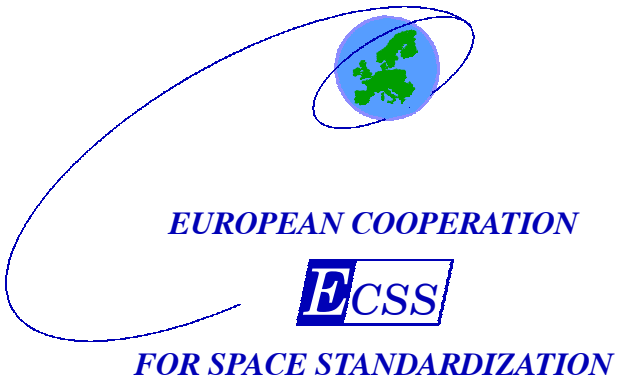
ECSS-E-40A
13 April 1999



Space engineering

Software

ECSS Secretariat
ESA-ESTEC
Requirements & Standards Division
Noordwijk, The Netherlands

Published by: ESA Publications Division
ESTEC, P.O. Box 299,
2200 AG Noordwijk,
The Netherlands

ISSN: 1028-396X

Price: DFl 35

Printed in The Netherlands

Foreword

This Standard is one of the series of ECSS Standards intended to be applied together for the management, engineering and product assurance in space projects and applications. ECSS is a cooperative effort of the European Space Agency, National Space Agencies and European industry associations for the purpose of developing and maintaining common standards.

Requirements in this Standard are defined in terms of what shall be accomplished, rather than in terms of how to organize and perform the necessary work. This allows existing organizational structures and methods to be applied where they are effective, and for the structures and methods to evolve as necessary without rewriting the standards.

The formulation of this Standard takes into account the existing ISO 9000 family of documents, and the ISO 12207:1995 standard.

This Standard has been prepared by the ECSS Software Engineering Working Group, reviewed by the ECSS Technical Panel and approved by the ECSS Steering Board.

(This page is intentionally left blank)

Contents list

Foreword **3**

1 Scope **7**

2 Normative references **9**

3 Terms, definitions and abbreviated terms **11**

3.1 Terms and definitions 11
3.2 Abbreviated terms 12

4 Space system software engineering **15**

4.1 Introduction 15
4.2 Organization of this standard 16
4.3 Space system software engineering processes 16
4.4 Relation to ECSS-M and ECSS-Q standards 21
4.5 Verification engineering 22
4.6 Tailoring of this standard 23

5 General requirements **25**

5.1 Introduction 25
5.2 System engineering processes related to software 25

5.3 Software management 29
5.4 Software requirements engineering process 33
5.5 Software design engineering process 35
5.6 Software verification and validation (qualification) process 39
5.7 Software operations engineering process 45
5.8 Software maintenance process 47

6 Special requirements 51

6.1 Introduction 51
6.2 Space segment software 51
6.3 Ground segment software 53
6.4 Software reuse 53
6.5 Man-machine interfaces 54
6.6 Critical software 55

Annex A (normative) Software documentation 57

A.1 Introduction 57
A.2 The Requirements Baseline (RB) 58
A.3 Technical Specification (TS) 59
A.4 Design Justification File (DJF) 60
A.5 Design Definition File (DDF) 62
A.6 System level documentation 63

Annex B (informative) Requirement cross references 65

Annex C (informative) References to other ECSS Standards 67

Bibliography 69

Figures

Figure 1: The recursive customer - supplier model 17
Figure 2: Overview of the software development processes 18
Figure 3: Process constraints 18
Figure 4: Accommodation of different software life cycles 19
Figure A-1: Overview of software engineering documents 57

1

Scope

This software engineering standard concerns the "Product software", i.e. software that is part of a space system product tree and developed as part of a space project.

This standard is applicable to all the elements of a space system, including the space segment, the launch service segment and the ground segment.

This standard covers all aspects of space software engineering including requirements definition, design, production, verification and validation, and transfer, operations and maintenance.

It defines the scope of the space software engineering process and its interfaces with management and product assurance, which are addressed in the Management (-M) and Product assurance (-Q) branches of the ECSS System, and explains how they apply in the software engineering process.

This standard reflects the specific methods used in space system developments, and the requirements for the software engineering process in this context. Together with the requirements found in the other branches of the ECSS Standards, this standard provides a coherent and complete framework for software engineering in a space project.

This standard is intended to help customers in formulating their requirements and suppliers in preparing their response and implementing the work.

This standard is not intended to replace textbook material on computer science or technology, and such material has been avoided in this standard. The readers and users of this standard are assumed to possess general knowledge of computer science.

The scope of this standard is the software developed as part of a space project, i.e. "Space system product software". It is not intended to cover software developments out of scope with the ECSS System of standards. An example is the development of commercial software packages, where software is developed for a (large) volume market and not just for a single customer, and the main requirement analysis consists of market analysis, combined with a marketing strategy.

Other classes of software products not covered are: management information systems (e.g. finance, planning), technical information systems (e.g. CAD/CAM, analysis packages) and supporting software products for documentation systems, database systems, spread-sheets. These usually result from the procurement or adaptation of existing commercial products, and not part of the space system development. Such software products will, however, often be part of a supporting infrastructure for space systems.

(This page is intentionally left blank)

2

Normative references

This ECSS Standard incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text, and publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these apply to this ECSS Standard only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

ECSS-P-001	Glossary of terms
ECSS-E-00	Space engineering - Policy and principles
ECSS-E-10	Space engineering - System engineering
ECSS-M-00	Space project management - Policy and principles
ECSS-M-10	Space project management - Project breakdown structures
ECSS-M-20	Space project management - Project organization
ECSS-M-30	Space project management - Project phasing and planning
ECSS-M-40	Space project management - Configuration management
ECSS-M-50	Space project management - Information/documentation management
ECSS-M-60	Space project management - Cost and schedule management
ECSS-M-70	Space project management - Integrated logistic support
ECSS-Q-20	Space product assurance - Quality assurance
ECSS-Q-80	Space product assurance - Software product assurance
ISO/IEC 12207:1995	Information technology - Software life cycle processes
ISO 8402:1994	Quality management and quality assurance - Vocabulary

These are the level 3 documents referenced by this standard.

ECSS-E-40-01	Space engineering - Space segment software (to be published)
ECSS-E-40-03	Space engineering - Ground segment software (to be published)

(This page is intentionally left blank)

3

Terms, definitions and abbreviated terms

3.1 Terms and definitions

Terms for which the ECSS-P-001 definitions have been further expanded to cover software specific issues (without changing the general definition in ECSS-P-001), and terms particular for ECSS-E-40:

3.1.1 (Top-level) Architecture

The highest level(s) structure of the components of a program or system, their interrelationships, and principles and guidelines governing their design and evolution over time.

3.1.2 Embedded software

There is no commonly agreed consistent definition of this term. It is sometimes used to denote the self-evident observation that software, at varying levels, is part of a system. The term is also sometimes used to emphasize that the software in question has extensive hardware interface requirements or real-time requirements. The term is purely descriptive. Use of the term is therefore discouraged.

3.1.3 Margin philosophy

The margin philosophy describes the rationale for margins allocated to the performance parameters and computer resources of a development, and how these margins shall be managed during the execution of the project.

3.1.4 Migration

Porting of a software product to a complete new opportunely environment.

3.1.5 Singular input

Individual parameter stress testing.

3.1.6 Software component

General term for a part of a software system. Components may be assembled and decomposed to form new components. In the production phase, components are implemented as modules, tasks or programs, any of which may be configuration items. This use of the term is more general than in ANSI/IEEE parlance, which defines a component as a “basic part of a system or program”; in ECSS-E-40, components may not be “basic” as they can be decomposed.

3.1.7 Software item

See Software product.

3.1.8 Software intensive system

A space software product where the dominant part of the constituents are software elements. In such systems, sub-systems consists mainly of software. For this type of system, the majority of interfaces are software-software interfaces.

3.1.9 Software observability

The property of a system for which observations of the output variables always is sufficient to determine the initial values of status variables.

3.1.10 Software product

The set of computer programs, procedures and possibly associated documentation and data.

3.1.11 Software unit

A separately compilable piece of code (ISO/IEC 12207). In ECSS-E-40 no distinction is made between a software unit and a database; both are covered by the same requirements.

3.1.12 Stress test

A test that evaluates a system or software component at or beyond the limits of its specified requirements.

3.1.13 Validation

Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled (ISO 8402:1994).

The validation process (for software): to ensure that the requirements baseline functions and performances are correctly and completely implemented in the final product.

3.1.14 Verification

Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled (ISO 8402:1994).

The verification process (for software): to establish that adequate specifications and inputs exist for any activity, and that the outputs of the activities are correct and consistent with the specifications and input.

3.2 Abbreviated terms

The following abbreviations are defined and used within this standard.

Abbreviation	Meaning
AR	Acceptance Review
NOTE	The term SW-AR may be used for clarity to denote ARs that solely involve software products.
CDR	Critical Design Review
NOTE	The term SW-CDR may be used for clarity to denote CDRs that solely involve software products.
CJF	Change Justification File

COTS Commercial off-the-shelf Software

NOTE It denotes finished software products, that are procured from third parties.

CPU Central Processing Unit

DDF Design Definition File

DJF Design Justification File

ICD Interface Control Document

IRB Interface Requirements Baseline

IRD Interface Requirements Document

ISV Independent Software Validation

ISVV Independent Software Verification and Validation

MMI Man Machine Interfaces

MOTS Modifiable off-the-Shelf

MP Maintenance Plan

OP Operational Plan

ORR Operational Requirements Review

PDR Preliminary Design Review

NOTE The term SW-PDR may be used for clarity to denote PDRs that solely involve software products.

QR Qualification Review

NOTE The term SW-QR may be used for clarity to denote QRs that solely involve software products.

RB Requirements Baseline

SDE Software Development Environment.

NOTE Software tools that are supporting the software engineering process.

SRR System Requirements Review

NOTE The term SW-SRR may be used for clarity to denote SRRs that solely involve software products.

SW Software

TS Technical Specification

(This page is intentionally left blank)

4

Space system software engineering

4.1 Introduction

This clause 4 introduces the structure of this standard and the framework of the space software engineering process that form its basis.

The context of space software engineering is the overall space system engineering process. This clause 4 defines the general relationships between the software engineering processes and the general engineering processes of space systems.

The software engineering standard differs from the other engineering disciplines covered by ECSS in one important aspect: software does not in itself produce heat, have mass or any other physical characteristics. The software engineering activity is a purely intellectual activity and a principle output of the activity is documentation. If the software code itself is considered as a specialized form of electronic documents, *all* visible outputs are in fact documentation.

It follows that this standard focuses on requirements for the structure and content of the documentation produced.

Software is used for the implementation of highly complex functions. The ability to deal with a high level of complexity in a flexible way makes software an essential and increasing part of space segment and ground segment products. In space systems, software engineering is found at all levels ranging from system level functions down to the firmware of a space system part.

Therefore the requirements engineering process, in which the software requirements and specifications are defined, has a special emphasis in this standard. The software requirements engineering process consumes a large and often underestimated amount of effort in the development of software for space systems.

As a result of the complexity of the functional and performance requirements, special measures and emphasis are required for software verification and validation, especially for space segment software. The functions assigned to software may be critical to the space mission.

The maintenance of software for space systems also poses special problems, because they imply operational lifetimes that far exceed what is expected of general computer software products. For the space segment, this is further complicated by the fact that software in general is the only part of the space segment that undergoes major maintenance and repair, sometimes even re-design, after launch. In ex-

treme cases, the space system mission itself is redesigned, implementing new space segment software after launch. Ground segment software is similarly characterized.

4.2 Organization of this standard

This standard is organized in two main parts:

- **General Requirements.** These are the core normative requirements for any space system software engineering activity.
- **Special Requirements.** These are additional requirements for specific application areas. These requirements are always applicable, but are only active in developments where the addressed disciplines or application areas occur. This separation serves to make the general requirements core compact and clear.

Software documentation summaries are included in annex A for information.

In the preparation of this standard the ISO/IEC 12207:1995 standard has been used extensively, providing a common internationally recognized framework for the terminology and engineering process description. For completeness, a cross-reference between ECSS-E-40 and ISO/IEC 12207:1995 is included in annex B.

4.3 Space system software engineering processes

The software engineering processes regulated by this standard are based on the definitions and requirements given in the ECSS-M series (in particular M-20, M-30, M-40 and M-50), and the general engineering process requirements of ECSS-E-00. These requirements have been used to define the top-level software engineering processes. This general framework defines the processes (that are later treated in detail in the following subclauses) and the top-level interface between the software engineering processes and other space development processes.

The fundamental principle of this standard is the *customer-supplier* relationship, assumed for all software developments. The organizational aspects of this are defined in ECSS-M-20. The customer is, in the general case, the procurer of two strongly associated products: the hardware and the software for a *system, subsystem, set, equipment or assembly* (see ECSS-E-00). The concept of the customer-supplier relationship is applied *recursively*, i.e. the customer may himself be a supplier to a higher level in the space system as shown in Figure 1. The software customer therefore has two important interfaces. The first interface is to his software and hardware suppliers and this includes the functional analysis required for the adequate allocation of function and performance requirements to his suppliers. The other where he is in his role as supplier at a higher level, where he shall ensure higher level system requirements are adequately taken into account.

The customer derives the functional and performance requirements for the hardware and software, based on system engineering principles and methods. The customer also controls the interface between the software and hardware. Software items are defined in the system breakdown at different levels. Nevertheless, it is important to manage the software-software interfaces irrespective of the level at which they occur. The *customer's requirements* are defined by this initializing process, and provide the starting point for the software engineering.

