, as detailed in clauses **38** and **39**, equivalent to those conducted on the initial release of the software.

42.8 The procedures for anomaly reporting should be detailed in the Software Maintenance Plan (see **annex B**).

Section Seven. Application of this Standard Across Differing Safety Integrity Levels.

43 Software of Differing Safety Integrity Levels

43.1 All SRS shall be developed within the framework of this Standard.

43.2 Annex D to Part 2 of this Standard categorises compliance to clauses and part clauses of this Standard for software of differing Safety Integrity Levels, as follows:

a) M indicates mandatory requirements. These shall be complied with.

b) J1 indicates requirements where, on a clause or part clause basis, justification shall be provided for any omission or non-compliance.

c) J2 indicates requirements where justification shall be provided for omission or non-compliance of the roles, methods, products or activities described within the clauses.

43.3 Any deviation from annex D to Part 2 of this Standard shall be justified in the Software
Safety Plan and referenced in the Software Safety Case.

Index

I

M

O

P

R

S

Definitions and Abbreviations

A.1 Definition of Terms

A.1.1 Abstraction. A view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information. For example, separation of the specification from the implementation.

A.1.2 Black Box testing. Techniques for test case generation which do not rely on knowledge of the internal structures of the item under test.

A.1.3 Block structured language. A design or programming language in which sequences of statements, called blocks, are defined, usually with `begin' and `end' delimiters, and variables or labels defined in one block are not recognized outside that block.

A.1.4 Complexity. The degree to which a system or component has a design or implementation that is difficult to understand and verify.

A.1.5 Configuration control. Activity to ensure that any change, modification, addition or amendment is prepared, accepted and controlled by set procedures.

A.1.6 Configuration item. An item that may be stored within the configuration management system.

A.1.7 Configuration management. Technical and administrative procedures to:

a) identify and document functional and physical characteristics of any part of the system;

b) control changes to these, and their interfaces;

c) record and report change, its process and implementation.

A.1.8 Defensive design or code. Design or code which prevents the propagation of errors, by incorporating checks even for circumstances which are believed to be infeasible.

A.1.9 Derived requirements. Additional requirements resulting from design or implementation decisions during the development process. Derived requirements are not directly traceable to higher level requirements; though derived requirements can influence higher level requirements.

A.1.10 Development process. A set of tasks which,
when carried out in a defined manner on identified inputs and outputs, constitute an identifiable element of the SRS development lifecycle.

A.1.11 Diversity. The existence of different means of performing a required function, for example, other physical principles, other ways of solving the same problem.

A.1.12 Encapsulation. A software development technique that consists of isolating a system function or a set of data and operations on that data within a module and providing precise specifications for the module.

A.1.13 English commentary. That part of the Software Specification or Software Design written in English.

A.1.14 Error. A system state, resulting from a fault or human mistake, that is liable to lead to a failure.

A.1.15 Executable prototype. An executable form of a formal specification that is used to validate the Software Specification.

A.1.16 Failure. The inability of a system or component to fulfil its operational requirements. Failures may be systematic or due to physical change.

A.1.17 Firmware. The combination of a hardware device and computer instructions and data that reside as read-only software on that device.

A.1.18 Formal arguments. Formal proofs or rigorous arguments.

A.1.19 Formal design. The part of the Software Design written using a formal notation.

A.1.20 Formal method. A software specification and production method, based on mathematics, that comprises: a collection of mathematical notations addressing the specification, design and development processes of software production; a well-founded logical inference system in which formal verification proofs and proofs of other properties can be formulated; and a methodological framework within which software may be developed from the specification in a formally verifiable manner.

A.1.21 Formal notation. The mathematical notation of a formal method.

A.1.22 Formal proof. The discharge of a proof obligation by the construction of a complete mathematical proof.

A.1.23 Formal specification. The part of the Software Specification written using a formal notation.

A.1.24 Hazard Log. The document, established in accordance with Def Stan 00-56, that acts as a record and directory for the safety justification for the system.

A.1.25 Linear Code Sequence and Jump (LCSAJ). A control flow sequence consisting of a series of sequential instructions, followed by a branch to a non-sequential instruction.

A.1.26 Maintainability. The ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changed environment.

A.1.27 Modified condition/decision coverage. A test coverage metric which considers the combinations of multiple predicates within a decision.

A.1.28 Preliminary validation. The tests that are performed on the Software Specification to ensure that the Software Specification correctly implements the Software Requirement.

A.1.29 Proof obligations. Mathematical statements arising during a formal design and development process that must be proved in order to verify the design or development step. Also known as verification conditions.

A.1.30 Reliability. The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

A.1.31 Requirements (software). Functional requirements express what the software must do. Nonfunctional requirements express properties of the final implementation, including accuracy, performance, reliability, user-interface etc.

A.1.32 Reverse engineering. The process of extracting software design information from the source code.

A.1.33 Rigorous argument. An outline of the way in which a proof obligation can be discharged, which presents the main steps but does not supply all the intervening detail.

A.1.34 Safety assurance. The degree of confidence required in the correctness of a particular process or tool.

