BSSC (2002)2 Issue 1.0
14 March 2002

ESA Ground Segment Software Engineering and Management Guide

Part B Management

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 B Management			
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

This page is intentionally left blank

PREFACE

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

This part, Part B, describes the management activities 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 the ESA Software Engineering Standards, ESA PSS-05-0. ESA now applies the European Co-operation for Space Standardisation (ECSS) 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 how to implement the requirements of the relevant ECSS-M and Q series standards on ground segment software projects undertaken by the various parts of ESA. The guide also:

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

The PSS-05 standards covered the development and management of software development projects. The ECSS-M and Q series of standards do not contain space system software engineering requirements; these are defined in ECSS E-40 Space Engineering: Software. Part A, of this Guide addresses these requirements.

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
Attention of Mr M Jones
ESOC
Robert Bosch Strasse 5
D-64293 Darmstadt

Germany

michael.jones@esa.int

BSSC/ESTEC Secretariat
Attention of Mr U Mortensen
ESTEC
Postbus 299
NL-2200 AG Noordwijk

The Netherlands

uffe.mortensen@esa.int

CHAPTER 1
INTRODUCTION

1.1 PURPOSE

This Software Engineering and Management Guide concerns the development and maintenance of ground segment software. This guide covers all aspects of software development for ground segment software including requirements definition, design, production, verification and validation, transfer, operations and maintenance.

This guide is a unified and complete description of how to implement the requirements of ECSS-E-40, ECSS-E-70, ECSS-Q-80 and the ECSS-M standards as concerns ground segment software. The guide also:

- carries over working practices from ESA PSS-05 [Ref 24, 25] and ESA Quality Management System, 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-E-40, Software [Ref. 11], and ECSS-Q-80, Software Product Assurance [Ref. 10], which are themselves based on Information Technology Software Life Cycle Processes 12207:1995 [Ref. 1]. As the software developed according to 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].

Managers, software engineers and assurance specialists applying this guide are thus conformant with the relevant ECSS standards.

1.2 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.

Process	Activities	End of Activity Review
Systems Engineering for Software	System Requirements Analysis	
	System Partitioning	
	System Level Requirements for Software Verification and Validation	
	System Level Requirements for Software Integration	
	System Requirements Review	System Requirements Review (SRR)
Software Requirements for Engineering	Software Requirements Analysis	Software Requirements Review (SWRR)
	Software Top Level Architectural Design	Preliminary Design Review (PDR)
	Software Verification and Validation	
Software Design Engineering	Design of Software Products	
	Coding and Testing	
	Integration	
	Validation Testing(1)	
	Critical Design Review	Critical Design Review (CDR)
Software Validation and Acceptance	Validation Testing against RB subset-1 (Factory Acceptance Tests: FAT(2))	
	Qualification Review	Qualification Review (QR)
	Delivery and Installation	

	Validation Testing against RB subset-2(3) (Preliminary Site Acceptance Tests: PSAT)	
	Validation Testing against RB(3) ((Final) Site Acceptance Tests: SAT)	
	Acceptance Review	Acceptance Review (AR)
Software Operation Engineering	Operational Planning	
	Operational Testing	
	System Operation	
	User Support	
Software Maintenance	Problem and Modification Analysis	
	Modification Implementation	
	Maintenance Review/Acceptance	
	Software Migration	
	Software Retirement	

(1) Can be carried out as Factory Acceptance Tests (2) Also known as Preliminary Acceptance Test

(3) Validation testing against RB subset-2 and against RB are known together as Operational Acceptance Tests

Table 1.1 Processes, Activities and Main Reviewing Tasks

Part C provides proposed templates for documents.

The management processes are associated with the E-40 Process definitions which are summarised in Table 1.1. For further details of these phases please refer to Part A.

Chapter 2 covers software project management. The ECSS-M standards define the requirements to be applied to the management of space projects. Although many of the practices will be directly applicable to ground segment software projects, the standards are tailored by this guide to suit the particular nature of software development. The guidance in this section is primarily derived from:

- ECSS-M-00 Policy and Principles [Ref. 3]
- ECSS-M-20 Project Organisation [Ref. 5]
- ECSS-Q-80 Product Assurance [Ref.10]

- ECSS-Q-00 Quality Assurance [Ref.21]
- ECSS-M-10 Project Breakdown Structures [Ref. 4]
- ECSS-M-60 Cost and Schedule Management [Ref. 8]
- ECSS-M-30 Project Phasing and Planning [Ref. 6].

Chapter 3 covers documentation and configuration management. The purpose of the documentation process is to record information produced by a lifecycle process. This process is concerned with the planning, production, distribution and maintenance of all documentation from the development project. The purpose of the configuration management process [Ref. 1] is to apply administrative and technical control throughout the software lifecycle to:

- Identify, define and baseline software products in a system
- Control modifications and releases of the products
- Record and report the status of the products and modification requests
- Ensure the completeness, consistency and correctness of the products storage, handling and delivery.

The guidance in this section is primarily derived from:

- ECSS-M-40 Configuration Management [Ref.7]
- ECSS-M-10 Project Breakdown Structures [Ref. 4]
- ECSS-M-50 Information/Document Management [Ref. 8]
- ECSS-Q-80 Software Product Assurance [Ref. 10]
- ECSS-M-00 Policy and Principles [Ref. 3].

Chapter 4 covers software product assurance. The purpose of the software product assurance process is to check that the software products and processes in the project lifecycle conform to their requirements and adhere to the established plans, procedures and standards. Product assurance should normally be independent (third party).

The guidance in this section is primarily derived from:

- ECSS-Q-80 Software Product Assurance [Ref. 10]
- ECSS-Q-00 Policy and Principles [Ref. 21]
- ECSS-Q-20 Quality Assurance [Ref. 22]
- ECSS-M-00 Policy and Principles [Ref. 3]
- ECSS-M-40 Configuration Management [Ref.7]
- ECSS-M-30 Project Phasing and Planning [Ref. 6].

Appendix A contains a glossary of acronyms and abbreviations. Appendix B contains a list of references. Appendix C contains document lifecycles for the typical project deliverables. For details of the content of the documents, please refer to E40 Document Requirement Definitions (Ref. 20). Appendix D provides a cross-reference index to the applicable requirements document.

CHAPTER 2
SOFTWARE PROJECT MANAGEMENT

2.1 INTRODUCTION

The ECSS-M standards define the requirements to be applied to the management of space projects. Although many of the practices will be directly applicable to ground segment software projects, the standards are tailored by this guide to suit software development for ground segments.

This guide has been prepared on the basis that work is normally done using a Firm Fixed Price (FFP) approach

2.2 THE SOFTWARE DEVELOPMENT PLAN

The customer shall specify the project management requirements in the system specification, and the supplier shall respond with a software development plan (part 1 of the technical specification) addressing:

- Project organisation
- Project phasing and planning
- Project work breakdown structures
- Cost and schedule management.

2.3 PROJECT MANAGEMENT TASKS

2.3.1 Organising the Project

All the work on the project shall be covered by a business agreement between the customer and supplier. A business agreement is either a contract or an internal agreement if the customer and supplier are part of the same organisation.

2.3.2 Tailoring

2.3.2.1 Introduction to tailoring

This guide describes the software engineering processes to be applied to all deliverable software products developed for ground segments. The processes are characterised in terms of activities and tasks. The tasks contain the requirements to be applied. As described in section 2.2 of Part A, the tasks are organised as mandatory practices, recommended practices and guidelines.

The tailoring process is the deletion of non-applicable processes, activities and tasks [Ref. 1]. The addition of unique or special processes, activities or tasks is permitted, as specified in the contract between customer and supplier, in the following referred to as “the Contract”.

This guide constitutes a tailoring of the ECSS-E-40 [Ref. 11], for applicability to the development of ground segment software, in accordance with the requirements for the selection and tailoring given in ECSS-M-00 [Ref. 3].

This guide, therefore, provides a tailoring framework for ground segment software development to be applied by customers. Within this framework, further tailoring of this guide can be undertaken.

All tailoring of this guide shall be made by the customer, in his request for proposal. The supplier shall describe how he intends to comply in his response.

2.3.2.2 Guidelines for tailoring this guide

The general guidelines for tailoring the standards applicable to a ground segment project are contained in ECSS-M-00 [Ref. 3]. The following is a summary of the requirements, as applicable to this guide.

Tailoring of this guide to specific project requirements shall be carried out according to a number of criteria, such as:

- The overall project risks, their criticality and their consequences
- The project category.

The project categories specified in Table 2.1 shall be considered by the customer to identify a framework appropriate for the project.