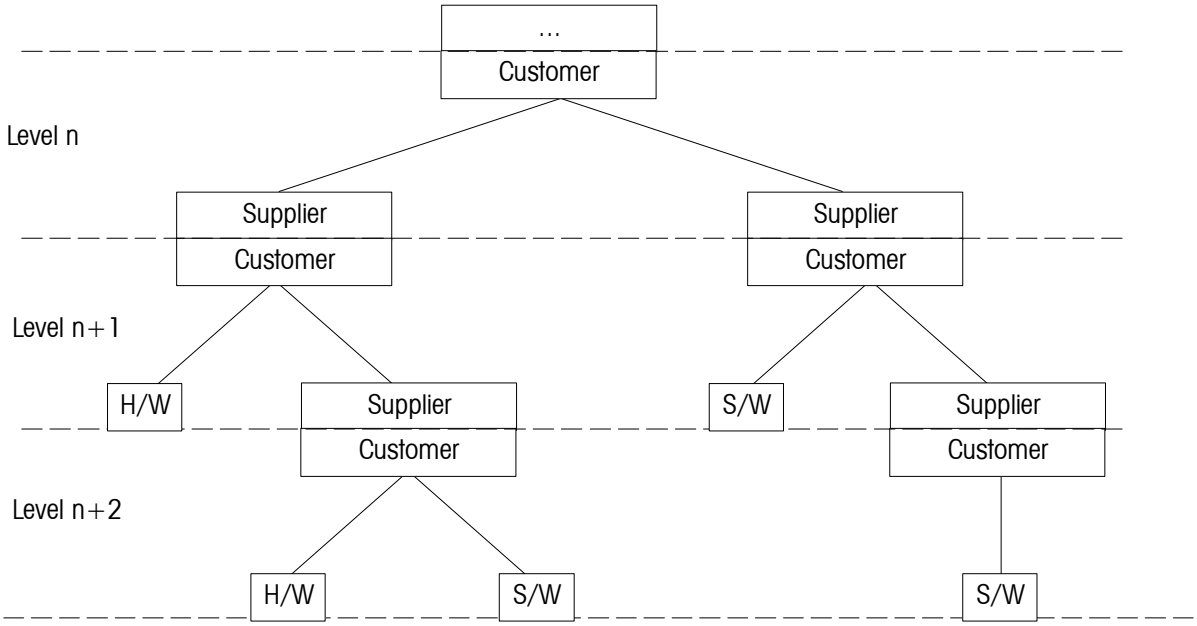


Figure 1: The recursive customer - supplier model

Reviews are the main interaction points between the customer and supplier. The reviews relevant to the software engineering process are the SRR, PDR, CDR, QR and AR, as defined by ECSS-M-30. All reviews are applicable to software. The reviews occur at different levels in the customer-supplier hierarchy and are sequenced according to the overall system level planning. This standard is designed to be applied at any level, without explicit assumptions of how these reviews shall be integrated with other reviews in the development of a space system. An overview is shown in Figure 2. The commonly designated mission phases (e.g. 0, A, B) are used for the overall mission phases, and play no direct role in the software engineering activities as such. This means that the software engineering processes, together with their reviews and attached milestones as defined in this standard, are not to be scheduled as the higher-level system mission phases. They should be planned in relation to the immediate higher level development processes.

The notion of engineering processes is fundamental to this standard, as the processes provide the means to describe the overall constraints and interfaces to the software engineering process at system level, and at the same time, provide the necessary freedom to the supplier to implement the individual activities implied by the processes. The freedom given to the supplier to implement the engineering processes is especially important for software engineering, because of the requirement to organize the work in accordance with a well defined software life cycle[1]. There is a requirement to accommodate different types of software life cycles, both for reasons of efficient organization of the work and also for reasons related to competitiveness and choice of software engineering technology. Different software life cycle types can be accommodated within the requirements in this standard. Figure 3 illustrates the constraints imposed. Figure 4 shows examples of variations within these constraints.

1 Software life cycles

There is an abundant technical literature on the different types of software life cycles. This standard does not recommend any specific life cycle. Instead the requirements to be used in the choice of a life cycle for a given software project are to be defined, such that the software life cycle remains consistent with the overall organization and management of the space project organization and management, and that the life cycle outputs remain consistent with the requirements of ECSS. Commonly known classes of software life cycles are:

- the Waterfall life cycle,
- the Incremental life cycle, and
- the Evolutionary life cycle.

This standard has been verified for compatibility with these, but the choice is not limited to these three types.

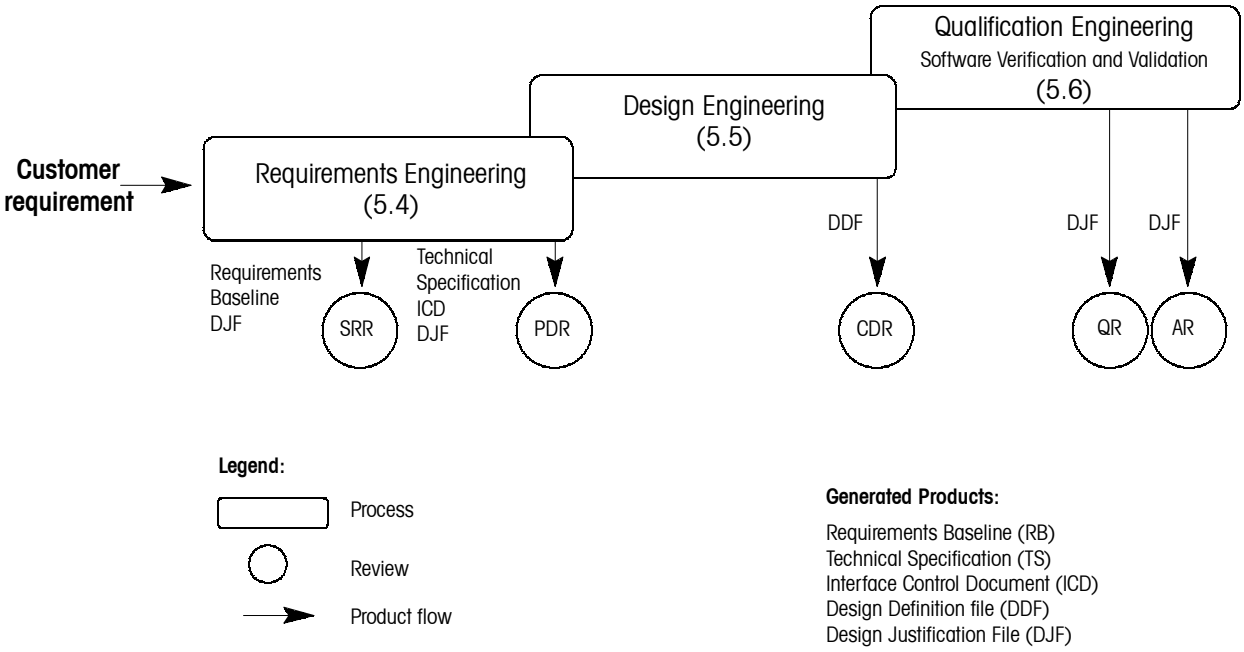


Figure 2: Overview of the software development processes

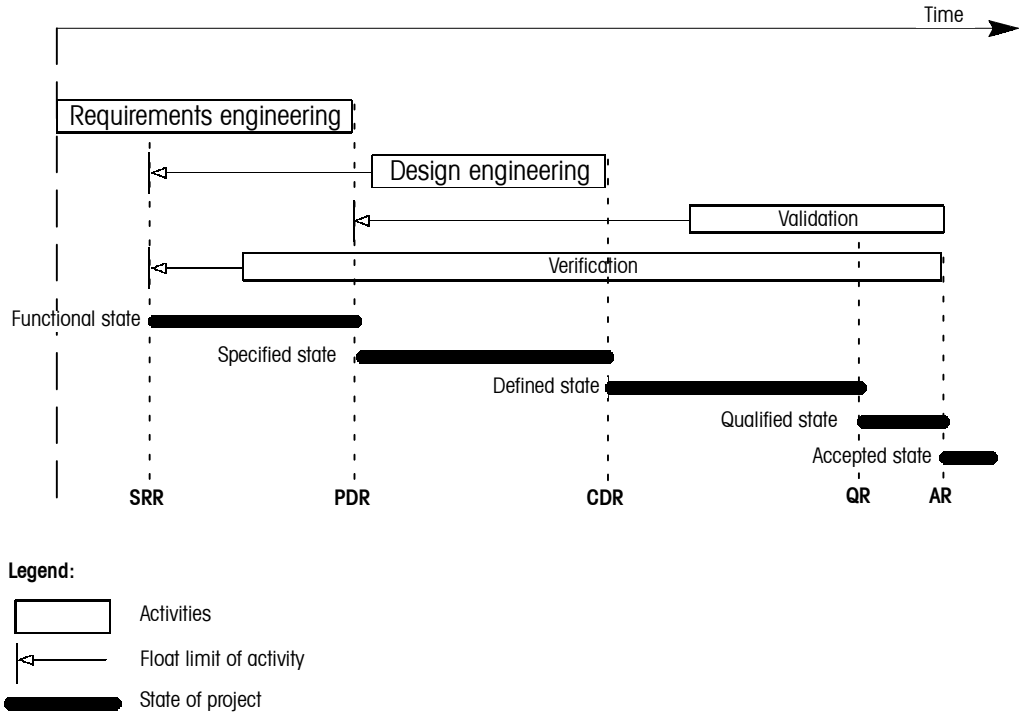


Figure 3: Process constraints

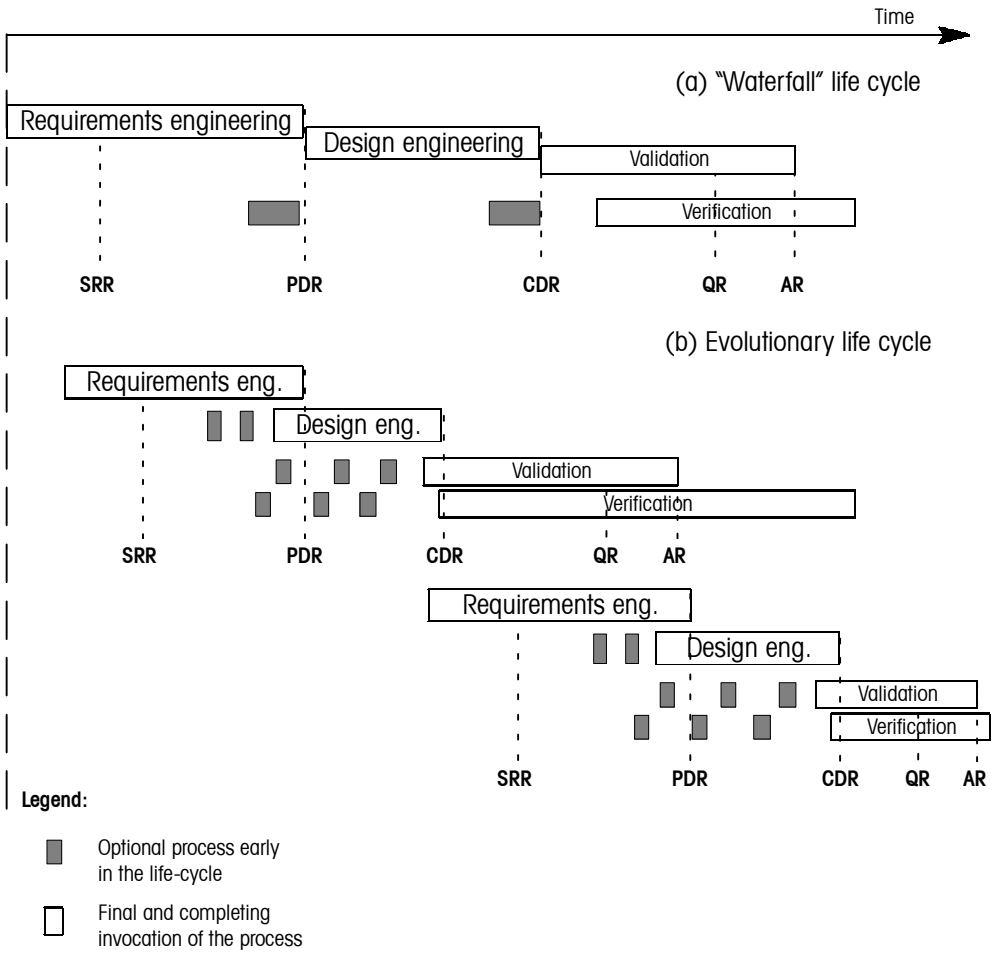


Figure 4: Accommodation of different software life cycles

4.3.1 Software requirements engineering process

The first part of the requirements engineering process produces the information required for input to the System Requirements Review (SRR). This establishes the functional and performance *requirements baseline* of the software development, and the preliminary interface requirements.

A second part of the software requirements engineering process is the elaboration of the *technical specification*, which is the supplier's response to the Requirements Baseline. This process may start in parallel or after the elaboration of the Requirements Baseline. The software product tree is defined by this process. The Technical Specification shall contain a precise and coherent definition of functions, performances, cost, schedule and implementation plans for all levels of the software to be developed. The preliminary Interface Control Document (ICD) is generated by this process.

