BSSC (2002)3 Issue 1.0
14 March 2002

ESA Ground Segment Software Engineering and Management Guide

Part C Document Templates

Prepared by:
ESA Board for Software Standardisation and Control (BSSC)

european space agency / agence spatiale européenne
8-10, rue Mario-Nikis, 75738 PARIS CEDEX, France

DOCUMENT STATUS SHEET

DOCUMENT STATUS SHEET
1. DOCUMENT TITLE: ESA Ground Segment Software Engineering and Management Guide : Part C Document Templates

2. ISSUE	3. REVISION	4. DATE	5. REASON FOR CHANGE
1	0	March 2002	First Issue

Approved, March 2002
Board for Software Standardisation and Control
M. Jones, BSSC co-chairman U. Mortensen, BSSC co-chairman

TABLE OF CONTENTS

PREFACE

This Guide comprises three parts: A, B and C.

This part, Part C, describes the document templates needed to produce software for space system ground segments, and is designed to be applied in all ground segment software engineering projects undertaken by the European Space Agency (ESA). In the past, ground segment software development projects undertaken by ESA and, especially, the European Space Operations Centre (ESOC) have been undertaken according to the ESA Software Engineering Standards, ESA PSS-05-0. ESA now applies the European Co-operation for Space Standardisation (ECSS) Engineering (E), Management (M) and Product Assurance (Q) series of standards for all space software projects. Requirements relating to ground segment software are also specified in the ECSS E-70 Space Engineering: Ground Systems and Operations standard.

This part of the guide describes templates based on the requirements of the relevant ECSS-E, M and Q series standards for ground segment software projects undertaken by the various parts of ESA. The templates also:

- carry over working practices from ESA PSS-05 and ESA Quality Management System, where they fully implement the requirements of the standards
- reflect the lessons learnt in the application of ESA PSS-05.

The Guide was reviewed and revised by the following BSSC Members: Michael Jones (co-chairman) Uffe Mortensen (co-chairman), Alessandro Ciarlo, Daniel de Pablo and Lothar Winzer, assisted by Eduardo Gomez. The BSSC wishes to thank John Brinkworth and John Barcroft for editing the final version.

Requests for clarifications, change proposals or any other comment concerning this guide should be addressed to:

BSSC/ESOC Secretariat	BSSC/ESTEC Secretariat
Attention of Mr M Jones	Attention of Mr U Mortensen
ESOC	ESTEC
Robert Bosch Strasse 5	Postbus 299
D-64293 Darmstadt	NL-2200 AG Noordwijk

Germany	The Netherlands
michael.jones@esa.int	uffe.mortensen@esa.int

This page intentionally left blank

INTRODUCTION

0.1 PURPOSE

This software engineering and management guide concerns the management of the development and maintenance of ground segment software. This guide covers all aspects of software development management for ground segment software including software engineering, documentation, configuration management, software product assurance, and software project management.

This guide describes how to implement the requirements of ECSS-E-40 and ECSS-E-70 on ground segment software projects undertaken by the various parts of ESA. The guide also:

- Carries over working practices from ESA PSS-05 [Ref 24, 25], where they fully implement the requirements of ECSS-E-40 and the other standards
- Reflects the lessons learnt in the application of ESA PSS-05.

The guide complies with the requirements of ECSS-Q-80, Software Product Assurance [Ref. 10], which is itself based on Information Technology Software Life Cycle Processes 12207:1995 [Ref. 1]. As the software developed by this guide is specifically used in ground segments, the guide is also compliant with the applicable requirements of ECSS E-70, Ground Systems and Operations [Ref. 13].

0.2 OVERVIEW

Part A provides an overview of the software life cycle process, in accordance with the E-40 and E-70 standards, and their application in the lifecycle of software for ground systems. It also gives advice on applying this guide to a particular project.

Part B of the guide covers the practices to implement effective management of the development, operation and maintenance of ground segment software. The majority of these practices are defined in the ECSS Management [Ref. 3 to 9] and

ECSS Product Assurance [Ref. 10] series of documents, rather than the ECSS E-40 and E-70 standards.

Part C (this part) provides proposed templates for documents. These templates have been derived mainly from [Ref 20].

0.3 TEMPLATE DOCUMENTS

The documents that are defined are shown in the second table below, together with their lifecycle. They are listed in the sequence in which their templates appear in the rest of the document. Also listed, with their titles shown in italics, are documents for which templates are not currently available.

Note that there are four stages of validation testing, as shown in the first table below, and that three types of document are produced in relation to these stages: plans, specifications and reports. There is a single template for each document type for all validation testing.

Name of Testing Stage	**Location & Platform**	**Sometimes known as**	
Validation Testing against Technical Specification	Supplier Premises	System Tests (can also be carried out as Factory Acceptance Tests: FAT)	
Validation Testing against Requirements Baseline Subset – 1	Supplier Premises	Preliminary Acceptance Tests. Factory Acceptance Tests (FAT)	
Validation Testing against Requirements Baseline Subset – 2	Customer Premises on Development Environment	Preliminary Site Acceptance Tests (PSAT)	Operational Acceptance Tests
Validation Testing against Requirements Baseline	Customer Premises on Operational Environment	(Final) Site Acceptance Tests (SAT)	

Stages of Validation Testing

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**													
					Sofware Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
RB	Interface Requirements Document	Customer defines software interface requirements	Completed (then maintained, under change control)													
RB	System Specification	Customer defines software requirements	Completed (then maintained, under change control)													
MGT	Software Development Plan	Supplier addresses management requirements	Drafted		Overall plan updated; detailed plans for architectural design activity		Overall plan updated; detailed plans for software design engineering		Overall plan updated; detailed plans for software validation and acceptance							
DJF	Software Verification Plan	Supplier defines arrangements for review activities	Drafted		Updated		Updated		Updated			Updated				
DJF	Software Validation Plan	Supplier defines arrangements for validation testing activities (normally against TS)	Drafted		Maintained		Maintained		Finalised							
DJF	Acceptance Test Plan (Software Validation Plan template)	Customer defines how acceptance tests will be performed										Finalised		Maintained		

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**													
					Sofware Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
TS	Interface Control Document	Supplier defines details of software interfaces	Drafted		Finalised		Maintained		Maintained							
TS	Software Requirements Specification	Supplier defines software requirements in response to customer requirements	Drafted		Finalised											
DDF	Software User Manual	Supplier describes what software does and how to achieve it							Created	Maintained		Maintained		Maintained		
DDF	Software Architectural Design	Supplier defines top-level design					Created		Maintained							
DDF	Software component designs	Supplier defines detailed design							Created							
DDF	Source code								Created	Maintained		Maintained				
DDF	*Software Installation Plan*	Identifies how the software is installed in the operational environment										Finalised				
DDF	Build code files													Delivered		
DDF	Executable code files													Delivered		
DJF	Software Requirements Verification Report	Provides evidence of requirements review			Created											Generated

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**													
					Sofware Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
DJF	*Design Choices and Trade-Offs*	Supplier justifies design decisions					Top-level design choices and trade-offs documented		Maintained	Maintained		Maintained		Maintained		
DJF	*Software Reuse Records*		Created		Created		Created									
DJF	Requirements Traceability / Compliance matrices	Trace implementation of requirements	System requirements to sub-system partitions		TS to RB completeness		Top-level architecture traceability		Traceability of detailed design to TS			Acceptance test to RB				
DJF	Software Validation Testing Specifications (against TS)	Defines tests, test cases and procedures for validation against TS							Finalised							
DJF	Preliminary Acceptance Test Specification (Software Validation Testing Specification template)	Defines tests, test cases and procedures for FAT										Created				

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**													
					Sofware Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
DJF	Operational Acceptance Test Specification (Software Validation Testing Specification template)	Defines tests, test cases and procedures for PSAT and SAT												Finalised		
DJF	Software Architecture and Interface Verification Report	Provides evidence of AD review					Created									Created (for design modifications)
DJF	Software Design Verification Report	Provides evidence of DD review							Created							
DJF	Software Code Verification Report	Provides evidence of code review							Created							
DJF	Software Documentation Verification Report	Provides evidence of document review							Created							
DJF	*Software Unit Test Verification Report*	Provides evidence of unit test results review							Created							

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**													
					Sofware Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
DJF	Software Integration Verification Report	Provides evidence of Integration review							Created							
DJF	Software Integration Plan	Supplier plans the integration task, including integration testing					Preliminary version drafted		Finalised							
DJF	Software Unit Test Plan	Provides details of Unit Testing							Created							
DJF	Software Unit Test Report	Provides results of unit testing							Created							
DJF	Integration Test Report	Provides results of integration testing							Created							
DJF	Software Validation Testing Report	Records results of Validation Testing against the Technical Specification. (Same template used for FAT and (P)SAT too.)								Created						
DJF	Preliminary Acceptance Test Results (Software Validation Testing Report template)	Records results of FAT										Created				

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**														
					Sofware Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
DJF	Operational Acceptance Test Results (Software Validation Testing Report template)	Records results of PSAT and SAT												Created		
DJF	Test Specification Evaluation Report	Provides evidence of review of a Test Specification								Created		Created		Created		
DJF	Software Design and Test Evaluation Report	Evaluates the detailed design and test s							Created							
DJF	Software Budget Report	Reports status of technical budget and margins	Produced		Updated		Updated		Updated			Updated		Updated		
DJF	Software Configuration File	Provides a definition of the configuration of the software at each milestone	Produced to identify System Engineering documents		Updated to include Requirements Analysis documents		Updated to include Architectural Design documents		Updated to include all software configuration Items	Updated to reflect current configuration status		Updated to reflect current configuration status		Updated to reflect current configuration status		Updated to reflect current configuration status
PAF	Product Assurance Report	Supplier reports on product assurance activities (on same cycle as other management reports)	produced on management report cycle		produced on management report cycle		produced on management report cycle		produced on management report cycle			produced on management report cycle		produced on management report cycle		

			Lifecycle													
					Sofware Requirements Engineering				Software Design Engineering			Software Validation and Acceptance				
File *key at end*	**Document** *italics means template not available*	**Purpose**	**System Engineering for Software**	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
PAF	Software Product Assurance Plan	Supplier defines plans for measuring and controlling product and process quality	Drafted		Maintained		Maintained		Maintained			Maintained				
O/M	Software Maintenance Plan	Supplier defines maintenance organisation, processes, etc.	May be drafted now or at agreed later stage						Drafted					Finalised		Maintained
O/M	Software Operations Plan	Operator defines approach to operational testing, operation and user support	Outline may be drafted now or at agreed later stage						Drafted					Finalised		Maintained
O/M	*Maintenance Records*	Customer maintains documentation related to problems and change requests														Generated
O/M	Migration Plan, including Migration Justification															Created
O/M	Software Retirement Plan															Created
MGT	Software Configuration Management Plan	Supplier defines how control of product configuration will be applied	Drafted		Maintained		Maintained		Maintained			Maintained				
MGT	Software Progress Report	Supplier reports project status (resource, schedule, financial)	produced on management report cycle		produced on management report cycle		produced on management report cycle		produced on management report cycle	produced on management report cycle		produced on management report cycle		produced on management report cycle		

File *key at end*	**Document** *italics means template not available*	**Purpose**	**Lifecycle**													
					Software Requirements Engineering				**Software Design Engineering**			**Software Validation and Acceptance**				
			System Engineering for Software	**SRR**	**Software Requirements Analysis**	**SWRR**	**Software Architectural Design**	**PDR**	**Design, Coding and Testing, Integration**	**Validation against Technical Specification**	**CDR**	**Preliminary Acceptance Tests**	**QR**	**Delivery, Installation and Operational Acceptance Tests**	**AR**	**Software Operations Engineering / Software Maintenance**
DJF / O/M	Software Problem Report	Records software problems during development or operation							Generated	Generated		Generated		Generated		Generated
DJF	Non-Conformance Report	Records failure to meet specified requirement			Generated		Generated		Generated	Generated		Generated		Generated		
DJF	*Observation Report*	Records unforeseen departures from planned tests and/or outcomes, but which are not failures								Created		Created		Created		
O/M	Problem Analysis Report	Provides assessment, diagnosis, and approach and estimate for rectification														Created
DJF / O/M	Software Change Request	Records a change required and its cost and schedule impacts			Generated		Generated		Generated	Generated		Generated		Generated		Generated
O/M	Software Release Note	Provides information on a new baseline, including changes applied and installation instructions														Created
DJF	Request for Waiver	Requests and grants (when signed) formal approval to ignore a requirement			Generated		Generated		Generated	Generated		Generated		Generated		
MGT	*Lessons Learned*	Supplier assesses lessons learned from the project												Finalised		

Key to File identifiers:

DDF	Design Definition File
DJF	Design Justification File
MGT	Management File
O/ M	Operations File / Maintenance File
PAF	Product Assurance File
RB	Requirements Baseline
TS	Technical Specification

This page intentionally left blank

1. Software Interface Requirements Document

TABLE OF CONTENTS

1. INTRODUCTION
This section should give an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software interface requirements document and specify its intended readership.

1.2 Scope
This section should:
- Identify the software products to which this IRD applies
- Identify the relationship of this document w.r.t. the other documentation specifying the software products.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SYSTEM OVERVIEW

A brief description of the system context in which the interface requirements are specified should be provided in this section. Use of diagrams for the identification of the relevant interfaces should be encouraged to enhance readability. A summary of the functionality of the software shall be also provided.

3 INTERFACE REQUIREMENTS

This section shall provide the requirements and specification of the software interfaces to external entities (software applications, operating system, hardware, etc…). This section shall specify any interface requirements imposed on the system. Requirements related to the following (as applicable) shall be listed:

- Communication interfaces
- Hardware interfaces
- Software interfaces
- HCI interactions.

This section shall be structured around each identified software external interface as shown below.

3.x [name] Interface

This section shall describe the interface protocol applicable to the specified interface

3 REQUIREMENTS FOR DATA PREPARATION

This section shall describe any special requirement for the preparation of data.

2 System Specification

TABLE OF CONTENTS

1. INTRODUCTION
This chapter should give an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this system specification and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to be produced by name
- Explain what the proposed software will do and what it won't
- Describe the relevant benefits, objectives and goals as precisely as possible.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. GENERAL DESCRIPTION
This chapter should describe the general factors that affect the product and its requirements. This chapter does not state specific requirements but makes those requirements easier to understand.

2.1 Product Perspective
This section puts the product into perspective with other related systems. If the product is to replace and existing system, the system should be described and referenced. Ancestors of the product that are no longer in use might be mentioned. If the product is "standalone", it should be stated here.

2.2 General Capabilities

This section should describe the main capabilities and why they are needed. This section should describe the process to be supported by the software, indicating those parts of the process where its is used.

2.3 General Constraints

This section should describe any item that will limit the developer's options for building the software. This section should not be used to impose specific requirements or specific design constraints, but should state the reasons why certain requirements or constraints exist.

2.4 User Characteristics

This section should describe those general characteristics of the users affecting the specific requirements. Many people may interact with the software during the operations and maintenance phase. Some of these people are users, operators and maintenance personnel. Certain characteristics of these people, such as educational level, language, experience and technical expertise impose important constraints on the software.

Software may be frequently used, but individuals may use it only occasionally. Frequent users will become experts whereas infrequent users may remain relative novices. It is important to classify the users and estimate the likely numbers in each category.

2.5 Operational Environment

This section should describe the real world the software is to operate in. Context diagrams may support this narrative description, to summarise external interfaces and system block diagrams, to show how the activity fits within the larger system. The nature of the exchanges with external systems should be specified.

If a system specification defines a product that is a component of a parent system or project, then this section should outline the activities that will be supported by external systems. References to the interface control documents that define the external interfaces with the other systems will be provided. The computer infrastructure to be used will be also described.

2.6 Assumptions and Dependencies

This section should list the assumptions that the specific requirements are based on. Risk analysis should be used to identify assumptions that may not prove to be valid.
A constraint requirement, for example, might specify an interface with a system that does not exist. If the production of the system does not occur when expected, this system specification may have to change.

3 SPECIFIC REQUIREMENTS

This chapter should define the system requirements that are conditions for the validation against the requirements baseline.

Specific requirements should be described in this section, which is the core of this specification. Each requirement must be uniquely identified. Each requirement shall be traced to the applicable higher level documentation.

Requirements maybe further classified as “essential” or “not essential”. The following provides guidelines on requirements “flags”. Essential requirements have to be met for the software to be acceptable. If a requirement is essential, it must be clearly flagged. Non-essential requirements should be marked with a measure of desirability (e.g. scale of 1, 2, and 3).
Some user requirements may be `suspended' pending resources becoming available. Such non-applicable user requirements must be clearly flagged.

The priority of a requirement measures the order, or the timing, of the related software becoming available. If the transfer is to be phased, so that some parts of the software come into operation before others, then each requirement must be marked with a measure of priority.

Unstable requirements should be flagged. The usual method for flagging unstable requirements is to attach the marker 'TBC' (to be confirmed).
The source of each user requirement must be stated and formally traced. The source may be defined using the identifier of a system requirement, or any other documentation considered applicable by the Project. Backward traceability depends upon each requirement explicitly referencing its source.

Each user requirement must be verifiable. Clarity increases verifiability. Each statement of user requirement should contain one and only one requirement. A user requirement is verifiable if some method can be devised for objectively demonstrating that the software implements it. For example statements such as:

"The software will work well";
"The product shall be user friendly";
"The output of the program shall usually be given within 10 seconds";

are not verifiable because the terms 'well', 'user friendly' and 'usually' have no objective interpretation.

A statement such as: '*the output of the program shall be given within 20 s of event X, 60% of the time; and shall be given within 30 s of event X, 99% of the time*', is verifiable because it refers to measurable quantities. If a method cannot be devised to verify a requirement, the requirement is invalid.

The user must describe the consequences of losses of availability and breaches of security, so that the developers can fully appreciate the criticality of each function.

3.1 Capability Requirements
This section shall list the requirements specifying the required system behaviour and its associated performances. The organisation of the capability requirements should reflect the problem, and no single structure will be suitable for all cases.
The capability requirements can be structured around a processing sequence, for example:

(a) RECEPTION OF IMAGE
(b) PROCESSING OF IMAGE
(c) DISPLAY OF IMAGE

perhaps followed by deviations from the baseline operation:

(d) HANDLING LOW QUALITY IMAGES

Each capability requirement should be checked to see whether the inclusion of capacity, speed and accuracy attributes is appropriate.

3.2 System Interface Requirements
This section shall specify any interface requirements imposed on the system. Requirements related to:
- Communication interfaces
- Hardware interfaces
- Software interfaces

- HCI interactions.

Should be specified in this section or a reference to the IRD can be made.