Category	Definition
1	Loss of mission would be unacceptable. The allocated budgets and development schedules shall be sufficient to obviate major technical deadlocks. This category normally does not apply to ground segment software.
2	A project aimed at achieving overall control of project risks. Project risk/total cost compromises that minimise risk are sought.
3	A project aimed at achieving overall containment of cost. Project risk/total cost compromises that minimise cost are sought. The level of accepted risk is higher.
4	A minimum cost project. The mission is only worth while if its cost is kept down.

Table 2.1 Project Category Definitions

The management and product assurance standards (the subject of this part of the guide) are the most likely to require tailoring. There will be relatively little effect on the selection of software standards, as the same basic engineering processes should be exercised in all cases [Ref. 15]. It is the level of detail to which the processes descend which is likely to be modified. For this reason, the tasks in this guide are defined in terms of mandatory, recommended and guideline practices.

In order to assist the tailoring process for a particular project, the following general characteristics are used to classify ground segment components. Table 2.2 gives additional guidance on the type of projects most appropriate to the categories.

Type	**Typical Elements**
On-line Operational	Mission Control System Ground Communication Subnet Ground Station System
Off-line Operational	Flight Dynamics Mission Planning Spacecraft Performance Payload Processing
Prototype	Simulators Study Software

Table 2.2 Classification of Ground Segment Components

Care must be taken with these categories, as their context may change at different points during the mission.

2.3.3 Management Interface

This guide has been prepared on the basis that work is normally done using a firm fixed price approach

If required the supplier shall provide the customer access to his facilities and relevant data, for audit, inspection, inquiry, or other exceptional events not covered by routine management interaction, within the framework of the contract.

The supplier shall consent to and support assessments and/or audits by the customer or by a third party agreed between the customer and the supplier. He shall co-operate with the assessors. Assessments shall result in a report prepared by the assessor containing the views of both parties.

Both the customer and supplier should define and implement an action monitoring system.

2.3.3 Roles, Responsibilities and Authority

Both customer and supplier shall appoint a project manager who shall have the authority to represent his organisation and make, or obtain all the decisions necessary.

The supplier shall ensure that the project organisation including internal and external interfaces is defined and documented. The definition of the organisation shall include responsibilities and authorities of key roles, and names of nominated staff in key roles.

The roles, responsibilities and authority of consultants or other specialists employed should be clearly defined.

The supplier should staff the project according to the personnel skills needed, develop team member skills by appropriate training and foster continuous improvement.

The customer may delegate all or part of the customer's prerogatives to a third party.

Where a supplier has lower level suppliers, he shall act as a customer.

The customer and supplier project Managers shall both define the objectives and priorities at each stage. They shall document the assumptions, dependencies and constraints that influence their decisions in the software development plan.

2.3.4 Risk Management

The supplier shall define the risk management policy in the software development plan taking into account project specific constraints. The policy shall define:

- How the probability of a risk occurring will be evaluated
- How the possible risk consequences will be evaluated
- How the risk acceptability will be assessed (i.e. at what point is an identified risk considered sufficiently small not to require containment or mitigation actions)[1]
- Who within the supplier team shall be owner for risks
- The interval at which risks should be re-assessed.

The normal approach is to report on risk status in the progress report.

The supplier shall provide an initial risk register in the software development plan, which shall:

- Identify each risk
- Identify the individual who owns the risk
- Assess its probability of occurrence
- Evaluate the consequences

---

[1] All risks should be identified in case the probability of occurrence or the consequences change later in the project

- Assess the acceptability of the risk
- Decide to accept or take action to reduce the likelihood or the severity of the consequences
- Monitor each risk.

The supplier shall review the risk register at the defined intervals in order to

- Update changes in risk probability or consequence
- Verify the effectiveness of all implemented actions and
- Identify new risks.

All significant changes in the risk register shall be reported to the customer.

The product assurance function shall contribute to the overall risk management activities by:

- Supporting the identification and risk evaluation of critical products for which major difficulties or uncertainties are expected in:
    - ♦ Demonstration of design performances
    - ♦ Development and qualification of new product, processes and technologies
    - ♦ Procurement, specification, design, coding, verification, validation and storage
    - ♦ Product utilisation or service implementation.
- By identifying the PA activities accompanying the individual risk reduction measures
- Monitoring and documenting the achievement of the specified risk reduction implementation and the corresponding verification measures throughout all project phases.

2.3.5 Technical Management

There are many methods and tools that can be applied throughout the software life cycle, which can greatly enhance the quality of the end product and their use is strongly recommended. Project management is responsible for selecting methods and tools, and for enforcing their use.

Technical management includes organising product assurance, software configuration management, and verification, validation and test activities.

The customer shall specify the minimum requirements to achieve the project objectives in the requirements baseline. The supplier shall produce a set of documents that form the technical specification, and demonstrate its compliance with the requirements baseline.

Technical Specification Section	Requirements Baseline Subject	For Further Details Refer to
Software Development Plan	Management Requirements	Section 2.2
Software Configuration Management Plan	Configuration Management Requirements	Section 3.5
Software Product Assurance Plan	Product Assurance Requirements	Section 4.5
Software Verification Plan	Verification Requirements	Part A Section 5.3.3
Software Validation Plan	Validation Requirements	Part A Section 5.3.3
Software Maintenance Plan	Maintenance Requirements	Section 2.4.4
Software Operations Plan	Operations Requirements	Section 2.4.4
Interface Control Document	Interface Requirements	Part A Section 5.3.1.1.2

Technical Specification Section	Requirements Baseline Subject	For Further Details Refer to
Software Requirements Specification	Software Requirements	Part A Section 5.3.1

Table 2.3 Content of the technical specification

Table 2.3 summarises the content of the technical specification. Note that the requirements baseline does not restate requirements from the ECSS standards and/or this Software Engineering and Management Guide, all it does in this context is to identify any tailoring or additional requirements.

The supplier shall provide the following in the software development plan:

- A product tree (i.e. a full set of project products and their inter-dependencies, (for example Client Software, middleware, business software, database software, test harness etc..), which shall be approved by the customer and maintained up to date under configuration control
- A work breakdown structure, maintained up to date giving for each work package, a unique identifier, its work contents, inputs, and outputs
- A price breakdown defining price elements in terms of agreed price categories (e.g. manpower, materials, travel...) and providing the framework for price summarisation
- If applicable, a business agreement structure i.e. which sub-contractor is performing which work packages.

2.3.6 Cost Management

The price shall be based on the work breakdown structure. A price should be associated to each work package broken down

by price category (e.g. manpower, missions, supplier overheads, materials etc.).

Prices may be refined using a stepwise approach, where an estimate of the overall cost is made, providing a budget for each of the phases and a firm fixed price for the first phase (if a phased lifecycle is used). The actual prices for subsequent phases are then produced before their start.

The contract shall include payment milestone plans that specify payment events, and associated payments. The conditions for the payment shall be clear. The payment milestone definition may also include penalised values (i.e. amount of penalty in case of late delivery) and associated conditions.

The prices are defined in the contract. Any change in baseline will require a contract change notice, or other appropriate mechanism.

2.3.7 Schedule Management

The customer shall define a set of project milestones. The supplier shall define a schedule to meet these milestones. The milestones will include payment milestones

The project schedule should

- Be linked to the work breakdown structure, showing all major milestones (including contractual ones) and required deliverables
- Establish task duration and planning by allocating appropriate resources; the tasks will correspond to the work packages, or a breakdown thereof.
- Include schedule margin contingency.

Both customer and supplier shall notify each other immediately of events that could significantly affect the achievement of schedule objectives (in particular payment milestones).

2.3.8 Reporting Project Progress

The supplier's project manager shall regularly report on work progress to his customer based on work packages and the corresponding schedules. This should be done based on the frequency and format defined in the management requirements or the contract. The reporting cycles depend on the type and size of the project. Typical reporting cycles are every month for development projects and once every 2 months for studies (a reporting cycle determines how often progress reports are delivered and progress meetings are held).

Progress reports should include:

- Assumptions and any changes in these assumptions or new assumptions arising during the reporting period
- The evolution of risk (see section 2.3.4)
- Progress on each current work package
- A report on schedule status including indication of any deviation (milestone trend charts can be used to illustrate this)
- A deliverable items list
- A report on the status of resources (if applicable)
- Work foreseen for the next reporting period
- A report on the configuration status (see Section 3.4.5)
- A report on product assurance status (see Section 4.2.4)
- Status report which could include optional report on expenditure of effort and/or changes in key personnel.

Progress meetings should be held at regular intervals. Special meetings may be held in the event of major deviations from the baseline plan, or occurrence of unforeseen events affecting planning.