During the software requirements engineering activity, the result of all significant trade-offs, feasibility analyses, make-or-buy decisions and supporting technical assessments shall be documented in a *Design Justification File* (DJF)

The software requirements engineering process is completed by the *Preliminary Design Review* (PDR). The input to the PDR is the technical specification, preliminary ICD and the DJF. The top-level architectural design is reviewed at the PDR. If the customer requires an additional review of the software architecture, this may be specifically included in the organization of the project.

The state of the software development after PDR is called "specified state".

4.3.2 Software design engineering process

The "Design and configuration engineering process" mentioned in ECSS-E-10 is in software developments referred to as the "Design engineering process".

This process should not start before SRR. It may start before the PDR, but it is after the PDR when the results of the Requirements Engineering Process are reviewed and baselined to be used as inputs to the Design Engineering Process.

The process produces the design of each element of the software product tree, in response to the requirements contained in the Technical Specification, ICD and DJF. All elements of the software design shall be documented in the *Design Definition File* (DDF). The DDF contains all the levels of design engineering results, including software code listings.

The rationale for important design choices, and analysis and test data that shows that the design meets all requirements, is added to the DJF by this process. The results of this process are the input to the *Critical Design Review* (CDR). The CDR signals the end of the design phase. For large software projects, all software subsystems shall undergo a CDR before they are integrated with the next highest level in the system hierarchy. Large software developments should be partitioned in smaller manageable projects that are managed like any other subsystem development in space projects.

The state of the software project after CDR is called "defined state".

4.3.3 Software verification and validation (qualification) process

The software verification and validation process may start any time after the SRR.

This process is intended to confirm that the customer's requirements have been properly addressed, that all requirements have been met and that all design constraints are respected.

The result of this process is included in the DJF.

The process shall include a *Qualification Review* (QR), with the DJF as input.

The state of the software project after QR is called the "qualified state".

A sub-process of this process is the transfer and acceptance of the software to the customer. This latter sub-process is completed by an *Acceptance Review* (AR), that may take place after QR. The Acceptance Review is a formal event in which the software product is evaluated in its operational environment. It should be carried out after the software product has been installed and transferred to the customer and installed on an operational basis. Software validation activities terminate with the Acceptance Review.

This state of the software after AR is called the "accepted state".

NOTE The term "qualification engineering" is often used synonymously with the term "verification engineering" in projects delivering hardware. For the sake of clarity, "qualification engineering" is used in this Standard to denote "the total set of verification and validation activities". This should be consistent with other ECSS Standards outside the software engineering discipline, and to avoid confusion with the general verification engineering activities that are invoked in many places in software projects.

4.3.4 Software operations engineering process

The operations process may start after completion of the Acceptance Review of the software. Since software products form an integrated part of a space system, the phasing and management of operations shall be determined by the overall system requirements and applied to the software products. The operations engineering processes are not directly connected to the overall mission phase E, but are, in-

stead, determined by the requirement at system level to operate the software product at a given time.

4.3.5 Software maintenance process

This separate process is started after the completion of the AR.

This process is activated when the software product undergoes any modification to code or associated documentation as a result of correcting an error, a problem or implementing an improvement or adaptation. The process ends with the retirement of the software product.

NOTE The software analysis process (as a general engineering process defined in the ECSS-M standards) is invoked by the requirements and design engineering processes. No separate output is produced by this process. The results produced by the analysis process are integrated with the requirements and design engineering outputs.

4.4 Relation to ECSS-M and ECSS-Q standards

This subclause discusses how this standard interfaces with other ECSS series, namely the ECSS-Q series of standards (Product assurance) and the ECSS-M series of standards (Project management).

4.4.1 Software product assurance

Requirements on software product assurance are defined in ECSS-Q-80, which is the entry level document of the ECSS-Q series (Product assurance) for software projects.

4.4.2 Software project management

ECSS-M standards define the requirements to be applied to the management of space projects. The following subclauses describe how the ECSS-M standards apply to the management of software projects.

In addition, normative requirements which cannot be found in M-series, because they are specific to software project management, are provided in subclause 5.3.

4.4.2.1 ECSS-M-00: Policy and principles

ECSS-M-00 is a top-level document which defines project management principles and general requirements to be applied to all aspects of a space project including software.

Risk management is covered by ECSS-M-00. Some risk factors, such as exceeding the assigned memory budget or CPU load, are specific to software.

The terms “customer” and “supplier” used in ECSS-E-40 are defined in ECSS-M-00A, subclause 5.2.

4.4.2.2 ECSS-M-10: Project breakdown structures

The provisions of ECSS-M-10 shall apply to software, taking account of the specific features of the software.

The products of a software project are usually documents (including code) but may also include computer devices in the case of software intensive systems.

“Model matrix” in ECSS-M-10A, subclause 5.2, is concerned with material models and therefore is not relevant to software. However, these models shall not be confused with logical and physical software models which may be produced as part of a software specification and design, respectively.

4.4.2.3 ECSS-M-20: Project organization

ECSS-M-20 provides a clear definition of the role and responsibility of each party to the project. ECSS-M-20 covers the requirements for software projects.

4.4.2.4 ECSS-M-30: Project phasing and planning

ECSS-M-30 defines the phasing and planning requirements for an entire space project. But some requirements also affect software development, because they are specified in ECSS-M-30 as applicable at any level of the project organization.

Project phases as defined in ECSS-M-30 are top-level (mission) phases, used to structure the whole space project. They do not apply recursively to software development. They should not be confused with the phases which are defined to give structure to software development life cycles, and for which no specific definition is imposed in ECSS-E-40.

Similarly, the reviews as defined in ECSS-M-30 do not apply directly to software even though the concept of review applies recursively to all levels of a space project.

The terms “SRR”, “PDR”, “CDR”, “QR” and “AR” are defined in ECSS-M-30 , and these are reused to define joint technical reviews for a software development as described in subclause 4.3.

These reviews shall be synchronized with higher level reviews in a way which is project dependant. In clause 6, interface requirements are given for particular types of software. Requirements concerning phasing and reviews, and which are specific to software are given in subclause 5.3.

4.4.2.5 ECSS-M-40: Configuration management

The requirements, also for software developments, are contained in ECSS-M-40.

One facet of software configuration management is that all configuration items may be regarded as documents (even code). Due to this, ECSS-M-40 is strongly connected to ECSS-M-50 when applied to software, making software configuration control a highly automated process, and which may be simplified with respect to the general approach.

4.4.2.6 ECSS-M-50: Information/documentation management

The objectives of information and documentation management are particularly required to ensure the accessibility of information to all parties of the project and to ensure the coherence of this information. These objectives also apply to software projects, and the relevant requirements are to be found in ECSS-M-50.

4.4.2.7 ECSS-M-60: Cost and schedule management

ECSS-M-60 contains requirements on software projects, although requirements on schedule management are more directly applicable to software, than costing requirements.

4.4.2.8 ECSS-M-70: Integrated logistic support

ECSS-M-70 is mainly of concern to large or software-intensive systems.

4.5 Verification engineering

Verification engineering supports requirements definition, design, coding, integration, operations and maintenance. The requirements of ECSS-E-10A, subclause 4.6, are applicable to the software with the following interpretation:

> “Verification engineering determines whether the software products fulfil the requirements and conditions imposed. Verification activities should be integrated, as early as possible, with requirements engineering and design engineering.”

Verification engineering activities may be executed with varying degrees of independence. The degree of independence may range from the same person, or different person in the same organization, to a person in a different organization, with varying degrees of separation. In the case where the processes are executed by an organization independent of the supplier, it is called Independent Software Verification and Validation (ISVV). The individual verification requirements for the various software engineering processes are given in clause 5.

4.6 Tailoring of this standard

The general requirements for selection and tailoring of applicable standards are defined in ECSS-M-00.

(This page is intentionally left blank)

5

General requirements

5.1 Introduction

This clause 5 defines the requirements for engineering software for space systems. They shall be applied to any space projects producing computer software.

Each requirement can be identified by a hierarchical number. The text of the requirement is followed, where necessary, by further explanation of the aim. For each requirement the associated output is given in the output section. With each output (e.g. “a.”, “b.”), the required destination (document) of the output is indicated in brackets together with the corresponding review. For example: “[DDF, DJF; QR]” denotes an output to the Design Definition File and the Design Justification File. The output in this example is required for the qualification review.

5.2 System engineering processes related to software

5.2.1 Introduction

This subclause 5.2 describes activities which are under the customer responsibility. The customer shall be responsible for the delivery of a system in which the developed software will be integrated (refer to the recursive customer-supplier model described in 4.3).

The customer activities described here are only those that require introduction of additional requirements particular for software development:

- system requirement analysis;
- system partitioning;
- system level requirements for software verification and validation;
- system level integration of software;
- software operations.

System level documentation is a prerequisite to the requirements engineering of the software. The aim of the requirements given in this subclause shall ensure the completeness and correctness of the customer’s system level documentation and to establish a complete and verified requirements baseline for the software project.

5.2.2 System requirements analysis

This activity consists of the following tasks:

- system requirements specification;
- system and functional criticality analysis.

5.2.2.1

System requirements shall be derived from an analysis of the specific intended use of the system to be developed. All system requirements shall be documented.

EXPECTED OUTPUT:
a. *functions and performance requirements of the system [RB; SRR];*
b. *operations and maintenance requirements [RB; SRR];*
c. *interface requirements [IRB; SRR];*
d. *design constraints and verification and validation requirements [RB; SRR];*
e. *identification of lower level software engineering standards that will be applied [RB; SRR]..*

5.2.2.2

System criticality analysis and critical functions analysis shall be performed for the system.

EXPECTED OUTPUT:
a. *overall safety and reliability requirements of the software to be produced [RB; SRR];*
b. *critical function identification and analysis [RB; SRR].*

5.2.3 System partitioning

5.2.3.1 Introduction

As part of the System Design process, a physical architecture and design (including HW, SW and manual operations) of the system shall be derived: this is called top-level partitioning of the system. This system design is derived from an analysis of the requirements on the system and its functions. Conformance to the system design with all system requirements shall be verified. All system requirements shall be allocated (and shall be traceable) to the different system design partitions.

5.2.3.2

A top-level partitioning of the system shall be established. This partitioning shall identify items of hardware, software and manual operations. It shall be ensured that all the system requirements are allocated to items. Hardware configuration items, software configuration items, and operating manual shall be subsequently identified from these items. The system partitioning and the system requirements allocated to the individual items shall be documented.

EXPECTED OUTPUT:
a. *system partition with definition of items [RB; SRR];*
b. *software/hardware interface requirements [IRD; SRR];*
c. *system configuration items list [RB; SRR];*
d. *traceability to system partitioning [DJF; SRR].*

5.2.4 System level requirements for software verification and validation

5.2.4.1 Introduction

The general ECSS approach to the verification process is described in ECSS-E-10A clauses 4 and 5, covering both verification and validation activities.

NOTE 1 It is assumed that for all space projects certain verification and validation activities will always be applied. Therefore the ISO/IEC 12207 requirements to determine if validation and verification is required have no equivalent here (tailoring of ISO/IEC 12207).

NOTE 2 The supplier process verification is handled as part of the ECSS management and is therefore not covered as part of the software activities (tailoring of ISO/IEC 12207:1995 6.4.2.2). In addition, ECSS-Q standards provide requirements related to the supplier process assessment which are not repeated in this standard.

5.2.4.2
The customer shall adapt the requirement for qualification engineering given in subclause 5.6 to system level requirements.

AIM: To identify the customer's verification and validation process requirements at system level, and to prepare for software acceptance and software integration by introducing the corresponding verification and validation process requirements in the requirements baseline.

EXPECTED OUTPUT: *Verification and validation process requirements [RB; SRR].*

5.2.4.3
The customer shall verify the requirements baseline.

In cases where the customer's product is an integrated hardware and software product, this shall be performed as required by the ECSS system engineering standards. In cases where the customer's product is a software product, the customer shall apply this standard in his role as "supplier" at a higher level in the product tree.

EXPECTED OUTPUT: *Requirements justifications [Customer DJF for system level].*

NOTE This output is a special case:

The output is not part of the customer-supplier interface to the software engineering processes, and is therefore not part of any milestone input. Instead the output is part of the customer's own system DJF, and should be used only by the customer in his role as supplier to the next higher level in the product tree. The output is mentioned here for completeness only.

5.2.5 System level integration of software

This activity consists of the following tasks, which the customer shall perform or support as required for system level activities, in the customer's role as supplier of the overall system:

- identification of required software observability for the support of software integration;
- required control and data interfaces for system level integration;
- data media requirements for integration;
- functional software integration support requirements;
- system level inputs required for the supplier's preparation of integration of the software;
- supplier software engineering outputs required for system level integration preparation;
- required supplier for system level integration.

The software product integration at system level takes place only after completion of the CDR of the software product, whereas the engineering, design and planning

activities supporting the later system level integration are completed for the product before CDR.

5.2.5.1
If a software product is integrated into a system, all software observability requirements, necessary to facilitate the software integration, shall be specified by the customer.

EXPECTED OUTPUT: *Software observability requirements [RB; SRR].*

5.2.5.2
If the software is integrated into a system, all the interfaces between the software and the system shall be specified by the customer, including the static and dynamic aspects, for nominal and degraded modes (e.g. behaviour in case of failure).

The external interfaces, specific to software integrated in a system, may be:

- software interface with other software on the system (operating system, files, database management system or other applications software);
- hardware interfaces to the specific hardware configuration;
- communication interfaces (particular network protocol for example).