3.3 Adaptation Requirements

This section shall list the data that may vary according to operational needs and any site-dependent data.

3.4 Computer Resource Requirements

This section shall specify the requirements on the computer resources

3.4.1 Computer Hardware Requirements

This section shall list the requirements on the computer hardware to be used

3.4.2 Computer Hardware Resources Utilisation Requirements

This section shall specify requirements on the computer hardware resource utilisation (processor capacity, memory capacity….) available for the software item (e.g. sizing and timing)

3.4.3 Computer Software Requirements

This section shall specify requirements on the software items, which have to be used by or incorporated into the system (or constituent software product).

3.5 Safety Requirements

This section should specify the safety requirements applicable to the system

3.6 Reliability Requirements

This section should specify the reliability requirements applicable to the system

3.7 Quality Requirements

This section shall specify the requirements relevant to the system quality factors (e.g. usability, reusability, and portability)

3.8 Design Requirements and Constraints

This section should include the requirements which constraint the design and production of the system. For example the following kind of requirements shall be included:

- Requirements on the software architecture.
- Utilisation of standards.
- Utilisation of existing components.
- Utilisation of Agency (or customer) furnished components or COTS components.
- Utilisation of design standard.
- Utilisation of data standards.
- Utilisation of a specific programming language.
- Utilisation of a specific naming convention.
- Flexibility and expansion.

4. VERIFICATION AND VALIDATION REQUIREMENTS

This section shall define verification and validation methods to be used to ensure the requirements have been met.

For each of the identified requirements in section 3, a verification method shall be specified. If verification of a given requirement is not requested for the software validation against the requirements baseline, then it should be clearly stated.

A verification matrix (requirements to verification method correlation table) can be utilised to define the verification/validation methods applicable to each requirement. This section can be further split into a section 4.1 for verification and validation requirements and a section 4.2 for acceptance requirements, if necessary, depending upon the project needs.

This page intentionally left blank

3 Software Development Plan

TABLE OF CONTENTS

1. INTRODUCTION
This chapter should give an introduction to this plan, providing scope, purpose, glossary and documentation references and also describing the document organisation. This chapter shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this plan and specify the intended readership of the plan.

1.2 Scope
This section should:
- Identify the software products to which this plan applies
- Define the relationships with other plans.

1.3 Project Overview
This section of the plan should provide a summary of the Project, specifying the following:
- Objectives
- Deliverables
- Life Cycle Approach
- Major Activities
- Milestones
- Resource Requirements
- Schedule
- Budget.

1.4 Project Deliverables
This section should list the project deliverables. Documentation and software releases to be delivered have to be listed. Any other required deliverable items, such as prototypes or tools should be included.

1.5 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.5.1 Acronyms
1.5.2 Definition of terms.

1.6 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.5.1 Applicable documents
1.5.2 Reference documents.

1.7 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. PROJECT ORGANISATION

2.1 Software Process Model

This section should define the activities to be performed in the project and their inputs and outputs. The definition should include the major project functions (i.e. activities that span the entire duration of the project, such as project management, configuration management, verification and validation, and quality assurance) and the major production activities needed to achieve the project objectives. The definition of the process model may be textual or graphic. The Process model shall be based on the ECSS SW standards and any related tailoring shall be justified and documented.

2.2 Organisational Structure

This section should describe the organisational structure to be used in the project. Lines of reporting, control and communication should be described using organigrams. As an example, roles that often appear in the software project organisational structure are:

- Project manager
- Software engineers
- Software librarian
- Software product assurance engineer.

(see ECSS-E-40; 5.9.2.3) and (ECSS-E-40; 5.9.3.3).

2.3 Organisational Boundaries and Interfaces

This section should describe the relationship between all the involved parties and the responsibility of each organisation/team, according to the established customer/supplier network. The following organisations/roles should be addressed to the extent applicable to the project:

- Customer organisation
- Client organisation
- End users
- Subcontractors
- Suppliers
- Independent verification and validation organisation
- Independent product assurance organisations.

The procedures and responsibilities for the control of each external interface should be summarised. For example:

- Name of Interface Control Document (ICD)
- Those responsible for the agreement of the ICD
- Those responsible for authorising the ICD.

2.4 Project Responsibilities

This section should define the roles identified in the organisational structure and its boundaries. For each role, the plan should briefly describe the purpose of the role and list the responsibilities.

3. SOFTWARE MANAGEMENT APPROACH

3.1 Management Objectives and Priorities

This section should define the management objectives of the software project and associated priorities. Objectives shall be discussed and trade-offs for their prioritisation documented.

3.2 Assumptions, Dependencies and Constraints

This section should state:

- The assumptions on which the plan is based
- The external events the project is dependent upon
- Constraints on the project.

Technical issues should only be mentioned if they have an effect on the plan.
Assumptions, dependencies and constraints are often difficult to distinguish. The best approach is not to categorise them but to list them. For example:

- Limitations on the budget
- Schedule constraints (e.g. launch dates, delivery dates)
- Constraints on the location of staff (e.g. they must work at developer's premises)
- Commercial hardware or software that will be used by the system
- Availability of simulators and other test devices
- Availability of external systems with which the system must interface.

3.3 Risk Management

This section should:

- Describe the approach to be followed for risk management, considering factors like stringent schedule constraints, highly changeable product specifications and project size. Guidelines specified in ECSS-M-00 and in ECSS-Q-20, clause 2.8 should be utilised.
- Identify and assess the risks to the specific project, and describe the actions that will be taken in this phase to manage them. A risk table may be used.

3.4 Monitoring and Controlling Mechanisms

This section of the plan should define the monitoring and controlling mechanisms for managing the work.
Possible monitoring and controlling mechanisms are:

- Work package descriptions
- Work package completion reports
- Progress reports
- Reviews
- Audits.

This section should define or reference the formats for all documents and forms used for monitoring and controlling the project.

This section should specify the:

- Frequency of progress meeting with initiators and management
- Frequency of submission of progress reports
- Tailoring (if any) to the software progress report DRD
- General policy regarding reviews and audits (the details will be in the software verification plan).

3.5 Staffing Plan

This section of the plan should specify the names, roles, skills and grades of staff that will be involved in the project. A staff profile giving the total number of staff on the project each month may also be given.

3.6 Supplier Management

This section should describe the approach to monitor and control the supplier activities.

3.7 Software Acquisition Approach

This section should define the utilised software acquisition process

4. TECHNICAL PROCESS

4.1 Software Development Life Cycle

This chapter specifies the applied software development life cycle for the project. Relationships with the system development life cycle are to be specified, definition of milestones and associated documentation to be produced.

4.1.1 Software Development Life Cycle identification

The definition of the software development life cycle shall include both the definition of the selected life cycle paradigm (e.g. waterfall, incremental, evolutive…) as well as the adopted software versioning approach.
This section shall cover the approach to be followed for implementation of all the activities/tasks relevant to the software processes:

- Software Requirements Engineering Process
- Software Design Engineering Process
- Software Verification and Validation Process
- Software Operations Process
- Software Maintenance Process.

4.1.2 Relationship with the System Development Life Cycle

This section shall describe the correspondence between the software life cycle and the system development life cycle. A GANNT representation may be used.

4.1.3 Reviews Identification and Associated Documentation

This section shall describe the scope and purpose of each review as well as the relevant deliverables and expected outputs.

4.2 Software Engineering Standards and Techniques

This section shall specify the applied methodologies and define the standards for each software process and relevant activity. Selected methodologies and standards have to be detailed here. Depending on the project size/complexity, a software engineering standard and techniques document might have to be produced; in this case simply a reference to this document has to be provided in this section.

The following items should be covered, as applicable to the project:

- Software requirements analysis methodology

  The requirements analysis method used should be defined and referenced. Reference to applied literature or other standards can be described here. Any tailoring of the requirements analysis methodology should be defined and justified.

- Design methodology and standards

  The selected design (architectural design and detailed design) methods should be defined and referenced. Reference to applied literature or other standards can be described here. Any tailoring of the method (e.g. deviations, extensions, and avoidance of utilisation of some methodology features…) should be defined and justified.

- Coding standards

This section should specify any specific coding standards, as well as any selected (if applicable) coding documentation standards. For example standard for module's headers and their completion instructions should be specified.

- Naming conventions
  This section should define all necessary and required naming conventions. Naming conventions shall be specified for files, programs, modules and data/commands, as necessary.

- Programming languages standard
  This section should specify the selected programming language(s) and relevant standard(s) for the software project.
  Applicable documentation recommended or imposed style guides have to be specified here, as well as any other relevant requirement.

- Verification/Validation methodology and standards

This section should specify any required method to conduct verification and validation testing activities. For example, the definition of any test procedure language for software validation has to be specified here.

- Data base management system standards
  This section should specify any required method to develop any required data base and the selected data base management system(s).

- Reusable software approach

This section should specify the adopted method both for software to be re-used and for software design for re-use, as applicable to the project.

- Handling of critical software

This section should define the applied approach to manage critical software (e.g. utilisation of specific design tools to design critical software,...)

- Utilisation of man-machine interfaces generators

This section should define any standard to be applied to the software development, if man-machine Interface code generators are utilised.

- Utilisation of code generators

This section should define any standard to be applied to the software development, if code generators are utilised (e.g. constraints to be imposed on the use of generators in terms of allowed specific features…)

- Software delivery format

This section should specify the selected software delivery format.

4.3 Software Development and Software Testing Environment

The software development environment and testing environment shall be defined and described. Hardware platforms and selected software tools to be utilised for the software development and testing shall be defined and described, with justification of their selection, w.r.t. the relevant software methodology and standards.
ECSS-Q-80; 5.7.3.2 and ECSS-Q-80, 6.1, 6.2, 6.3.3.1 and 6.3.3.4

4.3.1 Software Development Environment
4.3.2 Software Testing Environment

4.3.3 Software Libraries

4.4 Software Documentation Plan

This section should define all documentation relevant to the project and the documentation standards applied to the software project. The section should be then structured as follows:

4.4.1 Software Documentation Identification

This section should, for each document to be produced (both internal documents and deliverables), include the documentation plan specifying:

- Document name
- Delivery requirements
- Review requirements
- Approval requirements.

4.4.2 Software Documentation Standards

This section should specify the documentation standards applicable to the project. Any tailoring to applicable documentation standards shall be specified in this section. Therefore, depending on the specific project documentation requirements, this section should either refer to applicable documentation standards for format and layout (for example the level 2 and 3 ECSS software standards) or specify the proposed documentation DRD.

4.5 Project Support Functions

This section should contain an overview of the plans for the project support functions for:

- Software configuration management
- Software verification and validation
- Software product assurance.

This section will also provide references to the Software configuration management plan, software verification plan, software validation plan, and software product assurance plan.

5. WORK PACKAGES, SCHEDULE, AND BUDGETS

5.1 Work Packages

This section should describe the breakdown of the project activities into work packages.

This section may begin with a Work Breakdown Structure (WBS) diagram to describe the hierarchical relationships between the work packages. Each box should show the title and identifier of the work package. Alternatively, the work breakdown may be described by listing the work package titles and identifiers.

The full work package descriptions may be contained in this section or put in an appendix. Each work package description should define the:

- Work package title
- Work package reference number
- Responsible organisation
- Major constituent activity
- Work package manager
- Start event
- End event
- Inputs
- Activities
- Outputs.

Work packages should be defined for all activities, including project functions such as project management, configuration management, verification and validation and quality assurance.

5.2 Dependencies

This section should define the ordering relations between the work packages, using for instance a planning network technique, such as the Program Evaluation and Review Technique (PERT), to order the execution of work packages according to their dependencies. Dependency analysis should define the critical path to completion of the project and derive the 'float' for activities off the critical path.

5.3 Resource Requirements

This section should describe, for each work package:

- The total resource requirements
- The resource requirements as a function of time.

Labour resources should be evaluated in man-hours, man-days or man-months. Other resources should be identified (e.g. equipment).

5.4 Budget and Resource Allocation

The project budget allocation should be specified by showing how the available financial resources will be deployed on the project. This is normally done by providing a table listing the amount of money budgeted for each major work package. The manner in which this section is completed depends upon the contractual relationship.

5.5 Schedule

This section should define when each work package starts and ends. This is normally done by drawing a Gantt chart.
This section should describe the milestones in the project, providing for each milestone:

- An identifier
- A description (e.g. a list of deliverables)
- The planned date of achievement
- The actual date of achievement (for plan updates).

Milestones should be marked on or alongside the Gantt chart.

This page intentionally left blank

4 Software Verification Plan

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software verification plan:
- Identify the software project (and the actual software products) for which this software verification plan is written
- Identify the software products (documentation and code) to be verified and the documentation against which these will be verified
- Outline the goals of the verification activities
- Specify the intended readers of this software verification plan.

1.2 Scope
This section should:
- explain what the addressed verification activities will guarantee
- describe the relevant benefits, objectives and goals as precisely as possible

1.3 Glossary
This section should define all terms, acronyms and abbreviations used in this plan, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. VERIFICATION PROCESS OVERVIEW
This section shall describe the approach to be utilised to implement the verification process throughout the software life cycle, the required verification effort, and the level of required independence for the verification tasks.

This section should be structured as follows:

2.1 Organisation
This section should describe the organisation of the documentation review, proof and tracing activities. Topics that should be included are:

- Roles
- Reporting channels
- Levels of authority for solving problems
- Relationships to other activities such as project management, development, configuration management and product assurance
- Level of required and implemented independence.

2.2 Master Schedule
This section should identify the schedule for the planned verification activities. Information about reviews shall be also included.

2.3 Resources Summary
This section should summarise the resources needed to perform the verification activities such as staff, hardware and software tools.

2.4 Responsibilities
This section should define the specific responsibilities associated with the roles described in section 2.1.

2.5 **Tools, Techniques and Methods**
This section should identify the software tools, techniques and methods used for documentation and code reviews, walk-through, inspections, proofs and tracing.

2.6 **Personnel and Personnel Training Requirements**
This section should specify any requirement for verification personnel and any necessary training needs.

3. VERIFICATION ADMINISTRATIVE PROCEDURES

3.1 Problem Reporting and Resolution

This section should explain any method utilised to document and report problems found in the software and its documentation. This should clearly identify the problem reporting and resolution process utilised for the Software Project. The following list provides examples to be considered:

- External formal events (e.g. for formal reviews to supplier);
- Verification activities internal to the organisation (e.g. is the software requirements verification report utilised to improve the software requirements specification or are additional forms utilised by the Supplier to internally track and implement detected problems?)
- Internal Formal events (e.g. which are the procedures for tracking detected problems during internal inspections/walk-through?).

3.2 Task Iteration Policy
This section should define the criteria applied in order to decide whether a verification activity should be repeated when a change is approved for implementation. The criteria may include assessment of the change objective, the criticality of the function (affected), and any quality effect.

3.3 Deviation Policy

This section should define the procedures for deviating from the plan, and define the levels of authorisation required for approval of deviations. The information required for deviations should include task identification, deviation rationale and the effect on software quality.

3.4 Control Procedures

This section should identify the configuration management procedures of the products of review, proofs and tracing.

3.5 Standards, Practices and Conventions

This section should identify the standards, practices and conventions that govern the verification activities (e.g. documentation and code review, tracing, proof…). Reference to other documentation either project specific or internal to the responsible organisation could be made.

4 VERIFICATION ACTIVITIES

For each of the tasks pertaining to the verification process, this section shall describe which methods, tools and facilities are utilised and which outputs have to be produced. This includes the identification of internal reviews, walk-through, inspections etc. When the need for utilisation of specific methods and tools is identified (e.g. formal proof) it shall be specified

This section shall be structured in the following sub-sections and each one should contain the above specified information per each software product under verification.

4.1 Software Requirements Engineering Process Verification

4.1.1 Activities

This section should list the verification activities to be performed for this software process and how these are accomplished

4.1.2 Inputs

This section should list the required inputs (e.g. draft SRS; draft top level architectural design) to accomplish the verification activities for this software process.

4.1.3 Outputs

This section should list the intermediate and final outputs (e.g. software verification requirements report, top level architectural design to requirements traceability) documenting the verification activities for this software process.

4.1.4 Methodology, Tools and Facilities

This section should identify the methodologies, tools and facilities utilised to accomplish the verification activities. Reference to section 2.5 could be made.

4.2 Software Design Engineering Process Verification

4.2.1 Activities

This section should list the verification activities to be performed for this software process and how these are accomplished

4.2.2 Inputs

This section should list the required inputs (e.g. code, operations manual, software Integration plan) to accomplish the verification activities for this software process.

4.2.3 Outputs

This section should list the intermediate and final outputs (e.g. software code verification report, evaluation of software validation testing specification) documenting the verification activities for this software process.

4.2.4 Methodology, Tools and Facilities

This section should identify the methodologies, tools and facilities utilised to accomplish the verification activities. Reference to section 2.5 could be made.

5 SOFTWARE VERIFICATION REPORTING

This section should describe how the results of implementing the plan will be documented. Reports produced by the implementation of the verification process shall be identified. The reports identified in the ECSS-E-40 have then to be referenced, as implemented by the specific software project for which this plan is written as well as any problem report mechanism adopted (e.g. utilisation of RID form or problem report form). The forwarding procedure of this documentation to the customer shall be also addressed.

5 Software Validation Plan

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software validation plan and specify its intended readership.

1.2 Scope
This section should:
- Identify the software products under test
- Summarises the software validation testing campaign
- Specify the documentation driving the validation campaign
- Define the relationships to other plans.

In addition, this section shall clearly specify whether the Validation activities, mentioned in this plan, have to be performed against the technical specification as well as against the requirements baseline. The template is defined for use in both cases. In case the software supplier must conduct both validation campaigns (i.e. when the customer is not performing the validation against the requirements baseline) this plan will be structured to reflect the commitment. Therefore it will be divided in two parts, following from Chapter 2 onwards the same structure given in this DRD.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. VALIDATION PROCESS PLANNING

This section shall describe the approach to be utilised to implement the validation process, the required effort, and the level of required independence for the validation tasks. This section shall also clearly address if it is applicable to the software validation campaign against the requirements baseline, to the software validation campaign against the technical specification or to both.

2.1 Organisation

This section should describe the organisation of validation activities. Topics that should be included are:

- Roles
- Reporting channels
- Levels of authority for solving problems
- Relationships to other activities such as project management, development, configuration management and product assurance
- Level of required and implemented independence

2.2 Master Schedule

This section should identify the schedule for the planned validation activities. In particular test milestones identified in the software project schedule and all item delivery events. This section should specify:

- Any additional test milestones
- The time required for each testing task
- The schedule for each testing task and test milestone
- The period of use for the Test facilities.

