FIRST/ESA/R/0011.10

ESA
Board for Software Standardisation and Control
(BSSC)

BSSC(96)2
3 Sept. 1996

Distribution: All H/Departments, H/Division, H/Sections

Guide to applying the ESA Software Engineering Standards

Standards to Small Software Projects, BSSC(96)2

The ESA Software Engineering Standards, ESA PSS-05-0, define the software practices that must be applied in all the Agency's projects. While their application in large projects is quite straightforward, they can be too heavy when applied in small projects. Experience has shown that simplified approaches are appropriate for small software projects:

BSSC(96)2 (attached) defines the "small" software project and provides guidelines about how to apply the ESA Software Engineering Standards to small software projects.

M Jones

U Mortensen

Encl.

BSSC(96)2 Issue 1
May 1996

Guide to applying the ESA software engineering standards to small software projects

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: BSSC(96)2			
2. ISSUE	3. REVISION	4. DATE	5. REASON FOR CHANGE
1	0	1996	

Approved, May 8th, 1996
Board for Software Standardisation and Control
M. Jones and U. Mortensen, BSSC co-chairmen

TABLE OF CONTENTS

PREFACE

The ESA Software Engineering Standards, ESA PSS-05-0, define the software practises that must be applied in all the Agency's projects. While their application in large projects is quite straightforward, experience has shown that a simplified approach is appropriate for small software projects. This document provides guidelines about how to apply the ESA Software Engineering Standards to small software projects. The guide has accordingly been given the nickname of *'PSS-05 lite'*.

The following past and present BSSC members have contributed to the production of this guide: Michael Jones (co-chairman), Uffe Mortensen (co-chairman), Gianfranco Alvisi, Bryan Melton, Daniel de Pablo, Adriaan Scheffer and Jacques Marcoux. The BSSC wishes to thank Jon Fairclough for drafting and editing the guide. The authors wish to thank all the people who contributed ideas about applying ESA PSS-05-0 to small projects.

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

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

CHAPTER 1
INTRODUCTION

1.1 PURPOSE

ESA PSS-05-0 describes the software engineering standards to be applied for all deliverable software implemented for the European Space Agency (ESA) [Ref 1].

This document has been produced to provide organisations and software project managers with guidelines for the application of the standards to small software projects. Additional guidelines on the application of the standards are provided in the series of documents described in ESA PSS-05-01, 'Guide to the Software Engineering Standards' [Ref 2 to 12].

There are several criteria for deciding whether a software project is small, and therefore whether this guide can be applied. These are discussed in Chapter 2. Whatever the result of the evaluation, this guide should not be applied when the software is 'critical' (e.g. software failure would result in the loss of life, or loss of property, or major inconvenience to users) or when ESA PSS-01-21 'Software Product Assurance Requirements for ESA Space Systems' applies.

1.2 OVERVIEW

Chapter 2 defines what is meant by a small software project and discusses some simple strategies for applying ESA PSS-05-0 to them. Chapter 3 explains how the mandatory practices of ESA PSS-05-0 should be applied in a small software project. Appendix A contains a glossary of acronyms and abbreviations. Appendix B contains a list of references. Appendix C contains simplified document templates.

This page is intentionally left blank

CHAPTER 2
SMALL SOFTWARE PROJECTS

2.1 INTRODUCTION

ESA PSS-05-0 lists several factors to consider when applying it to a software project (see Introduction, Section 4.1). The factors that are related to the 'size' of a software development project are:

- project development cost
- number of people required to develop the software
- amount of software to be produced.

A software project can be considered to be small if one or more of the following criteria apply:

- less than two man years of development effort is needed
- a single development team of five people or less is required
- the amount of source code is less than 10000 lines, excluding comments.

One or more of the following strategies are often suitable for small projects producing non-critical software:

- combine the software requirements and architectural design phases
- simplify documentation
- simplify plans
- reduce the reliability requirements
- use the system test specification for acceptance testing.

The following sections review these strategies.

2.2 COMBINE THE SR AND AD PHASES

ESA PSS-05-0 requires that software requirements definition and architectural design be performed in separate phases. These phases end with formal reviews of the Software Requirements Document and Architectural Design Document. Each of the reviews normally involve the user, and can last two weeks to a month. For a small software project, separate reviews of the SRD and ADD lengthen the project significantly. An efficient way to organise the project is therefore to:

- combine the SR and AD phases into a single SR/AD phase
- combine the SR/R and AD/R into single formal review at the end of the SR/AD phase.

2.3 SIMPLIFY DOCUMENTATION

The document templates provided in ESA PSS-05-0 are based upon ANSI/IEEE standards, and are designed to cover the documentation requirements of all projects. Developers should use the simplified document templates provided in Appendix C.

Developers should combine the SRD and ADD into a single Software Specification Document (SSD) when the SR and AD phases are combined.

Developers should document the detailed design by putting most detailed design information into the source code and extending the SSD to contain any detailed design information that cannot be located in the source code (e.g. information about the software structure).