EXPECTED OUTPUT: *System level interface requirements [IRD; SRR].*

5.2.5.3
The customer shall identify the interface data medium and prepare the requirements accordingly.

For example, the interface data may be defined and structured in such a way that interface data may be automatically acquired by the supplier SDE. Trade-offs shall be performed, taking into account the number of software packages in the system, the evolution of interface data, and the number of interface data sets.

EXPECTED OUTPUT: *System level data interfaces [IRD; SRR].*

5.2.5.4
If necessary, the customer shall define specific development constraints on the supplier required to support the integration of the software into the system.

When the software is integrated into a system, some harmonization constraints may be required such as:

- specification of the operating system to be used;
- specification of COTS to be used (e.g. Database, MMI generator);
- specification of the SDE to be used.

EXPECTED OUTPUT: *Development constraints [RB; SRR].*

5.2.5.5
Where necessary, the customer shall identify and plan the specific inputs he shall provide to the supplier to support the integration of the software into the system, and he shall prepare the requirements baseline accordingly.

When the software is integrated into a system, the customer may provide the supplier with specific inputs for validating the software in a representative environment. These inputs can be:

- breadboard or computer model;
- a simulator of the hardware and/or software environment.

EXPECTED OUTPUT: *System level integration support products [IRD; SRR].*

5.2.5.6
The customer shall identify and plan the specific outputs which the supplier shall deliver to support the integration of the software into the system, and he shall prepare the requirements baseline accordingly.

When software is integrated into a system, some prototype versions or intermediate versions may be required by the customer to prepare for integration. Functionalities and delivery dates required for each of these versions shall be defined.

EXPECTED OUTPUT: *System level integration preparation requirements [IRD; SRR].*

5.2.5.7
The customer shall plan to support the software supplier in integration of the software at system level, and prepare the requirements baseline accordingly.

This can include activities such as: training, maintenance, configuration, test support.

EXPECTED OUTPUT: *System level integration support requirements [IRD; SRR].*

5.2.6 Software operations

Since software products are an integrated part of the space system, the phasing and management of operations shall be determined by the overall system requirements and shall be applied to software products.

5.2.6.1
For all software products the customer shall identify the supplier who performs the operations process (called “operator” hereafter).

5.2.6.2
The customer shall establish system requirements for the operation of software products. The supplier of the operation service (operator) shall prepare the response to the operational requirements, and document his response in accordance with the requirements in subclause 5.7. The supplier’s response shall be agreed with the customer in an Operational Requirements Review (ORR), intended to release the operational plans for execution.

5.3 Software management

5.3.1 Introduction

Most of the specific requirements for the management and control of space systems software projects exist in the ECSS-M series of documents. They are not repeated here. In addition, the software product assurance requirements described in ECSS-Q-80 are also used for the control of space systems software projects. The requirements described in this subclause 5.3 are necessary for the engineering and control of software development in a space systems project, and they bridge the gap between the other ECSS Standards mentioned above and the software engineering activities in space projects.

The management and control described in this subclause are:

- software life cycle;
- interface management;
- technical budget and margin management.

The requirements in this subclause 5.3 shall be applied to any type of software in a space project.

As defined in more detail in following subclauses, the software shall undergo the overall software milestone reviews SRR, PDR, CDR, QR and AR as a minimum. Further reviews (e.g. review of project plans, before the PDR) may be required by the customer and they should follow requirements mentioned in subclause 5.6.8.

5.3.2 Software life cycle

To assure effective phasing and planning, the software development life cycle shall be broken into phases, each having its associated milestones.

5.3.2.1
The software supplier shall define and follow a software development life cycle in accordance with subclause 4.3, and covering all activities from the statement of requirement to the entry of the software into service. The definition of the life cycle shall be associated with choices of techniques used during the development (e.g. database management system, extensive product reuse), with the risks inherent to the project (e.g. highly changeable specification, stringent schedule constraints) and with synchronization points with the upper level.

The choice of software life cycle shall be in accordance with the overall project requirements, and the process model of subclause 4.3 and ECSS-M-30 shall be used.

EXPECTED OUTPUT: *Project software development life cycle definition, included in the software project development plan [TS; PDR].*

5.3.2.2
The development life cycle shall define the input and output required for each phase and its associated milestones.

The output for each phase may consist of documents in complete or outline versions, including the results of verification of the technical outputs of the phase.

Milestones are the joint technical reviews required by the customer (SRR, PDR, CDR, QR and AR) and internal reviews at the supplier level.

The outputs for each milestone are documents submitted for examination and which are explicitly listed in the software life cycle definition.

5.3.2.3
The interface between development and maintenance (e.g. documents to be produced, tools to be kept in maintenance) shall be identified for the software life cycle.

AIM: Define and prepare during development input necessary for maintenance process for the software product. See subclause 5.8.

EXPECTED OUTPUT: *Elements of the software maintenance plan [TS; PDR].*

5.3.2.4
The customer's release of the software requirements baseline shall be included in the material submitted to the SRR.

The software requirements baseline results from a system requirements analysis and a system partitioning conducted by the customer. It represents the customer's requirements towards the software to be developed:

- customer's requirements;
- external interfaces of the software.

EXPECTED OUTPUT: *a. Customer approval of requirements baseline [RB; SRR]*

b. SRR Milestone Review Report [DJF; SRR].

5.3.2.5
A software technical specification phase shall be included at the beginning of the development life cycle.

AIM: To establish the technical specification baseline for the project. This is the software suppliers response to the requirements baseline. The technical specification captures all technical requirements for the software product, and it is aimed to establish the technical specification early in the project.

EXPECTED OUTPUT: *a. technical specification of the software [TS; PDR];*

b. top-level architectural design [DDF; PDR];

c. interface control document [ICD; PDR];

d. top-level design trade-offs [DJF; PDR].

5.3.2.6
On completion of the specification phase, the software supplier shall hold a Preliminary Design Review (PDR) to which the customer shall be invited to attend.

AIM:
- Agree with the customer or their representatives that all requirements with respect to the requirements baseline are captured in the technical specification.
- Review the top-level software architecture.

EXPECTED OUTPUT: *Customer approval of technical specification and top-level architecture [TS, DDF, ICD, DJF; PDR].*

5.3.2.7
At the end of the design, the software supplier shall hold a Critical Design Review (CDR) to which the customer shall be invited to attend.

AIM: During the CDR, the design definition file, operations manual and the associated design justification file are reviewed.

The completeness of the verification and validation plan and the availability of necessary supporting resources (e.g. test case specification, simulators) shall be reviewed.

EXPECTED OUTPUT: *a. customer approval of the design definition file (e.g. architectural design and detailed design, code) [DDF; CDR];*

b. customer approval of the design justification file (e.g. results of unit and integration tests) [DJF; CDR];

c. customer approval of the design of system level interfaces and the system level integration plan [DDF, DJF; CDR];

d. customer approval of the operations manual [TS; CDR].

5.3.2.8
To ensure that the software product conforms with its technical specification, verification and validation shall be carried out at the end of the development life cycle.

AIM: To ensure, by means of verification and validation processes in a representative environment, that the software product conforms to its technical specification before integration in the system.

5.3.2.9
The software supplier shall hold a Qualification Review to verify that the software product meets all of its specified requirements.

AIM: To verify that the software meets all of its specified requirements, and in particular that verification and validation process outputs enable transition to “qualified state” for the software products.

During QR, a summary of tests reports and operations manual are reviewed. The consistency of all software documentation (TS, DDF, ICD, DJF, operations manual) shall be verified.

EXPECTED OUTPUT: *Customer’s approval of qualified state [DJF; QR].*

5.3.2.10
After the Qualification Review, the customer shall hold an Acceptance Review.

AIM: Acceptance of the software with respect to the intended operational environment.

EXPECTED OUTPUT: *Customer's approval of accepted state [DJF; AR].*

5.3.2.11
Each supplier shall define the software engineering standards he intends to follow for his application area. These standards shall be approved by the customer as being fit for the application under development.

AIM: Define the software engineering standards applicable to the project.

EXPECTED OUTPUT:
a. *documentation standards [RB; SRR];*
b. *design standards [RB; SRR];*
c. *verification and validation standards [RB; SRR].*

5.3.3 Interface management

Interfaces shall be defined in the requirements baseline in an interface requirements document, which defines the requirements applicable to various elements of the system product tree.

Interface management procedures shall be defined in accordance with ECSS-M-40 requirements.

AIM: Define procedures which guarantee the consistency of the system interfaces.

EXPECTED OUTPUT:
a. *interface management procedures [RB; SRR];*
b. *part of configuration management plan [RB; SRR].*

5.3.4 Technical budget and margin management

Software budgets considered in this subclause are those associated with computer resources (CPU load, maximum memory size) and performance requirements.

5.3.4.1
Technical budget targets and margin philosophy dedicated to the software shall be specified by the customer in the requirements baseline.

AIM: Define the limits to be considered by the supplier.

EXPECTED OUTPUT: *Technical budgets and margin philosophy for the project [RB; SRR].*

5.3.4.2
The supplier shall manage margins regarding the technical budgets and present their status at each milestone.

The margins shall be established by analysis in the early phases of development and consolidated by performance measurements commensurate with the software implementation.

Hypothesis with which analysis are performed shall be described as part of the evaluation results.

EXPECTED OUTPUT: *Margins and technical budgets status [DJF; PDR, CDR, QR, AR].*

5.4 Software requirements engineering process

5.4.1 Introduction

The software requirements engineering process consists of the following activities:

- software requirements analysis;
- software top-level architectural design;
- software verification and validation.

5.4.2 Software requirements analysis

For each software item, this activity consists of the following tasks:

- establish and document software requirements;
- identify each requirement;
- evaluate the software requirements.

5.4.2.1
The supplier shall establish and document software requirements, including the software quality requirements.

EXPECTED OUTPUT: *(Technical specification baseline)*

a. *functional and performance specifications, including hardware characteristics, and environmental conditions under which the software item will execute [TS; PDR];*

b. *interfaces external to the software item [ICD; PDR];*

c. *verification plan (by invoking subclause 5.6.4.5) [DJF; PDR];*

d. *safety specifications, including those related to methods of operation and maintenance, environmental influences, and personnel injury [TS; PDR];*

e. *security specifications, including those related to factors which might compromise sensitive information [TS; PDR];*

f. *human-factors engineering (ergonomics) specifications, including those related to manual operations, human-equipment interactions, constraints on personnel, and areas requiring concentrated human attention, that are sensitive to human errors and training [TS; PDR];*

g. *data definition and database requirements [TS; PDR];*

h. *installation and acceptance requirements of the delivered software product at the operation and maintenance site(s) [TS; PDR];*

i. *identification of lower level software engineering standards that will be used and that will document their compatibility with this current standard [TS; PDR].*

5.4.2.2
Each requirement shall be separately identified in order to allow for traceability.

5.4.2.3
The supplier shall evaluate the software requirements considering the criteria listed below invoking the process “verification of software requirements” (subclause 5.6.6.1). The results of the evaluation shall be documented:

a. traceability to system partitioning;

b. external consistency with system requirements;

c. internal consistency;

d. verifiability;

e. feasibility of software design;

f. feasibility of operations and maintenance.

EXPECTED OUTPUT: *a. requirement traceability matrices [DJF; PDR];*

b. requirements verification report [DJF; PDR].

5.4.3 Software top-level architectural design

For each software item, this activity consists of the following tasks:

- transformation of software requirements into an architecture;
- development and documentation of the top-level design of the software interfaces (external and internal);
- development and documentation of preliminary versions of the operations manual;
- definition and documentation of preliminary test requirements and a software integration plan;
- evaluation of the top-level architectural design;
- conducting a Preliminary Design Review.

5.4.3.1

The supplier shall transform the requirements for the software item into an architecture that describes its top-level structure and identifies the software components. It shall be ensured that all the requirements for the software item are allocated to its software components and further refined to facilitate detailed design. The top-level architecture of the software item shall be documented.

EXPECTED OUTPUT: *a. software architectural design [DDF; PDR];*

b. top-level architectural design to requirements traceability matrices [DJF; PDR].

5.4.3.2

The supplier shall develop and document a top-level design for the interfaces external to the software item and between the software components of the software item.

EXPECTED OUTPUT: *a. preliminary (top-level) external interfaces design [ICD; PDR];*

b. preliminary (top-level) internal interfaces design [ICD; PDR].

5.4.3.3

The supplier shall develop and document preliminary versions of the operations manual.

EXPECTED OUTPUT: *Preliminary version of operations manual [TS; PDR].*

5.4.3.4

The supplier shall define and document preliminary test requirements and the plan for software integration.

EXPECTED OUTPUT: *Preliminary software integration plan [DJF; PDR].*

5.4.3.5

The supplier shall evaluate the architecture of the software item and the interface designs considering the criteria listed below. The results of the evaluations shall be documented. The verification process "verification of the software design" (subclause 5.6.6.2) shall be invoked for verification of the architectural design:

a. traceability from the software requirements to the software item;

b. external consistency with the requirements of the software item;

c. internal consistency between the software components;

d. appropriateness of design methods and standards to be used for the item;

e. feasibility of detailed design;

f. feasibility of operation and maintenance.

EXPECTED OUTPUT: *Architecture and interface verification report [DJF; PDR].*

5.4.3.6
The supplier shall conduct a Preliminary Design Review (PDR) in accordance with subclause 5.3.2.6. The successful completion of the review establishes a baseline for the development of the software item.

EXPECTED OUTPUT: *PDR milestone report [DJF; PDR].*

5.4.4 Software verification and validation

5.4.4.1
The technical specifications shall include specification of verification and validation of the software product. These specifications are determined by the customer's requirements baseline (subclause 5.2.4.2) and by invoking the relevant verification and validation processes.