2.4 PROJECT MANAGEMENT AND THE SOFTWARE LIFECYCLE

Each planning document exists in outline from the start of the project. It is updated prior to the start of each phase to provide increasing levels of detail, so that each phase is fully planned prior to its commencement. For a summary of documents in the lifecycle see Appendix C.

Please note that for ground segment Software it is unusual for a contract to cover the entire development. It is more usual to have one or more processes contracted separately based on the knowledge obtained from earlier processes.

2.4.1 System Engineering

During Systems Engineering the requirements baseline is developed including:

- Management requirements
- Maintenance requirements
- Operations requirements.

The supplier's offer will include a draft software development plan providing a draft of the overall plan covering all the activities of the development with a full description of the processes that will be carried out by the supplier.

For ground segments the processes of ground segment Software Maintenance and Operations are normally handled separately from the software development. At an agreed date in the project, the supplier (or different supplier(s) if Software Maintenance and Operations have been awarded to a different supplier) should respond with

- The software maintenance plan
- The software operations plan.

The system requirements review shall include an assessment of the software development plan, and if applicable and available

the software maintenance plan and outline software operations plan to ensure that they are able to meet the requirements

2.4.2 Software Requirements Engineering

At the end of the process the supplier shall provide detailed plans for the architectural design activity based on the knowledge of the software requirements, and shall update the overall software development plan in the light of detailed plans and a better understanding of the requirements

The Software Requirements Review (SWRR) shall include an assessment of the updated software development plan to ensure that it complies with the management requirements.

Note that it is possible for the supplier of the Software Requirements Engineering process to be part of the customer organisation.

2.4.2.1 Top level architectural software design

At the end of the process the supplier shall provide detailed plans for Software Design Engineering based on the knowledge of the architectural design and update the overall software development plan in the light of detailed plans and a better understanding of the design.

The preliminary design review shall include an assessment of the updated software development plan and ensure that it complies with the management requirements.

2.4.4 Software Design Engineering

As the design work proceeds to lower levels, the WBS and schedule should be refined to reflect this.

As an output from this process the supplier shall provide detailed plans for the Validation and Acceptance process and update the overall software development plan in the light of detailed plans and the known detailed design.

The supplier shall also supply draft software operations and maintenance plans.

The critical design review shall include an assessment of the updated software development plan, draft software maintenance plan and draft software operations plan to ensure that they meet the management requirements.

2.4.5 Validation and Acceptance

At the end of technical validation, the supplier shall hold a Qualification Review (QR), chaired by the customer.

The supplier shall prepare final software maintenance and software operations plans.

The Acceptance Review (AR) shall include an assessment of the software maintenance and operations plans to ensure that it is capable of meeting the requirements.

At the AR the supplier shall deliver an end–of–project assessment of the lessons learned.

2.4.6 Software Operations Engineering and Maintenance Processes

These processes cover pre-launch operations, launch, and in-flight operations (see Part A Section 2.7.5)

The software operation and maintenance plans shall be kept up to date based upon experience from operations.

The customer shall prepare the software retirement plan, detailing the end of operations and system disposal.

The customer shall carry out the retirement process in conformance with the software retirement plan. Software retirement is the equivalent of the hardware disposal, and covers the removal of a software product from service (see Part A section 9.3.5).

It is recommended to hold a review of the software retirement plan in conjunction with the ECSS-E-70 [Ref 13] Mission Completion Operation Review (MCOR).

This page is intentionally left blank

CHAPTER 3
CONFIGURATION MANAGEMENT

3.1 INTRODUCTION

This chapter covers configuration and document management.

The purpose of the configuration management process [Ref. 1] is to apply administrative and technical control throughout the software lifecycle to:

- Identify, define and baseline software products in a system
- Control modifications and releases of the products
- Record and report the status of the products and modification requests
- Ensure the completeness, consistency and correctness of the products
- Storage, distribution handling and delivery of the products.

3.2 CONFIGURATION MANAGEMENT PROCESSES

3.2.1 Configuration Identification

A Configuration Item (CI) is an aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process. A configuration item may be any kind of software product, for example: a product, a component, a unit, a document, or a set of CIs.

The term version is used to define a stage in the evolution of a CI. Each stage is marked by a version number. When the CI changes, the version number changes. The identifier of a CI shall include a version number.

Each configuration item shall be allocated, and marked with a unique identifier that is traceable to its technical description.

The configuration item version number shall include an issue number and a revision number. Issue numbers are used to mark major changes and revision numbers are used to mark minor changes. Major changes usually require formal approval. The issue number and revision number together mark the version of the CI.

The configuration identification method shall be capable of accommodating new CIs, without requiring the modification of the identifiers of any existing CIs.

3.2.2 Choosing Configuration Items

A distinction should be made between CIs which are subject to configuration management (only) which involves effective identification, storage and retrieval, and configuration control which comprises CIs subject to configuration management plus change control.

The following items shall be under configuration management:

- Software requirements verification report
- Software requirements traceability matrix report
- Software architecture and interface verification report
- Software design verification report
- Software code verification report
- Software documentation verification report
- Software integration verification report
- Software unit test report
- Software integration test report
- Software validation test report
- Test specification evaluation report

- Software design and test evaluation report
- Software budget report
- Software product assurance reports.

The following items shall be under configuration control:

- System specification
- Interface requirements document
- Software development plan
- Software configuration plan
- Software product assurance plan
- Verification plan
- Validation plan
- Software maintenance plan
- Operations plan
- Interface control document
- Software requirements specification
- Software design document
- Software validation testing specification
- Software integration plan
- Software unit test plan
- Software user manual
- Change justification file
- Migration plan
- Retirement plan
- Any other document that describes a software component
- Customisable parts of code generation tools

- Source code
- Executable code
- Tools
- Test software and data
- Configuration and set up instructions for packages and tools
- Software configuration file.

Not all CIs need to be placed under configuration control with the disciplines and costs that this implies; for example it may be decided that test harnesses should not be placed under configuration management.

3.2.3 Configuration Item Content

The delivery format for configuration items shall be specified and shall conform to international standards (e.g. PDF) where available.

Configuration items shall be prepared to an agreed standard format and structure, particularly where multiple location information retrieval is planned.

Each document shall carry objective evidence of its current category, i.e. for approval, for agreement, for information, etc. and shall be verified before being made available via the information system.

All customer and supplier project staff shall, in a timely manner, be informed of the existence of, and have access to, items they need for the project.

3.2.4 Configuration Item Storage and Distribution

The supplier's software configuration management system shall store software, document or data in a manner that:

- Enables speedy retrieval

- Provides access control according to the level of confidentiality
- Protects against foreseeable hazards (corruption, fire, flood, viruses)
- Identifies documents unambiguously, including issue state
- Makes any version of an item available for the period of the business agreement
- Ensures the retrieval of the correct release using version verification (see section 3.2.6).

The access rules to information shall be specified

The customer will define in the requirements baseline the means by which information flows in a project. The customer should not impose changes on existing suppliers information systems unless the project information system performance or interface requirements cannot otherwise be met. Regular training should be provided in the use of the system. The project information systems shall be an integral part of the project management process. Care should be taken to avoid duplication of information

3.2.5 Baselines

A baseline is a set of configuration items that has been formally reviewed and agreed upon, and is a basis for further development. Formal change control procedures are required to modify a baseline.

Software integration strategies should make use of baselines to track the inputs and outputs of the integration process. The following description illustrates the relationship between units of modules, baselines and releases. Modules, after successful unit testing, are integrated into existing baselines. Incorporation into a baseline can only occur after successful integration tests. Baselines must be validated before being transferred to users as a ‘release’ of the software.

Releases will be of three categories depending on the impact the release has on the end user:

- Major, where significant new functionality has been added
- Minor, where minor functionality additions and/or solutions to problems have been incorporated
- Patch, where a solution to specific problem is provided without re-delivery of the whole system.

After delivery and installation, releases of the software undergo acceptance testing by users.

3.2.6 Forming and Maintaining Baselines

For each software product the supplier shall define a list of software configuration items that make up the software product.

The total list of configuration item identifiers for a product will form a baseline for that product.

The supplier shall submit the baseline in the software configuration file for the product to the customer for approval at each Review, and maintain it through the development.

The customer shall ensure that the baseline is maintained during product maintenance. The customer may delegate this responsibility to a supplier.

The software configuration file provides a definition of the configuration of the Software at each Milestone and release. It includes

- The category of the release (see below)
- An inventory of materials released (e.g. media etc)
- An inventory of software contents forming the baseline (documents and files) including for each Item:
  - Scope
  - Intended distribution

  - An indication if the document has to be kept in hard copy for legal reasons
  - A complete design description for developed products
  - Procurement specification or a list of performances and interface characteristics if it is a COTS product.
- A description of the software baseline and any changes to it since the last release
- A description of the changes to the software
- A description of installation and configuration of the software on the site
- A description of known or potential problems.