The production of a Project History Document is considered to be good practice but its delivery is optional.

2.4 SIMPLIFY PLANS

In small software projects it is highly desirable to plan all phases at the start of the project. This means that the phase sections of the Software Project Management Plan, Software Configuration Management Plan and Software Verification and Validation Plans are combined. The plans may be generated when writing the proposal.

The purpose of the software quality assurance function in a project is to check that the project is adhering to standards and plans. In a small

software project this is normally done informally by the project manager. The practice of producing a Software Quality Assurance Plan may be waived.

ESA PSS-05-0 requires that the test approach be outlined in the test plan and refined in the test design. In small software projects, it is sufficient to outline the test approach only. This means that the Test Design sections of the SVVP can be omitted.

Developers may find it convenient to combine the technical process section of the Software Project Management Plan, the Software Configuration Management Plan, the Software Verification and Validation Plan and the Software Quality Assurance Plan into a single document (called a 'Quality Plan' in ISO 9000 terminology).

2.5 REDUCE THE RELIABILITY REQUIREMENTS

The reliability requirements on software should be set by trading-off the cost of correcting defects during development against the cost of:

- correcting the defects during operations
- penalties resulting from failure during operation
- loss of reputation suffered by producing a faulty product.

Reliability is built into software by:

- designing the software to be reliable
- reviewing documents and code
- testing the code.

ESA PSS-05-0 requires that every statement be executed at least once. This is sometimes known as the 'statement coverage requirement'. There is good evidence that significant numbers of defects remain in the software when the statement coverage falls below 70% [Ref 14], and that the number of defects becomes very low when the statement coverage exceeds 90%. However achieving high coverage in testing can be quite expensive in effort; test cases have to be invented to force execution of all the statements. The cost of this exhaustive testing may exceed the cost of repairing the defects during operations.

Developers should:

- set a statement testing target (e.g. 80% coverage)
- review every statement not covered in testing.

Software tools are available for measuring test coverage. They should be used whenever possible.

2.6 USE THE SYSTEM TEST SPECIFICATION FOR ACCEPTANCE TESTING

When the developer is responsible for producing the acceptance test specification, the acceptance tests often repeat selected system test cases and procedures. One approach to documenting acceptance tests is simply to indicate in the system test specification which test cases and procedures should be used in acceptance testing.

CHAPTER 3
PRACTICES FOR SMALL SOFTWARE PROJECTS

3.1 INTRODUCTION

Figure 3.1 illustrates a small software project life cycle approach with the following features:

- the software requirements and architectural design phases have been combined
- the simplified document templates described in Appendix C are used
- detailed design information has been included in the source code
- plans have been combined
- reliability requirements have been reduced
- no test design documentation is produced
- selected system tests are used for acceptance testing
- all SQA practices are not applicable
- the Project History Document is not a required deliverable.

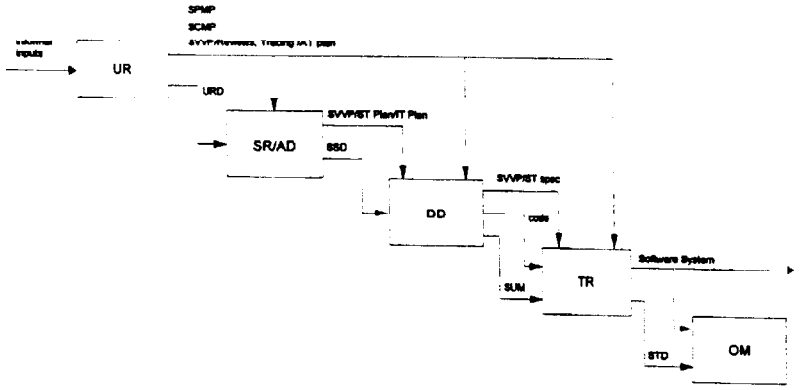

Figure 3.1: Small software project life cycle

The following sections describe the small software project mandatory practices. The practices are tabulated with their ESA PSS-05-0 identifier. Guidance (in *italics*) is attached where appropriate to explain how the practice is applied in a small project. This guidance is based upon the strategies described in Chapter 2.

3.2 SOFTWARE LIFE CYCLE

SLC01 The products of a software development project shall be delivered in a timely manner and be fit for their purpose.

SLC02 Software development activities shall be systematically planned and carried out.

SLC03 All software projects shall have a life cycle approach which includes the basic phases shown in Part 1, Figure 1.1 (of PSS-05):

UR phase - Definition of the user requirements
SR phase - Definition of the software requirements
AD phase - Definition of the architectural design
DD phase - Detailed design and production of the code
TR phase - Transfer of the software to operations
OM phase - Operations and maintenance

In small projects, the SR and AD phases are combined into an SR/AD phase.

3.3 UR PHASE

UR01 The definition of the user requirements shall be the responsibility of the user.

UR02 Each user requirement shall include an identifier.

UR03 Essential user requirements shall be marked as such.

UR04 For incremental delivery, each user requirement shall include a measure of priority so that the developer can decide the production schedule.