A.1.35 Safety critical software. Software, including firmware, used to implement a function or component of safety integrity level S4.

A.1.36 Safety related software. Software, including firmware, used to implement a function or component of safety integrity levels S1, S2, S3 or S4.

A.1.37 Safety feature. A feature that is required to remove unacceptable safety risks and constrain the other risks from the defence equipment to a tolerable level (includes safety functions and safety properties).

A.1.38 Safety function. A function that the software must perform in order to maintain the safety of the system.

A.1.39 Safety integrity. The likelihood of a system achieving its required safety features under all the stated conditions within a stated measure of use.

A.1.40 Safety integrity level. An indicator, assigned in accordance with Def Stan 00-55, of the required level of safety integrity.

A.1.41 Safety property. A property that is required to hold during execution of the software to achieve safety.

A.1.42 Safety requirement. A requirement for a safety feature to be met by the SRS.

A.1.43 Segregation unit. A physical unit of the system designed such that the immediate effects of the failure of a component that resides in it are restricted to the unit.

A.1.44 Software. Computer programs, procedures and associated documentation and data pertaining to the operation of a computer system.

A.1.45 Software Requirement. The system level document defining the requirements which, if met, will ensure that the SRS performs safely and according to operational need.

A.1.46 SRS development process. A set of development processes and their relationships in terms of their inputs, outputs and relative timing which together define the means by which the SRS is to be developed.

A.1.47 Static analysis. The process of evaluating software without executing the software.

A.1.48 Strongly typed. A feature of a programming language that requires the type of each data item to be declared, precludes the application of operators to inappropriate data types and prevents the interaction of data items of incompatible types.

A.1.49 Structured design method. A disciplined approach that facilitates the partitioning of a software design into a number of manageable stages through decomposition of data and function. Includes object oriented design methods.

A.1.50 Validation. The process of evaluating software at the end of the software development process to ensure compliance with the Software Requirement. See also preliminary validation.

A.1.51 Verification. The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

A.1.52 White Box Testing. Testing techniques which use knowledge of the internal structure of the item under test. Techniques based on knowledge of the control flow are most common.

A.2 Definition of Abbreviations

A.2.1 ALARP. As low as reasonably practicable

A.2.2 DRACAS. Data reporting analysis and corrective action system.

A.2.3 LCSAJ. Linear Code Sequence and Jump.

A.2.4 MOD PM. MOD Project Manager.

A.2.5 SCS. Safety critical software.

A.2.6 SRS. Safety related software.

A.2.7 V&V. Verification and validation.

Collation Page

Documentation

This annex lists the documentation set to be produced in accordance with this Standard. Annex B to Part 2 of this Standard details the required content of each document.

The Software Design Authority may decide, in conjunction with the Design Authority and the MOD, that the information would be better presented using a different document set to that presented here. For example a number of documents may be incorporated into a single document, provided, firstly, that the complete document set shall incorporate all the contents listed in annex B to Part 2 of this Standard and, secondly, that there is easy traceability between these requirements and their incorporation into the actual document set.

The Software Design Authority in conjunction with the Design Authority and the MOD should decide on whether paper or electronic format is most appropriate for each of these products, and also whether they are to be delivered, or simply retained and made accessible.

B.1 Documentation Set

The following documents comprise the documentation set required by this Standard:

a) Software Safety Plan.

b) Software Safety Case.

c) Software Safety Records Log.

d) Software Safety Audit Plan.

e) Software Safety Audit Report.

f) Software Quality Plan.

g) Software Development Plan.

h) Software Risk Management Plan.

i) Software Verification and Validation Plan.

j) Software Configuration Management Plan.

k) Software Configuration Record.

l) Code of Design Practice.

m) Software Specification.

n) Specification Record.

B.1 (Cont)

o) Software Design.

p) Design Record.

q) Code.

r) Test Record.

s) Test Schedule.

t) Test Report.

u) Acceptance Test Specification.

v) Acceptance Test Report.

w) User Manuals.

x) Software Maintenance Plan.

DEF STAN 00-55(PART 1)/2

Published by and obtainable from:
Ministry of Defence
Directorate of Standardization
Kentigern House
65 Brown Street
GLASGOW G2 8EX

Tel No: 0141 224 2531

The following Defence Standard file reference relates to the work on this Standard - D/D Stan/303/12/3

Contract Requirements

When Defence Standards are incorporated into contracts users are responsible for their correct application and for complying with contract requirements.

Revision of Defence Standards

Defence Standards are revised when necessary by the issue either of amendments or of revised editions. It is important that users of Defence Standards should ascertain that they are in possession of the latest amendments or editions. Information on all Defence Standards is contained in Def Stan 00-00 (Part 3) Section 4, Index of Standards for Defence Procurement - Defence Standards Index published annually and supplemented periodically by Standards in Defence News. Any person who, when making use of a Defence Standard encounters an inaccuracy or ambiguity is requested to notify the Directorate of Standardization without delay in order that the matter may be investigated and appropriate action taken.