The software configuration file shall be up to date and available at each project milestone.

For each release, documentation and code shall be consistent. The policy for retaining old releases should be determined by the customer project manager, depending upon the situation. Where possible, the previous release should be kept online during a change-over period, to allow comparisons, and as a fallback. Older releases may be archived. Only one release should be in operational use at any one time, although there may be circumstances where this is not possible (e.g. a mission uses an old version of re-usable software, which cannot be upgraded to a newer version).

Some form of software version verification capability is recommended for controlled source and binary code to avoid use of incorrect releases. The strength of this protection depends on the criticality of use of the product. In general, each release should be self-identifying (e.g., operator dialogue or printed output) or preferably self-verifying (e.g. checksum).

The software configuration file shall be approved

- At the end of software requirements review

- At the end of the critical design reviews
- At the qualification review and acceptance review
- At each subsequent release of the software.

The supplier shall deliver configuration status reports in the project progress reports (see Section 2.3.8). These shall include:

- Deliveries made with version identification
- SPR status and statistics for delivered version.

The configuration management system shall take into account the following aspects of software developed for re-use. Software developed for re-use will normally be under independent CM from the projects it serves for these reasons:

- Longer lifetime of the components developed for re-use compared to the other components of the project
- Evolution or change of the development environment for the next project that intends to use the components
- Transfer of the configuration and documentation management information to the next project.

3.2.7 Media

Media delivered to the customer shall be marked with

- The software name
- The version number
- The reference to the software configuration file.

3.2.8 Configuration Change Control

As a configuration item passes through unit, integration, system and acceptance tests, a higher level of authority is needed to approve changes. This is called the promotion of a CI. Just as programmers sign off unit tests, team leaders sign off integration tests, project leaders sign off system tests, and the customer signs off acceptance tests.

A formal procedure shall be established to document requested changes to agreed baselines.

A Configuration Control Board (CCB), under the control of the customer normally controls the change process once the product has been accepted. The CCB should comprise representatives of the end user and supplier

The proposed change shall then be subject to an evaluation made by the CCB in a formal process to which each of the parties involved has to contribute.

Such changes shall be reported in accordance with business agreement requirements and shall have formal or contractual back up.

Where necessary the supplier may request permission to waive requirements. In addition permission for non-conformances to processes (e.g. not to have completed 100% branch coverage in testing) may be requested. If requested, the supplier shall assess the impact of the waiver request. The impact analysis shall be made available to the CCB so that a decision can be made on permission for non-conformance to process requirements.

All physical and logical interfaces shall be maintained in an interface control document under Configuration Control, and changes to that interface controlled by procedures that involve all actors who contribute to that interface.

Changed documents require the same authorisation and circulation requirements as the original document

The supplier shall ensure that all changes are authorised and implemented according to the software configuration management plan

3.3 THE SOFTWARE CONFIGURATION MANAGEMENT PLAN

The supplier shall prepare a software configuration management plan that describes the organisation, methods, means and procedure implemented to manage the configuration in accordance with the project requirements baseline for configuration management.

The supplier's software configuration management plan shall:

- Exist for each phase of a project but be appropriately scoped to the items to be controlled for that phase
- Specify the procedures employed for approval or agreement of documents
- Define the related document management activities
- Define the process for receipt, assessment, authorisation and distribution of documents.

A product's configuration baseline shall be identified by approved documents. The establishment of a configuration baseline determines the starting point for a formal evolution procedure for products and configuration documents.

3.4 CONFIGURATION MANAGEMENT AND THE LIFE CYCLE

The following table indicates when key documents shall be brought under configuration control. For a summary of documents in the lifecycle see Appendix C.

Document	Review at Which Document is Brought under Configuration Control
Requirements Baseline Documents	SRR

Document	Review at Which Document is Brought under Configuration Control
Technical Specification Documents	PDR
Design Definition File Documents	CDR
Design Justification File Documents	Review for Which they Provide Evidence
Production Master File	CDR
Product Assurance File Documents	Review for which they provide evidence
Production/Maintenance File Documents	Qualification Review

Table 3.1 Key documents under configuration control

This page is intentionally left blank

CHAPTER 4
SOFTWARE PRODUCT ASSURANCE

4.1 INTRODUCTION

This chapter covers software product assurance. The purpose of software product assurance is to provide adequate assurance that the software product and processes in the project lifecycle conform to their specified requirements and adhere to the established plans.

Product assurance are met by measuring and controlling the quality of:

- The product indirectly, by using metrics
- The process directly, by ensuring documented and verified processes are used, and that a system of process improvement is employed.

The direct control of product quality is the process of Software Validation and Acceptance, as described in Part A.

Product Assurance may be internal or external depending on whether evidence of product or process quality is demonstrated to the management of the supplier (internal) or the customer (external).

4.2. PRODUCT ASSURANCE MANAGEMENT

4.2.1 Organisation

A software product assurance manager shall be appointed from the product assurance organisation. He shall report functionally to the project manager but shall have unimpeded access to the head of the product assurance organisation, and shall have sufficient authority and independence to perform this role.

The supplier shall produce a software product assurance plan, approved by the customer, which addresses all the product assurance requirements. The software product assurance plan shall be maintained throughout the software lifecycle, and shall be updated before each milestone to ensure that product assurance is fully defined for the next phase.

4.2.2 Reviews and Audits

The supplier shall plan and undertake audits of his own activities, using established and maintained procedures, to ensure compliance with, and effectiveness of the software product assurance plan. These audits shall be conducted by personnel not directly involved in the work being performed.

Audits shall be performed on a planned regular basis at a frequency appropriate to the size and criticality of the project. Where failure, consistent poor quality, or other problems become apparent in the project, extra unplanned audits shall be performed.

4.2.2.1 Auditing of secondary suppliers (sub contractors)

The supplier shall regularly maintain and provide the customer with a plan for the auditing of secondary suppliers on the project. The customer has the right to attend these audits, and shall be given notice of their scheduled occurrence. In addition the customer may conduct his own audit of secondary suppliers at any time, such audits shall be arranged by the supplier and the next/higher level of the audited supplier as relevant.

4.2.3 Training

The supplier shall

- As part of initial planning, conduct a review of the project requirements to establish and make timely provision for acquiring or developing the resources and skills required by the management and technical staff

- Establish a documented training programme for all personnel whose performance determines or affects product quality
- Maintain training records.

4.2.4 Product Assurance Reporting

The supplier shall report on the status and progress of product assurance activities using the same cycle as other management progress reporting, including an assessment of the current quality of the product, based on:

- Verification activities performed
- Problems reported and resolved
- Measurements of the metrics as defined in the software product assurance plan.

4.3 PRODUCT ASSURANCE

This section deals with the assurance of the product by

- Defining product quality objectives and metrication
- Defining product standards and procedures
- Ensuring product dependability and safety.

4.3.1 Product Quality Objectives and Metrics

The supplier shall define assurance activities in the software product assurance plan to ensure that the product meets the quality requirements as specified in the technical specification. Quality models shall be used to specify the quality objectives, which shall be derived from the reliability, safety, maintainability and quality requirements of the system. Quality models are defined as the set of characteristics and the relationships between them, which provides the basis for specifying quality requirements and evaluating quality

To achieve these objectives the supplier shall apply rules on design, code and documentation and define a set of metrics as a means to verify and prove that the requirements are met. The metrics shall provide the means of assessing the quality characteristics of the product against the quality requirements. Metrics shall be collected throughout development, and used to define corrective action as required. The results shall be reported to the customer through software product assurance reports (see Ref. 20) and used to provide an insight into the level of quality obtained.

An example of a quality model is the one defined in [Ref. 23]. This quality model is based on three inter-dependent concepts:

- Property: a characteristic of the software that contributes to its overall quality (for example reliability). Properties may require special evaluation methods in order to achieve evaluation.
- Metrics: the defined measurement method and the measurement scale (for example MTBF) metrics are used for evaluating each property
- Evaluation method: a technique or methodology that, if applied, allows the evaluator to find data for calculating the metric value (for example failure data collection). Evaluation methods may be required for providing data for calculating metrics

The following is a list of possible metrics:

- Size (design, code)
- Complexity (design, code)
- Fault density (the number of SPRs per 1000 lines of code)
- Failure intensity (the percentage of Critical SPRs compared to all SPRs)
- Test coverage (percentage lines of code covered in tests)
- Number of observation or anomaly reports
- Number of SPRs (See Section 4.4.10).