UR05 The source of each user requirement shall be stated.

UR06 Each user requirement shall be verifiable.

UR07 The user shall describe the consequences of losses of availability, or breaches of security, so that developers can fully appreciate the criticality of each function.

UR08 The outputs of the UR phase shall be formally reviewed during the User Requirements Review.

UR09 Non-applicable user requirements shall be clearly flagged in the URD.

UR10 An output of the UR phase shall be the User Requirements Document (URD).

UR11 The URD shall always be produced before a software project is started.

UR12 The URD shall provide a general description of what the user expects the software to do.

UR13 All known user requirements shall be included in the URD.

UR14 The URD shall describe the operations the user wants to perform with the software system.

UR15 The URD shall define all the constraints that the user wishes to impose upon any solution

UR16 The URD shall describe the external interfaces to the software system or reference them in ICDs that exist or are to be written.

3.4 SR/AD PHASE

SR01 SR phase activities shall be carried out according to the plans defined in the UR phase.

SR02 The developer shall construct an implementation-independent model of what is needed by the user.

SR03 A recognised method for software requirements analysis shall be adopted and applied consistently in the SR phase.

This practice applies to the SR/AD phase.

SR04 Each software requirement shall include an identifier.

SR05 Essential software requirements shall be marked as such.

SR06 For incremental delivery, each software requirement shall include a measure of priority so that the developer can decide the production schedule.

SR07 References that trace software requirements back to the URD shall accompany each software requirement

SR08 Each software requirement shall be verifiable.

SR09 The outputs of the SR phase shall be formally reviewed during the Software Requirements Review.

The outputs of software requirements definition activities are reviewed in the SR/AD phase review.

SR10 An output of the SR phase shall be the Software Requirements Document (SRD).

This practice applies to the SR/AD phase.

The SRD information is placed in the Software Specification Document (SSD).

SR11 The SRD shall be complete.

This practice applies to the SSD.

SR12 The SRD shall cover all the requirements stated in the URD.

This practice applies to the SSD.

SR13 A table showing how user requirements correspond to software requirements shall be placed in the SRD.

This practice applies to the SSD.

SR14 The SRD shall be consistent.

This practice applies to the SSD.

SR15 The SRD shall not include implementation details or terminology, unless it has to be present as a constraint.

This practice applies to the SSD.

SR16 Descriptions of functions ... shall say what the software is to do, and must avoid saying how it is to be done.

SR17 The SRD shall avoid specifying the hardware or equipment, unless it is a constraint placed by the user.

This practice is not applicable.

SR18 The SRD shall be compiled according to the table of contents provided in Appendix C.

This practice applies to the SSD. The SSD should be compiled according to the table contents in Appendix C of this guide.

AD01 AD phase activities shall be carried out according to the plans defined in the SR phase.

In small projects, the AD plans should be made in the UR phase.

AD02 A recognised method for software design shall be adopted and applied consistently in the AD phase.

This practice applies to the SR/AD phase.

AD03 The developer shall construct a 'physical model', which describes the design of the software using implementation terminology.

AD04 The method used to decompose the software into its component parts shall permit a top-down approach.

AD05 Only the selected design approach shall be reflected in the ADD.

This practice applies to the SSD.

For each component the following information shall be detailed in the ADD:

AD06 • data input;

AD07 • functions to be performed;

AD08 • data output.

AD06, 7, 8 apply to the SSD.

AD09 Data structures that interface components shall be defined in the ADD.

This practice applies to the SSD.

Data structure definitions shall include the:

AD10 • description of each element (e.g. name, type, dimension);

AD11 • relationships between the elements (i.e. the structure);

AD12 • range of possible values of each element;

AD13 • initial values of each element.

AD10, 11, 12, 13 apply to the SSD.

AD14 The control flow between the components shall be defined in the ADD.

This practice applies to the SSD.

AD15 The computer resources (e.g. CPU speed, memory, storage, system software) needed in the development environment and the operational environment shall be estimated in the AD phase and defined in the ADD.

This practice applies to the SSD.

AD16 The outputs of the AD phase shall be formally reviewed during the Architectural Design Review.

The outputs of software architectural design activities are reviewed in the SR/AD phase review.

AD17 The ADD shall define the major components of the software and the interfaces between them.

This practice applies to the SSD.

AD18 The ADD shall define or reference all external interfaces.

This practice applies to the SSD.

AD19 The ADD shall be an output from the AD phase.

This practice applies to the SSD.

AD20 The ADD shall be complete, covering all the software requirements described in the SRD.

This practice applies to the SSD.

AD21 A table cross-referencing software requirements to parts of the architectural design shall be placed in the ADD.

This practice applies to the SSD.

AD22 The ADD shall be consistent.

This practice applies to the SSD.

AD23 The ADD shall be sufficiently detailed to allow the project leader to draw up a detailed implementation plan and to control the overall project during the remaining development phases.

This practice applies to the SSD.

AD24 The ADD shall be compiled according to the table of contents provided in Appendix C.