3.7 Resources Summary

This section should summarise the resources needed to perform the validation activities such as staff, hardware and software tools.

3.8 Responsibilities

This section should define the specific responsibilities associated with the roles described in section 2.1. In particular this section should identify the groups responsible for managing, designing, preparing, executing, witnessing and checking tests. Groups may include developers, operational staff, user representatives, technical support staff, and product assurance staff.

3.9 Tools, Techniques and Methods

This section should identify the software tools, techniques and methods used for validation activities.

3.10 Personnel and Personnel Training Requirements

This section should specify any requirement for software validation personnel (level of independence) and any necessary training needs.

3.11 Risks and contingencies

This section should identify risks to the software validation campaign. Contingency plans should be specified.

3. SOFTWARE VALIDATION TASKS IDENTIFICATION

This section shall describe the software validation tasks to be performed for the identified software item.

This section shall specify which are the tasks and the items under tests, as well as the criteria to be utilised. The section shall be structured as follows:

3.1 Software Products Under Test
This section should identify the software products under test, providing a summary of their functions. Reference to their documentation can be done.

3.2 Features to Be Tested
This section shall identify the features to be tested, making references to the applicable documentation (e.g. sections of the TS, section of the RB as applicable)

3.3 Features not to Be Tested
This section should identify all the features and significant combinations that will not be tested.

3.4 Test Pass/Fail Criteria
This section shall specify the criteria to be used to determine whether or not tests are passed.

3.5 Suspension Criteria and Resumption Requirements
This section should specify the criteria used to suspend all, or a part of, the testing activities on the test items associated with the plan. The section should specify the testing activities that have to be repeated when testing is resumed.

3.6 Deliverable Items
This section should identify the items to be delivered. This section should clearly address deliverable items internal to the software validation organisation (what, when and how) and deliverable items to the Customer (what, when and how), in accordance with the established software project development plan.
The following list provide an example of deliverable items:

- Software validation testing specification (including test cases, test design, test procedures)
- Software validation testing report.

4. SOFTWARE VALIDATION TESTING APPROACH
This section shall identify the overall requirements applicable to the software validation testing activities, providing definition of overall requirements, guidelines on the kinds of tests to be executed. The following will guide the definition of the test cases/test procedures defined in the software validation testing specification for the validation tasks to be executed to verify the technical specification or the requirements baseline, as applicable. The following kind of information shall be specified:

- Test level
- Test classes
- General test conditions
- Data recording, reduction and analysis.

5. ANALYSIS AND INSPECTION APPROACH
This section shall specify the selected approach to accomplish validation of those software specification requirements, which need to be verified by inspection and analysis.

6. SOFTWARE VALIDATION TESTING FACILITY
This section shall describe the test environment needed to execute the software validation testing activity, whose approach is defined by this plan.
This section shall describe the configuration of the selected validation facility in terms of software (e.g. tools and programs, simulation), hardware (e.g. platforms, target computer target), test equipment (e.g. bus analyser, etc) and communication networks. Reference to other documentation describing the facility can be done. If the validation testing against the

RB and the validation testing against the TS will use different environments, this has to be clearly stated and specified.

7. SOFTWARE VALIDATION TESTING PROCEDURES

7.1 Problem Reporting and Resolution

This section should explain method utilised to document and report problems found in the software during validation testing. This should clearly identify the problem reporting and resolution process utilised for the software under test and its support software. Definition of the software control board and forwarding procedures to the customer should also be addressed.

7.2 Deviation Policy

This section should define the procedures for deviating from the plan, and define the levels of authorisation required for approval of deviations. The information required for deviations should include task identification, deviation rationale and the effect on software quality.

7.3 Control Procedures

This section should identify the configuration management procedures, specifying the level of configuration control to be implemented on the software under test and its test support software.

6 Software Interface Control Document

TABLE OF CONTENTS

1. INTRODUCTION
This chapter should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This chapter shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software interface control document and specify its intended readership.

1.2 Scope
This section should :
- Identify the software products to which this ICD applies
- Identify the relationship of this document w.r.t. other documents specifying the software products.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:
1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:
1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3. INTERFACES DESIGN
This section shall define the design of the external interfaces (software, hardware...) of the software product.
Representation of external interfaces shall be made utilising an interface diagram

3.1 Interface Identification

This section shall identify the external interfaces to the software item. Interface diagrams can be used to identify the external interfaces of the software item.

3.x. [name] Interface description
For each identified interface this section shall be organised as follows:

3.x.1 Data Elements
For each data element the following information shall be provided:
- Name
- Unique identifier
- Description
- Source item
- Destination item
- Unit of measure
- Limit/range
- Accuracy
- Precision
- Frequency
- Rate
- Legality checks
- Data type
- Data representation
- Priority

3.x.2 Message Description
This section shall describe the messages transmitted across the interface by name and unique identifier and shall describe the assignment of data elements to each message.

3.x.3 Communication Protocol
This section shall describe the protocol applicable to the interface or make reference to applicable documentation.

7 Software Requirements Specification

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software requirements specification and specify its intended readership.

1.2 Scope
This section should:
- Identify the software products to be produced by name and allocated CI number
- Describe the main capabilities provided by the software product.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:
1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:
1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 GENERAL DESCRIPTION
This section should describe the general factors that affect the product and its requirements. It does not state specific requirements; it only makes those requirements easier to understand.

2.1 Relationship to Current Projects
This section should describe the context of the project in relation to the established customer/supplier relationships. For instance, the project may be independent of other projects or part of a larger project. Any parent projects should be identified. A detailed description of the interface of the product to the larger system should, however, be deferred until Section 2.5.

2.2 Relationship to Predecessor and Successor Projects
This section should describe the context of the project in relation to past and future projects.

This section should identify projects that the developer has carried out for the initiator in the past and also any projects the developer may be expected to carry out in the future, if known, within the established customer/supplier chain.
If the product is to replace an existing product, the reasons for replacing that system should be summarised in this section.

2.3 Function and Purpose
This section should discuss the purpose of the product. It should expand upon the points made in Section 1.2.

2.4 Environmental Considerations
This section should summarise:

- The physical environment of the target system, i.e. where is the system going to be used and by whom.
- The hardware environment in the target system, i.e. what computer(s) does the software have to run on?
- The operating environment in the target system, i.e. what operating systems are used?
- The hardware environment in the development system, i.e. what computer(s) does the software have to be developed on?
- The operating environment in the development system, i.e. what software development environment is to be used? Or what software tools are to be used?

2.5 Relation to Other Systems
This section should describe in detail the product's relationship to other systems, for example is the product:

- An independent system?
- A component of a larger system?
- Replacing another system?

If the product is for instance a component of an integrated HW-SW Product, then this section should:

- Summarise the essential characteristics of this larger product.
- Identify the other HW or SW components the software will interface with
- Summarise the computer hardware and peripheral equipment to be used.

A block diagram may be presented showing the major components of the larger system or project, interconnections, and external interfaces.

2.6 General Constraints
This section should describe any items that will limit the developer's options for building the software. It should provide background information and seek to justify the constraints.

2.7 Model Description
This section should include a top-down description of the logical model of the software. Diagrams, tables and explanatory text may be included.
The functionality at each level should be described, to enable the reader to `walkthrough' the model level-by-level, function-by-function, and flow-by-flow. A bare-bones description of the software product in terms of data flow diagrams and low-level functional specifications needs supplementary information. Natural language is recommended.

3. SPECIFIC REQUIREMENTS
The software requirements are detailed in this chapter. Each requirement must include an identifier. Essential requirements must be marked as such. For incremental delivery, each software requirement must include a measure of priority so that the developer can decide the production schedule, unless this is not specified by (or with) the Customer. References that trace the software requirements back to the higher level specification must accompany each

software requirement. Any other sources should be stated. Each software requirement must be verifiable. If no requirements are identified for a section, then this section shall explicitly report the “None” keyword.

The functional requirements should be structured top-down in this chapter. Non-functional requirements can appear at all levels of the hierarchy of functions, and, by the inheritance principle, apply to all the functional requirements below them. Non-functional requirements may be attached to functional requirements by cross-references or by physically grouping them together in the document.
If a non-functional requirement appears at a lower level, it supersedes any requirement of that type that appears at a higher level. Critical functions, for example, may have more stringent reliability and safety requirements than those of non-critical functions.
Specific requirements may be written in natural language. This makes them understandable to non-specialists, but permits inconsistencies, ambiguity and imprecision to arise. These undesirable properties can be avoided by using requirements specification languages (e.g., Z, VDM, and LOTOS), according to established policies in the SW development plan.

Each software requirement must have a unique identifier. Forward traceability to subsequent activities in the life cycle depends upon each requirement having a unique identifier.
Essential software requirements have to be met for the software to be acceptable. If a software requirement is essential, it must be clearly flagged. Non-essential software requirements should be marked with a measure of desirability (e.g. scale of 1, 2, and 3).

The priority of a requirement measures the order, or the timing, of the related functionality becoming available. If the transfer is to be phased, so that some parts come into operation before others, each requirement must be marked with a measure of priority.

Unstable requirements should be flagged. The usual method for flagging unstable requirements is to attach the marker ‘TBC'.
The source of each software requirement must be stated using the identifier of requirements of the higher level specification.
Each software requirement must be verifiable. Clarity increases verifiability. Ensuring that each software requirement is well separated from the others enhances clarity. A software requirement is verifiable if some method can be devised in order to demonstrate objectively that the requirement is correctly implemented.

The chapter shall be structured as follows:

3.1 Functional Requirements
(ECSS-E-40; 5.4.2.1 (a) and ECSS-Q-80; 6.3.1.3)
This section specifies the functionality to be provided by the software product under definition. Each functional requirement shall be uniquely identified and grouped by subject. It is recommend that each requirement definition is organised in accordance with the following:
- General description
- Inputs
- Outputs
- Processing.

3.2 Performance Requirements
(ECSS-E-40; 5.4.2.1 (a) and ECSS-Q-80; 6.3.1.3)
This section specifies any specific requirement relevant to the desired performance of the software product under definition.

3.3 Interface Requirements
(ECSS-E-40; 5.4.2.1(b) , 5.4.2.1 (f) and 6.5.4)

The software item external interfaces shall be identified and specified in this chapter.
The following interfaces shall be fully specifi ed either in this section or by reference to another document (e.g. ICD):

- Interfaces between the software item and other software items.
- Interfaces between the software item and hardware products.
- Interface requirements relating to the man/machine interaction (as applicable)

3.4 Operational Requirements

This section specifies any specific requirement relevant to the operation of the software in its intended environment. This should include for instance any specified operational mode and mode transition for the software, and in case of Man-Machine Interaction, the intended use scenario(s).

Diagrams should be used as necessary, to show the intended operations and related modes/transitions.

3.5 Resources Requirements

This section shall specify all the resource requirements relevant to the software and shall specify also the hardware requirements (target hardware on which the software is required to operate)

3.5.1 Computer Hardware Requirements

This section shall list requirements relevant to the hardware environment in which the software is required to operate (ECSS-E-40; 5.4.2.1(a)).

3.5.1 Computer Hardware Resources Utilisation

This section shall list the sizing and timing requirements applicable to the software product under specification (Response to ECSS-E-40; 5.3.4.1).

3.5.2 Computer Software Requirements

This section shall specify which computer software shall be used with the software under specification or incorporated into the software product (e.g. OS, software items to be re-used…).

3.6 Design and Implementation Constraints

This section shall list any requirements driving the design of the software item under specification and any identified implementation constraint. Requirements relevant to the following items should be included, as applicable:

- Software standards (e. g. applicable coding standards, (ECSS-Q-80; 6.3.3.4))
- Design requirements
- Specific design methods to be applied to minimise the number of critical software components (ECSS-Q-80; 6.2.2.6)
- Requirements relevant to numerical accuracy management (e.g. for AOCS products) (ECSS-Q-80; 7.1.11)
- Design requirements relevant to the "in-flight modification" of the software product (ECSS-E-40;5.8.5.2)
- Specific design requirements to be applied if the software product has to be designed for reuse. ECSS-E-40; 6.4.1.2.

3.7 Security and Privacy Requirements

This section shall specify any security and privacy requirement applicable to the software product. (ECSS-E-40; 5.4.2.1(e))

3.8 Portability Requirements

This section shall specify any portability requirement applicable to the software product under specification.

3.9 Software Quality Requirements
This section shall specify any quality requirement applicable to the software product. (ECSS-Q-80; 3.3.1 (d) and ECSS-Q-80; 4.2.1 (a)).

3.10 Software Reliability Requirements
This section shall specify any reliability requirement applicable to the software product. (ECSS-Q-80; 3.3.1 (d))

3.11 Software Maintainability Requirements
This section shall specify any maintainability requirement applicable to the software item. (ECSS-Q-80; 3.3.1 (d))

3.12 Safety Requirements
This section shall specify any safety Requirement applicable to the software. (ECSS-E-40; 5.4.2.1(d) and ECSS-Q-80; 3.3.1 (d))

3.13 Software Configuration and Delivery Requirements
This section shall specify any requirement, relevant to the selected delivery medium and any software configuration applicable to the software item. (ECSS-Q-80; 3.2.3 (j) and ECSS-Q-80; 3.3.1 (d))

3.14 Personnel-related Requirements
This section shall specify any requirement relevant to the software, regarding personnel, as applicable to the specific software product under definition. This section should contain (as applicable) requirements relevant to: manual operations, human-equipment interactions, constraints on personnel, areas needing concentrated human attention and that are sensitive to human errors and training as well as relevant to human factors engineering requirements (ECSS-E-40; 5.4.2.1(f))

3.15 Adaptation and Installation Requirements
This section shall specify any requirement relevant to the adaptation of data as well as specific installation requirements. The section can be structured as follows:

3.15.1 Adaptation Requirements
3.15.2 Installation Requirements
(ECSS-E-40; 5.4.2.1(h))

3.16 Others Requirements
This section contains any additional requirements not covered in the previous sections.

4 VERIFICATION, VALIDATION AND ACCEPTANCE REQUIREMENTS

4.1 Verification and Validation Requirements
This section shall define verification and validation methods to ensure that the requirements have been met.
For each requirement uniquely identified in Section 3, the method of verification/validation shall be specified.
Verification methods can be, for example, test, analysis, inspection, review of documentation, demonstration. (ECSS-Q-80 3.3.1(d) and ECSS-Q-80; 4.2.1 (d)).

A verification matrix (requirements to verification methods correlation table) can be utilised to define the verification/validation methods applicable to each requirement.

4.2 Acceptance Requirements

This section shall identify the set of requirements subject to acceptance testing and the associated verification method. (ECSS-E-40; 5.4.2.1(h)).

A matrix (requirements to verification methods correlation table) can be utilised to define the verification/validation methods applicable to each requirement, which will be subject to acceptance testing.

8 Software User Manual

TABLE OF CONTENTS

1. INTRODUCTION
This chapter should provide an introduction to this document, providing intended readership, purpose, glossary and documentation references and also describing the document organisation. This chapter shall be organised in the following subsections:

1.1 Intended Readership
This section should define the categories of user (e.g., end user and operator) and for each category:

- define the level of experience assumed
- state which sections of the user manual are most relevant to their needs

1.2 Applicability Statement
This section should define the software releases to which the current issue of the user manual refers.

1.3 Purpose
This section should define both the purpose of this user manual and the purpose of the software. It should name the process to be supported by software and the role of the user manual in supporting that process.

1.4 Related Documents
This section should list related documents and define the relationship of each document to the others. All document trees to which the user manual belongs should be defined in this section. If the user manual is a multi-volume set, each volume of the set should be separately identified.

1.5 Conventions
These sections should summarise symbols, stylistics conventions, and command syntax conventions used in the document.
An example of stylistic conventions is using boldface and Courier font to distinguish user input. Examples of syntax conventions are the rules for combining commands, keywords and parameters, or how to use a WIMP interface.

Problem Reporting Instructions

This section should summarise the procedures for reporting software problems.

1.7 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.7.1 Acronyms
1.7.2 Definition of Terms

1.8 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or

reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.8.1 Applicable Documents
1.8.2 Reference Documents

1.9 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised. In particular the intended use of each section and the relationship between sections should be addressed.

2 OVERVIEW SECTION

This chapter should give an overview of the software to all users and summarises:

- The process to be supported by the software
- The fundamental principles of the process
- What the software does to support the process
- What the user needs to supply to the software
- The required hardware environment.

The software functionality should be described from an external ‘black box' point of view. The discussion should be limited to functions, inputs and outputs that the user sees.
The overview can often become much clearer if a good metaphor is used for the system. Some GUIs, for example, use the ‘office system' metaphor of desk, filing cabinets, folders, indexes etc. Often a metaphor will have been defined very early in the system development, to help capture the requirements. As another example, for real-time software, the control systems metaphor may provide an effective approach.

3 INSTRUCTION SECTION

This chapter should aim to teach new users how to use the software. The viewpoint is that of the trainee.
The chapter should begin by taking the user through short and simple sessions. Each session should have a single purpose, such as ‘to enable the user to edit a data description record' or to use a specific software capability. The sessions should be designed to accompany actual use of the system, and should contain explicit references to the user's actions and the system's behaviour. The trainee should derive a sense of achievement by controlling a practical session that achieves a result.
Diagrams, plans, tables, drawings, and other illustrations should be used to show what the software is doing, and to enable the novice to form a clear and accurate mental model of the system.
The tutorial may be structured in any convenient style. It is wise to divide it into sessions lasting 30-40 minutes. Longer sessions are difficult to absorb.
A session could have the goal of introducing the user to a set of ‘advanced' facilities in a system.
It may not be practical or desirable to present a whole session. It may be more appropriate to give a ‘tutorial tour'. Even so, it is still helpful to fit all the examples into a single framework.
For each session, this chapter should provide:

- Functional description
  A description of what the session is supposed to achieve, in the user's terms.
- Cautions and warnings
  A list of precautions that the user may need to take; the user is not concerned with all possibilities at this stage, but with gaining a working understanding of some aspect of the system.
- Procedures
  - Set-up and initialisation operations
    A description of how to prepare for and start the task

- Input operations
  A step-by-step description of what the user must do and a description of the screens or windows that the system shows in response
- What results to expect
  A description of the expected final results
- Likely errors and possible causes
  An informal description of the major errors that are possible in this task, and how to avoid them. The aim is to give the user the confidence to carry on when problems arise. This chapter should not simply provide a list of errors, but should set them in their context.

4 REFERENCE SECTION

This chapter should give comprehensive information about all the software capabilities. The point of view is that of the operator or expert user.