The processes invoked are:

a. verification process implementation (subclause 5.6.4);

b. validation process implementation (subclause 5.6.5).

EXPECTED OUTPUT:

a. software verification plan - criticality and effort [DJF; PDR];

b. software verification plan - methods and tools [DJF; PDR];

c. software verification plan - organization [TS; PDR];

d. software validation plan - effort and independence [DJF; PDR];

e. software validation plan - methods and tools [DJF; PDR];

f. software validation plan - independent validation [DJF; PDR];

g. software validation plan - organization [TS; PDR].

5.5 Software design engineering process

5.5.1 Introduction

The software design engineering process consists of the following activities:

- design of software items;
- coding and testing;
- integration.

5.5.2 Design of software items

For each software item, this activity consists of the following tasks:

- design of each software component;
- development and documentation of a design for the interfaces;
- updating of the operations manual;
- definition and documentation of a unit test specification and plan;
- updating the test specification and the schedule for integration;
- evaluation of the software detailed design and test specification.

5.5.2.1
The supplier shall develop a detailed design for each component of the software. Each software component shall be refined into lower levels containing software units that can be coded, compiled, and tested. It shall be ensured that all the software requirements are allocated from the software components to software units. The design shall be documented.

EXPECTED OUTPUT: *Software components design documents [DDF; CDR].;*

5.5.2.2
The supplier shall develop and document a detailed design for the interfaces external to the software item, between the software components, and between the software units. The detailed design of the interfaces shall permit coding without the requirement for further information.

EXPECTED OUTPUT: *a. external ICDs (update) [ICD; CDR];*
b. internal ICDs (update) [ICD; CDR].

5.5.2.3
The supplier shall update the operations manual as necessary.

EXPECTED OUTPUT: *operations manual (update) [TS; CDR].*

5.5.2.4
The supplier shall define and document test requirements and plan for testing software units. The test specifications shall include stressing the software at the limits of its requirements.

EXPECTED OUTPUT: *Software test plan [DJF; CDR].*

5.5.2.5
The supplier shall update the test requirements and the plan for software integration.

EXPECTED OUTPUT: *Integration test plan (update) [DJF; CDR].*

5.5.2.6
The supplier shall evaluate the software design and test specifications considering the criteria listed below. The process in "verification of software design" (subclause 5.6.6.2) shall be invoked for this. The results of the evaluations shall be documented:

a. traceability to the architectural design of the software item;
b. external consistency with architectural design;
c. internal consistency between software components and software units;
d. appropriateness of integration test methods and standards used;
e. feasibility of testing;
f. feasibility of operation and maintenance.

EXPECTED OUTPUT: *a. design verification report [DJF; CDR];*
b. design traceability matrices [DJF; CDR].

5.5.3 Coding and testing

For each software item, this activity consists of the following tasks:

- development and documentation of software units, test procedures and test data;
- testing of each software unit and database;
- updating the operations manual;
- updating the test requirements and the schedule for integration;
- evaluation of software code and test results.

5.5.3.1
The supplier shall develop and document the following:

a. the coding of each software unit;

b. test procedures and data for testing each software unit.

EXPECTED OUTPUT: *a. software component design documents and code (update) [DDF; CDR];*

b. unit test plan (update) [DJF; CDR].

5.5.3.2
The supplier shall test each software unit ensuring that it satisfies its requirements, using the process "verification of code" (subclause 5.6.6.3). The test results shall be documented.

EXPECTED OUTPUT: *a. software component design document and code (update) [DDF; CDR];*

b. unit test reports [DJF; CDR].

5.5.3.3
The supplier shall update the operations manual as necessary.

EXPECTED OUTPUT: *Operations manual (update) [TS; CDR].*

5.5.3.4
The supplier shall update the test requirements and the plan for software integration.

AIM: To make the test requirements and integration plan consistent with the results of the code design process.

EXPECTED OUTPUT: *Software integration plan (update) [DJF; CDR].*

5.5.3.5
The supplier shall evaluate software code and test results, invoking the process "verification of code" (subclause 5.6.6.3) and taking account of the criteria listed below. The results of the evaluation shall be documented:

a. traceability between the design of the software item and the code of the software units;

b. external consistency with the requirements and design of the software item;

c. internal consistency between unit requirements;

d. test coverage of units;

e. conformance to coding methods and standards;

f. feasibility of software integration and testing;

g. feasibility of operation and maintenance.

EXPECTED OUTPUT: *a. software code verification report [DJF; CDR];*

b. software code traceability matrices [DJF; CDR].

5.5.4 Integration

The following refers to the software integration of the software product, i.e. the software product delivered by the supplier to the customer. The integration process shall include preparation for validation testing of the integrated product.

The integration of the software product has a relation to the integration of the software in a space system. At system level, which is the next higher level in the product tree, the system level integration takes place. The system level integration nominally takes place after completion of the QR for the software product to be integrated with the system. However, depending on the system level life cycle and

risk sharing approach, the system integration process may be specified invoked earlier (see for example subclause 5.2.5.6), but not earlier than the software CDR.

For each software item, this activity consists of the following tasks:

- development of an integration plan;
- integration and testing of the software units and software components;
- updating the operations manual;
- evaluation of the integration plan, design, code, tests, test results, and operations manual;
- conducting a joint review (CDR) before starting validation.

5.5.4.1
The supplier shall develop an integration plan to integrate the software units and software components into the software item. The plan shall include test requirements, procedures, data, responsibilities, and schedule. The plan shall be documented.

EXPECTED OUTPUT: *Software integration plan [DJF; CDR].*

5.5.4.2
The supplier shall integrate the software units and software components and test, as the aggregates are developed, in accordance with the integration plan. It shall be ensured that each aggregate satisfies the requirements of the software item and that the software item is integrated at the conclusion of the integration activity. The integration and test results shall be documented.

EXPECTED OUTPUT: *Software integration test report [DJF; CDR].*

5.5.4.3
The supplier shall update the operations manual as necessary.

EXPECTED OUTPUT: *operations manual (update) [TS; QR].*

5.5.4.4
The supplier shall evaluate the integration plan, design, code, tests, test results, and operations manual considering the criteria listed below. The results of the evaluation shall be documented.

The verification process "verification of software integration" (subclause 5.6.6.4) shall be invoked:

a. traceability to the system requirements;
b. external consistency with the system requirements;
c. internal consistency;
d. test coverage of the requirements of the software item;
e. appropriateness of test standards and methods used;
f. conformance to expected results;
g. feasibility of software validation testing;
h. feasibility of operation and maintenance.

EXPECTED OUTPUT: Software integration verification report containing the reports a., b., c. and d. below:

a. software integration verification report [DJF; CDR];

b. software documentation verification report [DJF; CDR];

c. evaluation of test completeness and code conformance [DJF; CDR];

d. feasibility confirmation of validation testing, operations and maintenance [DJF; CDR];

e. *software validation testing specification [DJF; CDR].*

5.5.4.5
The supplier shall conduct a Critical Design Review (CDR) in accordance with subclause 5.3.2.7. All outputs required for CDR shall be prepared and verified by the process "verification of software documentation" (subclause 5.6.6.5) in preparation of the CDR.

AIM: That the supplier baselines his design documentation for the project to transit from "specified state" to the "defined state", thereby achieving the milestone of a completed design.

EXPECTED OUTPUT: *CDR milestone report [DJF; CDR].*

5.6 Software verification and validation (qualification) process

5.6.1 Introduction

This verification and validation processes may be executed with varying degrees of independence. The degree of independence may range from the same person, or different person in the same organization, to a person in a different organization, with varying degrees of separation. In the case where the processes are executed by an organization independent of the supplier, it is called Independent Software Verification and Validation (ISVV); or Independent Software Validation (ISV), if only the Validation Process is independent.

The requirements of this subclause 5.6 are in two parts. The first part concerns engineering activities for the QR and AR milestones (5.6.2), the second part (5.6.3 onwards) contains requirements called up by other processes.

NOTE 1 It is assumed that for all space projects certain verification and validation activities are always applied. Therefore the requirements do not address whether or not these activities are required (tailoring of ISO/IEC 12207).

NOTE 2 The supplier process verification is handled as part of the ECSS management and is therefore not covered as part of the software activities (tailoring of ISO/IEC 12207:1995 6.4.2.2).

5.6.2 Milestones

5.6.2.1
The Qualification Review (QR) shall be conducted in accordance with subclause 5.3.2.9.

AIM: To verify that the software meets all the requirements, and in particular that verification and validation process outputs enable transition to "qualified state" for the software products.

EXPECTED OUTPUT:
a. *preliminary software acceptance data package [DJF; QR];*
b. *preliminary software release documentation [DDF; QR];*
c. *preliminarily software delivery on specified data medium [DDF; QR];*
d. *software design and test evaluation report [DJF; QR];*
e. *validation testing report [DJF; QR];*
f. *test specification evaluation [DJF; QR];*
g. *qualification review records [DJF; QR].*

5.6.2.2

The Acceptance Review (AR) shall be conducted in accordance with subclause 5.3.2.10. The software supplier's acceptance support process shall support the customer's acceptance activities in preparation of the AR.

AIM: To ensure that the customer will receive adequate supplier support to perform his acceptance and integration activities in preparation of the AR. The process "support to acceptance reviews and testing" (subclause 5.6.7.7), is invoked for this.

EXPECTED OUTPUT:
a. *final software acceptance data package [DJF; AR];*
b. *acceptance testing documentation [DJF; AR];*
c. *acceptance review records [DJF; AR];*
d. *software release documentation [DDF; AR];*
e. *software delivery on specified data medium [DDF; AR].*

5.6.3 Verification and validation processes

The following subclauses are intended to be invoked by other parts of this standard. For this reason the output destination is not noted explicitly.

The software verification and validation engineering processes consist of:

- verification process implementation;
- validation process implementation;
- verification process;
- validation process;
- joint technical reviews.

5.6.4 Verification process implementation

This activity consists of the following tasks:

- determination of the verification effort for the project;
- establishment of verification process;
- selection of organization responsible for conducting the verification;
- determination of life cycle activities and software products requiring verification;
- development and documentation of a verification plan.

5.6.4.1 Determination of the verification effort for the project

A determination shall be made concerning the verification effort and the degree of organizational independence of that effort required. ECSS-M-00A subclause 6.3 (management of risks), and ECSS-Q-80A subclauses 3.2.2 (software dependability and safety) and 3.2.5.p (independent software verification and validation) shall be checked for applicability. The project requirements shall be analysed for criticality. Criticality may be gauged in terms of:

a. the potential of an undetected error in a system or software requirement for causing death or personal injury, mission failure, or financial or catastrophic equipment loss or damage;
b. the maturity of and risks associated with the software technology to be used;
c. availability of funds and resources.

EXPECTED OUTPUT: *Software verification plan - criticality and effort.*

5.6.4.2 Establishment of the verification process

A verification process shall be established to verify the software product(s).

5.6.4.3 Selection of the organization responsible for conducting the verification

If the project warrants an independent verification effort, a qualified organization responsible for conducting the verification shall be selected. This organization shall be assured of the independence and authority to perform the verification activities. ECSS-Q-80A subclause 3.2.5.p (independent software verification and validation), ECSS-M-00A subclause 7.2.3 and ECSS-M-20 (project organization) contain further requirements relevant for this subclause.

AIM: A coherent and consistent approach to project organization within each project.

EXPECTED OUTPUT: *Appropriate element of project requirements documents dealing with project organization.*

5.6.4.4 Determination of life cycle activities and software products requiring verification

Based upon the scope, magnitude, complexity, and criticality analysis above mentioned, target life cycle activities and software products requiring verification shall be determined. Verification activities and tasks defined in subclause 5.6.6, including associated methods, techniques, and tools for performing the tasks, shall be selected for the target life cycle activities and software products.

EXPECTED OUTPUT: *Verification plan - methods and tool.*

5.6.4.5 Development and documentation of a verification plan covering the software verification activities

Based upon the verification tasks as determined, a verification plan shall be developed and documented. The plan shall address the life cycle activities and software products subject to verification, the required verification tasks for each life cycle activity and software product, and related resources, responsibilities, and schedule. The plan shall address procedures for forwarding verification reports to the customer and other involved organizations.

EXPECTED OUTPUT: *Verification plan - organization.*

5.6.5 Validation process implementation

This activity consists of the following tasks:

- determination of the validation effort for the project;
- establishment of the validation process;
- selection of validation organization;
- development and documentation of the validation plan.

5.6.5.1 Determination of the validation effort for the project

The validation effort and the degree of organizational independence of that effort shall be determined, coherent with ECSS-Q-80A subclause 3.2.5.p.

EXPECTED OUTPUT: *Validation plan - effort and independence.*

5.6.5.2 Establishment of a validation process

The validation process shall be established to validate the software product. Validation tasks defined in subclause 5.6.7, including associated methods, techniques, and tools for performing the tasks, shall be selected.

EXPECTED OUTPUT: *Validation plan - methods and tools.*

5.6.5.3 Selection of a validation organization

If the project warrants an independent validation effort, a qualified organization responsible for conducting the effort shall be selected. The conductor shall be assured of the independence and authority to perform the validation tasks. This subclause shall be applied with ECSS-M-00A subclause 7.2.3.

EXPECTED OUTPUT: *a. appropriate element of project requirements documents dealing with project organization.*

b. independent validation plan.

5.6.5.4 Development and documentation of a validation plan

A validation plan shall be developed and documented. The plan shall include, but shall not be limited to, the following:

a. items subject to validation;

b. validation tasks to be performed;

c. resources, responsibilities, and schedule for validation;

d. procedures for forwarding validation reports to the customer and other parties.