This practice applies to the SSD. The SSD should be compiled according to the table contents in Appendix C of this guide.

3.5 DD PHASE

DD01 DD phase activities shall be carried out according to the plans defined in the AD phase.

DD phase plans are contained in the plans made in the UR phase and updated, as appropriate during the project.

The detailed design and production of software shall be based on the following three principles:

DD02 • top-down decomposition;

DD03 • structured programming;

DD04 • concurrent production and documentation.

DD05 The integration process shall be controlled by the software configuration management procedures defined in the SCMP.

DD06 Before a module can be accepted, every statement in a module shall be executed successfully at least once.

Acceptance should be performed by the project manager after unit testing and by the customer at the end of the phase.

Software projects should:

- set a statement testing target (e.g. 80% coverage)

- review every statement not covered in testing.

Software tools are available for measuring test coverage. They should be used whenever possible.

DD07 Integration testing shall check that all the data exchanged across an interface agrees with the data structure specifications in the ADD.

This practice applies to the SSD.

DD08 Integration testing shall confirm that the control flows defined in the ADD have been implemented.

This practice applies to the SSD.

DD09 System testing shall verify compliance with system objectives, as stated in the SRD.

This practice applies to the SSD.

DD10 When the design of a major component is finished, a critical design review shall be convened to certify its readiness for implementation.

DD11 After production, the DD Review (DD/R) shall consider the results of the verification activities and decide whether to transfer the software.

DD12 All deliverable code shall be identified in a configuration item list.

DD13 The DDD shall be an output of the DD phase.

Not applicable. Detailed design information is placed in the SSD and source code.

DD14 Part 2 of the DDD shall have the same structure and identification scheme as the code itself, with a 1:1 correspondence between sections of the documentation and the software components.

Not applicable.

DD15 The DDD shall be complete, accounting for all the software requirements in the SRD.

This practice means that all requirements in the SSD must be implemented in the code.

DD16 A table cross-referencing software requirements to the detailed design components shall be placed in the DDD.

The traceability matrix in the SSD must be updated instead.

DD17 A Software User Manual (SUM) shall be an output of the DD phase.

3.6 TR PHASE

TR01 Representatives of users and operations personnel shall participate in acceptance tests.

TR02 The Software Review Board (SRB) shall review the software's performance in the acceptance tests and recommend, to the initiator, whether the software can be provisionally accepted or not.

TR03 TR phase activities shall be carried out according to the plans defined in the DD phase.

TR phase plans are established in the UR phase and updated as appropriate.

TR04 The capability of building the system from the components that are directly modifiable by the maintenance team shall be established.

TR05 Acceptance tests necessary for provisional acceptance shall be indicated in the SWP.

TR06 The statement of provisional acceptance shall be produced by the initiator, on behalf of the users, and sent to the developer.

TR07 The provisionally accepted software system shall consist of the outputs of all previous phases and modifications found necessary in the TR phase.

TR08 An output of the TR phase shall be the STD.

TR09 The STD shall be handed over from the developer to the maintenance organisation at provisional acceptance.

TR10 The STD shall contain the summary of the acceptance test reports, and all documentation about software changes performed during the TR phase.

3.7 OM PHASE

OM01 Until final acceptance, OM phase activities that involve the developer shall be carried out according to the plans defined in the SPMP/TR.

OM02 All the acceptance tests shall have been successfully completed before the software is finally accepted.

OM03 Even when no contractor is involved, there shall be a final acceptance milestone to arrange the formal hand-over from software development to maintenance.

OM04 A maintenance organisation shall be designated for every software product in operational use.

OM05 Procedures for software modification shall be defined.

OM06 Consistency between code and documentation shall be maintained.

OM07 Resources shall be assigned to a product's maintenance until it is retired.

OM08 The SRB ... shall authorise all modifications to the software.

OM09 The statement of final acceptance shall be produced by the initiator, on behalf of the users, and sent to the developer.

OM10 The PHD shall be delivered to the initiator after final acceptance.

Not applicable. However developers are encouraged to produce a PHD for internal use.

3.8 SOFTWARE PROJECT MANAGEMENT

SPM01 All software project management activities shall be documented in the Software Project Management Plan (SPMP).

SPM02 By the end of the UR review, the SR phase section of the SPMP shall be produced (SPMP/SR).

The SPMP does not have phase sections: there is a single plan for all the SR/AD, DD and TR phases.

SPM03 The SPMP/SR shall outline a plan for the whole project.

This practice applies to the SPMP.

SPM04 A precise estimate of the effort involved in the SR phase shall be included in the SPMP/SR.

This practice applies to the SPMP.

SPM05 During the SR phase, the AD phase section of the SPMP shall be produced (SPMP/AD).

Not applicable.

SPM06 An estimate of the total project cost shall be included in the SPMP/AD.

This practice applies to the SPMP.

SPM07 A precise estimate of the effort involved in the AD phase shall be included in the SPMP/AD.

This practice applies to the SPMP.