The chapter should contain a list of operations sorted for easy access. The order of precedence may be alphabetical (e.g. for command-driven systems) or correspond directly to the structure of the user interface (e.g. for menu-driven systems).

In contrast to the informal style of the instruction section, the reference section should be formal, rigorous and exhaustive.

4.1 Operations

For each operation, this section should provide:

- Functional description
  A concise description of what the operation achieves.
- Cautions and warnings
  A list of cautions and warnings that apply to the operation.
- Formal description, including as appropriate:
  - Required parameters
  - Optional parameters
  - Defaults
  - Syntax & semantics.

This section should describe precisely what the operation does and how it is used. The means of doing this should be decided in advance. Syntax may be defined formally using the Backus-Naur Form (BNF). Semantics may be described by means of tables, diagrams, equations, or formal language.

4.2 Examples

This section gives one or more worked examples to enable operators to understand the format at glance. Commands should be illustrated with several short examples, giving the exact form of the most common permutations of parameters and qualifiers in full, and stating what they achieve.

4.3 Possible Errors and Their Causes

This section lists all the errors that are possible for this operation and state what causes each one. If error numbers are used, it will explain what each one means.

4.4 References to Related Operations

This section gives references to other operations, which the operator may need to complete a task, and to logically related operations (e.g. it should refer to RETRIEVE and DELETE when describing the INSERT operation).

The operations should be described in a convenient order for quick reference: for example, alphabetically, or in functionally related groups. If the section is separately bound, then it should contain its own tables of error messages, glossary, and index; otherwise they should be provided as appendices to the body of the user manual. These appendices are described below.

Appendix A - Error Messages and Recovery Procedures
This appendix should list all the error messages. It should not simply repeat the error message: referral to this section means that the user requires help. For each error message the section should give a diagnosis, and suggest recovery procedures. For example:

File ‘TEST.DOC' does not exist.

There is no file called ‘TEST.DOC' in the current directory and drive. Check that you are in the correct directory to access this file.

If recovery action is likely to involve loss of data, remind the user about possible backup or archiving procedures.

Appendix B – Glossary
A glossary should be provided if the manual contains terms that operators are unlikely to know, or that are ambiguous.
Care should be taken to avoid redefining terms that operators usually employ in a different sense in a similar context. References may be provided to fuller definitions of terms, either in the operational manual itself or in other sources.
Pairs of terms with opposite meanings, or groups of terms with associated meanings, should be cross-referenced within the glossary. Cross-references may be indicated with a highlight, such as italic type.
List any words used in other than their plain dictionary sense, and define their specialised meanings.
All the acronyms used in the user manual should be listed with brief explanations of what they mean. This glossary is intended as an extension of the glossary provided in Section 1 of this operational manual.

Appendix C – Analytical Index
An index helps to make manuals easier to use. Manuals over 40 pages should have an index, containing a systematic list of topics from the user's point of view.
Indexes should contain major synonyms and variants, especially if these are well known to users but are not employed in the operational manual for technical reasons. Such entries may point to primary index entries.
Index entries should point to topics in the body of the manual by:

- Page number
- Section number
- Illustration number
- Primary index entry (one level of reference only).
-

Index entries can usefully contain auxiliary information, especially cross-references to contrasting or related terms. For example, the entry for INSERT could say `see also DELETE'.
Indexes can be made particularly helpful if attention is drawn primarily to important keywords, and to important locations in the body of the manual. This may be achieved by highlighting such entries, and by grouping minor entries under major headings. Indexes should not contain more than two levels of entry.

If a single index points to different kinds of location, such as pages and illustration numbers, these should be unambiguously distinguished, e.g.
Page 35, Figure 7
Since the use of highlighting (35, 7) is not for instance sufficient to prevent confusion in this case

9 Software Design Document

TABLE OF CONTENTS

1. INTRODUCTION

This chapter should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This chapter shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this software design document and specify its intended readership.

1.2 Scope

This section should:

- Identify the software products to be produced by name
- Explain what the software will do and what it will not do
- Describe the relevant objectives and goals as precisely as possible.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SYSTEM DESIGN OVERVIEW

This chapter should briefly introduce the system context and design, and discuss the background to the project.
This section may summarise the costs and benefits of the selected architecture, and may refer to trade-off studies and prototyping exercises.

3. SYSTEM INTERFACES CONTEXT

This section should define all the external interfaces. This discussion should be based on system block diagram or context diagram to illustrate the relationship between this system and other systems.

4. DESIGN STANDARDS, CONVENTIONS AND PROCEDURES

This section should briefly state, and if needed should summarise, the software standards adopted for the architectural and the detailed design. Since this information is already provided in the software project development plan, a reference to this document can be given instead. This section can be utilised by the designer to provide further information. This section is therefore not intended as a mandatory one.
The following provides an example of the data that should be included in this section:

4.1 Architectural Design Method

The design method used should be named and referenced. A brief description may be added for those readers not familiar with the method. Any deviations and extensions of the method should be explained and justified.

4.2 Detailed Design Method

Detailed design method and utilised language should be described.

4.3 Code Documentation standards

This section should describe the format, style, and tools adopted by the project for software design and code documentation. Headers, footers, section formats and typefaces should be specified. They may be prepared as word processor template files, for automatic inclusion in all project documents. If the formats are new, they should be prototyped and reviewed. This section should contain the standard module header and contain instructions for its completion

4.4 Naming Conventions

This section should explain all naming conventions used. A table of the file types and the permitted names or extensions for each is recommended for quick reference.
Conventions for naming files, programs, modules, and possibly other structures such as variables and messages, should all be reported here.

4.5 Programming Standards

This section should state the project programming language and relevant standards adopted by the project.

5. SOFTWARE TOP-LEVEL ARCHITECTURAL DESIGN

This chapter should describe the Software Top-Level Architectural Design and it shall be provided by the PDR. The top-level structure of the software product design shall be described, identifying the software components, their hierarchical relationships, any dependency and the interfaces between them. The following structure is recommended.

5.1 Overall Architecture

The software architecture should be summarised describing its components. This should be presented for instance as structure charts or object diagrams showing the hierarchy, control flow (AD14) and data flow between the components.
Components can be organised in various ways to provide the views needed by different members of the development organisation.
Organisation of this section depends upon the selected design method. The following provides an example in accordance with IEEE Std 1016, 'Recommended Practice For Software Design Descriptions'. That standard describes three possible ways of presenting the software design. They are the 'decomposition view', the 'dependency view' and the 'interface view'. Ideally, all views should be provided.
The decomposition view shows the component breakdown. It defines the system using one or more of the identity, type, purpose, function and subordinate parts of the component description.

Suitable methods are tables listing component identities with a one-line summary of their purpose. The intended readership consists of managers, for work package definition, and development personnel, to trace or cross-reference components and functions.
The dependency view emphasises the relationships among the components. It should define the system using one or more of the identity, type, purpose, dependency and resource parts of the component description. Suitable methods of presentation are structure charts and object diagrams. The intended readership of this description consists of managers, for the formulation of the order of implementation of work packages, and maintenance personnel, who need to assess the impact of design changes.
The interface view emphasises the functionality of the components and their interfaces. It should define the system using one or more of the identity, functions and interface parts of the component description. Suitable methods of presentation are interface files and parameter tables. The intended readership of this description comprises designers, programmers and testers, who need to know how to use the components in the system.

This section should also contain a summary of the computer resources required to build, operate and maintain the software

5.2 Software Product Components
This section shall describe:
- The software components, constituting the software product. Each software component shall be uniquely identified and software requirements allocation shall be provided, for each software component.
- The relationship between the software components
- The purpose of each software component. If the software component is written for re-use, its provided functionality has to be described from an external point of view and its external interfaces shall be also well documented.
- Identification, for each software component the type of development (e.g., new development, software to be re-used) should be provided.

The descriptions of the components should be laid out hierarchically. There should be subsections dealing with the following aspects of each component:

5.2.n Component Identifier
5.2.n.1 Type
5.2.n.2 Purpose
5.2.n.3 Function
5.2.n.4 Subordinates
5.2.n.5 Dependencies
5.2.n.6 Interfaces
5.2.n.7 Resources
5.2.n.8 References
5.2.n.9 Processing
5.2.n.10 Data

The number 'n' should relate to the place of the component in the hierarchy.

5.2.n Component Identifier
Each component should have a unique identifier for effective configuration management. The component should be named according to the rules of the programming language or operating system to be used. Where possible, a hierarchical naming scheme should be used that identifies the parent of the component (e.g. ParentName_ChildName)
The identifier should reflect the purpose and function of the component and be brief yet be meaningful. If abbreviation is necessary, abbreviations should be applied consistently and

without ambiguity. Abbreviations should be documented. Component identifiers should be mutually consistent (e.g. if there is a routine called READ_RECORD then one might expect a routine called WRITE_RECORD, not RECORD_WRITING_ROUTINE).

5.2.n.1 Type

Component type should be defined by stating its logical and physical characteristics. The logical characteristics should be defined by stating the package, library or class that the component belongs to. The physical characteristics should be defined by stating the type of component, using the implementation terminology (e.g. task, subroutine, subprogram, package, and file).

The contents of some component description sections depend on the component type. For the purpose of this guide the categories: executable (i.e. contains computer instructions) or non-executable (i.e. contains only data) are used.

5.2.n.2 Purpose

The purpose of a component should be defined by tracing it to the software requirements that it implements.

Backward traceability depends upon each component description explicitly referencing the requirements that justify its existence.

5.2.n.3 Function

The function of a component must be defined in the top-level architectural design. This should be done by stating what the component does.

The function description depends upon the component type. Therefore it may be a description of:

- The process;
- The information stored or transmitted.

Process descriptions may use such techniques as structured English, precondition-postcondition specifications and state-transition diagrams.

5.2.n.4 Subordinates

The subordinates of a component should be defined by listing the immediate children. The subordinates of a module are the modules that are 'called by' it. The subordinates of a database could be the files that 'compose' it. The subordinates of an object are the objects that are 'used by' it.

5.2.n.5 Dependencies

The dependencies of a component should be defined by listing the constraints placed upon its use by other components. For example:

'What operations must have taken place before this component is called?'
'What operations are excluded when this operation is taking place?'
'What components have to be executed after this one?'

5.2.n.6 Interfaces

Both control flow and data flow aspects of an interface need to be specified for each 'executable' component. Data aspects of 'non-executable' components should be defined in Section 5.3.

The control flow to and from a component should be defined in terms of how execution of the component is to be started (e.g. subroutine call) and how it is to be terminated (e.g. return). This may be implicit in the definition of the type of component, and a description may not be necessary. Control flows may also take place during execution (e.g. interrupt) and these should be defined, if they exist.

The data flow input to and output from each component must be identified too. Data structures should be identified that:

- Are associated with the control flow (e.g. call argument list)

- Interface components through common data areas and files.

One component's input may be another's output and to avoid duplication of interface definitions, specific data components should be defined and described separately (e.g. files, messages). The interface definition should only identify the data component and not define its contents.

The interfaces of a component should be defined by explaining 'how' the component interacts with the components that use it. This can be done by describing the mechanisms for:

- Invoking or interrupting the component's execution
- Communicating through parameters, common data areas or messages.

If a component interfaces to components in the same system then the interface description should be defined here. If a component interfaces to components in other systems, the interface description should be defined in an Interface Control Document (ICD).

5.2.n.7 Resources

The resources a component requires should be defined by itemising what the component needs from its environment to perform its function. Items that are part of the component interface are excluded. Examples of resources that might be needed by a component are displays, printers and buffers.

5.2.n.8 References

Explicit references should be inserted where a component description uses or implies material from another document.

5.2.n.9 Processing

The processing that a component needs to do should be defined by summarising the control and data flow within it. For some kinds of component (e.g. files) there is no such flow. In practice it is often difficult to separate the description of function from the description of processing. Therefore a detailed description of function can compensate for a lack of detail in the specification of the processing. Techniques of process specification more oriented towards software design are program design language, pseudo code and flow charts.

The top-level design should not provide a complete, exhaustive specification of the processing to be done by bottom-level components; this should be done in the detailed design chapter.

The processing of higher-level components, especially the control flow, data flow and state transitions should be specified.

Software constraints may specify that the processing be performed by means of a particular algorithm (which should be stated or referenced).

5.3 Internal Interfaces Design

This section should specify the internal interfaces among the identified software components. This section shall be initially provided at the PDR. It will be updated by CDR as necessary.

5.3.1 Internal Interfaces Identification

This section shall provide a unique identification for all the interfacing software components. It shall be provided by PDR and then updated as necessary at CDR. The interface data defined in the previous section, by component, will be organised in this section showing the complete internal interfaces map, using as appropriate diagrams or matrices supporting their cross-checking.

5.3.1.x Internal Interface Specification.

For each identified internal interface, all the defined data elements shall be specified.

The amount of detail required depends strongly on the type of component. The logical and physical data structure of files that interface major components should be defined in detail.

The specification of the data internal to a major component should be postponed to the detailed design. Data structure definitions must include the:

- Description of each element (e.g. name, type, dimension)
- Relationships between the elements (i.e. the structure)
- Range of possible values of each element
- Initial values of each element.

Subsections will be provided as necessary to specify all identified internal interfaces.

6. SOFTWARE COMPONENTS DETAILED DESIGN

This section shall be defined as long as the software detailed design is defined. It can be then updated to include any feedback from the unit testing and software integration. This section shall be provided by CDR.

Each software component and software unit of the software product shall be completely described.
This section shall be structured per software component and, in turn, in software units.

The descriptions of the components should be laid out hierarchically following the same structure as in previous chapter

6.n.1 Type
6.n.2 Purpose
6.n.3 Function
6.n.4 Subordinates
6.n.5 Dependencies
6.n.6 Interfaces
6.n.7 Resources
6.n.8 References
6.n.9 Processing
6.n.10 Data
The number `n' should relate to the place of the component in the hierarchy. For each section, subsections can be added as necessary to define the detailed design of lower level units.

6.n.1 Component Identifier

Each component unique identifier will be restated here for effective configuration management. The component should be named according to the rules of the programming language or operating system to be used. Where possible, a hierarchical naming scheme should be used that identifies the parent of the component (e.g. ParentName_ChildName)
The identifier should reflect the purpose and function of the component and be brief yet meaningful. If abbreviation is necessary, abbreviations should be applied consistently and without ambiguity.
Abbreviations should be documented. Component identifiers should be mutually consistent (e.g. if there is a routine called READ_RECORD then one might expect a routine called WRITE_RECORD, not RECORD_WRITING_ROUTINE).
A naming style that clearly distinguishes objects of different classes is good programming practice. In Pascal, for instance, it is traditional to use upper case for user-defined types, mixed case for modules, and lower case for variables, giving the following appearance:

- procedure Count_Chars; {a module}
- type SMALL_INT = 1..255; {a type}
- var count: SMALL_INT; {a variable}

Other styles may be appropriate in other languages. The naming style should be consistent throughout a project. It is wise to avoid styles that might confuse maintenance programmers accustomed to standard industrial practices.

6.n.2 Type

The component type should be defined by stating its logical and physical characteristics. The logical characteristics should be defined by stating the package, library or class that the component belongs to. The physical characteristics should be defined by stating the type of component, using the implementation terminology (e.g. task, subroutine, subprogram, package, and file). The contents of some component-description sections depend on the component type. For this guide the categories: executable (i.e. contains computer instructions) or non-executable (i.e. contains only data) are used.

6.n.3 Purpose

The purpose of a component should be defined by tracing it to the software requirements that it implements.
Backward traceability depends upon each component description explicitly referencing the requirements that justify its existence.

6.n.4 Function

The function of a component must be defined. This should be done by stating what the component does.
The function description depends upon the component type. Therefore it may be a description of the:

- Process
- Information stored or transmitted.

Process descriptions may use such techniques as structured English, precondition-postcondition specifications and state-transition diagrams.

6.n.5 Subordinates

The subordinates of a component should be defined by listing the immediate children. The subordinates of a program are the subroutines that are `called by' it. The subordinates of a database could be the files that ‘compose' it. The subordinates of an object are the objects

6.n.6 Dependencies

The dependencies of a component should be defined by listing the constraints placed upon its use by other components. For example:

‘What operations must have taken place before this component is called?'
‘What operations are excluded while this operation is taking place?'
‘What operations have to be carried out after this one?’

6.n.7 Detailed Interfaces Design

Both control flow and data flow aspects of each interface need to be specified for each ‘executable' component. Data aspects of ‘non-executable' components should be defined in Subsection 6.n.10.
The control flow to and from a component should be defined in terms of how execution of the component is to be started (e.g. subroutine call) and how it is to be terminated (e.g. return). This may be implicit in the definition of the type of component, and a description may not be necessary. Control flows may also take place during execution (e.g. interrupt) and these should be defined, if they exist.
The data flow input to and output from each component must be detailed here. Data structures should be identified that:

- Are associated with the control flow (e.g. call argument list)
- Interface components through common data areas and files.

One component's input may be another's output and to avoid duplication of interface definitions, specific data components should be defined and described separately (e.g. files, messages). The interface definition should only identify the data component and not define its contents.

The interfaces of a component should be defined by explaining 'how' the component interacts with the components that use it. This can be done by describing the mechanisms for:

- Invoking or interrupting the component's execution
- Communicating through parameters, common data areas or messages.

If a component interfaces to components in the same system, the interface description should be defined here (if not already in the Chapter 5). If a component interfaces to components in other systems, the interface description should be defined in an Interface Control Document (ICD).

6.n.8 Resources

The resources a component requires should be defined by itemising what the component needs from its environment to perform its function. Items that are part of the component interface are excluded. Examples of resources that might be needed by a component are displays, printers and buffers.

6.n.9 References

Explicit references should be inserted where a component description uses or implies material from another document.

6.n.10 Processing

This section should describe in detail how processing is carried out. Algorithms that are fully described elsewhere may be summarised briefly, provided that their sources are properly referenced.

The processing that a component needs to do should be defined by defining the control and data flow within it. For some kinds of component (e.g. files) there is no such flow. In practice it is often difficult to separate the description of function from the description of processing.

Therefore a detailed description of function can compensate for a lack of detail in the specification of the processing. Techniques of process specification more oriented towards software design are program design language, pseudo-code and flow charts.

Software constraints may specify that the processing has to be performed using a particular algorithm (which should be stated or referenced).

6.n.11 Data

The data internal to a component should be defined. The amount of detail required depends strongly on the type of component. The logical and physical data structure of files that interface components should have been defined in the DDD (files and data structures that interface major components will have been defined in Chapter 5). The data structures internal to a program or subroutine should also be specified (in contrast to Chapter 5, where it is omitted).