EXPECTED OUTPUT: *Validation plan - organization*

5.6.6 Verification process

This activity consists of the following engineering tasks:

- verification of software requirements;
- verification of the software design;
- verification of code;
- verification of software integration;
- verification of software documentation;
- evaluation of test specifications;
- problem and nonconformance handling.

5.6.6.1 Verification of software requirements

The requirements shall be verified considering the criteria listed below:

a. The software requirements are consistent (not implying formal proof consistency), feasible, verifiable, and accurately reflect system requirements.

b. The software requirements related to safety, security, and criticality are correct as shown by suitably rigorous methods.

EXPECTED OUTPUT: *a. requirement traceability matrices;*

b. requirements verification report.

5.6.6.2 Verification of the software design

The design shall be verified considering the criteria listed below:

a. The design is correct and consistent with and traceable to requirements and interfaces.

b. The design implements a proper sequence of events, inputs, outputs, interfaces, logic flow, allocation of timing and sizing budgets, and error definition, isolation, and recovery.

c. The selected design can be derived from requirements.

d. The design implements safety, security, and other critical requirements correctly as shown by suitable rigorous methods.

EXPECTED OUTPUT: *a. top-level architectural design to requirements traceability matrice;*

b. architecture and interface verification report;

c. *design traceability matrice;*

d. *design verification report.*

5.6.6.3 Verification of code

The code shall be verified taking into consideration the criteria listed below:

a. The code is traceable to design and requirements, testable, correct, and in conformity to software requirements and coding standards.

b. The code implements proper event sequence, consistent interfaces, correct data and control flow, completeness, appropriate allocation timing and sizing budgets, and error definition, isolation, and recovery.

c. Selected code can be derived from design or software requirements.

d. The code implements safety, security, and other critical requirements correctly as shown by suitable rigorous methods.

EXPECTED OUTPUT: *a. software code traceability matrices;*

b. software code verification report.

5.6.6.4 Verification of software integration

The integration shall be verified considering that the software components and units of each software item have been completely and correctly integrated into the software item.

EXPECTED OUTPUT: *Software integration verification report*

5.6.6.5 Verification of software documentation

The documentation shall be verified considering the criteria listed below:

a. The documentation is adequate, complete, and consistent.

b. Documentation preparation is timely.

c. Configuration management of documents follows specified procedures.

EXPECTED OUTPUT: *Software documentation verification report*

5.6.6.6 Evaluation of test specifications

Test requirements, test cases, and test specifications shall demonstrate the coverage of all software requirements of the technical specification.

EXPECTED OUTPUT: *Test specification evaluation.*

5.6.6.7 Problem and nonconformance handling

Problems and nonconformances detected by the software verification effort shall be entered into the problem resolution process (ECSS-Q-80A subclause 2.3.5 and 2.3.6). All problems and nonconformances shall be resolved. Results of the verification activities shall be made available to the customer and other involved organizations.

EXPECTED OUTPUT: *Problem and nonconformance reports.*

5.6.7 Validation process

This activity consists of the following tasks:

- development and documentation of validation testing specification;
- conducting the validation tests;
- updating the operations manual;
- evaluation of the design, code, tests, test results, and operations manual;
- updating and preparation of the deliverable software product;
- problem and nonconformance handling;
- support to acceptance reviews and testing;

- provision of initial and continued training and support.

5.6.7.1 Development and documentation of a software validation testing specification

The supplier shall develop and document, for each validation requirement of the software item, a set of tests, test cases (inputs, outputs, test criteria), and test procedures for conducting software validation testing. The supplier shall ensure that the integrated software item is ready for software validation testing.

EXPECTED OUTPUT: *Software validation testing specification.*

5.6.7.2 Conducting the validation tests

The validation tests shall be conducted as specified in the output of subclause 5.6.7.1 above, including:

a. testing with stress, boundary, and singular inputs;

b. testing the software product for its ability to isolate and minimize the effect of errors; that is graceful degradation upon failure, request for operator assistance upon stress, boundary and singular conditions;

c. testing that the software product can perform successfully in a representative operational environment.

EXPECTED OUTPUT: *Validation testing report.*

5.6.7.3 Updating the operations manual

The supplier shall update the operations manual as necessary.

EXPECTED OUTPUT: *Operations manual (update).*

5.6.7.4 Evaluation of the design, code, tests, test results, and operations manual

The supplier shall evaluate the design, code, tests, test results, and operations manual considering the criteria listed below. The results of the evaluations shall be documented.

a. test coverage of the requirements of the software item;

b. conformance to expected results;

c. feasibility of system integration and testing, if conducted;

d. feasibility of operation and maintenance.

EXPECTED OUTPUT: *Software design and test evaluation report.*

5.6.7.5 Updating and preparation of the deliverable software product

Upon successful completion of the audits, if conducted, the supplier shall:

a. update and prepare the deliverable software product as specified in the requirements baseline for system integration, system validation testing, software installation, or software acceptance support as applicable;

b. update the established baseline for the design and code of the software item.

EXPECTED OUTPUT: *a. software delivery on specified data medium*

b. software release documentation;

c. software acceptance data package.

5.6.7.6 Problem and nonconformance handling

Problems and nonconformances detected during the validation shall be the subject of a problem resolution process (ECSS-Q-80A subclauses 2.3.5 and 2.3.6). All problems and nonconformances shall be resolved. Results of the validation activities shall be made available to the customer and other involved organizations.

EXPECTED OUTPUT: *Problem and nonconformance report.*

5.6.7.7 Support to acceptance reviews and testing

The supplier shall support the customer's acceptance reviews and testing of the software product. Acceptance reviews and testing shall consider the results of the joint reviews (ECSS-Q-20A subclauses 4.6.4.4 and 8.3), audits (ECSS-Q-20A subclause 2.6), software validation testing (ECSS-Q-80A subclause 3.3.4), and system validation testing (if performed). The results of the acceptance reviews and testing shall be documented.

EXPECTED OUTPUT: *Acceptance testing documentation.*

5.6.7.8 Provision of initial and continued training and support

The supplier shall provide initial and continuing training and support to the customer as specified in the technical specification.

EXPECTED OUTPUT: *Training material.*

5.6.8 Joint technical review process

The joint review process is a process for evaluating the status and products of an activity of a project as appropriate. Joint reviews shall be held throughout the life cycle of the software. This process may be employed by any two parties, where one party (reviewing party) reviews another party (reviewed party).

5.6.8.1 Support to software reviews

The software support of joint technical reviews shall be related to project phasing and planning (refer to ECSS-M-30). Therefore software shall undergo the overall software milestone reviews SRR, PDR, CDR, QR and AR as a minimum. Further reviews may be required by the customer.

EXPECTED OUTPUT: *Milestone review reports.*

5.6.8.2

Technical reviews (including milestone reviews) shall be held to evaluate the software products or services under consideration and provide evidence that:

a. they are complete;

b. they conform to applicable standards and specifications;

c. changes are properly implemented and affect only those areas identified by the configuration management process;

d. they adhere to applicable schedules;

e. they are ready for the next activity;

f. the development, operation, or maintenance is being conducted according to the plans, schedules, standards, and guidelines laid down for the project.

Reviews shall be planned of each identified software product within its defined software life cycle according to the criteria above.

EXPECTED OUTPUT: *Technical review reports.*

5.7 Software operations engineering process

5.7.1 Introduction

The operation process may start after completion of software acceptance. Since software products are an integrated part of the space system, the phasing and management of operation should be determined by the overall system requirements and applied to the software products. The operation engineering processes are therefore not directly connected to the overall mission phase E, but are determined by the system level requirement to operate the software product at a given time. Ground segment software products are for example in extensive operational use to qualify the ground segment, well before the actual mission operation occur. Similarly, for flight segment software, extensive ground operations are, in general,

required for testing flight equipment long before space system flight operations begin.

Both the documents and the reviews identified as outputs by the subclauses of 5.7 are therefore part of the operations activities for the space systems, and the requirements for these reviews and their documentation forms part of the space system operations engineering requirements covered in other ECSS Standards. The provisions of this subclause 5.7 are intended to produce the required software engineering inputs for the system level activities.

5.7.2 Operation process

The operation process comprises the activities and tasks of the operator. The process covers the operation of the software product and operational support to users. Because operation of a software product is integrated into the operation of the system, the activities and tasks of this process shall refer to the system.

The operator manages the operation process at the project level following the management process (ECSS-M-30). This process consists of the following activities:

- process implementation;
- operational testing;
- system operation;
- user support.

5.7.3 Process implementation

This activity consists of the following tasks:

- development of operational plans and set standards;
- definition of procedures for problem handling;
- definition of operational testing specifications.

5.7.3.1

The operator shall develop a plan and set operational standards for performing the activities and tasks of this process. The plan shall be documented and executed.

EXPECTED OUTPUT: *Operational plan - plan and standards [OP; ORR].*

5.7.3.2

The operator shall establish procedures for receiving, recording, resolving, tracking problems, and providing feedback. Whenever problems are encountered, they shall be recorded in accordance with the change control established and maintained in conformance with ECSS-M-40.

EXPECTED OUTPUT: *Operational plan - procedures for problem handling [OP; ORR].*

5.7.3.3

The operator shall establish procedures for testing the software product in its operation environment, for entering problem reports and modification requests to the maintenance process (subclause 5.8), and for releasing the software product for operational use in accordance with the change control established and maintained in conformance to ECSS-M-40.

EXPECTED OUTPUT: *Operational plan - operational testing specifications [OP; ORR].*

5.7.4 Operational testing

For each release of the software product, the operator shall perform operational testing in accordance with the change control established and maintained in conformance to ECSS-M-40. On satisfying the specified criteria, the software product shall be released for operational use.

5.7.5 System operation

The system shall be operated in its intended environment according to the operations manual.

5.7.6 User support

This activity consists of the following tasks:

- user assistance and consultation;
- handling of user requests for software maintenance;
- provision of work-around solutions.

5.7.6.1
The operator shall provide assistance and consultation to the users as requested. These requests and subsequent actions shall be recorded and monitored.

5.7.6.2
The operator shall forward user requests, as necessary, to the maintenance process for resolution. These requests shall be addressed and the actions that are planned and taken shall be reported to the originators of the requests. All resolutions shall be monitored to conclusion.

5.7.6.3
If a reported problem has a temporary work-around before a permanent solution can be released, the originator of the problem report shall be given the option to use it. Permanent corrections, releases that include previously omitted functions or features, and system improvements shall be applied to the operational software product using the maintenance process (subclause 5.8).

5.8 Software maintenance process

5.8.1 Introduction

The maintenance process contains the activities and tasks of the maintainer. This process shall be activated when the software product undergoes modifications to code and associated documentation due to a problem or the requirement for improvement or adaptation. The objective is to modify an existing software product while preserving its integrity. This process shall include the migration and retirement of the software product. The process shall end with the retirement of the software product.

The activities provided in this subclause 5.8 are specific to the maintenance process; however, the process may utilize other processes in this standard. If the software engineering process (subclause 4.3) is utilized, the term supplier there is interpreted as maintainer.

The maintainer shall manage the maintenance process at the project level following the management process (ECCS-M-10), which is instantiated for software in this process.

Both the documents and the reviews identified as outputs by the subclauses in this subclause 5.8 are part of the general maintenance activities for the space systems, and the requirements for these reviews and documentation is part of the space system maintenance engineering requirements, covered in other ECSS Standards. The provisions of this subclause 5.8 shall produce the required software engineering inputs for these system level activities.

This process consists of the following activities:

- process implementation;
- problem and modification analysis;
- modification implementation;
- maintenance review/acceptance;

- software migration
- software retirement.

5.8.2 Process implementation

This activity consists of the following tasks:

- maintenance procedure development and planning;
- implementation of a configuration control process for problem reporting and handling.

5.8.2.1
The maintainer shall develop, document, and execute plans and procedures for conducting the activities and tasks of the maintenance process.

EXPECTED OUTPUT: *Maintenance plan - plans and procedures [MP; System].*

5.8.2.2
The maintainer shall establish procedures for receiving, recording and tracking problem reports and modification requests from the users and providing feedback to the users. Whenever problems are encountered, they shall be recorded and entered in accordance with the change control established and maintained in conformance to ECSS-M-40.

EXPECTED OUTPUT: *Maintenance plan - problem reporting and handling [MP; System].*

5.8.2.3
The maintainer shall implement (or establish organizational interface with) the configuration management process (ECSS-M-40) for managing modifications.

5.8.3 Problem and modification analysis

This activity consists of the following tasks:

- problem analysis;
- problem verification;
- development of options for modifications;
- documentation of problems, analysis and implementation options;
- obtaining customer approval for selected modification option.

5.8.3.1
The maintainer shall analyse the problem report or modification requests for its impact on the organization, the existing system, and the interfacing systems for the following:

a. type (e.g. corrective, improvement, preventive, or adaptive to new environment);

b. scope (e.g. size of modification, cost involved, time to modify);

c. criticality (e.g. impact on performance; safety, or security).

5.8.3.2
The maintainer shall reproduce or verify the problem.

5.8.3.3
Based upon the analysis, the maintainer shall develop options for implementing the modification.

5.8.3.4
The maintainer shall document the problem/modification request, the analysis results and implementation options.

EXPECTED OUTPUT: *Change justification file - problem analysis report [CJF].*

5.8.3.5
The maintainer shall obtain approval for the selected modification option in accordance with procedures agreed with the customer.