SPM08 During the AD phase, the DD phase section of the SPMP shall be produced (SPMP/DD).

Not applicable. However the SPMP should be updated when the SSD has been produced.

SPM09 An estimate of the total project cost shall be included in the SPMP/DD.

This practice applies to the SPMP.

SPM10 The SPMP/DD shall contain a WBS that is directly related to the decomposition of the software into components.

This practice applies to the SPMP.

SPM11 The SPMP/DD shall contain a planning network showing relationships of coding, integration and testing activities.

A bar chart (i.e. Gantt Chart) should be used to show the relationship.

SPM12 No software production work packages in the SPMP/DD shall last longer than 1 man-month.

This practice applies to the SPMP.

SPM13 During the DD phase, the TR phase section of the SPMP shall be produced (SPMP/TR).

This practice applies to the SPMP.

3.9 SOFTWARE CONFIGURATION MANAGEMENT

SCM01 All software items, for example documentation, source code, object or relocatable code, executable code, files, tools, test software and data, shall be subjected to configuration management procedures.

SCM02 The configuration management procedures shall establish methods for identifying, storing and changing software items through development, integration and transfer.

SCM03 A common set of configuration management procedures shall be used.

Every configuration item shall have an identifier that distinguishes it from other items with different:

SCM04 • requirements, especially functionality and interfaces;

SCM05 • implementation.

SCM06 Each component defined in the design process shall be designated as a CI and include an identifier.

SCM07 The identifier shall include a number or a name related to the purpose of the CI.

SCM08 The identifier shall include an indication of the type of processing the CI is intended for (e.g. filetype information).

SCM09 The identifier of a CI shall include a version number.

SCM10 The identifier of documents shall include an issue number and a revision number.

SCM11 The configuration identification method shall be capable of accommodating new CIs, without requiring the modification of the identifiers of any existing CIs.

SCM12 In the TR phase, a list of configuration items in the first release shall be included in the STD.

SCM13 In the OM phase, a list of changed configuration items shall be included in each Software Release Note (SRN).

SCM14 An SRN shall accompany each release made in the OM phase.

As part of the configuration identification method, a software module shall have a standard header that includes:

SCM15 • configuration item identifier (name, type, version);

SCM16 • original author;

SCM17 • creation date;

SCM18 • change history (version/date/author/description).

All documentation and storage media shall be clearly labelled in a standard format, with at least the following data:

SCM19 • project name;

SCM20 • configuration item identifier (name, type, version);

SCM21 • date;

SCM22 • content description.

To ensure security and control of the software, at a minimum, the following software libraries shall be implemented for storing all the deliverable components (e.g. documentation, source and executable code, test files, command procedures):

SCM23 • Development (or Dynamic) library;

SCM24 • Master (or Controlled) library;

SCM25 • Static (or Archive) library.

SCM26 Static libraries shall not be modified.

SCM27 Up-to-date security copies of master and static libraries shall always be available.

SCM28 Procedures for the regular backup of development libraries shall be established.

SCM29 The change procedure described (in Part 2, Section 3.2.3.2.1) shall be observed when changes are needed to a delivered document.

SCM30 Software problems and change proposals shall be handled by the procedure described (in Part 2, Section 3.2.3.2.2).

SCM31 The status of all configuration items shall be recorded.

To perform software status accounting, each software project shall record:

SCM32 • the date and version/issue of each baseline;

SCM33 • the date and status of each RID and DCR;

SCM34 • the date and status of each SPR, SCR and SMR;

SCM35 • a summary description of each Configuration Item.

SCM36 As a minimum, the SRN shall record the faults that have been repaired and the new requirements that have been incorporated.

SCM37 For each release, documentation and code shall be consistent.

SCM38 Old releases shall be retained, for reference.

SCM39 Modified software shall be retested before release.

SCM40 All software configuration management activities shall be documented in the Software Configuration Management Plan (SCMP).

SCM41 Configuration management procedures shall be in place before the production of software (code and documentation) starts.

SCM42 By the end of the UR review, the SR phase section of the SCMP shall be produced (SCMP/SR).

This practice applies to the SCMP.

SCM43 The SCMP/SR shall cover the configuration management procedures for documentation, and any CASE tool outputs or prototype code, to be produced in the SR phase.

This practice applies to the SCMP

SCM44 During the SR phase, the AD phase section of the SCMP shall be produced (SCMP/AD).

Not applicable.

SCM45 The SCMP/AD shall cover the configuration management procedures for documentation, and CASE tool outputs or prototype code, to be produced in the AD phase.

This practice applies to the SCMP

SCM46 During the AD phase, the DD phase section of the SCMP shall be produced (SCMP/DD).

Not applicable.

SCM47 The SCMP/DD shall cover the configuration management procedures for documentation, deliverable code, and any CASE tool outputs or prototype code, to be produced in the DD phase.

This practice applies to the SCMP.

SCM48 During the DD phase, the TR phase section of the SCMP shall be produced (SCMP/TR).

Not applicable.