Data structure definitions must include the:

- Description of each element (e.g. name, type, dimension)
- Relationships between the elements (i.e. the structure)
- Range of possible values of each element
- Initial values of each element.

7. SOFTWARE CODE LISTINGS

This section should contain the software code listings or precisely reference the medium containing the code listings. It shall be provided by CDR.

10 Software Validation Testing Specification

TABLE OF CONTENTS

1 INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software validation testing specification and specify its intended readership.

1.2 Scope
This section should:

- Specify the software item, validated utilising this software validation testing specification
- Specify whether this software validation testing specification is written based on the software requirements of the requirements baseline or on the requirements of the technical specification
- Specify in which step of the validation campaign it will be utilised.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 TEST DESIGN
The overall section is organised according to each identified test design.

3.n Validation Test "n" Design

3.n.1 Test Design Identifier
The title of this section should specify the test design uniquely. The content of this section should briefly describe the test design.

3.n.2 Features to Be Tested
This section should identify the test items and describe the features to be tested. This section shall identify the software requirements (either of the technical specification or requirements baseline) which are going to be validated by the test.

3.n.3 Approach Refinements
This section should identify the test approach and the test class (e.g. stress test, black-box test…) for the specific test design, as defined in the software validation plan.
This description should also provide the rationale for test case selection and grouping into test procedures. The method for analysing test results should be also identified (e.g. compare with expected output, compare with old results…). Configuration of the software validation testing facility (both hardware and software) required to execute the identified test shall be also described.

3.n.4 Test Case Identification
This section should list the test cases associated with the test design and provide a summary description of each one.

3.n.5 Test Pass/Fail Criteria
This section should specify the criteria to decide whether the test was successful or not.

4 TEST CASE SPECIFICATION

4.n Test Case "n"

4.n.1 Test Case Identifier
The title of this section should specify the test case uniquely. Short description of the test case purpose shall be provided.

4.n.2 Test Items
This section should identify the test items. Reference to the appropriate documentation shall be provided.
Information for tracing the tests down to the software requirements shall be provided.

4.n.3 Inputs Specification
This section should specify the inputs required to execute the test case.

4.n.4 Outputs Specifications
This section should specify the expected outputs from executing the test case.

4.n.5 Environmental Needs
4.n.5.1 Hardware
This section should specify the exact configuration and the set-up of the facility used to execute the test case as well as the utilisation of any special test equipment (e.g. bus analyser).

4.n.5.2 Software
This section should specify the configuration of the software utilised to support the test (e.g. identification of the simulation configuration).

4.n.5.3 Other
This section should specify any other special requirement.

4.n.6 Special Procedural Requirements
This section should describe any special constraints on the test procedures used.

4.n.7 Inter-Case Dependencies
This section should identify all the test cases that need to be executed before this test case.

4.n.8 Test Script
This section should specify or identify the test script used to execute the test case. The test scripts can be included in Appendix A.

5 TEST PROCEDURES

5.n Test Procedure “n”

5.n.1 Test Procedures Identifier
The title of this section should specify the test procedure uniquely. Related test shall also be addressed.

5.n.2 Purpose
This section should describe the purpose of this procedure. A reference to each test case used by the test procedure should be given.

5.n.3 Special Requirements
This section should identify any special requirement for the execution of this procedure.

5.n.4 Procedure Steps
This section should define the steps described in the subsections below as applicable

5.n.4.1 Log
This section should describe any special methods or format for logging the results of test execution, the incidents observed, and any other event pertinent to this test

5.n.4.2 Set-up
This section should describe the sequence of actions necessary to begin execution of this procedure

5.n.4.3 Start
This section should describe the actions necessary to begin execution of this procedure

5.n.4.4 Proceed
This section should describe the actions necessary during the execution of the procedure

5.n.4.5 Measure
This section should describe how the test measurements will be made.

5.n.4.6 Shut Down
This section should describe the actions necessary to suspend testing when interruption is forced by unscheduled events.

5.n.4.7 Restart

This section should identify any procedural restart points and describe the actions necessary to restart procedure at each of these points

5.n.4.8 Stop
This section should describe the actions necessary to bring execution to an orderly halt.

5.n.4.9 Wrap-up
This section should describe the actions necessary to terminate testing.

5.n.4.10 Contingencies
This section should describe the actions necessary to deal with anomalous events that may occur during execution.

6. TRACEABILITY MATRICES

6.1 Test to Requirements Traceability Matrix

This section should provide correlation information between all the identified tests and software requirements (either of the requirements baseline or of the technical specification, as applicable)

6.2 Requirements to Test Traceability Matrix

This section should provide correlation information between software requirements (either of the requirements baseline or of the technical specification, as applicable) and the identified tests.

APPENDICES

Appendix A- Test Scripts

All the test scripts utilised to execute the test cases should be included in this section or provided in electronic format (giving a reference in this appendix).

11 Software Requirements Verification Report

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this software requirements verification report and specify its intended readership.

1.2 Scope

This section should:

- Identify the software products to which this report refers
- Identify the software requirements documentation (e.g. the software requirements specification)
- Describe the benefits, objectives and goals of the reported activity as precisely as possible.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3. SOFTWARE REQUIREMENTS VERIFICATION RESULTS

3.1 External Consistency Evaluation Results

This chapter shall document the consistency evaluation with system requirements and system partitioning

3.2 Internal Consistency Evaluation Results

This chapter shall document the consistency evaluation between software requirements

3.3 Software Requirements Evaluation Results

This chapter shall document the completeness of the software. This chapter shall address the following:

3.3.1 Correctness of the software requirements related to safety, security and criticality
3.3.2 Verifiability of software requirements
3.3.3 Correctness of the methods used for software requirements definition.

3.4 Software Design Feasibility Evaluation

For the software item, the software design feasibility shall be assessed and reported regarding the software requirements evaluation.

3.5 Operations and Maintenance Feasibility Evaluation

For the software item, the operation and maintenance feasibility shall be reported regarding the software requirements evaluation results.

4. SOFTWARE REQUIREMENTS EVALUATION RESULTS

4.1 Software Requirements Evaluation Summary

The summary of all evaluation tasks shall be included in this section, clearly identifying the product characteristics crucial to its safety.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended solution.

12 Software Requirements Traceability Matrices Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software requirements traceability matrices report and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to which this software traceability matrix report applies
- Identify all the reference software documentation.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE REQUIREMENTS TRACEABILITY

2.1 Software Requirements Specification to System Level Specification Traceability Matrix

This section shall include a traceability matrix for software requirements contained in the software requirements specification to the system level specification requirements.
This section shall be provided by PDR.

2.2 System Level Specification to Software Requirements Specification Traceability Matrix

This section shall include a traceability matrix from system level specification requirements to software requirements specification.
This section shall be provided by PDR.

3. TOP LEVEL ARCHITECTURE TRACEABILITY

3.1 Top Level Architectural Design to Software Requirements Traceability Matrix

This section shall include a traceability matrix from software components identified in the top level architectural design to software requirements specification.
This section shall be provided by PDR.

3.2 Software Requirements to Top Level Architectural Design Traceability Matrix

This section shall include a traceability matrix from software requirements specification to software components identified in the top level architectural design.
This section shall be provided by PDR.

4. SOFTWARE DESIGN TRACEABILITY

4.1 Software Units to Software Components Traceability Matrix

This section shall include a traceability Matrix from each of the identified software units to the software components of the top level architectural design.
This section shall be provided by CDR.

4.2 Software Components to Software Units Traceability Matrix

This section shall include a traceability Matrix from the software components identified in the top level architectural design to the software units
This section shall be provided by CDR.

4.3 Software Units Traceability

This section shall include a traceability matrix from software units to the software requirements allocated to it.
This section shall be provided by CDR.

5. VALIDATION TRACEABILITY

5.1 Software Test Specifications Traceability

This section shall include a traceability matrix from the software requirements to the validation w.r.t. the technical specification tests allocated to it.
This section shall be provided by CDR.

5.2 System Test Specifications Traceability

This section shall include a traceability matrix from the requirements baseline to the validation w.r.t. requirements baseline tests allocated to it.
This section shall be provided by QR and/or AR.

13 Software Architecture and Interface Verification Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this Architecture and Interface Verification Report and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to which this report refers
- Identify the verified architecture and interface documentation (e.g. the software design document)
- Describe the benefits, objectives and goals of the reported activity as precisely as possible.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 ARCHITECTURE AND INTERFACE VERIFICATION EVALUATION

3.1 External Consistency Evaluation Results
This chapter shall document the consistency evaluation with requirements

3.1 Internal Consistency Evaluation Results

This section shall document the design consistency evaluation between software components

3.2 Architecture Design Evaluation Results

This section shall document the adherence of the software architecture to the applicable requirements and standards and correctness. This chapter shall address the following:

- Architecture design compliance with software requirements
- Architecture design compliance with design methods and standards
- The appropriateness of selected methods shall be also evaluated
- Architecture design compliance with safety, security and other critical requirements
- Architecture design correctness.

This paragraph will focus on the verification of the correct implementation into the software design of the following aspects: proper sequence of events, inputs/outputs, interfaces, logic flow, allocation of timing and sizing budgets and error definition, isolation and recovery.

3.4 Software Detailed Design Feasibility Evaluation

This section should report the assessment on the testing feasibility with respect to software architecture.

3.5 Operations and Maintenance Feasibility Evaluation

For the software item, the operation and maintenance feasibility shall be reported with respect to software architecture evaluation results.

4 SOFTWARE ARCHITECTURE EVALUATION RESULTS

4.1 Software Architecture Evaluation Summary

The summary of all evaluation tasks shall be included in this section, clearly identifying the product characteristics, which are crucial to its safety.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended solution.

14 Software Design Verification Report

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this software design verification report and specify its intended readership.

1.2 Scope

This section should:

- Identify the software products to which this report refers
- Identify the verifi ed software design documentation (e.g. the software design document)
- Describe the benefits, objectives and goals of the reported activity as precisely as possible.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 DESIGN VERIFICATION

3.1 External Consistency Evaluation Results

This chapter shall document the consistency evaluation with architectural design.

3.2 Internal Consistency Evaluation Results

This chapter shall document the design consistency evaluation between software components and software units

3.3 Design Evaluation Results

This chapter shall document the adherence of the software design to the applicable requirements and standards and correctness. This chapter shall address the following:

- Design compliance with software requirements
- Design compliance with design methods and standards
- Design compliance with safety, security and other critical requirements
- Design correctness.

This paragraph will focus on the verification of the correct implementation into the software design of the following aspects: proper sequence of events, inputs/outputs, interfaces, logic flow, allocation of timing and sizing budgets and error definition, isolation and recovery.

3.4 Software Integration Testing Evaluation Results

This chapter shall document the assessment on the appropriateness of the integration test methods and applied standards

- Appropriateness of integration test methods
- Appropriateness of integration standards used.

3.5 Software Testing Feasibility Evaluation

For the software item, testing feasibility shall be assessed and reported with respect to software design evaluation results.

3.6 Operations and Maintenance Feasibility Evaluation

For the software item, the operation and maintenance feasibility shall be reported with respect to software design evaluation results.

4 SOFTWARE DESIGN EVALUATION RESULTS

4.1 Software Design Evaluation Summary

The summary of all evaluation tasks shall be included in this section, clearly identifying the product characteristics, which are crucial to its safety.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended resolution

15 Software Code Verification Report

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this software code verification report and specify its intended readership.

1.2 Scope

This section should:

- identify the software products to which this report refers.
- Identify the verified software code documentation (e.g. code listings)
- describe the benefits, objectives and goals of the reported activity

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE CODE EVALUATION REPORT

3.1 External Consistency Evaluation Results

This chapter shall document the consistency evaluation with the requirements and design of the software item

3.2 Internal Consistency Evaluation Results

This chapter shall document the consistency evaluation between unit's requirements

3.3 Code evaluation Results

This chapter shall document the adherence of the code to the applicable requirements and standards. This chapter shall address the following:

- Code compliance with software design and software requirements
- Code compliance with coding methods and standards
- Code compliance with safety, security and other critical requirements
- Code correctness.

3.4 Software Unit Testing Evaluation Results

This chapter shall document the coverage of the software unit tests and the code testability

3.5 Software Integration and Testing Feasibility Evaluation

For the software items, the integration and testing feasibility shall be assessed and reported with respect to coding and unit testing evaluation.

3.6 Operations and Maintenance Feasibility Evaluation

For the software items, the operation and maintenance feasibility shall be reported with respect to coding and unit testing evaluation results.

4 SOFTWARE CODE EVALUATION RESULTS

4.1 Software Code Evaluation Summary

The summary of all performed evaluation tasks shall be included in this section, clearly identifying the product characteristics, which are crucial to its safe.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended resolution

16 Software Documentation Verification Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software documentation verification report and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to which this report refers
- Identify the verified software documentation
- Describe the benefits, objectives and goals of the reported activity.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE DOCUMENTATION EVALUATION

3.1 Software Documentation evaluation

This section shall document the evaluation of the documentation, considering its timely preparation, adequacy, completeness and consistency.

3.2 Configuration Management

This section shall document the evaluation of the assessed documentation, considering configuration management.

4 SOFTWARE DOCUMENTATION EVALUATION RESULTS

4.1 Software Documentation Evaluation Summary

The summary of all performed evaluation tasks shall be included in this section

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended resolution.

17 Software Integration Verification Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software integration verification report and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to which this report refers
- Identify the verified software integration documentation (e.g. the Software Integration Test Plan)
- Describe the benefits, objectives and goals of the reported activity.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE INTEGRATION EVALUATION RESULTS

3.1 External Consistency Evaluation Results
This chapter shall document the consistency evaluation with system requirements

3.2 Internal Consistency Evaluation Results

This chapter shall document the internal consistency evaluation

3.3 Software Testing Evaluation Results

This chapter shall document the evaluation of completeness of the integration testing. This chapter shall address the following:

3.3.1 Traceability to requirements
3.3.2 Test coverage of the software requirements
3.3.3 Software integration testing correctness and completeness
3.3.4 Evaluation of conformance of expected results and of code
3.3.5 Appropriateness of test standards and methods

3.4 Software Validation Testing Feasibility Evaluation

For the software item, software validation testing feasibility shall be assessed and reported with respect to software integration evaluation results.

3.5 Operations and Maintenance Feasibility Evaluation

For the software item, the operation and maintenance feasibility shall be reported with respect to software integration evaluation results.

4 SOFTWARE INTEGRATION EVALUATION RESULTS

4.1 Software Integration Evaluation Summary

The summary of all evaluation tasks shall be included in this section.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended resolution.

18 Software Integration Plan

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software integration test plan and specify its intended readership.

1.2 Scope
This section should:
- Define the software products to which this plan applies
- Summarise the planned activities.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:
1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:
1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE INTEGRATION TESTING PLANNING
This section shall specify responsibility and schedule information for the software integration testing. This chapter shall be preliminary submitted at the PDR and finalised by CDR. This section shall be structured as follows:

2.1 Organisation
This section should describe the organisation of software integration activities. Topics that should be included are:
- Roles
- Reporting channels
- Levels of authority for solving problems
- Relationships to other activities such as project management, development, configuration management and product assurance.

2.2 Master Schedule
This section should identify the schedule for the planned integration activities. In particular test milestones identified in the software project schedule and all item delivery events. This section should specify:

- Any additional test milestones and state the time required for each testing task
- The schedule for each testing task and test milestone
- The period of use for the test facilities.

2.3 Resources Summary
This section should summarise the resources needed to perform the software integration activities such as staff, hardware and software tools.

2.4 Responsibilities
This section should define the specific responsibilities associated with the roles described in section 2.1. In particular this section should identify the groups responsible for managing, designing, preparing and executing the tests. Groups may include developers, technical support staff and product assurance staff.

2.5 Tools, Techniques and Methods
This section should identify the hardware platforms, software tools, techniques and methods used for software integration activities.

2.6 Personnel and Personnel Training Requirements
This section should specify any requirement for software integration personnel and any necessary training needs.

2.7 Risks and Contingencies
This section should identify risks to the software integration testing campaign. Contingency plans should be specified.

3 SOFTWARE INTEGRATION TESTING PROCEDURES

3.1 Problem Reporting and Resolution
This section should explain the method utilised to document and report problems found in the software during software integration testing. This should clearly identify the problem reporting and resolution process utilised for the software under test and its support software. Definition of the software control board and the forwarding procedures to the customer should also be addressed.

3.2 Deviation Policy
This section should define the procedures for deviating from the plan, and define the levels of Authorisation required for approval of deviations. The information required for deviations should include task identification, deviation rationale and the effect of software quality.

3.3 Control Procedures
This section should identify the configuration management procedures, specifying the level of configuration control to be implemented on the software under test and its test support software.

4. SOFTWARE INTEGRATION TESTING TASKS IDENTIFICATION
This section shall describe the approach to be utilised for the integration testing. This section shall be preliminary submitted at the PDR, based on the top-level architectural design and then finalised at the CDR.

This section shall specify which are the tasks and the items under tests, as well as the criteria to be utilised. The section shall be structured as follows:

4.1 Software Items Under Test
This section should identify the software items under test, providing a summary of their functions. Reference to their documentation can be done.

4.2 Features to Be Tested
This section shall identify the features to t, making references to the applicable documentation (e.g. making reference to the appropriate sections of the software design document).

4.3 Features not to Be Tested
This section should identify all the features and significant combinations that will not be tested.

4.4 Test Pass/Fail Criteria
This section shall specify the criteria to be used to determine whether or not tests are passed.

4.5 Suspension Criteria and Resumption Requirements
This section should specify the criteria used to suspend all, or part of, the testing activities on the test items associated with the plan. The section should specify the testing activities that have to be repeated when testing is resumed.

4.6 Deliverable Items
This section should identify the items to be delivered. This section should clearly address deliverable items internal to the software development organisation (what, when and how) as well as items to be delivered to the customer (what, when and how), in accordance with the established software project development plan.
The following list provides examples of deliverable items:

- Software integration test plan (including test cases, test design, test procedures)
- Software integration testing report.

5. SOFTWARE INTEGRATION TESTS DESIGN

The overall section is organised according to each identified test design. This section should provide the definition of software integration test design. This section is required by the CDR.

5.n Test Design “n”

5.n.1 Test Design Identifier
The title of this section should specify the test design uniquely. The content of this section should briefly describe the test design.

5.n.2 Features to Be Tested
This section should identify the test items and describe the features to be tested. This section shall identify the associated design documentation (appropriate sections of the software design document) which is going to be validated by the test.