The supplier shall regularly:

- Collect metrics
- Store them
- Analyse them against target values and/or quality objectives
- Report them
- Take remedial action as required to ensure conformance to quality requirements.

Where numerical accuracy is important (e.g. for an attitude and orbit control subsystem) the supplier shall define rules on design and coding to ensure that the specified level of accuracy is obtained. Numerical errors shall be estimated and checked during development.

4.3.2 Procedures and Standards

Development procedures and project standards shall be put in place for all types of software on the project to cover:

- How metrics are collected and recorded
- How data is analysed to identify statistics and trends in software behaviour (e.g. failures, corrections, duration of runs)
- How to determine whether further actions are needed to improve the software
- How the functional criticality of software is classified (see Part A section 3.3.1.2)
- Program design language standard, if used
- Coding standards [Ref. 17][Ref. 18].

All procedures and standards shall be reviewed by all types of their users to ensure suitability, and shall be in place prior to the start of the activity in which they are used.

4.3.3 Software Dependability and Safety

The supplier shall conduct an analysis of software criticality. For further details see Part A. Section 3.3.1.2

The supplier shall ensure that failure of non-critical software, which is not subject to the above-stated assurance measures, does not cause failure of critical software.

4.4 PROCESS ASSURANCE

4.4.1 Process Improvement

The supplier shall implement and document a system to improve his software processes by learning from experience of their use. An assessment procedure shall be put in place that:

- Collects data on the process quality
- Collects data on the product quality (as defined in section 4.3.1)
- Analyses the data to reveal strengths and weaknesses of the process
- Identifies process improvements to address these weaknesses
- Feeds back improvements into the relevant procedural documentation.

Records of the process improvement shall be kept.

The metrics to be applied are further detailed in section 4.4.9.

4.4.2 Sub-Supplier Control

The experience and record of suppliers are key factors in assessing the risks to a project. Indicators of maturity are the:

- Track record of successful development of similar systems

- Use of software engineering standards
- Up to date ISO 9000 certification with a scope statement that is appropriate to their role in the project
- Existence of a software process assessment and improvement programme (CMM, SPICE, ISO15504).

Experience of developing similar systems is an essential qualification for a project team. Lack of experience can result in poor estimates, avoidable errors, and higher costs (because lower productivity and/or extra work is required to correct the errors).

Standards not only include software development standards but also coding standards and organisational standards. Standards and procedures may exist within an organisation but ignorance of how best to apply them may prevent their effective use. Project managers should ensure that standards and/or procedures are understood, accepted and applied. Project managers may require additional support from software quality assurance staff to achieve this.

A software process improvement programme measures process maturity. Such a programme should be led by a software process group that is staffed with experienced software engineers.

The supplier shall maintain procurement records from sub-suppliers, and shall furnish the content of these records to the customer on request.

If a supplier is to delegate any product assurance function to a sub-supplier he shall:

- Require the sub-supplier to implement a software product assurance plan addressing all the relevant product assurance controls

- Monitor the sub-supplier for compliance with the product assurance requirements including the review and approval of the sub-supplier's software product assurance plan, the continuing verification of process products and the monitoring of the final validation of the productEnsure that the subcontracted software complies with the applicable dependability and safety criticality requirements.

4.4.3 Purchasing COTS Software

The customer shall identify the acquisition approach (such as IEEE Standard 1062) for projects where the use of COTS/OTS/MOTS is intended.

The criteria for selecting COTS are defined in Part A Section 10.3.3. All COTS software that is selected shall be registered in a component list in the design justification file for customer approval. The list shall:

- Describe the component
- Specify the ordering criteria (versions, options, extensions, service packs, patches, etc.)
- Specify receiving inspection criteria (e.g. test all systems affected or those using the COTS)
- Specify arrangements for maintenance and upgrades to new releases (e.g. maintenance contract, testing required for patches/upgrades)
- Specify back-up solutions if the product becomes unavailable
- Specify licensing arrangements.

The COTS item shall be subjected to configuration control.

On receipt of the COTS item, the supplier shall conduct the specified receiving inspection, and generate a report that shall detail any problems encountered.

4.4.4 Re-use of Software

Software re-use is discussed in Part A chapter 10.

The supplier shall finalise investigating the use of existing software and software design (e.g. architectural design). The software to be re-used may include software which has been previously developed by the supplier, for which the supplier holds the Intellectual Property Rights (IPR), in which case the software should be compatible with product assurance requirements. The investigation may refine earlier work, and define the associated actions required to meet quality requirements. The investigation shall include:

- A check of the validity of reusing the original development tools and methods
- Identification of any adaptation required
- A check of the level of validation of all components reused
- A check of the documentation status
- A check of the quality status (non-conformances, waivers, etc)
- A certification from the supplier that all the relevant tests have been carried out on the required platform.

The results of the investigations shall be recorded in the software re-use records (part of the design justification file), which shall contain:

- A summary of the re-used components and an assessment of the level of re-use
- Assumptions made and the method used to perform the assessment
- Adaptations and/or corrections required to re-use the component.

The supplier shall adopt the same standards for re-used software as for procured non-COTS software.

If the levels of product assurance of the re-used software do not meet the product assurance requirements then the supplier may:

- Analyse software life cycle data from the previous development to ensure the adequacy of the development processes
- Reverse engineer to generate required software documentation
- Use the product service history which includes information concerning:
  - Configuration management and change control of the software product
  - Effectiveness of problem reporting records
  - Stability and maturity of the software as shown in the change control records
  - Relevance for the product service history for the new environment (for example most errors occurred in areas not used in new target environment, or slow performance not relevant as a faster processor is now available)
  - Actual error rates and maintenance records
  - Impact of modifications.

If this analysis leaves doubts about the fitness of the software for present purposes, the supplier may agree with the customer additional verification activities to permit confidence in re-use of the software.

Based on the re-use records, the customer shall approve the reuse of the software.

The supplier shall separate information related to components developed for reuse in the quality assurance plan, the technical specification (including requirements for maintainability,

portability and verification of re-used components), and the design justification file.

4.4.5 Tools, Techniques and Methods and Supporting Environment

The supplier shall select and procure hardware for the development environment taking into account:

- Performance
- Maintenance
- Durability
- Consistency with the operational equipment
- Availability of adequate support documentation
- Acceptance and warranty conditions
- Conditions of installation, preparation, training and use
- Maintenance conditions, including the possibilities of evolution.

The supplier shall select development methods and tools taking into account:

- Maturity
- Compatibility of the methods and tools with the development team's experience and training
- Suitability of each tool to support the concerned lifecycle activity
- The availability of the tool throughout the lifecycle of the product.

The supplier shall ensure that the software development environment required for each process of the lifecycle is set up before the start of that process.

If automatic code generation is to be used the supplier should address the following questions:

- How will the tool evolve in relation to other tools such as compilers?
- Can the tool be customised to comply with project coding or other standards?
- Is the generated code sufficiently portable?
- How will the required metrics be collected?
- How will the software be verified? (The software needs to be verified to the same level as manually generated code)
- How will configuration control be applied, including control of customisation input parameters?

4.4.6 Hardware Environment for Operational System

Software and Hardware may be procured separately, or alternatively they may be procured as a turnkey system

In the latter cases, the supplier shall choose the hardware taking into account

- Constraints of both development and use
- Performance
- Ease of purchase/ market availability
- Growth capability
- Compatibility (e.g. with the customer's hardware platform policy and /or environment)
- Maintenance and durability.
- Reliability.

The hardware maintenance policy and ease of maintenance proposed by the manufacturer, including any substitutions, should be compatible with the specified service life and operational constraints.

The supplier shall justify the choice of ground computer equipment in the software product assurance plan.

4.4.7 Process Metrics

The supplier shall document in the software product assurance plan the software metrics to be applied to manage the development and to assess the quality of the development process used. Metrics shall be collected, stored, analysed, reported in the regular software product assurance reports, and submitted to each milestone from the start of design to the completion of acceptance.

The following metrics shall be applied:

- Actual duration of phases/tasks compared to plan
- Effort used in phases/tasks compared to the plan
- Number of SPRs generated during verification (i.e. review or audit) (this metric shall be reported to the customer)
- Number of SPRs generated during integration and validation testing and use (this metric shall be reported to the customer).

4.4.8 Verification and Validation

All verification activities shall be specified in the verification plan (part of the design justification file). All Validation activities shall be specified in the validation plan (part of the design justification file).

The supplier shall verify the outputs of each activity against its inputs to ensure that they:

- Conform to the appropriate standards
- Contain (or reference) acceptance criteria before proceeding to the next dependent activity
- Identify product characteristics that will ensure its safe and proper functioning. (e.g. margin philosophy of the computing resources or performances of operating systems on which the application i runs).