SCM49 The SCMP/TR shall cover the procedures for the configuration management of the deliverables in the operational environment.

Not applicable.

3.10 SOFTWARE VERIFICATION AND VALIDATION

SVV01 Forwards traceability requires that each input to a phase shall be traceable to an output of that phase.

SVV02 Backwards traceability requires that each output of a phase shall be traceable to an input to that phase.

SVV03 Functional and physical audits shall be performed before the release of the software.

SVV04 All software verification and validation activities shall be documented in the Software Verification and Validation Plan (SVVP).

The SVVP shall ensure that the verification activities:

SVV05 • are appropriate for the degree of criticality of the software;

SVV06 • meet the verification and acceptance testing requirements (stated in the SRD);

SVV07 • verify that the product will meet the quality, reliability, maintainability and safety requirements ...;

SVV08 • are sufficient to assure the quality of the product.

SVV09 By the end of the UR review, the SR phase section of the SVVP shall be produced (SVVP/SR).

This practice applies to SVVP.

SVV10 The SVVP/SR shall define how to trace user requirements to software requirements, so that each software requirement can be justified.

This practice applies to SVVP.

SVV11 The developer shall construct an acceptance test plan in the UR phase and document it in the SVVP.

SVV12 During the SR phase, the AD phase section of the SVVP shall be produced (SVVP/AD).

Not applicable.

SVV13 The SVVP/AD shall define how to trace software requirements to components, so that each software component can be justified.

This practice applies to SVVP.

SVV14 The developer shall construct a system test plan in the SR phase and document it in the SVVP.

SW15 During the AD phase, the DD phase section of the SWP shall be produced (SWP/DD).

Not applicable.

SW16 The SWP/AD shall describe how the DDD and code are to be evaluated by defining the review and traceability procedures.

This practice applies to the SWP.

SW17 The developer shall construct an integration test plan in the AD phase and document it in the SWP.

This is done in the SR/AD phase.

SW18 The developer shall construct a unit test plan in the DD phase and document it in the SWP.

SW19 The unit, integration, system and acceptance test designs shall be described in the SWP.

Not applicable.

SW20 The unit integration, system and acceptance test cases shall be described in the SWP.

SW21 The unit, integration, system and acceptance test procedures shall be described in the SWP.

SW22 The unit, integration, system and acceptance test reports shall be described in the SWP.

3.11 SOFTWARE QUALITY ASSURANCE

SQA01 An SQAP shall be produced by each contractor developing software.

Not applicable.

SQA02 All software quality assurance activities shall be documented in the Software Quality Assurance Plan (SQAP).

Not applicable.

SQA03 By the end of the UR review, the SR phase section of the SQAP shall be produced (SQAP/SR).

Not applicable.

SQA04 The SQAP/SR shall describe, in detail, the quality assurance activities to be carried out in the SR phase.

Not applicable.

SQA05 The SQAP/SR shall outline the quality assurance plan for the rest of the project.

Not applicable.

SQA06 During the SR phase, the AD phase section of the SQAP shall be produced (SQAP/AD).

Not applicable.

SQA07 The SQAP/AD shall cover in detail all the quality assurance activities to be carried out in the AD phase.

Not applicable.

SQA08 During the AD phase, the DD phase section of the SQAP shall be produced (SQAP/DD).

Not applicable.

SQA09 The SQAP/DD shall cover in detail all the quality assurance activities to be carried out in the DD phase.

Not applicable.

SQA10 During the DD phase, the TR phase section of the SQAP shall be produced (SQAP/TR).

Not applicable.

SQA11 The SQAP/TR shall cover in detail all the quality assurance activities to be carried out from the start the TR phase until final acceptance in the OM phase.

Not applicable.

APPENDIX A
GLOSSARY

A.1 LIST OF ACRONYMS AND ABBREVIATIONS

AD	Architectural Design
ADD	Architectural Design Document
ANSI	American National Standards Institute
AT	Acceptance Test
DD	Detailed Design and production
DDD	Detailed Design Document
ESA	European Space Agency
IEEE	Institute of Electrical and Electronics Engineers
IT	Integration Test
PHD	Project History Document
PSS	Procedures, Standards and Specifications[1]
SCMP	Software Configuration Management Plan
SPMP	Software Project Management Plan
SQAP	Software Quality Assurance Plan
SR	Software Requirements
ST	System Test
STD	Software Transfer Document
SRD	Software Requirements Document
SSD	Software Specification Document
SUM	Software User Manual
SVVP	Software Verification and Validation Plan
URD	User Requirements Document
UT	Unit Test
WBS	Work Breakdown Structure

[1] Not 'Programming Saturdays and Sundays'

This page is intentionally left blank.