5.n.3 Approach Refinements
This section should identify the test approach and the test class (e.g. component integration sequence) for the specific test design.
This description should also provide the rationale for test cases selection and grouping into test procedures. The method for analysing test results should be also identified (e.g. comparison with expected output, comparison with old results…). Configuration of the

software integration testing facility (both hardware and software) required to execute the identified test shall be also described.

5.n.4 Test Case Identification
This section should list the test cases associated with the test design and provide a summary description of each one.

5.n.5 Test Pass/Fail Criteria
This section should specify the criteria to decide whether the test was successful or not.

6. SOFTWARE INTEGRATION TEST CASES SPECIFICATION

This section should provide the identification of software integration test cases. For each uniquely identified test, the following shall be specified by the CDR

6.n Test Case "n"

6.n.1 Test Case Identifier
The title of this section should specify the test case uniquely. Short description of the test case purpose shall be provided.

6.n.2 Test Items
This section should identify the test items. Reference to the appropriate documentation shall be provided.

6.n.3 Inputs Specification
This section should specify the inputs required to execute the test case.

6.n.4 Outputs Specifications
This section should specify the expected outputs obtained after executing the test case.

6.n.5 Environmental Needs

6.n.5.1 Hardware
This section should specify the exact configuration and the set-up of the facility used to execute the test case as well as the utilisation of any special test equipment.

6.n.5.2 Software
This section should specify the configuration of the software utilised to support the test (e.g. identification of the stubs and drivers).

6.n.5.3 Other
This section should specify any other special requirement.

6.n.6 Special Procedural Requirements
This section should describe any special constraints on the test procedures used

6.n.7 Inter-Case Dependencies
This section should identify all the test cases that need to be executed before this test case.

7. SOFTWARE INTEGRATION TEST PROCEDURES

This section should provide the identification of software integration test procedures. For each of the uniquely identified test procedures, the following shall be specified by the CDR

7.n Integration Procedure “n”

7.n.1 Test procedures Identifier

The title of this section should specify the test procedure uniquely. Related tests shall also be addressed.

7.n.2 Purpose

This section should describe the purpose of this procedure. A reference to each test case used by the test procedure should be given.

7.n.3 Special Requirements

This section should identify any special requirement for the execution of this procedure.

7.n.4 Procedure Steps

This section should define the steps described in the subsections below as applicable

7.n.4.1 Log

This section should describe any special methods or format for logging the results of test execution, the incidents observed, and any other event pertinent to this test

7.n.4.2 Set-up

This section should describe the sequence of actions necessary to begin execution of this procedure

7.n.4.3 Start

This section should describe the actions necessary to begin execution of this procedure

7.n.4.4 Proceed

This section should describe the actions necessary during the execution of the procedure

7.n.4.5 Measure

This section should describe how the test measurements will be made

7.n.4.6 Shut Down

This section should describe the actions necessary to suspend testing when interruption is forced by unscheduled events.

7.n.4.7 Restart

This section should identify any procedural restart points and describe the actions necessary to restart procedure at each of these points

7.n.4.8 Stop

This section should describe the actions necessary to bring execution to an orderly halt.

7.n.4.9 Wrap-up

This section should describe the actions necessary to terminate testing.

7.n.4.10 Contingencies

This section should describe the actions necessary to deal with anomalous events that may occur during execution

This page intentionally left blank

19 Software Unit Test Plan

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this software unit test plan and specify its intended readership.

1.2 Scope

This section should:

- Define the software items to which this plan applies
- Summarise the planned activities.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE UNIT TESTING PLANNING

This section shall specify responsibility and schedule information for the software unit testing. This section shall be structured as follows:

2.1 Organisation

This section should describe the organisation of software unit testing activities. Topics that should be included are:

- Roles
- Reporting channels
- Levels of authority for resolving problems
- Relationships to other activities such as project management, development, configuration management and product assurance

2.2 Master Schedule

This section should identify the schedule for the software unit testing activities. In particular the test milestones identified in the software project schedule and all the item delivery events. This section should specify:

- Any additional test milestones and state the time required for each testing task
- The schedule for each testing task and test milestone
- The period of use for the Test facilities.

2.3 Resources Summary

This section should summarise the resources needed to perform the software unit testing activities such as staff, hardware and software tools.

2.4 Responsibilities

This section should define the specific responsibilities associated with the roles described in section 2.1. In particular this section should identify the groups responsible for managing, designing, preparing and executing the tests. Groups may include developers, technical support staff and product assurance staff.

2.5 Tools, Techniques and Methods

This section should identify the hardware platforms, software tools, techniques and methods used for software unit testing activities.

2.6 Personnel and Personnel Training Requirements

This section should specify any requirement for software integration personnel and any necessary training needs.

2.7 Risks and Contingencies

This section should identify risks to the software integration testing campaign. Contingency plans should be specified.

3 SOFTWARE UNIT TESTING PROCEDURES

3.1 Problem Reporting and Resolution

This section should explain method utilised to document and report problems found in the software during software unit testing. This should clearly identify the problem reporting and resolution process utilised for the software under test and its support software. Definition of the software control board and the forwarding procedures to the customer should also be addressed.

3.2 Deviation Policy

This section should define the procedures for deviating from the plan, and define the levels of authorisation required for approval of deviations. The information required for deviations should include task identification, deviation rationale and the effects on software quality.

3.3 Control Procedures

This section should identify the applied configuration management procedures, specifying the level of configuration control to be implemented on the software under test and its test support software.

4 SOFTWARE INTEGRATION TESTING TASKS IDENTIFICATION

This section shall describe the approach to be utilised for the software unit testing.

This section shall specify which are the tasks and the items under tests, as well as the criteria to be utilised. The section shall be structured as follows:

4.1 Software Items under Test
This section should identify the software items under test, providing a summary of their functions. Reference to their documentation can be done.

4.2 Features to Be Tested
This section shall identify the features to be tested, making references to the applicable documentation (e.g., making reference to the appropriate sections of the software design document).

4.3 Features not to Be Tested
This section should identify all the features and significant combinations that will not be tested.

4.4 **Test Pass/Fail Criteria**
This section shall specify the criteria to be used to determine whether or not tests are passed.

4.5 Suspension Criteria and Resumption Requirements
This section should specify the criteria used to suspend all, or a part of, the testing activities on the test items associated with the plan. The section should specify the testing activities that have to be repeated when testing is resumed.

4.6 Deliverable Items
This section should identify the items to be delivered. This section should clearly address deliverable items internal to the software development organisation (what, when and how) as well as items to be delivered to the customer (what, when and how), in accordance with the established software project development plan.
The following list provides examples of deliverable items:

- Software unit test plan (including test cases, test design and test procedures)
- Software unit testing report

5 Software Unit Tests Design
The overall section is organised according to each identified test design. This chapter should provide the definition of software unit test design.

5.n Unit Test Design “n”

5.n.1 Test Design Identifier
The title of this section should specify the test design uniquely. The content of this section should briefly describe the test design.

5.n.2 Features to Be Tested
This section should identify the test items and describe the features to be tested. This section shall identify the associated design documentation (appropriate sections of the software design document) which is going to be validated by the test.

5.n.3 Approach Refinements
This section should identify the test approach and the test class (e.g. component integration sequence) for the specific test design.
This description should also provide the rationale for test case selection and grouping into test procedures. The method for analysing test results should be also identified (e.g. compare with expected output, compare with old results…). Configuration of the software unit testing facility (both hardware and software) required to execute the identified test shall be also described.

5.n.4 Test Case Identification
This section should list the test cases associated with the test design and provide a summary description of each one.

5.n.5 Test Pass/Fail Criteria
This section should specify the criteria to decide whether the test was successful or not.

6. SOFTWARE UNIT TEST CASES SPECIFICATION
This section should provide the identification of software unit test cases.

6.n Unit Test Case "n"

6.n.1 Test Case Identifier
The title of this section should specify the test case uniquely. A short description of the test case purpose shall be provided.

6.n.2 Test Items
This section should identify the test items. Reference to appropriate documentation shall be provided.

6.n.3 Inputs Specification
This section should specify the inputs required to execute the test case.

6.n.4 Outputs Specifications
This section should specify the expected outputs from executing the test case.

6.n.5 Environmental Needs

6.n.5.1 Hardware
This section should specify the exact configuration and the set-up of the facility used to execute the test case as well as the utilisation of any special test equipment.

6.n.5.2 Software
This section should specify the configuration of the software utilised to support the test (e.g. identification of the stubs and drivers).

6.n.5.3 Other
This section should specify any other special requirement.

6.n.6 Special Procedural Requirements
This section should describe any special constraints on the test procedures used

6.n.7 Inter-Case Dependencies
This section should identify all the test cases that need to be executed before this test case.

7. SOFTWARE UNIT TEST PROCEDURES
This section should provide the identification of software unit test procedures.

7.n Unit Test Procedure "n"

7.n.1 Test procedures Identifier
The title of this section should specify the test procedure uniquely. Related test shall also be addressed.

7.n.2 Purpose

This section should describe the purpose of this procedure. A reference to each test case used by the test procedure should be given.

7.n.3 Special Requirements

This section should identify any special requirement for the execution of this procedure.

7.n.4 Procedure Steps

This section should define the steps described in the subsections below as applicable.

7.n.4.1 Log

This section should describe any special methods or format for logging the results of test execution, the incidents observed, and any other event pertinent to this test.

7.n.4.2 Set-up

This section should describe the sequence of actions necessary to begin execution of this procedure.

7.n.4.3 Start

This section should describe the actions necessary to begin execution of this procedure.

7.n.4.4 Proceed

This section should describe the actions necessary during the execution of the procedure.

7.n.4.5 Measure

This section should describe how the test measurements will be made.

7.n.4.6 Shut Down

This section should describe the actions necessary to suspend testing when interruption is forced by unscheduled events.

7.n.4.7 Restart

This section should identify any procedural restart points and describe the actions necessary to restart procedure at each of these points.

7.n.4.8 Stop

This section should describe the actions necessary to bring execution to an orderly halt.

7.n.4.9 Wrap-up

This section should describe the actions necessary to terminate testing.

7.n.4.10 Contingencies

This section should describe the actions necessary to deal with anomalous events that may occur during execution

This page intentionally left blank

20 Software Unit Test Report

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this Software Unit Test Report and specify its intended readership.

1.2 Scope

This section should:

- Identify the software products to which this report is relevant
- Describe objectives and goals of the reported activities.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE UNIT TEST RESULTS REPORT

The section has to be organised by software unit test. For each of the uniquely identified software unit tests, this section shall report the detailed results.

3.n.1 Software Unit Test [Unique identifier] Report

This section should uniquely identify the software unit test (referencing the appropriate documentation), whose results are here reported.

3.n.2 Description

This section should identify the software unit under test, providing also its version/release identifier. The utilised test environment should also be reported, including identification of utilised support software.

3.n.3 Activity and Event Entries

This section should define the start and end time of each activity and event, pertaining to the unit test execution. The unit test responsible should be identified. One or more of the descriptions in the following subsections should be included as necessary.

3.n.3.1 Execution Description

This section should identify the unit test procedure being executed, with reference to its documentation. Reference to witnesses (if any) should be reported.

3.n.3.2 Unit Test Procedure Results

For each execution, this section should record the observable results (e.g. error messages, aborts and requests for test operator action). The outputs and the result of the test should be recorded.
References to generated problem reports should be recorded.

3.n.3.3 Environmental Information

This section should record any environmental conditions specific to this entry, particularly deviations from the normal.

4 SOFTWARE UNIT TESTS RESULTS EVALUATION

4.1 Software Unit Tests Results Summary

This section shall summarise the results of the unit testing. The status of the test completion shall be reported as well as the chronological recording for the execution of the software unit tests.

4.2 Recommendations

In all cases, the results of the software unit testing campaign lead to the need of corrective actions, these will be identified and reported together with a resolution planning.

21 Software Integration Test Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software integration test report and specify its intended readership

1.2 Scope
This section should:
- identify the software products to which this report is relevant;
- describe objectives and goals of the reported activities

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. .A summary of the software functionality, software configuration , its operational environment and its external interfaces shall be provided.

3 SOFTWARE INTEGRATION TESTING RESULTS REPORT
This section has to be organised by software integration test. For each of the uniquely identified software integration test, this section shall report the detailed results.

3.n.1 Software Integration Test [Unique identifier] Report
This section should uniquely identify the Software Integration Test (referencing the appropriate documentation), whose results are here reported.

3.n.2 Description

This section should identify the software product under test, providing also its version/release identifier. The utilised test environment should also be reported, including identification of utilised support software.

3.n.3 Activity and Event Entries

This section should define the start and end time of each activity and event, pertaining to the software integration test execution. The software integration test responsible should be identified. One or more of the descriptions in the following subsections should be included as necessary.

3.n.3.1 Execution Description

This section should identify the software integration test procedure being executed, with reference to its documentation. Reference to witnesses (if any) should be reported

3.n.3.2 Software Integration Test procedure Results

For each execution, this section should record the observable results (e.g. error messages, aborts and requests for test operator action). The outputs and the result of the test should be recorded.

References to generated problem reports should be recorded.

3.n.3.3 Environmental Information

This section should record any environmental conditions specific to this entry, particularly deviations from the normal.

4. SOFTWARE INTEGRATION TESTS RESULTS EVALUATION

4.1 Software Integration Testing Results Summary

This section shall summarise the results of the software integration testing. The status of the test completion shall be reported as well as chronological record of the software integration testing campaign execution

4.2 Recommendations

In all cases, the results of the software integration testing campaign lead to the need of corrective actions, these will be identified and reported together with a resolution planning.

22 Software Validation Testing Report

This report shall be submitted at the critical design review, when written to report the validation testing activities against the technical specification and at the qualification review when written to report the validation testing activities against the requirements baseline

Note: this report can be used either to report the result of the validation tests performed against the technical specification or those conducted against the requirements baseline. This shall be clearly stated in Chapter 1.1.

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this validation testing report and specify its intended readership.

1.2 Scope

This section should:

- identify the software products to which this report is relevant
- describe objectives and goals of the reported activities

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE VALIDATION TESTING RESULTS REPORT

This section has to be organised by software validation test as specified in the software validation testing specification.

For each of the uniquely identified software validation test, this section shall report the detailed results.

3.n.1 Software Validation Test [Unique identifier] Report

This section should uniquely identify the software validation test (referencing the appropriate documentation), whose results are here reported.

3.n.2 Description

This section should identify the software item under test, providing also its version/release identifier. The utilised test environment should also be reported, including identification of utilised test support software (e.g. simulation, test procedures).

3.n.3 Activity and Event Entries

This section should define the start and end time of each activity and event, pertaining to the software validation test execution. The software validation test responsible should be identified. One or more of the descriptions in the following subsections should be included as necessary.

3.n.3.1 Execution Description

This section should identify the software validation test procedure being executed, with reference to its documentation. Reference to witnesses should be reported

3.n.3.2 Software Validation Test Procedure Results

For each execution, this section should record the observable results (e.g. error messages, aborts and requests for test operator action). The outputs and the result of the test, should be recorded.
References to generated problem reports should be recorded.

3.n.3.3 Environmental Information

This section should record any environmental conditions specific to this entry, particularly deviations from the normal.

4. SOFTWARE VALIDATION TESTS RESULTS EVALUATION

4.1 Software Validation Tests Results Summary

This section shall summarise the results of the software validation testing. The status of the test completion shall be reported as well as chronological record of the validation test execution. When test results lead to the detection of problems, then a reference to the open problem report has to be provided.

4.2 Recommendations

In all cases, the results of the software validation tests lead to the need of corrective actions, these will be identified and reported together with a resolution planning.

23 Software Test Specification Evaluation Report

Note: this report can be used either to report the evaluation of the validation tests performed against the technical specification (submitted at the CDR) or those conducted against the requirements baseline (submitted at the QR). This shall be clearly stated in Chapter 1.1.

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this test specification evaluation report and specify its intended readership.

1.2 Scope

This section should:

- Identify the software products to which this report refers
- Identify the test specification documentation (e.g. the software validation testing specification)
- Describe the benefits, objectives and goals as precisely as possible of the reported activity.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE REQUIREMENTS COVERAGE

This section shall document the assessment of the software validation testing specification w.r.t. the software requirements testing coverage

4 SOFTWARE VALIDATION TESTING SPECIFICATION EVALUATION

4.1 Evaluation Summary

The summary of all performed evaluation tasks shall be included in this section.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended resolution.

24 Software Design and Test Evaluation Report

Note: this report can be used either to report the evaluation of the validation tests performed against the technical specification (submitted at the CDR) or those conducted against the requirements baseline (submitted at the QR). This shall be clearly stated in Chapter 1.1.

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software design and test evaluation report and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to which this report is relevant
- Summarise the software design and test evaluation activities documented in this report
- Identify the documentation utilised to perform such verification activities (e.g. software design documentation)
- Describe the relevant benefits, objectives and goals.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3 SOFTWARE DESIGN AND TEST EVALUATION

3.1 Test Coverage Evaluation Results

For the software item, this section shall report an assessment regarding the test coverage of the software requirements and the conformance of the software item to the software requirements.

3.2 System Integration and Testing Feasibility Evaluation

For the software item, testing feasibility shall be assessed and reported with respect to software design and testing documentation evaluation results.

3.3 Operations and Maintenance Feasibility Evaluation

For the software item, the operation and maintenance feasibility shall be reported with respect to design, code, test and results of the evaluation of the user manual.

4 EVALUATION RESULTS

4.1 Evaluation Summary

The summary of all performed evaluation tasks shall be reported in this section.

4.2 Recommendations

In all cases, the results of evaluation lead to the need of corrective actions, these will be identified and reported together with their criticality and recommended resolution.

25 Software Product Assurance Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software product assurance report and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products for which this Software Product Assurance Report is relevant
- Describe the main items addressed by this report.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3. SOFTWARE ASSESSMENT ACTIVITIES
This section summarises the undertaken verification activities throughout the life cycle to assess the software quality, the detected problems and identified corrective actions and their implementation status.

- Activities summary
- Detected problems and solutions.

4. SOFTWARE METRICS REPORTING

This section shall report metrics on the software process and on the software product, relevant to the reporting period.

- Software process metrics
- Software product metrics.

26 Software Budget Report

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 **Purpose**
This section should briefly define the purpose of this software budget report and specify its intended readership.

1.2 Scope
This section should:

- identify the software products to which this software budget report applies
- identify the phase of the life cycle in which this report is produced.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE OVERVIEW

A brief description of the software system and its context should be provided in this chapter. A summary of the software functionality, software configuration, operational environment and external interfaces shall be provided.

3. MARGINS AND TECHNICAL BUDGETS SUMMARY

3.1 Margins and Technical Budget Requirements
This section shall report the applicable requirements for computer resources utilisation.