5.8.4 Modification implementation

This activity consists of the following tasks:

- analysing and documenting which products require modification;
- invoking the software development process to implement the modifications.

5.8.4.1
The maintainer shall conduct analysis and determine which documentation, software units, and versions thereof shall be modified. These shall be documented.

EXPECTED OUTPUT: *Change justification file - modification identification [CJF; System].*

5.8.4.2
The maintainer shall enter the software engineering process (subclause 4.3) to implement the modifications. The requirements of the development process shall be supplemented as follows:

a. Test and evaluation criteria for testing and evaluating the modified and the unmodified parts (software units, components, and configuration items) of the system shall be defined and documented.

b. The complete and correct implementation of the new and modified requirements shall be ensured. It also shall be ensured that the original, unmodified requirements were not affected. The test results shall be documented.

5.8.5 Maintenance review/acceptance

The maintainer shall conduct joint review(s) with the organization authorizing the modification to determine the integrity of the modified system. Upon successful completion of the reviews, a baseline for the change shall be established.

EXPECTED OUTPUT: *Change justification file - baseline for changes [CJF].*

5.8.6 Software migration

This activity consists of the following tasks:

- coherent application of standards for migration;
- developing, documenting and executing a migration plan;
- notifying the space system users of migration plans and activities;
- provision of training, and parallel operations of existing and migrated system where required;
- notification of transition to migrated system;
- performance of technical review to assess impact of transition to new environment;
- maintaining data of former systems.

5.8.6.1
If a system or software product (including data) is migrated from an old to a new operational environment, it shall be ensured that any software product or data produced or modified during migration are in accordance with this Standard.

5.8.6.2
A migration plan shall be developed, documented, and executed. The planning activities shall include users. Items included in the plan shall include the following:

a. requirements analysis and definition of migration;

b. development of migration tools;
c. conversion of software product and data;
d. migration execution;
e. migration verification;
f. support for the old environment in the future.

EXPECTED OUTPUT: *Migration plan*

5.8.6.3
Users shall be given notification of the migration plans and activities.

Notifications shall include the following:

a. statement of why the old environment is no longer to be supported;
b. description of the new environment with its date of availability;
c. description of other support options available, if any, once support for the old environment has been removed.

EXPECTED OUTPUT: *Migration justification file*

5.8.6.4
Parallel operation of the old and new environments may be conducted for smooth transition to the new environment. During this period, training shall be provided as necessary and specified in the operational plan.

5.8.6.5
When the scheduled migration takes place, notification shall be sent to all concerned. All associated old environment's documentation, logs, and code shall be placed in archives.

5.8.6.6
A post-operation review shall be performed to assess the impact of changing to the new environment. The results of the review shall be sent to the appropriate authorities for information, guidance, and action.

5.8.6.7
Data used by or associated with the old environment shall be accessible in accordance with the requirements for data protection and audit applicable to the data.

5.8.7 Software retirement

Refer to the disposal activities described in ECSS-M-30.

6

Special requirements

6.1 Introduction

This clause 6 defines the specific requirements for engineering software for space systems. They are special in the sense that they are only to be applied where the software engineering disciplines or technologies identified in this clause are exploited in the project.

6.2 Space segment software

The space segment software calls for special engineering requirements, due to the highly specialized environment and because the software implements functions that directly relate to space system dependability.

The detailed technical engineering requirements are found expanded in ECSS-E-40-01, at this level the main top-level requirements are defined.

6.2.1 Critical functions

6.2.1.1
The system criticality analysis produced by the customer (subclause 5.2.2.2) shall be produced as a separate input to the SRR.

AIM: To partition the system such that the criticality of software elements is agreed between the customer and supplier already at the functional state of the project. The different levels of criticality to be defined for the project as required by ECSS-Q-80. See subclause 6.6 for further explanation.

EXPECTED OUTPUT: *Software criticality analysis report [DJF; SRR].*

6.2.1.2
The supplier's response to the criticality requirements baseline shall be documented as a separate input to the PDR.

AIM: To completely specify the methods and means the supplier will apply to achieve the required response to the criticality related requirements. All elements of the software life cycle as well as the criticality of COTS and of the engineering tools required to produce the software shall be included.

EXPECTED OUTPUT: *Specification of design response to criticality requirements [TS, DJF; PDR].*

6.2.1.3
The supplier shall demonstrate the achieved level of dependability as a separate input to the CDR.

AIM: To ensure that a sufficient level of dependability has been achieved by the design, already at CDR.

EXPECTED OUTPUT: *Preliminary design verification, validation results and supporting analysis [DJF; CDR].*

6.2.1.4
The customer shall include requirements for validation of all elements of the software at system level, including validation at mission level.

In general, no prototype flights are possible, the software shall be fully operational at first flight. Therefore the aim of this subclause is to ensure the software is validated at system level with realistic mission data and operational environments, and to minimize the functions that can only be validated by actual flight.

EXPECTED OUTPUT: *Functional requirements for support to system and mission level validation [RB; SRR].*

6.2.2 System interfaces

The Interface Requirements Document (IRD) and the Interface Control Document (ICD) shall be produced as separate inputs to SRR and PDR, respectively.

AIM: Space segment software is in general integrated with highly specialized processors and electrical equipment. The IRC and ICD therefore have a special importance and shall be controlled separately to ensure consistent design throughout the hardware and software life cycle.

EXPECTED OUTPUT:
a. *IRD for space segment software as separate output [IRD; SRR];*
b. *ICD for space segment software as separate output [ICD; PDR].*

6.2.3 In-flight software modifications

6.2.3.1
For space segment software where ability to perform software modifications in flight is required, the special customer requirements for this shall be documented in the requirements baseline, and the supplier's response shall be documented in the technical specification baseline.

AIM: In addition to software maintenance, space segment software may be required to support reprogramming in flight. In addition to software maintenance, this implies the software design process is re-invoked to include changed or added requirements. Therefore, the means to re-invoke the complete design process is required to be maintained for these cases additional to the means for maintaining the software end-product.

Space segment software shall not always be reprogrammable in flight. In cases where reprogramming is required, the appropriate requirements shall be captured already at SRR, and the supplier's design baseline shall incorporate the corresponding design at PDR. Due to the long life-time often encountered with space segment software, special requirements may also exist to ensure the supporting tools (e.g. compilers, engineering tools) can support the reprogramming in orbit during the required life-time.

EXPECTED OUTPUT: *Requirements of in-flight modification capabilities [RB; SRR].*

6.2.3.2
For space segment software requiring in-flight modification, the supplier shall perform analysis of the specific implications for the software design processes and include the necessary functional and performance requirements in the Technical Specification.

EXPECTED OUTPUT: *Specifications for in-flight software modifications [TS; PDR].*

6.3 Ground segment software

No special requirements concerning the software engineering processes have been identified at this level. Detailed special software engineering requirements for ground segments are found in the level 3 standard ECSS-E-40-03.

6.4 Software reuse

The following subclauses shall be applied in the software engineering process for projects where:

- it is intended to reuse the software products being developed for other space projects;
- it is intended to reuse software products from other space projects and third-party "commercial off-the-shelf tools" are intended to be part of the software product.

6.4.1 Developing software for intended reuse

6.4.1.1
The customer shall specify the special constraints that apply for the development, to enable future reuse of the software.

AIM: Specification of the customer's generic application domain for the parts where the customer requires reuse of developed software components. This may for example include requirements on software architecture for given target computers and operating systems, the interfaces required for reuse and the level where reuse is required (e.g. function, sub-system, code modules).

EXPECTED OUTPUT: *Design for reuse requirements [RB; SRR].*

6.4.1.2
The supplier shall define procedures, methods and tools for reuse, and he shall apply these to the software engineering processes to conform to the reusability requirements for the software development. In particular, this implies:

a. An evaluation of the reuse potential of the software to be conducted at PDR and CDR.

b. Design choices for testing and documentation that shall support the future existence of software reuse components as independent sub-products of the software engineering processes.

c. Specification of any special configuration management required for the reuse items.

EXPECTED OUTPUT:
a. *Design for reuse specification [TS; PDR];*
b. *Design for reuse - justification of methods and tools [DJF; PDR];*
c. *Design for reuse - evaluation of reuse potential [DJF; PDR, CDR];*
d. *Design for reuse - organization and content of the user manual or equivalent document to help people in reusing software [DJF; PDR, CDR].*

6.4.2 Reusing software from other projects

6.4.2.1
The supplier shall consider the "reuse" of already developed, commercial off-the-shelf and modifiable off-the-shelf software, if required by the customer. The analysis of the potential reusability of existing software components shall be performed through:

a. Identification of the reuse components with respect to the functional requirements baseline.

b. A quality evaluation of these components, invoking ECSS-Q-80A subclause 3.2.7

NOTE There are no special requirements concerning the verification and validation requirements for reused software. The requirements are the same as for software developed without reuse. The difference is that some already existing verification and validation plans and results might be available with the reused products. However, the full verification and validation requirement apply to reused software as for any other part of the software development.

EXPECTED OUTPUT: *a. Specification of intended reuse [TS; PDR];*

b. Justification of reuse with respect to Requirements Baseline [DJF; PDR].

6.4.2.2
For projects, in which the customer intends to use commercial off-the-shelf software products or modified versions thereof, the customer shall tailor the acquisition process of software destined for reuse as described in document IEEE Standard 1062-1993, referenced in annex C, (or in a similar standard, if available) to the project requirements. The tailored acquisition process shall be documented in the Requirements Baseline.

AIM: To baseline the procurement process requirements for commercial products supporting the software development (e.g. compilers, operating system, development tools).

EXPECTED OUTPUT: *Software acquisition process for COTS and MOTS Products [RB; SRR]..*

6.4.2.3
The supplier shall implement the software acquisition process for reused software, and document the process in the Technical Specification.

EXPECTED OUTPUT: *Software acquisition process implementation [TS; PDR].*

6.5 Man-machine interfaces

6.5.1

Software projects which include the development of a significant interactive direct interface to a human user or operator, require the specialized software engineering disciplines covering this field, and the requirements of this subclause 6.5 shall be applied.

The reason for the special subclauses is that modern MMI technology (e.g. graphical user interfaces, multi-layered choice menus), is not feasible to specify or design using conventional software engineering documentation methods. The non-linear and multi-dimensional nature of modern MMI cannot be described adequately only using two-dimensional documents that by nature are linear in structure. This is very similar to other systems with significant human factors requirements, such as cars, airplanes, buildings. In those cases a mock-up or model is implemented

during the "requirements engineering". An analogous approach in software engineering is required for software with extensive human interaction requirements.

6.5.2

For software that requires interface to human operator(s), the customer shall, based on the complexity and requirements of the MMI, determine if a software mock-up of the MMI is required to support the requirements engineering processes.

EXPECTED OUTPUT: *MMI software mock-up requirements [RB; SRR].*

6.5.3

The customer shall determine if general MMI standards or guidelines shall be applicable to the software project and include these requirements in the requirements baseline.

AIM: To ensure for example that appropriate guidelines and style-guides are selected for projects in cases where a common MMI style and functionality is required for several suppliers' products.

EXPECTED OUTPUT: *MMI general requirements and guidelines [RB; SRR]*

6.5.4

For developments requiring a software mock-up of the MMI, the supplier shall develop a software mock-up to support the requirements engineering process. The supplier shall use the mock-up to prototype the specifications of man-machine interfaces for the software, such that MMI specifications are consolidated and evaluated with respect to human factors and use.

The aim of this subclause includes:

- proper consideration of human factors;
- that the man-machine interface reach an acceptable state of definition during requirements engineering activities;
- that the technical performance of the man-machine interface is verified.

NOTE Depending on the nature of the project, the supplier might opt to later upgrade the software mock-up of the MMI to become part of the final software product. However, unless the mock-up is later upgraded to become part of the final product tree, the mock-up is not required to be a formal delivery to the customer.

EXPECTED OUTPUT:
a. *MMI specifications for software [TS; CDR];*
b. *Report on evaluation of MMI specifications using a software mock-up [DJF; PDR].*

6.6 Critical software

This topic is dealt with in depth in ECSS-Q-80. Here the basic principles are repeated together with the effects it has on the software engineering process.

Software criticality is not inherent to software by itself; criticality is determined by factors outside the software development. These factors can be related to requirements that are not functional ones, e.g. a development schedule, but other critical factors of more direct influence on the software engineering process are those where functions allocated to software implementation are critical system functions. In all cases, the criticality requirements originate externally with respect to the software project. Therefore, it is by definition, that the customer shall perform the criticality analysis (see subclause 5.2.2.2), whereas the supplier implements the corresponding special measures, as agreed with the customer in the Technical Specification Baseline.

The criticality analysis results in a classification according to criticality levels, such that each software product corresponds to a well defined criticality level (ref. ECSS-Q-80). For each level of criticality, appropriate measures shall be specified by the supplier and agreed with the customer. With each level of criticality, the precise measures (e.g. formal proof, independent validation, test coverage completeness and analysis, development of redundant software) shall be defined, and the implementation shall be introduced into the engineering processes at SRR.

"Software errors" are understood as errors that are introduced by any of the software engineering processes covered by this standard. It is important to note, that the validation of specifications as well as code and testing processes shall be thereby included.

With reference to other ECSS-Q standards, it is important to note that software errors are not stochastic in nature, and do not follow the statistical laws often used to determine hardware reliability. Software errors are all, by definition, engineering errors, and persist until the design is corrected. The software testing effort should reduce the probability of remaining design errors, not to be confused with reliability or "mean-time-between-failures".