The supplier shall include a summary of verification activities and results in software product assurance reports.

The supplier shall record all verification results, referencing software problem reports generated and further actions required, and shall monitor completion of all actions.

In the case of software containing deactivated, or configurably activated code, the supplier shall verify either that it cannot be accidentally activated, or that any such activation cannot harm the system or its operation.

The supplier shall test all different configurations of the software. For certain types of software e.g. generic or infrastructure software, it may not be possible on resource or schedule grounds to test all configurations. In this case, the supplier shall demonstrate that a representative set of tests has been selected.

The supplier shall ensure that:

- All items are adequately tested to demonstrate their compliance with assigned requirements
- All verifications are performed according to the verification plan
- The planned verification activities include full verification of software identified as critical, at each stage of its development.

The supplier shall conduct Reviews and inspections according to written procedures that shall specify:

- Item under review
- The person responsible for carrying out the review or inspection (who shall be independent of the author)
- All participants
- How the item is to be inspected (tools, checklist, etc.)
- How the inspection is to be reported.

The supplier shall keep a record of all reviews and inspections. The record shall identify the:

- Item under review
- Author of the review record
- All participants in the review
- Review criteria
- Review result.

All traceability matrices provided to demonstrate compliance with assigned requirements shall be verified at the completion of each activity.

Independent (third party) software verification shall be performed for critical software.

4.4.9 Software Problem Reporting

The supplier shall define procedures for registering, analysing and correcting software problems during development.

Anomalies can be reported at any stage in the life cycle. Anomalies can fall into a number of categories according to the degree of regression in the life cycle.

Anomaly categories are:

- Communications link failure
- Hardware failure
- Operations error
- User documentation does not conform to code
- Code does not conform to design
- Design does not conform to requirements
- New or changed requirements.

Anomalies that are due to problems with the software shall have Software Problem Reports generated. Those for new or changed requirements shall have change proposals generated. Those that are due to communications, hardware or operator failure are outside the scope of this document.

Software problems and change proposals are handled by the procedure described below. This change procedure requires a formal review to be held

A Software Problem Report (SPR) must be completed for each detected problem, giving all information about

- The symptoms
- The operating environment and circumstances
- The priority of the problem, expressed in two dimensions: criticality (critical/non-critical) and urgency (urgent/routine)
- And the version of the software exhibiting the problem.

A problem is critical if the software or an essential feature of the software is not available. A problem is urgent if the solution is required as soon as possible. An urgent SPR will normally have an operational work around, but must be resolved as soon as possible (for example because the work around is manpower intensive).

Evidence, such as listings of results, may be attached. A problem does not formally exist until an SPR has been written.

The SPR is passed to the Configuration Control Board (CCB) or Software Review Board who will assign it to the relevant authority for analysis. A Software Change Request form (SCR) must be completed for each software change found necessary. This describes the changes required and includes an assessment of the cost and schedule impacts.

The configuration control board then reviews each SCR and, if appropriate, assigns someone to carry out the change.

The interface with the non-conformance system (i.e. the circumstances under which a problem qualifies as a non-conformance) shall be defined (see section 4.4.10). The supplier shall ensure adherence to these procedures.

4.4.10 Non-Conformance Reporting

A non conformance is defined in [Ref. 2] as a non fulfilment of a specified requirement. The definition covers the departure or absence of one or more quality characteristics (including dependability characteristics), or quality system elements from specified requirements.

The supplier shall provide, in the software product assurance plan, a system for handling non-conformances. The system should help the supplier to:

- Identify and segregate non-conforming items
- Report their existence
- Record their status
- Analyse and review their impact
- Define and implement corrective action
- Analyse trends
- Define the authority and responsibilities assigned to his sub-suppliers for handling non-conformances
- Define the point in the lifecycle non-conformance processing should be applied.

Non-conformances shall be classified as major or minor. Any non-conformance should be classified as major if:

- It affects the safety of people or equipment
- It significantly affects the operational, functional or contractual requirements

- It significantly affects the reliability, maintainability, availability
- It significantly affects the lifetime of the end product
- It affects the interfaces with hardware and/or external software
- It involves incorrect qualification or acceptance test procedures or non-compliant test results.

Other non-conformances may be classified as minor. In addition anomalies with limited impact, such as the non-availability of a minor operational requirement with a manual work around, or an infrequent failure anomaly. However if in doubt, the non-conformance shall be classified as major.

The impact of multiple non-conformances on the same item shall be analysed.

The supplier shall inform the customer of all major non-conformances, including non-conformances reported by secondary suppliers that are classified as major.

A non-conformance review board chaired by the customer's quality manager could be an appropriate mechanism to investigate, review and classify all non-conformances, identify root causes and ensure the implementation of corrective action to prevent recurrence.

The review of non-conformances should be undertaken judiciously, as frequent reviews will reduce their impact.

The waiver processing procedure as defined in ECSS–M–40 shall handle all proposed use of non-conforming items.

4.4.11 Assurance of the Configuration Management Process

*[JB7]*The software product assurance manager shall ensure that configuration management is defined and applied both internally and by secondary suppliers. He (or his representative) shall attend all boards established to review the release of items.

The supplier's software product assurance manager shall ensure that all delivered software complies with requirements and design documentation.

The software product assurance manager shall conduct audits to ensure that the configuration control requirements are be applied through the product lifecycle.

4.5 SOFTWARE PRODUCT ASSURANCE PLAN

The software product assurance plan shall specify or reference:

- Quality objectives (measurable whenever possible)
- The software development life cycle, (including milestones and the input and output criteria for each phase)
- Verification and validation activities (including definition, schedules, resources and approval authorities)
- Responsibilities for quality activities such as reviews and tests configuration management and change control, non conformance control and corrective action
- Methods, tools and rules to be used
- The procedures for determining the criticality category of software processes and items.

The supplier shall document the software lifecycle, including milestones, in the software product assurance plan. The choice of lifecycle shall be reviewed to ensure:

- It satisfies the contractual requirements
- It satisfies the product assurance requirements
- Adequate resources are available for its implementation.

In the definition of the lifecycle, associated milestones and documents the quality objectives shall be taken into account.

The plan shall identify the expected state of completion of phase outputs at the end of each phase.

Those characteristics that are crucial to the products safe and proper functioning shall be identified in each phase outputs.

The plan shall document the role of the customer at milestones.

In order to confirm that the software is ready for validation and that the necessary resources, software product assurance plans, test case specifications and procedures, simulators or technical means are available, the supplier should schedule a milestone at the following stages:

- Immediately after the CDR (i.e. before factory acceptance tests)
- Before preliminary site acceptance tests
- And before the site acceptance tests.

The software product assurance plan shall list which plans are intended, and the schedule for their production. The following activities shall be covered in the relevant plans (see Part A):

- development
- specification, design and user documents to be produced
- configuration and documentation management
- verification and validation activities (including testing)
- maintenance.

It shall be assured that each plan has been:

- Finalised before the phase to which it is applied
- Maintained up to date
- Reviewed against contractual requirements.

The product assurance plan shall include a compliance matrix to the contractual product assurance requirements.

4.6 PRODUCT ASSURANCE AND THE SOFTWARE LIFE CYCLE

For a summary of documents in the lifecycle see Appendix C.

4.6.1 System Engineering for Software

It shall be assured that:

- The customer shall document in the system specification (part of the requirements baseline) a complete, unambiguous, and verifiable set of software quality requirements together with methods for verification/validation where appropriate
- The supplier shall draft the product assurance plan in response to the product assurance requirements in the requirements baseline
- The customer shall place any verification and validation requirements in the requirements baseline
- The supplier shall respond by specifying verification activities in the verification plan and validation activities in the validation plan.

The product assurance function shall:

- Ensure that contracts include suitable product assurance provisions based upon the knowledge of the products and on customer's requirements
- Be involved in the preparation and negotiations of the product assurance provisions
- Participate in the detailed review of contract
- Be involved in the assessment and review of all changes to the contractual requirements.

4.6.2 Software Requirements Engineering

The supplier shall finalise the product assurance plan, and shall assure the finalisation of the verification plan and validation plan.

Depending on its criticality, the product assurance manager shall ensure that an appropriate dependability and safety analysis is produced indicating in-service dependability including interfaces, taking worst case operating conditions into account.

The product assurance engineer shall ensure that the technical specification:

- Includes all functional, operational and performance requirements
- Includes all other types of requirements to cover the customer's needs
- Addresses the results from the dependability analysis, if applicable.