3.2 Technical Budgets Status

This section shall provide budgeting estimation or budget measurements, in accordance with the applicable software development life cycle. The following data shall be specified:

– Analysis and measurement approach description
– Assumptions and conditions (worst-case utilisation, typical utilisation)
– Special considerations affecting the utilisation
– Units of measure utilised (e.g., cycles per second, bytes of memory, kilobytes per second…)
– Level (s) at which the estimates or measures are made (e.g. software, and software component).

27 Software Configuration File

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software configuration file and specify its intended readership.

1.2 Scope
This section should:

- Identify the version/release of the software product to which its applies.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

3. SOFTWARE RELEASE INFORMATION

3.3 Inventory of Materials Released
This section shall list by unique identifiers all physical media and associated documentation released with the software item. In case the software is not delivered via a physical medium, this shall be specified with a reference to section 3.2, which specify the delivered files.

3.2 Inventory of Software Contents
This section shall list by unique identifier all files constituting the software item under release. This section should include or make reference to any software release documentation utilised in order to formally release the software product. Checksum value of the software product shall also be specified.

3.3. Software Baseline Description

3.3.1 Software Documentation

This section shall describe the software baseline, identifying all applicable documentation, their issue, date and relevant unique identifier.

3.3.2 Software Baseline Changes

This section shall report the status of all the change proposals, engineering waivers, deviations and change requests. This section can be structured as follows:

3.3.2.1 Change Proposal
3.3.2.2 Waivers/Deviations
3.3.2.3 Change Request

3.4 Implemented Changes

This section shall list all changes incorporated in the software since the previous version.
Different classes of changes shall be properly classified (e.g. changes due to implementation of change request, non conformance reports, software problems reports...).

3.5 Adaptation Data

This section shall identify or make reference to all unique-to-site data contained in the released software.

3.6 Installation Instructions

This section shall specify all the instruction to install the software item, including:

- Procedures to install the software in the target environment
- Procedures to verify the correct execution of the installation
- Adaptation data, security issues relevant to the installation.

3.7 Possible Problems and known errors

This section shall report all the known open points relevant to the software product as well as any relevant known workaround solution.

28 Software Product Assurance Plan

TABLE OF CONTENTS

1. INTRODUCTION
This chapter should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This chapter shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software product assurance plan and specify its intended readership for the relevant software product.

1.2 Scope
This section should:

- Identify the software products to which this plan applies
- Explain the relationship to other software plans.

The compliance with ECSS SW Standards will be stated in this section together with any applied tailoring policy, with reference to the appropriate sections providing detailed compliance traceability and tailoring definition.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable documents
1.4.2 Reference documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE SYSTEM OVERVIEW
A description of the software product(s) or system should be provided, assuming an SPA perspective in order to define the context for definition and application of this plan.

3 Software Product Assurance Management
This section shall describe the organisation, structure and responsibilities for software product assurance in the relevant software processes. The following sections should be provided:

3.1 Organisation
This section shall describe the organisation for software product assurance activities. Topics that should be included are:

- Roles
- Reporting channels
- Level of authority for solving problems
- Relationships to other activities such as, project management, development, configuration management.

3.2 Responsibilities
This section should define the specific responsibilities associated to roles defined in section 3.2.

3.3 Personnel
This section should specify any requirements for software product assurance personnel.

3.4 Resources
This section should summarise the resources needed to perform the software product assurance activities.

3.5 Documentation
This section should list all the documentation to be provided in order to report the software product assurance activities.
This section shall also state how the documents are to be checked for adequacy and identify the method by which this check is performed.

4 SOFTWARE PROCESS DEFINITION

This chapter shall specify or make reference to other project documentation, the selected project software life cycle and definition of the software process. This chapter should also address identification of relevant milestones.

5 SOFTWARE PRODUCT ASSURANCE ACTIVITIES

This chapter shall specify the assurance activities to be performed throughout the software process with identification of documentation to be produced.
The SPA activities relevant to the following should be also addressed:
- Software product and process metrics
- Software criticality definition
- Verification and validation with schedule information
- Reviews
- Audit planning
- Change control
- Non conformance process definition, control and corrective action
- Supplier control.

5.1 SPA Activities for Software Development Processes
This section shall describe the product assurance activities that will ensure the quality of all the software development activities. The assessment criteria to be used and the assessment methods and tools to be employed have to be specified.

5.1.1 Assessment of plans, standards, procedures, tools and facilities
This section shall describe the activities and plans to assess all software plans and procedures (e.g., software development plan, software configuration management plan) to determine whether the selected methodologies, standards, procedures, tools and facilities comply with the project requirements and are adequate to support the software project.

5.1.2 Assessment of Configuration Management

This section shall describe the planned activities for evaluating the software configuration management system applied to the Software project. This should include evaluation of configuration identification, configuration control, configuration status accounting and reporting and preparation for configuration audits.

5.1.3 Assessment of Software Libraries

This section shall describe the planned activities to assess libraries and software configuration management tools, utilised to control the software under development. This section should also state plans to evaluate content, procedures, controls and status accounting and reporting mechanisms provided by the Libraries

5.1.4 Assessment of Documentation and Software Distribution Mechanisms

This section shall describe plans and activities for evaluating the control and distribution of documentation and the software distribution mechanism (e.g. physical media or electronic distribution).

5.1.5 Assessment of Storage and Handling

This section shall describe the plans and activities for evaluating the system of storing and handling software media and documentation. It shall include the plans for evaluating protection of materials, security of data, and backup procedures.

5.1.6 Assessment of Non-deliverables

This section shall describe the plans and activities to assess non deliverable products.

5.1.7 Assessment of Risk Management

This section shall describe the plans to asses risk management procedures and risk management activities results.

5.1.8 Assessment of Supplier Products

This section shall describe the plans and the activities for assuring software and documentation received from suppliers.

5.1.9 Assessment of Reusable Software, Commercial Off The Shelf and Customer Furnished Software

This section shall describe the plans and the activities to assess the utilisation of reusable software, COTS and customer furnished software.

5.1.10 Software Product Assurance Reports

This section shall describe the plans and activities to document and report software product assurance activities in the Software Product Assurance Reports.

5.1.11 Software Problem Resolution Process

This section shall describe the plans an the activities to monitor and control the correct implementation of the established software problem resolution process.

5.1.12 Certification

This section shall describe the plans and activities to certify that the deliverable products comply with the requirements.

5.2 Software PA Process Activities

This section shall specify the software product assurance activities, as software support process for the software life cycle.

6 SOFTWARE PRODUCT ASSURANCE PROCEDURES, METHODS, TOOLS AND FACILITIES

This chapter shall identify procedures, methods, tools and facilities that will be used to accomplish the software assurance activities. These shall be specified in accordance with the software development life cycle specified for the project. Company Standards can be referenced.

The SPA process shall be documented in detail, according to applicable ECSS SW standards, with any tailoring applied documented and justified. Charts showing the process elements could be provided to enhance readability.

This chapter shall also define the applied approach for the following:

- Software process and software product metrics definition
- Software criticality definition
- Design verification.

7 SOFTWARE DEVELOPMENT AND TESTING ENVIRONMENT ASSESSMENT

This section should assess and provide a description of the adequacy of the selected software development and testing environment.

8 FIRMWARE PROCEDURES

This section shall specify or make reference to specific procedures for firmware device programming, as applicable.

9 SOFTWARE PRODUCT ASSURANCE REQUIREMENTS COMPLIANCE

This section shall include a matrix showing the compliance with applicable software PA.

29 Software Maintenance Plan

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this software maintenance plan and specify its intended readership.

1.2 Scope
This section should:

- Identify the software products to which this software maintenance plan applies.
- Specify the scope of the maintenance activities
- Specify schedule information
- Relationship with the software development process.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. SOFTWARE MAINTENANCE PLANNING
This section shall describe the organisational support needed to execute the maintenance activities. This section shall be structured as follows:

2.1 Organisation
This section should describe the organisation of the maintenance activities. Topics that should be included are:

- Roles
- Reporting channels
- Levels of authority for resolving problems

2.2 Master Schedule
This section should identify the schedule for the planned maintenance activities.

2.3 Resources Summary
This section should summarise the estimated resources needed to perform the maintenance activities such as staff, hardware and software tools.

2.4 Responsibilities
This section should define the specific responsibilities associated with the roles described in section 2.1.

2.5 Personnel Requirements
This section should specify any requirement for maintenance personnel.

2.6 Procedures
This section should define the following kinds of maintenance procedures:
- Problem handling and reporting
- Change control
- Handling of routines and emergency situations
- Maintenance activities reporting.

3 SOFTWARE MAINTENANCE DEFINITION

3.1 Software Maintenance Scope
This section shall exactly identify the planned software maintenance and provide the definition of the kind of maintenance supported (e.g. corrective maintenance, evolutionary maintenance, improving maintenance, evolutionary maintenance, adaptive maintenance, preventive maintenance) and relevant activities.

3.2 Software Maintenance Activities Identification
This section should describe all the software processes activities to exploit the planned software maintenance activities. For each of the kind of planned kind of maintenance, the software engineering activities, verification and validation activities and software assurance activities

3.3 Software Maintenance Standards, Methods
This section should describe methods and standards used to support the software maintenance activities

3.4 Software Maintenance Activities documentation
This section should define the software documentation associated with software maintenance activities. Both the dedicated documentation (e.g. MF) and the updating of inherited documentation from the software development process should be addressed.

4 SOFTWARE MAINTENANCE FACILITY
This section shall describe the facilities (hardware and software) to be utilised to support the planned maintenance activities. Any inheritance from the software development process should be addressed.

30 Software Operations Plan

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of this operation plan and specify its intended readership.

1.2 Scope
This section should:
- identify the software products to which this operation plan applies.
- explain the major items addressed by this Plan

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. OPERATIONAL PLANNING
This section shall describe the organisational support needed to execute the operational activities.

2.1 Organisation
This section should describe the organisation of operational activities. Roles, Reporting Channels and relevant responsibilities should be also addressed. In particular relationships with the software development processes and any other involved organisation, external to the operation organisation should be described in this section.

2.2 Master Schedule

This section should identify the schedule for the planned operational activities.

2.3 Resources Summary
This section should summarise the estimated resources needed to perform the operational activities such as staff, hardware and software tools.

2.4 Personnel Requirements
This section should specify any requirement for personnel.

2.5 Procedures
This section shall specify the procedures for problem handling and relationship with the software development or maintenance process.

3 OPERATIONAL ACTIVITIES AND PROCEDURES
This section shall identify the operational activities and applied standards.

3.1 Operational Activities
This section should describe all the planned operational activities.

3.2 Operational Standards, Methods and Techniques
This section should describe standards, methods, tools (hardware and software) to be utilised for the operational activities.

4 OPERATIONAL TESTING SPECIFICATION
This section shall identify the kind of tests to be executed by the operator. This section should be organised as a software validation testing specification or a dedicated specification could be utilised. In the latter a simple reference to the independent documents has to be made in this section.

31 Software Migration Plan

TABLE OF CONTENTS

1. INTRODUCTION

This section should provide an introduction to this document, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of this migration plan and specify its intended readership.

1.2 Scope

This section should:

- Identify the software products to which this Migration Plan applies
- Explain rational and define the items affected by the planned migration
- Describe the relevant benefits, objectives and goals as precisely as possible.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in this document, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2. MIGRATION PLANNING

This section shall describe the planning for executing the migration, the schedule information for the execution, and the provided support.

2.1 Organisation

This section should identify the organisation responsible for the implementation of the migration and the organisations responsible for the execution of the migration. Relationships among the involved organisations should be addressed.

2.2 Migration Schedule

This section should identify the schedule for the execution of the migration activities.

2.3 Resource Summary
This section should summarise the estimated resources needed to perform the migration activities, such as staff, hardware and software tools.

2.4 Responsibilities
This section should define the responsibilities of all the involved organisations in the migration execution.

2.5 Planned Support for the Old Environment
This section should identify the planned support for the old environment, until migration completion.

3 MIGRATION DEFINITION

3.1 Software Migration Scope
This section shall exactly identify the need and requirements for the software migration.

3.2 Software Maintainer Activities Identification
This section should describe all the software processes activities necessary to implement the migration. In particular software implementation, verification and validation activities should be identified; needed tools and software conversion should be identified.

3.3 Software Users Activities Identification
This section should describe all the activities necessary to implement the migration. In particular any specific validation activities should be identified, needed tools and software conversion should be identified.

32 Software Migration Justification

This document is normally a section in the migration plan

TABLE OF CONTENTS

1. INTRODUCTION
This section should provide an introduction to the file, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose
This section should briefly define the purpose of migration justification file and specify its intended readership.

1.2 Scope
This section should identify the software item(s) to be migrated and the organisations the plan applies to.

1.3 Glossary
This section should define all terms, acronyms, and abbreviations used in the plan, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References
This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview
This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 Old Environment
Statement of why the old environment can no longer be supported

3 New Environment
Description of the new environment with its date of availability

4 Other Support Options available
Description of other support options available, if any, once support for the old environment has been removed

This page intentionally left blank

33 Software Retirement Plan

TABLE OF CONTENTS

0 INTRODUCTION

This section should provide an introduction to the plan, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of software retirement plan and specify its intended readership.

1.2 Scope

This section should identify the Software item(s) to be retired and the organisations the plan applies to.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in the plan, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.3 Acronyms
1.3.4 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.3 Applicable Documents
1.4.4 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 Cessation of Operations

This section shall describe the timeline of activities required to end operations of all or part of the software system, and the removal of support for that system

3 **Archiving**

Detail what software and documentation is to be archived, the archive medium, and how and where it is to be stored.

4 **Residual Support**

Detail who is responsible for any residual support issues.

5 **Transition to New Product**.

Detail how the transition to the new replacement software is to take place (if applicable), including the transfer of any data.

6 Archive Access

Describe how the archive can be accessed in the future.

34 Software Configuration Management Plan

TABLE OF CONTENTS

1 INTRODUCTION

This section should provide an introduction to the plan, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should briefly define the purpose of the particular SCMP and specify the intended readership of the SCMP.

1.2 Scope

This section should identify the configuration items to be managed, the configuration management activi ties in this plan; organisations the plan applies to.

1.3 Glossary

This section should define all terms, acronyms, and abbreviations used in the plan, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.5 Acronyms
1.3.6 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

5.4.1 Applicable Documents
5.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 SOFTWARE CONFIGURATION MANAGEMENT APPROACH

This section should describe the organisation of configuration management, and the associated responsibilities. It should also define the roles. IEEE Std 828**,** ‘Standard for Software Configuration Management Plans' recommends the following structure to be used for this section (provided only as an example).

2.1 Organisation

This section should:

- Identify the organisational roles that influence the software configuration management function (e.g. project managers, programmers, quality assurance personnel and review boards)
- Describe the relationships between the organisational roles
- Describe the interface with the user organisation.

Relationships between the organisational roles may be shown by means of an organigram. This section may reference the software project development plan.

2.2 SCM Responsibilities

This section should identify the:

- Software configuration management functions each organisational role is responsible for (e.g. identification, storage, change control, status accounting)
- Responsibilities of each organisational role in the review, audit and approval processes
- Responsibilities of the users in the review, audit and approval processes.

2.3 Interface Management

This section should define the procedures for the management of external hardware and software interfaces. In particular it should identify the:

- External organisations responsible for the systems or subsystems with which the software interfaces
- Points of contact in the external organisations for jointly managing the interface
- Groups responsible for the management of each interface.

2.4 SCMP Implementation

This section should establish the key events in the implementation of the SCMP, for example the:

- Readiness of the configuration management system for use
- Establishment of the Software Review Board
- Establishment of baselines
- Release of products.

The scheduling of the software configuration management resources should be shown (e.g. availability of software librarian, software configuration management tools and SRB). This section may cross-reference the software project development plan.

2.5 Applicable Policies, Directives and Procedures

This section should:

- identify all applicable software configuration management policies, directives or procedures to be implemented as part of this plan (corporate software configuration management documents may be referenced here, with notes describing the parts of the documents that apply);
- describe any software configuration management polices, directives or procedures specific to this project, for example:
  - project-specific interpretations of corporate software configuration management documents;
  - level of authority required for each level of control;
  - level of review, testing or assurance required for promotion.

3 CONFIGURATION IDENTIFICATION

This section should describe the conventions for identifying the software items and then define the baselines used to control their evolution.

3.1 Conventions

This section should:

- Define the project software configuration item(s) naming conventions.
- Define or reference software configuration item labelling conventions.

3.2 Baselines

For each baseline, this section should give the:

- Identifier of the baseline

- The contents, i.e. the software itself
- Tools for making derived items in the baseline (e.g. compiler, linker and build procedures)
- Test software (e.g. data, harnesses and stubs)
- RIDs, SPRs etc that relate to the baseline
- ICDs, if any, which define the interfaces of the software
- Review and approval events, and the acceptance criteria, associated with establishing each baseline
- Participation required of developers and users in establishing baselines.

Because the SCMP is a plan, the precise contents of each baseline may not be known when it is written (e.g. names of modules before the detailed design is started). When this occurs, procedures for getting a report of the contents of the baseline should be defined (e.g. a directory list). An example of a report may be supplied in this section.
If appropriate, the description of each baseline should:

- Distinguish software being developed from software being reused or purchased
- Define the hardware environment needed for each software configuration
- Trace the software configuration Items to the deliverable items listed in the software project development plan, and, for code, to the software components described in the software design documentation.

4. CONFIGURATION CONTROL

This section should provide information on code control, media control and configuration item change control.

4.1 Code (and Document) Control

This section should describe the software library handling procedures. Identification and definition of the software libraries shall also be addressed (e.g. development library, master library, archive library).
Ideally the same set of procedures should be used for each type of library.
While Section 6 of the SCMP, ‘Tools, Techniques and Methods', gives background information, this section should describe, in a stepwise manner, how these are applied.

4.2 Media Control

This section should describe the procedure for handling the hardware on which the software resides, such as:

- Magnetic disk
- Magnetic tape
- Read-Only Memory (ROM)
- Erasable Programmable Read-Only Memory (EPROM)
- Erasable Programmable Read-Only Memory (EEPROM)
- Optical disk.

Whatever media is used, this section should describe the procedures for:

- labelling media
- Storing the media (e.g. fire-proof safes, redundant off-site locations)
- Recycling the media (e.g. always use new magnetic tapes when archiving).

4.3 Change Control

This section should define the procedures for processing changes to baselines described in Section 3.2 of the plan.

4.3.1 Levels of Authority

This section should define the level of authority required to authorise changes to a baseline (e.g. software librarian, project manager).