APPENDIX B
REFERENCES

1. ESA Software Engineering Standards, ESA PSS-05-0 Issue 2 February 1991.

2. Guide to the ESA Software Engineering Standards, ESA PSS-05-01 Issue 1 October 1991.

3. Guide to the User Requirements Definition Phase, ESA PSS-05-02 Issue 1 October 1991.

4. Guide to the Software Requirements Definition Phase, ESA PSS-05-03 Issue 1 October 1991.

5. Guide to the Software Architectural Design Phase, ESA PSS-05-04 Issue 1 January 1992.

6. Guide to the Software Detailed Design and Production Phase, ESA PSS-05-05 Issue 1 May 1992.

7. Guide to the Software Transfer Phase , ESA PSS-05-06 Issue 1 October 1994.

8. Guide to the Software Operations and Maintenance Phase, ESA PSS-05-07 Issue 1 December 1994.

9. Guide to Software Project Management, ESA PSS-05-08 Issue 1 June 1994.

10. Guide to Software Configuration Management, ESA PSS-05-09 Issue 1 November 1992.

11. Guide to Software Verification and Validation, ESA PSS-05-10 Issue 1 February 1994.

12. Guide to Software Quality Assurance, ESA PSS-05-11 Issue 1 July 1993.

13. Software Engineering Economics, B. Boehm, Prentice-Hall (1981)

14. Achieving Software Quality with Testing Coverage Measures, J.Horgan, S.London and M.Lyu, Computer, IEEE, September 1994.

This page is intentionally left blank.

APPENDIX C
DOCUMENT TEMPLATES

All documents should contain the following service information:

a - Abstract
b - Table of contents
c - Document Status Sheet
d - Document Change Records made since last issue

If there is no information pertinent to a section, the section should be omitted and the sections renumbered.

Guidelines on the contents of document sections are given in italics. Section titles which are to be provided by document authors are enclosed in square brackets.

C.1 URD TABLE OF CONTENTS

C.2 SSD TABLE OF CONTENTS

1 Introduction
- 1.1 Purpose of the document
- 1.2 Definitions, acronyms and abbreviations
- 1.3 References
- 1.4 Overview of the document

2 Model description
Describe the logical model using a recognised analysis method.

3 Specific Requirements
List the specific requirements, with attributes.
Subsections may be regrouped around high-level functions.
- 3.1 Functional requirements
- 3.2 Performance requirements
- 3.3 Interface requirements
- 3.4 Operational requirements
- 3.5 Resource requirements
- 3.6 Verification requirements
- 3.7 Acceptance testing requirements
- 3.8 Documentation requirements
- 3.9 Security requirements
- 3.10 Portability requirements
- 3.11 Quality requirements
- 3.12 Reliability requirements
- 3.13 Maintainability requirements
- 3.14 Safety requirements

4 System design
- 4.1 Design method
  Describe or reference the design method used.
- 4.2 Decomposition description
  Describe the physical model, with diagrams.
  Show the components and the control and data flow between them.

5 Component description
Describe each component.
Structure this section according to the physical model.
5.n [Component identifier]
5.n.1 Type
Say what the component is (e.g. module, file, program etc)
5.n.2 Function
Say what the component does.
5.n.3 Interfaces
Define the control and data flow to and from the component.
5.n.4 Dependencies
Describe the preconditions for using this component.
5.n.5 Processing
Describe the control and data flow within the component. Outline the processing of bottom-level components.
5.n.6 Data
Define in detail the data internal to components, such as files used for interfacing major components. Otherwise give an outline description.
5.n.7 Resources
List the resources required, such as displays and printers.

6 Feasibility and Resource Estimates
Summarise the computer resources required to build, operate and maintain the software.

7 User Requirements vs Software Requirements Traceability matrix
Provide tables cross-referencing user requirements to software requirements and vice versa.

8 Software Requirements vs Components Traceability matrix
Provide tables cross-referencing software requirements to components and vice versa.

C.3 SOURCE CODE DOCUMENTATION REQUIREMENTS

Each module should contain a header that defines:

Module Name
Author
Creation Date
Change History
    Version/Date/Author/Description
Function
    Say what the component does.
Interfaces
    Define the inputs and outputs
Dependencies
    Describe the preconditions for using this component.
Processing
    Summarise the processing using pseudo-code or a PDL

C.4 SUM TABLE OF CONTENTS

1 Introduction
  1.1 Intended readership
      Describe who should read the SUM.
  1.2 Applicability statement
      State which software release the SUM applies to.
  1.3 Purpose
      Describe the purpose of the document.
      Describe the purpose of the software.
  1.4 How to use this document
      Say how the document is intended to be read.
  1.5 Related documents
      Describe the place of the SUM in the project documentation.
  1.6 Conventions
      Describe any stylistic and command syntax conventions used.
  1.7 Problem reporting instructions
      Summarise the SPR system for reporting problems.

2 [Overview section]
Describe the process to be supported by the software, and what the software does to support the process, and what the user and/or operator needs to supply to the software.

3 [Installation section]
Describe the procedures needed to the software on the target machine.

4 [Instruction section]
From the trainee's viewpoint, for each task, provide:

(a) Functional description
What the task will achieve.

(b) Cautions and warnings
Do's and don'ts.

(c) Procedures
Include:
Set-up and initialisation
Input operations
What results to expect

(d) Probable errors and possible causes
What to do when things go wrong.