The product assurance engineer shall ensure that the Design Justification file contains a complete, reviewed compliance matrix to demonstrate compliance with the requirements baseline.

4.6.3 Software Design Engineering

The supplier shall define the following in the software product assurance plan:

- The appropriate design method to be used
- The appropriate mandatory and advisory standards to be applied (including coding standards)
- The complexity checks to be applied
- The tools to be used.

Product assurance shall review these against the requirements baseline and check their adherence to them.

The design shall facilitate testing to meet the non-functional requirements, and shall meet the quality requirements as documented in the technical specification.

The supplier shall describe in the product assurance report the synthesis of results obtained and corrective actions.

The supplier shall review the design documentation to ensure that it is of adequate quality for maintenance.

The supplier shall justify in the software product assurance plan the use of any low level computer languages (i.e. assembler or machine languages).

It shall be assured that the supplier specifies in the software verification plan:

- Metrics of code complexity to be collected
- Coding standards to be applied
- Verification tools to be used
- The degree of independence of verification from coding.

The supplier shall report the results of these tests and any corrective actions in the software product assurance reports

Product assurance shall assure that all formal tests are conducted on code that is subject to configuration control.

4.6.4 Verification and Validation (including Validation against TS)

It shall be assured that in the software validation plan the supplier specifies the strategy for each level of testing including:

- The types of tests to be performed, e.g. functional, boundary, performance, usability, compliance with standards
- Test coverage goals according to the software criticality.

This shall be agreed with the customer.

Minimum test coverage shall be as follows:

- For unit level testing branch and decision coverage
- For integration testing each control flow between integrated modules should be checked to ensure correct functionality of the call and parameter passing
- For system level testing all requirements and external interfaces
- Using nominal, boundary, zero/nil, and out of range data values.

Note that for some types of software (e.g. communications protocols) coverage of all states of the state machine might be needed.

If the supplier is using automatically generated code the test strategy shall achieve the same test coverage as for manually generated code, unless explicitly required otherwise. Test code generation tools shall only be used if they have been formally qualified for the purpose.

It shall be assured that the supplier reviews all test procedures and data to ensure they satisfy requirements and are adequate, feasible and traceable.

It shall be assured that the supplier holds a test readiness review prior to the start of each key test activity.

In the software product assurance plan the supplier shall define how the required test coverage measurements are to be performed and collected. The level of coverage obtained shall be reported in the software product assurance reports.

It shall be assured that the supplier documents and reports any software problems and non conformances detected during testing.

The test coverage of configurable code shall be checked to ensure that the stated requirements are met in each test configuration.

The supplier shall ensure that problem reports and subsequent actions are properly closed.

It shall be assured that the supplier makes provision for the witnessing of tests by the customer (for the factory acceptance tests) and by independent supplier personnel. The supplier shall for all tests ensure that:

- They are conducted using approved test procedures and data
- The configuration under test is correct
- The tests are documented
- Test reports are up to date
- The tests are repeatable by verifying the storage / recording of tested software, support software, test environment, supporting documents and problems found

The supplier shall confirm in writing correct completion of the tests.

The supplier shall hold review boards on the completion of key test phases. The supplier shall ensure that the software is re-tested, and the test documentation updated:

- When the code is changed
- Following changes to other code if analysis shows the code is affected
- Following changes to hardware if analysis shows the code is affected
- Following change of software generation tools (e.g. compilers). if there is any possibility that the code may be affected.

It shall be assured that staff that was not involved in the design or code carries out the validation.

It shall be assured that independent software validation (i.e. involving a third party contractor) is performed for highly critical software.

The supplier shall review the test documentation to ensure that it is up to date and adequate for use during the maintenance phase.

It shall be assured that the detailed test documentation:

- Specifies procedures, data and expected results
- Specifies criteria for the completion of the test and any contingency steps
- Is consistent with the strategy defined in the test plan
- Covers the test environment, tools and test software, personnel required and associated training requirements
- Identifies the hardware and software configuration.

It shall be assured that any requirement not subjected to a test has a verification report drawn up documenting or referring to the verification activities performed (e.g. A requirement that all source code shall be written in ADA cannot be validated by test, but can be verified by inspection).

The supplier shall report on the execution and the results of all assurance, verification and validation activities in software product assurance reports.

4.6.5 Software Validation and Acceptance

It shall be assured that the installation of the software shall be performed in accordance with the installation plan, in which the roles, responsibilities and obligations of supplier and customer are defined.

The customer shall define a software acceptance test plan, specifying the intended acceptance tests.

*[JB15]*Before presenting the software for acceptance, the supplier shall ensure:

- That the delivered software complies with all requirements
- That the object code corresponds to the delivered source code
- That all agreed changes (e.g. as a result of technical validation) have been implemented
- That all non-conformances are either resolved or declared.

The customer shall ensure that the acceptance tests are performed in compliance with the approved software acceptance test plan.

It shall be assured that the executable code was regenerated from configuration managed source code components and installed on the target environment following pre-defined procedures.

Any discovered problems shall be documented in non-conformance reports.

On completion of the acceptance tests a report shall be generated certifying conformance to procedures and stating the test result. This report shall be signed by:

- Customer representatives
- Customer product assurance representatives, if applicable
- Supplier representatives
- Supplier product assurance representatives, if applicable
- Maintenance organisation representatives, if applicable.

4.6.6 Software Operations Engineering

The software conformance to operational requirements shall be demonstrated and shall cover:

- Availability and maintainability
- Safety features

- Human/computer interfaces
- Operating procedures
- Mission product quality requirements.

The required quality of the mission products related to software during operations shall be agreed with the customer and /or users.

Software operations engineering:

- Mans the helpdesk
- Receives anomaly reports from users of the system
- Prepares SPRs for those anomalies that are found to be faults, and
- Attends configuration control board meetings.

4.6.7 Maintenance Process

The organization responsible for maintenance shall be identified sufficiently early to allow a smooth transition into operations and maintenance.

The maintenance organisation shall specify the assurance, verification and validation activities applicable to maintenance interventions in the software maintenance plan.

The software maintenance plan shall be verified against requirements for maintenance in the requirements baseline.

Product assurance shall be represented on the SW review board to ensure that improvements or corrections to the Software during operations and maintenance do not compromise the product quality

Product assurance shall review migration plans (if any) and retirement plan of the software to ensure that product assurance is maintained throughout its required operational life.

APPENDIX A GLOSSARY

DEFINITIONS

Operational Software

Operational software is any software used in operating the space system that can be traced back to requirements in the requirements baseline.

Non-operational Software

Non-operational software is any other software, which could include test scripts, test programs and simulators. Simulators would become operational in the event that they are included in operational processes, for example to check command sequences or operational procedures, as an essential step before carrying out the operations.

ABBREVIATED TERMS

The following additional acronyms are used:

AR	Acceptance Review
BSSC	Board for Software Standardisation and Control
CASE	Computer Aided Software Engineering
CCB	Configuration Control Board
CDR	Critical Design Review
CI	Configuration Item
COTS	Commercial Off-the-Shelf Software
CPU	Central Processor Unit
DDF	Design Definition File
DJF	Design Justification File
ECSS	European Cooperation for Space Standardisation
EGSE	Electrical Ground Support Equipment

ESA	European Space Agency
ESOC	European Space Operations Centre
FAT	Factory Acceptance Tests
FFP	Fixed Firm Price
GCS	Ground Communication Sub-net
GSCDR	Ground Segment Critical Design Review
GSPDR	Ground Segment Preliminary Design Review
GSRR	Ground Segment Requirements Review
GSTS	Ground Station System
GSTVVRR	Ground Segment Technical Verification and Validation Review
HCI	Human Computer Interface
ICD	Interface Control Document
IRD	Interface Requirement Document
IOQR	In-Orbit Qualification Review
ISO	International Standards Organisation
LEOP	Launch and Early Orbit Operations
MCOR	Mission Close-Out Review
MCS	Mission Control System
MES	Mission Exploitation System
MMI	Man-Machine Interface
MTBF	Mean Time Between Failure
MOTS	Modified Off-The-Shelf Software
OTS	Off-the-Shelf Software
PA	Product Assurance
PCS	Payload Control System
PDR	Preliminary Design Review
PSAT	Provisional Site Acceptance Tests
PSS	Procedures, Standards and Specifications
QR	Qualification Review
RAM	Random Access Memory
RB	Requirements Baseline
RF	Radio Frequency
SAT	Site Acceptance Tests
SPR	Software Problem Report
SRB	Software Review Board

SRR	System Requirements Review
SUM	Software User Manual
SVF	Software Validation Facility
SWRR	Software Requirements Review
TS	Technical Specification
WBS	Work Breakdown Structures

APPENDIX B 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
11. ECSS-E-40 Space Engineering: Software, Issue B
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-00 Policy and Principles April 1996

22. ECSS-Q-20 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

APPENDIX C DOCUMENT LIFECYCLES

Documents are shown in the order in which their templates appear in Part C.