4.3.2 Change Procedures

This section should define the procedures for processing change proposals to software.
The documentation change process shall be described and shall be in accordance with the project applicable requirements and standards.
The code change process should be described and shall be in accordance with the project applicable requirements and standards.

4.3.3 Review Board
This section should define the:
- Membership of the review board (e.g. PDR, SRB)
- Levels of authority required for different types of change.

The second point implies that the review board may delegate some of their responsibilities for change.

4.3.4 Interface Control
This section should define the procedures for controlling the interfaces. In particular it should define the:
- change control process for ICDs
- Status accounting process for ICDs.

4.3.5 Support Software Change Procedures
This section should define the change control procedures for support software items. Support software items are not necessarily produced by the developer. They may be components of the delivered system or tools used to make it. Examples are:
- Commercial software
- Reused software
- Compilers
- Linkers.

4 CONFIGURATION STATUS ACCOUNTING
This section should define how the project will keep an audit trail of changes, as follows:
- Define how configuration item status information is to be collected, stored, processed and reported
- Identify the periodic reports to be provided about the status of the CIs, and their distribution
- State what dynamic inquiry capabilities, if any, are to be provided
- Describe how to implement any special status accounting requirements specified by users.

6 TOOLS, TECHNIQUES AND METHODS FOR SCM
This section should describe the tools, techniques and methods to support:
- Configuration identification (e.g. controlled allocation of identifiers)
- Configuration item storage (e.g. source code control systems)
- Configuration change control (e.g. online problem reporting systems)
- Configuration status accounting (e.g. tools to generate accounts).

7 SUPPLIER CONTROL
This section should describe the requirements for software configuration management to be placed on external organisations such as:
- Subcontractors
- Suppliers of commercial software (i.e. vendors).

This section should reference the review and audit procedures in the software verification plan for evaluating the acceptability of supplied software. This section may reference contractual requirements.

8 RECORDS COLLECTION AND RETENTION

This section should:

- Identify the software configuration management records to be retained (e.g. CIDLs, RIDs,, SPRs, SCRs, SMRs, configuration status accounts)
- State the methods to be used for retention (e.g. fire-proof safe, on paper, magnetic tape)
- State the retention period (e.g. retain all records for all baselines or only the last three baselines).

This page intentionally left blank

35 Software Progress Report

Note: This report is not a constituent of any specific documentation folder. It shall be submitted as specified by the applicable contract (e.g. monthly).

1. INTRODUCTION

This section should provide an introduction to the software progress report, providing scope, purpose, glossary and documentation references and also describing the document organisation. This section shall be organised in the following subsections:

1.1 Purpose

This section should describe the purpose of the report. This section should include:

- Name of the project
- Reference to the applicable plan
- Reference to the applicable contract
- Specification of the reporting period.

1.2 Summary

This section should summarise the main activities and achievements during the reporting period.
Any change in the total cost to completion should be stated in this section. Changes to work package resource estimates should be summarised.

1.3 Glossary

This section should define all terms, acronyms or abbreviations used in the report, or refer to other documents where the definitions can be found. This section can be organised as follows:

1.3.1 Acronyms
1.3.2 Definition of Terms

1.4 References

This section should provide a complete list of all the applicable and reference documents, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included. Therefore the section can be structured as follows:

1.4.1 Applicable Documents
1.4.2 Reference Documents

1.5 Document Overview

This section should describe what the rest of the document contains and explain how the rest of the document is organised.

2 . TECHNICAL STATUS

2.1 Work Package Technical Status

This section should list the work packages completed during the reporting period. There should be some indication of whether each work package was completed successfully or not.
This section should list the work packages continued or started during the reporting period. The outstanding tasks for each work package should be identified.

2.2 Configuration Status

This section should summarise the changes to the configuration status in the reporting period, e.g.:

- RID statistics
- Issues and revisions of documents
- SCR, SPR and SMR statistics
- Releases of software.

2.3 Planning for Next Reporting Period

This section should state what technical progress is expected on each work package in the next reporting period.

Any events that are expected to take place in the reporting period should be described (e.g. milestones, deliveries).

3 RESOURCE STATUS

3.1 Staff Utilisation

This section should comprise:

- A table of hours booked by each member of the team
- The hours staff have booked for a number of past reporting periods (e.g. from the beginning of the current phase, or beginning of the year, or beginning of the project, as appropriate)
- A forecast for the man-hours staff will book for future reporting periods (e.g. up to the end of the current phase, or the end of the project, or other milestone).

3.2 Work Package Resource Status

This section should contain a work package progress table summarising for each work package:

Previous estimate of the effort required for completion

- Effort expended in this period
- Cumulative effort expenditure up to the end of the reporting period
- Estimation of remaining effort required for completion.

3.3 Resource Status Summary

This section should present the aggregated effort expenditure for the reporting period for:

- The project;
- Its subsystems (for medium and large-size projects).

4 SCHEDULE STATUS

4.1 Milestone Trend Charts

This section should provide an updated milestone trend chart for all the milestones identified in the applicable plan and relevant to the reporting period. All changes to milestone dates made in this reporting period should be explained and justified.

4.2 Schedule Summary

This section should provide an updated bar chart (i.e. Gantt chart) showing work packages completed and milestones achieved. Progress on partially completed work packages may also be shown.

5 PROBLEMS

This section should identify any problems, which are affecting or could affect progress. These may include technical problems with the software under development, environmental problems (e.g. computers, communications, and accommodation) and resource problems (e.g. staffing problems). This list is not exhaustive.

6 FINANCIAL STATUS REPORT

The financial status report (sometimes called the cost report) provides the total amounts, in the adopted currency units, to be billed during the reporting period. The financial status report may be provided separately.

6.1 Costs for the Reporting Period

This section should provide a table listing the hours worked, rates and costs (hours worked multiplied by rate) for each team member. Total labour costs should be stated.
For ESA contracts, labour costs may have to be calculated in base rates and price variation rates, determined by applying a contractually specified price-variation formula to the base rate.
Non-labour costs should be listed and a total stated.

6.2 Cost to Completion

This section should state the accumulated cost so far and the planned cost to completion of the project and project phase.
A progress chart plotting the cost to completion values reported since the start of the project or project phase may be provided.

6.3 Limit of Liability

This section states:
- The Limit Of Liability (LOL)
- How much has been spent
- How much there is to go.

6.4 Payments

This section should state what payments are due for the reporting period (e.g. contractual stage payments on achievement of a given milestone).

The following documentation DRDs, are provided as examples of tailoring of the documentation templates.

- Element System Specification, this can be used to specify requirements for a flight or ground system or sub-system. It contains the software requirements relevant to both the to-be-implemented functions and the main software design requirements. This documentation DRD is provided just as an example of possible tailoring of the defined system specification for software, in the case software requirements are defined within a system context.

- Software Design Document, instantiated to implement the HOOD design methodology and to include other support methodology.

This page intentionally left blank

36 Software Problem Report

SOFTWARE PROBLEM REPORT	**SPR NO**	
	DATE	
	ORIGINATOR	
1. SOFTWARE ITEM TITLE:		**VERSION/RELEASE NO:**
2. SOFTWARE PROBLEM TITLE:		
3. CRITICALITY: CRITICAL/ROUTINE (underline choice)		
4. URGENCY: URGENT/ROUTINE (underline choice)		
5. SOFTWARE TYPE:		
6.PROBLEM DISCOVERED BY: REVIEW/COMPILATION/TESTING/VALIDATION/OPERATION (underline choice)		
7. PROBLEM DESCRIPTION:		
8. DESCRIPTION OF ENVIRONMENT:		
9. RECOMMENDED SOLUTION:		
1. SOFTWARE REVIEW BOARD DECISION:		
1. ATTACHMENT:		

SOFTWARE PROBLEM REPORT

SPR NO: sequence number of the SPR;

Originator: initials and surname of the originator of the SPR;

Date: when the SPR was written;

1.Software Item Title and Version/Release No: Software CI name and Software CI version/release number;

2.Software Problem Title: clear and descriptive name of the problem;

3.Criticality: underline one of the words as follows
Critical – the software or an essential feature is not available
Non-critical – all other problems
4.Urgency: underline one of the words as follows
Urgent – solution required as soon as possible
Routine – all other problems;

5.Software Type: specify one of the following option adding further peculiar description
GUI, Real Time, Database, Internet/Intranet;

6.Problem Discovered by: underline one the words to indicate how/when the problem was found;

7.Problem Description: describe the problem clearly and concisely, giving the precise context so that the problem can easily be re-created and located;

8.Description of Environment: describe exactly the hardware and software environment;

9.Recommended Solution: suggest a technical approach to fix the problem;

10. Software Review Board Decision: record the SPR’s progress through Software Control Boards;

11.Attachment: indicate attachment to support the SPR.

37 Software Non Conformance Report

SOFTWARE NON CONFORMANCE REPORT	**NCR NO**	
	DATE	
	ORIGINATOR	
1. SOFTWARE ITEM TITLE:		**VERSION/RELEASE NO:**
2. SOFTWARE PROBLEM TITLE:		
3. CRITICALITY: CRITICAL/ROUTINE (underline choice)		
4. URGENCY: URGENT/ROUTINE (underline choice)		
5. SOFTWARE TYPE:		
6. PROBLEM DISCOVERED BY: REVIEW/COMPILATION/TESTING/VALIDATION/OPERATION (underline choice)		
7. PROBLEM DESCRIPTION:		
8. DESCRIPTION OF ENVIRONMENT:		
9. RECOMMENDED SOLUTION:		
10. SOFTWARE REVIEW BOARD DECISION:		
11. ATTACHMENT:		

NON CONFORMANCE REPORT

NCR NO: sequence number of the NCR;

Originator: initials and surname of originator of NCR;

Date: when NCR was written;

1.Software Item Title and Version/Release No: Software CI name and Software CI version/release number;

2.Non Conformance Title: clear and descriptive name of the problem;

3.Criticality: underline one of the words as follows
Critical – the software or an essential feature is not available
Non-critical – all other problems

4.Urgency: underline one of the words as follows
Urgent – solution required as soon as possible
Routine – all other problems;

5.Software Type: specify one of the following option adding further peculiar description
GUI, Real Time, Database, Internet/Intranet;

6. Non Conformance Discovered by: underline one the words to indicate how/when the problem was found;

7.Non Conformance Description: describe the problem clearly and concisely, giving the precise context so that the problem can easily be re-created and located;

8.Description of Environment: describe exactly the hardware and software environment;

9.Recommended Solution: suggest a technical approach to fix the problem;

10. Software Review Board Decision: record the NCR's progress through Software Control Boards;

11.Attachment: indicate attachment to support the NCR.

38 Software Problem Analysis Report

SOFTWARE PROBLEM ANALYSIS REPORT	SPAR NO	
	DATE	
	ORIGINATOR	
1. RELATED SPR(s):		
2. AFFECTED REQUIREMENTS:		
3. CAUSE OF PROBLEM:		
4. RECOMMENDATIONS:		
5. DECISION AND SIGNATURE:		

SOFTWARE PROBLEM ANALYSIS REPORT

SPAR NO: sequence number of the SPAR;

Originator: initials and surname of the originator of the SPAR;

Date: when the SPAR was written;

1.Related SPR(s): title and number given on each SPR covered by this SPAR;

2.Affected Requirements: identify uniquely all requirements to be changed, giving both their number and their titles;

3.Cause of Problem: describe concisely what causes the problem described on the SPR(s), identifying exactly the mechanism involved;

4.Recommentations: state clearly what action you recommend to solve the problem, detailing the options if necessary and making a trade-off between them;

5.Decisi on and Signatures: summarise the action agreed by the Software Control Board; complete the field with each Software Control Board level disposing the SPAR (date and signature as a minimum);

39 Software Change Request

SOFTWARE CHANGE REQUEST	**SCR NO**	
	DATE	
	ORIGINATOR	
1. SOFTWARE ITEM TITLE:		**VERSION/RELEASE**
2. PRIORITY: CRITICAL/URGENT/ROUTINE (underline choice):		
3. CHANGES REQUIRED:		
4. RESPONSIBLE STAFF:		
5. ESTIMATED START DATE, END DATE AND MANPOWER EFFORT:		
6. ATTACHMENT:		

SOFTWARE CHANGE REQUEST

SCR NO: sequence number of the SCR;

Originator: initials and surname of originator of SCR;

Date: when SCR was written;

1.Software Item Title and Version/Release No: Software CI name and Software CI version/release number;

2.Priority: underline one of the words as follows
Critical
Urgent
Routine;

3.Changes required: describe precisely the problem whose solution is addressed by this SCR;

4.Responsible Staff: people responsible for implementing the SCR;

5. Estimated Start Date, End Date And Manpower Effort: self-explained;

6.Attachment: brief description to indicate that a dump, a listing, IO data, a Command Procedure, or other data to support the SCR is attached.

40 Software Release Note

SOFTWARE RELEASE NOTE	**SRN NO**	
	DATE	
	ORIGINATOR	
1. SOFTWARE ITEM TITLE:		**VERSION/RELEASE**
2. CHANGE IN THIS RELEASE:		
3. CONFIGURATION ITEMS INCLUDED IN THIS RELEASE:		
4. INSTALLATION INSTRUCTIONS:		

SOFTWARE RELEASE NOTE

SRN NO: sequence number of the SRN;

Originator: initials and surname of originator of SRN;

Date: when SRN was written;

1.Software Item Title and Version/Release No: Software CI name and Software CI version/release number;

2.Change in This Release: list related SPR/SPAR/SCR/SMR with title, document number, issue, revision, and date as applicable;

3. Configuration Items Included In This Release: Configuration Items number affected by this SRN;

4.Installation Instruction: describe the installation instruction in case of external software delivery.

41 Review Item Discrepancy

REVIEW ITEM DISCREPANCY	**RID NO**	
	DATE	
	ORIGINATOR	
1. DOCUMENT TITLE:		
2. DOCUMENT REFERENCE NUMBER:		
3. DOCUMENT ISSUE/REVISION NUMBER:		
4. PROBLEM LOCATION:		
5. PROBLEM DESCRIPTION:		
6. RECOMMENDED SOLUTION:		
7. AUTHOR'S RESPONSE:		
8. REVIEW DECISION: CLOSE/UPDATE NEEDED/ACTION (underline choice)		

REVIEW ITEM DISCREPANCY

RID NO: sequence number of the RID;

Originator: initials and surname of originator of RID;

Date: when RID was written;

1.Document Title: title of the Document affected;

2.Document Reference Number: number of the Document affected;

3.Document Issue/Revision Number: issue/revision number of the Document affected;

4.Problem Location: page, section, figure, table of the Document affected;

5.Problem Description: describe briefly and precisely what the problem is;

6.Recommended Solution: state how the discrepancy can in practice be remedied;

7.Author's Response: describe briefly and precisely what it should be done with the recommendation action;

8.Review Decision: Close/Update Needed/Action (underline choice): the review team make and underline the decision about the RID.

42 Request for Waiver

REQUEST FOR WAIVER	N.C.R./SPR reference. No.:	DATE: PAGE 139 of 1
3 Title of Waiver:		

4 Other Systems / CL's affected ☐ YES ☐ NO	**5** Recurring Waiver ☐ YES ☐ NO	**6** Baseline Affected YES

7 Documentation affected related to this waiver

DOC NUMBER	ISSUE	REV	DATE	TITLE	APPL. PARAGRAPH

8 Effect on Cost/Price	**9** Effect on Delivery Schedule	**10** Qty	**11** Effect on Performance

12 Description of request	**13 Need /Reason for request**
14 Proposal / Recovery Action	

15	C. I. NUMBER	C. I. NOMENCLATURE	MODEL	**16**
Effectivity				Date on which waiver approval is needed

16 Notes and limitations:

17 AUTHORITY DISPOSITION	CONFIGURATION CONTROL	ENGINEERING RESPONSIBLE	PRODUCT ASSURANCE	PROGRAM MANAGER
Accepted / Rejected	Signature Date	Signature Date	Signature Date	Signature Date
SUPPLIER (THIRD LEVEL). Accepted / Rejected	Signature Date	Signature Date	Signature Date	Signature Date
SUPPLIER (SECOND LEVEL). Accepted / Rejected	Signature Date	Signature Date	Signature Date	Signature Date
SUPPLIER (FIRST LEVEL). Accepted / Rejected	Signature Date	Signature Date	Signature Date	Signature Date

CUSTOMER (AGENCY) Accepted / Rejected	ENGINEERING RESPONSIBLE Signature Date	PRODUCT ASSURANCE Signature Date	PROGRAM MANAGER Signature Date

This page intentionally left blank

APPENDIX A REFERENCES

1. Information Technology Software Life Cycle Processes ISO/IEC 12207:1995.

2. ECSS-P-001 Glossary of Terms June 1997

3. ECSS-M-00 Policy and Principles April 1996

4. ECSS-M-10 Project Breakdown Structures April 1996

5. ECSS-M-20 Project Organisation April 1996

6. ECSS-M-30 Project Phasing and Planning April 1996

7. ECSS-M-40 Configuration Management April 1996

8. ECSS-M-50 Information/Document Management April 1996

9. ECSS-M-60 Cost and Schedule Management April 1996

10. ECSS-Q-80 Software Product Assurance Issue B Draft

11. ECSS-E-40 Space Engineering: Software, Issue B Draft, 3rd April 2000

12. ISO 9000-3:1997 Guidelines for the application of ISO 9001:1994 to the development, supply, installation and maintenance of computer software

13. ECSS-E-70 Space Engineering : Ground Systems and Operations, Part 1: Principles and Requirements, April 2000

14. ISO 9126 Information Technology – Software product evaluation - Quality characteristics and guidelines for their use

15. ECSS-E-00 Space Engineering: Policy and Principles April 1996

16. ECSS-E-10 Space Engineering: System Engineering April 1996

17. BSSC C/C++ Coding Standard, BSSC99(1), Issue 1

18. BSSC Ada Coding Standard, BSSC98(3), Issue 1

19. ECSS-E-40-3 Space Engineering: Ground Segment Software, Issue 1.0 Draft, September 2000

20. ECSS-E40-DRD Software - Document Requirements Definitions Issue 1 Draft 1 May 2000

21. ECSS-Q-00A Policy and Principles April 1996

22. ECSS-Q-20A Quality Assurance April 1996

23. SPEC/TN3 Issue:3.0 Draft A 5 November1999

24. PSS-05-0 Software Engineering Standards, October 1992

25. PSS-05-XX Software Engineering Guides, May 1995

26. ECSS-M-00-02 A Tailoring of Space Standards, April 2000

27. ECSS-M-00-03 A Risk Management, April 2000