5 [Reference section]
From the expert's viewpoint, for each basic operation, provide:

(a) Functional description
What the operation does.

(b) Cautions and warnings
Do's and don'ts.

(c) Formal description
Required parameters
Optional parameters
Default options
Parameter order and syntax

(d) Examples
Give worked examples of the operation.

(e) Possible error messages and causes
List the possible errors and likely causes.

(f) Cross references to other operations
Refer to any complementary, predecessor or successor operations.

Appendix A Error messages and recovery procedures
List all the error messages.

Appendix B Glossary
List all terms with specialised meanings.

Appendix C Index (for manuals of 40 pages or more)
List all important terms and their locations.

C.5 STD TABLE OF CONTENTS

1 Introduction
- 1.1 Purpose of the document
- 1.2 Definitions, acronyms and abbreviations
- 1.3 References
- 1.4 Overview of the document

2 Build Report
Describe what happened when the software was built from source code

3 Installation Report
Describe what happened when the software was installed on the target machine.

4 Configuration Item List
List all the deliverable configuration items.

5 Acceptance Test Report Summary
For each acceptance test, give the:
user requirement identifier and summary
test report identifier in the SVVP/AT/Test Reports
test result summary

6 Software Problem Reports
List the SPRs raised during the TR phase and their status at STD issue.

7 Software Change Requests
List the SCRs raised during the TR phase and their status at STD issue.

8 Software Modification Reports
List the SMRs completed during the TR phase.

C.6 SPMP TABLE OF CONTENTS

1 Introduction
- 1.1 Purpose of the document
- 1.2 Definitions, acronyms and abbreviations
- 1.3 References
- 1.4 Overview of the document

2 Project organisation
- 2.1 Organisational roles and responsibilities
  Describe project roles, responsibilities and reporting lines.
- 2.2 Organisational boundaries and interfaces
  Describe the interfaces with customers and suppliers.

3 Technical Process
- 3.1 Project inputs
  List what documents will be input to the project, when and by who
- 3.2 Project outputs
  List what will be delivered, when and where
- 3.3 Process model
  Define the life cycle approach to the project.
- 3.4 Methods and tools
  Describe or reference the development methods used.
  Define and reference the design and production tools.
  Describe the format, style and tools for documentation.
  Define and reference the coding standards.
- 3.5 Project support functions
  Summarise the procedures for SCM and SVV and identify the related procedures and plans

4 Work Packages, Schedule, and Budget
- 4.1 Work packages
  For each work package, describe the inputs, tasks, outputs, effort requirements, resources allocated and verification process.
- 4.2 Schedule
  Describe when each work package will be start and end (Gantt Chart)
- 4.3 Budget
  Describe the total cost of the project.

C.7 SCMP TABLE OF CONTENTS

C.8 SVVP TABLE OF CONTENTS

Reviews and tracing section

1 Introduction
- 1.1 Purpose of the document
- 1.2 Definitions, acronyms and abbreviations
- 1.3 References
- 1.4 Overview of the document

2 Reviews
Describe the inspection, walkthrough and technical review procedures.

3 Tracing
Describe how to trace phase inputs to outputs.

Unit, Integration, System and Acceptance Testing Sections

1 Introduction
- 1.1 Purpose of the document
- 1.2 Definitions, acronyms and abbreviations
- 1.3 References
- 1.4 Overview of the document

2 Test Plan
- 2.1 Test items
  List the items to be tested.
- 2.2 Features to be tested
  Identify the features to be tested.
- 2.3 Test deliverables
  List the items that must be delivered before testing starts.
  List the items that must be delivered when testing ends.
- 2.4 Testing tasks
  Describe the tasks needed to prepare for and carry out the tests.
- 2.5 Environmental needs
  Describe the properties required of the test environment.
- 2.6 Test case pass/fail criteria
  Define the criteria for pass or failing a test case

3 Test Case Specifications (for each test case...)
- 3.n.1 Test case identifier
  Give a unique identifier for the test case.
- 3.n.2 Test items
  List the items to be tested.
- 3.n.3 Input specifications
  Describe the input for the test case.
- 3.n.4 Output specifications
  Describe the output required from the test case.
- 3.n.5 Environmental needs
  Describe the test environment.

4 Test Procedures (for each test procedure...)
- 4.n.1 Test procedure identifier
  Give a unique identifier for the test procedure.
- 4.n.2 Purpose
  Describe the purpose of the procedure.
  List the test cases this procedure executes.
- 4.n.3 Procedure steps
  Describe how to log, setup, start, proceed, measure, shut down, restart, stop, wrap-up the test, and how to handle contingencies.

5 Test Report template
- 5.n.1 Test report identifier
  Give a unique identifier for the test report.
- 5.n.2 Description
  List the items being tested.
- 5.n.3 Activity and event entries
  Identify the test procedure.
  Say when the test was done, who did it, and who witnessed it.
  Say whether the software passed or failed each test case run by the procedure.
  Describe the problems.