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle												
			System Engineering for Software	SRR	Sofware Requirements Engineering: Software Requirements Analysis	SWRR	Sofware Requirements Engineering: Software Architectural Design	PDR	Software Design Engineering: Design, Coding and Testing, Integration	Software Design Engineering: Validation against Technical Specification	CDR	Software Validation and Acceptance: Preliminary Acceptance Tests	QR	Software Validation and Acceptance: 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				

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle													
			System Engineering for Software	SRR	Sofware Requirements Engineering			PDR	Software Design Engineering		CDR	Software Validation and Acceptance			AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis	SWRR	Software Architectural Design		Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests		
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		
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							

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle														
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance				AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests			
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
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										

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle													
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance			AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests		
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													
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance			AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests		
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							

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle													
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance			AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests		
DJF	*Software Unit Test Verification Report*	Provides evidence of unit test results review							Created							
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						

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle													
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance			AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests		
DJF	Preliminary Acceptance Test Results (Software Validation Testing Report template)	Records results of FAT										Created				
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 tests							Created							
DJF	Software Budget Report	Reports status of technical budget and margins	Produced		Updated		Updated		Updated			Updated		Updated		

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle														
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance				AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests			
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			
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

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle													
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance	QR	Delivery, Installation and Operational Acceptance Tests	AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests				
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		
DJF / O/M	Software Problem Report	Records software problems during development or operation							Generated	Generated		Generated		Generated		Generated

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle														
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance				AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests			
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

File *key at end*	Document *italics means template not available*	Purpose	Lifecycle														
			System Engineering for Software	SRR	Sofware Requirements Engineering	SWRR	Software Architectural Design	PDR	Software Design Engineering		CDR	Software Validation and Acceptance				AR	Software Operations Engineering / Software Maintenance
					Software Requirements Analysis				Design, Coding and Testing, Integration	Validation against Technical Specification		Preliminary Acceptance Tests	QR	Delivery, Installation and Operational Acceptance Tests			
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												Finallised			

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

APPENDIX D CROSS REFERENCE TO ECSS STANDARDS

ECSS-M-00 5.4.1, 13
ECSS-M-00 5.4.2, 13
ECSS-M-00 6.3.6 a., 11
ECSS-M-00 6.3.6 b., 12
ECSS-M-00 6.3.6 c., 12
ECSS-M-00 6.3.6b, 11
ECSS-M-00 6.3.6c, 11
ECSS-M-00 6.3.6d, 11
ECSS-M-00 7.1.3 a., 6
ECSS-M-00 7.2.3 a., 6
ECSS-M-00 7.3.3 a., 6
ECSS-M-00 7.4.3 a., 32
ECSS-M-00 7.5.3 a., 27
ECSS-M-00 7.6.3 a., 6
ECSS-M-00 7.6.3 b., 16
ECSS-M-00 7.7.3 a., 6
ECSS-M-00 8.1.1, 10
ECSS-M-00 8.2.1, 10
ECSS-M-00 8.3.1, 10
ECSS-M-10 5.1.1, 15
ECSS-M-10 5.1.2, 15
ECSS-M-10 5.1.3, 15
ECSS-M-10 5.3.1, 15
ECSS-M-10 5.3.2, 15
ECSS-M-10 5.3.4, 15
ECSS-M-10 5.3.5, 15
ECSS-M-10 5.4.1, 15
ECSS-M-10 5.4.2, 15
ECSS-M-10 5.4.3, 28
ECSS-M-10 5.4.4, 15
ECSS-M-10 5.5.1, 15
ECSS-M-20 4.1.4, 10
ECSS-M-20 4.1.5, 10
ECSS-M-20 4.1.6, 6
ECSS-M-20 4.1.7, 6
ECSS-M-20 4.2.10, 10
ECSS-M-20 4.2.2, 17
ECSS-M-20 4.2.3, 17
ECSS-M-20 4.2.4, 17
ECSS-M-20 4.2.5, 17
ECSS-M-20 4.2.6, 10
ECSS-M-20 4.2.7, 10
ECSS-M-20 4.2.8, 10
ECSS-M-20 4.2.9, 10
ECSS-M-20 5.1.2, 20
ECSS-M-20 5.1.3, 20
ECSS-M-20 5.1.4, 10
ECSS-M-20 5.1.5, 10
ECSS-M-20 5.5.1, 11
ECSS-M-20 6.1.1, 6
ECSS-M-20 6.2.1, 6
ECSS-M-30 6.6.1, 20
ECSS-M-30 6.6.3, 20
ECSS-M-30 6.6.4, 20
ECSS-M-30 6.6.6, 20
ECSS-M-30 6.6.8, 28
ECSS-M-30 6.7.1, 20
ECSS-M-30 6.7.2, 20
ECSS-M-30 7.4, 19
ECSS-M-30 7.5, 19
ECSS-M-30 7.6, 20
ECSS-M-30 7.7, 20
ECSS-M-30A 6.4.4, 56
ECSS-M-40 5.1.1, 28
ECSS-M-40 5.1.10, 24
ECSS-M-40 5.1.11, 24
ECSS-M-40 5.1.12, 24
ECSS-M-40 5.1.2a, 32
ECSS-M-40 5.1.2b, 32
ECSS-M-40 5.1.2c, 32
ECSS-M-40 5.1.2d, 32
ECSS-M-40 5.1.2e, 32
ECSS-M-40 5.1.3, 28
ECSS-M-40 5.1.4, 28
ECSS-M-40 5.1.5, 24
ECSS-M-40 5.1.6, 28
ECSS-M-40 5.1.7, 28
ECSS-M-40 5.1.8, 28
ECSS-M-40 5.1.9, 28
ECSS-M-40 5.2.1, 32
ECSS-M-40 5.2.10, 30
ECSS-M-40 5.2.11, 31
ECSS-M-40 5.2.12, 31
ECSS-M-40 5.2.13, 31
ECSS-M-40 5.2.14, 31
ECSS-M-40 5.2.2, 31
ECSS-M-40 5.2.3, 29
ECSS-M-40 5.2.4, 30
ECSS-M-40 5.2.5, 30
ECSS-M-40 5.2.6, 30
ECSS-M-40 5.2.7, 30
ECSS-M-40 5.2.8, 30
ECSS-M-40 5.2.9, 30
ECSS-M-40 5.3.1, 30
ECSS-M-40 5.3.2, 30
ECSS-M-40 5.4.1, 29, 53
ECSS-M-40 5.5.1, 32
ECSS-M-50 5.1.1, 27
ECSS-M-50 5.1.2, 27
ECSS-M-50 5.1.3, 27
ECSS-M-50 5.1.4, 27
ECSS-M-50 5.2.1, 27
ECSS-M-50 5.2.2, 27
ECSS-M-50 5.3.1, 27
ECSS-M-50 5.3.2, 27
ECSS-M-50 5.3.3, 27
ECSS-M-50 5.3.4, 27
ECSS-M-50 5.5.1, 26, 27
ECSS-M-50 6.1.1, 26
ECSS-M-50 6.1.10, 28
ECSS-M-50 6.1.2, 26
ECSS-M-50 6.1.3, 26
ECSS-M-50 6.1.4, 26
ECSS-M-50 6.1.5, 26
ECSS-M-50 6.1.7, 26
ECSS-M-50 6.1.8, 28
ECSS-M-50 6.1.9, 26
ECSS-M-50 6.2.1, 26
ECSS-M-50 6.3.1, 26
ECSS-M-50 6.3.2, 26
ECSS-M-50 6.4.1, 26
ECSS-M-50 6.5.1, 32
ECSS-M-50 6.5.2, 32
ECSS-M-50 6.5.5, 32
ECSS-M-50 6.5.6, 32
ECSS-M-50 6.5.7, 32
ECSS-M-60 5.1, 15
ECSS-M-60 5.12, 16
ECSS-M-60 5.2, 15
ECSS-M-60 5.20, 16
ECSS-M-60 5.21, 16
ECSS-M-60 5.22, 16
ECSS-M-60 5.5, 15
ECSS-M-60 7.1, 16
ECSS-M-60 7.2, 16
ECSS-M-60 7.3, 16
ECSS-M-60 7.4, 16
ECSS-M-60 7.5, 16
ECSS-M-60 7.6, 16
ECSS-M-60 7.7, 17
ECSS-M-60 7.8, 16