With reference to the fault tree analysis defined in ECSS-Q standards, it shall be taken into account, that no software of significant size can be made completely free of design errors. It is beyond current state of the art. Fault tree analysis shall therefore assume software design errors exist for large software, and the system engineering processes shall introduce appropriate measures to overcome this problem.

Annex A (normative)

Software documentation

A.1 Introduction

This annex defines the contents of the software engineering documents to be produced. The contents are defined by the outputs of the clauses in this standard, and the list of the outputs for each milestone of the project is provided below. The detailed structure of the software documents (e.g. table of contents, number of volumes) are not defined here, but left open to be determined by the size and nature of the individual software projects. The overall structure is given in Figure A-1.

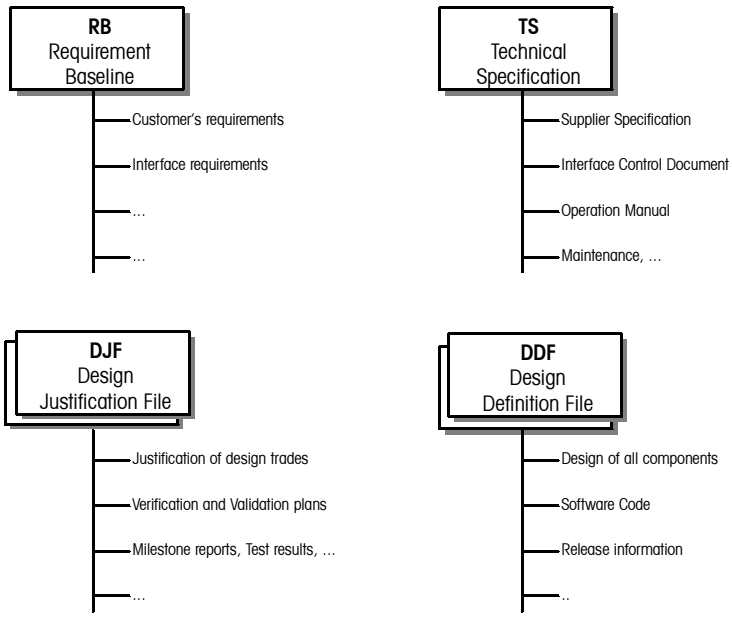


Figure A-1: Overview of software engineering documents

A.2 The Requirements Baseline (RB)

The RB expresses the customer's requirements. It is generated by the requirements engineering processes, and it is the primary input to the SRR review process.

A.2.1 Requirements baseline contents at SRR

Requirement	RB Contents at SRR Milestone
5.2.2.1-a.	Functions and performance requirements of the system [RB; SRR]
5.2.2.1-b.	Operations and maintenance requirements [RB; SRR]
5.2.2.1-d.	Design constraints and verification and validation requirements [RB; SRR]
5.2.2.1-e.	Identification of lower level software engineering standards that will be applied [RB; SRR]
5.2.2.2-a.	Overall safety and reliability requirements of the software to be produced [RB; SRR]
5.2.2.2-b.	Critical function identification and analysis [RB; SRR]
5.2.3.2-a.	System partition with definition of items [RB; SRR]
5.2.3.2-c.	System configuration items list [RB; SRR]
5.2.4.2	Verification and validation process requirements [RB; SRR]
5.2.5.1	Software observability requirements [RB; SRR]
5.2.5.4	Development constraints [RB; SRR]
5.3.2.4-a.	Customer approval of requirements baseline [RB; SRR]
5.3.2.11-a.	Documentation standards [RB; SRR]
5.3.2.11-b.	Design standards [RB; SRR]
5.3.2.11-c.	Verification and validation standards [RB; SRR]
5.3.3-a.	Interface management procedures [RB; SRR]
5.3.3-b.	Part of configuration management plan [RB; SRR]
5.3.4.1	Technical budgets and margin philosophy for the project [RB; SRR]

A.2.2 Interface Requirements Document (IRD)

The IRD expresses the customer's interface requirements for the software to be produced by the supplier. It is required in all cases where the software product is intended for integration with the customer's hardware or software products. This document is part of the requirements baseline. Depending on the size and nature of the project, the IRD sub-document may form separate clauses or separate volumes of the RB.

Requirement	IRD contents at SRR milestone
5.2.2.1-c.	Interface requirements [IRB; SRR]
5.2.3.2-b.	Software/hardware interface requirements [IRD; SRR]
5.2.5.2	System level interface requirements [IRD; SRR]
5.2.5.3	System level data interfaces [IRD; SRR]
5.2.5.5	System level integration support products [IRD; SRR]
5.2.5.6	System level integration preparation requirements [IRD; SRR]
5.2.5.7	System level integration support requirements [IRD; SRR]

A.3 Technical Specification (TS)

The TS contains the supplier's response to the requirements baseline, and is the primary input to the PDR review process. It includes the plans defined as part of the software development processes.

Depending on the size and nature of the project, the following sub-documents can be separate clauses or separate volumes of the TS.

Requirement	**TS contents at PDR**
5.3.2.1	Project software development life cycle definition, included in the software project development plan [TS; PDR]
5.3.2.5-a.	Technical specification of the software [TS; PDR]
5.3.2.6	Customer approval of technical specification and top-level architecture [TS, DDF, ICD, DJF; PDR]
5.4.2.1-a.	Functional and performance specifications, including hardware characteristics, and environmental conditions under which the software item will execute [TS; PDR]
5.4.2.1-d.	Safety specifications, including those related to methods of operation and maintenance, environmental influences, and personnel injury [TS; PDR]
5.4.2.1-e.	Security specifications, including those related to factors which might compromise sensitive information [TS; PDR]
5.4.2.1-f.	Human-factors engineering (ergonomics) specifications, including those related to manual operations, human-equipment interactions, constraints on personnel, and areas requiring concentrated human attention, that are sensitive to human errors and training [TS; PDR]
5.4.2.1-g.	Data definition and database requirements [TS; PDR]
5.4.2.1-h.	Installation and acceptance requirements of the delivered software product at the operation and maintenance site(s) [TS; PDR]
5.4.2.1-i.	Identification of lower level software engineering standards that will be used and that will document their compatibility with this current standard [TS; PDR]
5.4.4.1-c.	Software verification plan - organization [TS; PDR]
5.4.4.1-g.	Software validation plan - organization [TS; PDR]
Requirement	**TS contents at CDR**
5.5.3.4	Software integration plan (update) [DJF; CDR]
5.5.4.1	Software integration plan [DJF; CDR]

A.3.1 Interface Control Document (ICD)

The ICD is the suppliers response to the IRD, and is part of the TS.

Requirement	ICD contents at PDR
5.3.2.5-c.	Interface Control Document [ICD; PDR]
5.3.2.6	Customer approval of technical specification and top-level architecture [TS, DDF, ICD, DJF; PDR]
5.4.2.1-b.	Interfaces external to the software item [ICD; PDR]
5.4.3.2-a.	Preliminary (top-level) external interfaces design [ICD; PDR]
5.4.3.2-b.	Preliminary (top-level) internal interfaces design [ICD; PDR]

Requirement	ICD contents at CDR
5.5.2.2-a.	External ICDs (update) [ICD; CDR]
5.5.2.2-b.	Internal ICDs (update) [ICD; CDR]

A.3.2 Software maintenance plan

Requirement	Maintenance plan contents at PDR
5.3.2.3	Elements of the software maintenance plan [TS; PDR]

A.3.3 Operations manual

Requirement	Operations manual contents at PDR
5.4.3.3	Preliminary version of operations manual [TS; PDR]

Requirement	Operations manual contents at CDR
5.3.2.7-d.	Customer approval of the operations manual [TS; CDR]
5.5.2.3	Operations manual (update) [TS; CDR]
5.5.3.3	Operations manual (update) [TS; CDR]

Requirement	Operations manual contents at QR
5.5.4.3	Operations manual (update) [TS; QR]

A.4 Design Justification File (DJF)

The DJF is generated and reviewed at all stages of the development and review processes. It contains the documents that describe the trade-offs, design choice justifications, test procedures, test results, evaluations and any other documentation called for to justify the design of the supplier's product. The DJF is a primary input to the QR and AR milestones, and supporting input for the other milestones.

Requirement	DJF contents at SRR Milestone
5.2.3.2-d.	Traceability to system partitioning [DJF; SRR]
5.3.2.4-b.	SRR milestone review report [DJF; SRR]

Requirement	DJF contents at PDR milestone
5.3.2.5-d.	Top-level design trade-offs [DJF; PDR]
5.3.2.6	Customer approval of technical specification and top-level architecture [TS, DDF, ICD, DJF; PDR]
5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]
5.4.2.3-a.	Requirement traceability matrices [DJF; PDR]

5.4.2.3-b.	Requirements verification report [DJF; PDR]
5.4.3.1-b.	Top-level architectural design to requirements traceability matrices [DJF; PDR]
5.4.3.4	Preliminary software integration plan [DJF; PDR]
5.4.3.5	Architecture and interface verification report [DJF; PDR]
5.4.3.6	PDR milestone report [DJF; PDR]
5.4.4.1-a.	Software verification plan - criticality and effort [DJF; PDR]
5.4.4.1-b.	Software verification plan - methods and tools [DJF; PDR]
5.4.4.1-d.	Software validation plan - effort and independence [DJF; PDR]
5.4.4.1-e.	Software validation plan - methods and tools [DJF; PDR]
5.4.4.1-f.	Software validation plan - independent validation [DJF; PDR]
Requirement	**DJF contents at CDR milestone**
5.5.2.4	Software test plan [DJF; CDR]
5.5.2.6-a	Design verification report [DJF; CDR]
5.5.2.6-b.	Design traceability matrices [DJF; CDR]
5.3.2.7-b.	Customer approval of the design justification file (e.g. results of unit and integration tests) [DJF; CDR]
5.3.2.7-c.	Customer approval of the design of system level interfaces and the system level integration plan [DDF, DJF; CDR]
5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]
5.5.3.1-b.	Unit test plan (update) [DJF; CDR]
5.5.3.2-b.	Unit test reports [DJF; CDR]
5.5.3.5-a.	Software code verification report [DJF; CDR]
5.5.3.5-b	Software code traceability matrices [DJF; CDR]
5.5.4.1	Software integration plan [DJF; CDR]
5.5.4.2	Software integration test report [DJF; CDR]
5.5.4.4-a.	Software integration verification report [DJF; CDR]
5.5.4.4-b.	Software documentation verification report [DJF; CDR]
5.5.4.4-c.	Evaluation of test completeness and code conformance [DJF; CDR]
5.5.4.4-d.	Feasibility confirmation of validation testing, operations and maintenance [DJF; CDR]
5.5.4.4-e.	Software validation testing specification [DJF; CDR]
5.5.4.5	CDR milestone report [DJF; CDR]
Requirement	**DJF Contents at QR Milestone**
5.6.2.1-a.	Preliminary software acceptance data package [DJF; QR]
5.6.2.1-d.	Software design and test evaluation report [DJF; QR]
5.6.2.1-e.	Validation testing report [DJF; QR]
5.6.2.1-f.	Test specification evaluation [DJF; QR]
5.6.2.1-g.	Qualification review records [DJF; QR]

5.3.2.9	Customer's approval of qualified state [DJF; QR]
5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]
Requirement	**DJF contents at AR milestone**
5.3.2.10	Customer's approval of accepted state [DJF; AR]
5.6.2.2-a.	Final software acceptance data package [DJF; AR]
5.6.2.2-b.	Acceptance testing documentation [DJF; AR]
5.6.2.2-c.	Acceptance review records [DJF; AR]
5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]

A.5 Design Definition File (DDF)

The DDF is a supplier-generated document that documents the result of the design engineering processes. The DDF is the primary input to the CDR review process which shall contain all the documents called for by the design engineering requirements.

Requirement	**DDF contents at PDR**
5.3.2.5-b.	Top-level architectural design [DDF; PDR]
5.3.2.6	Customer approval of technical specification and top-level architecture [TS, DDF, ICD, DJF; PDR]
5.4.3.1-a.	Software architectural design [DDF; PDR]
Requirement	**DDF contents at CDR**
5.5.2.1	Software components design documents [DDF; CDR]
5.5.3.1-a.	Software component design documents and code (update) [DDF; CDR]
5.5.3.2-a.	Software component design document and code (update) [DDF; CDR]
5.3.2.7-a.	Customer approval of the design definition file (e.g. architectural design and detailed design, code) [DDF; CDR]
5.3.2.7-c.	Customer approval of the design of system level interfaces and the system level integration plan [DDF, DJF; CDR]
Requirement	**DDF contents at QR**
5.6.2.1-c.	Preliminarily software delivery on specified data medium [DDF; QR]
5.6.2.2-d.	Software release documentation [DDF; AR]
Requirement	**DDF contents at AR**
5.6.2.2-d.	Software release documentation [DDF; AR]
5.6.2.2-e.	Software delivery on specified data medium [DDF; AR]

A.6 System level documentation

A.6.1 Introduction

The system level documentation is governed by the ECSS system engineering standard. The relevant input for software elements at system level are found in subclause 5.2. In the special case where the customer is himself a software supplier (a product consisting solely of software) for the next higher level in the system product tree, the *customer* becomes a *supplier* at that level and the requirements of this current standard are applied recursively for that case.

A.6.2 Operations, maintenance, migration and retirement documentation

The operations, maintenance, migration and retirement processes are system level activities, defined by the customer's requirements for the space system. The corresponding software engineering processes are therefore not independent engineering activities, but are support processes at system level. Hence, the output of the processes are contributions to system level outputs, and the outputs below will therefore either be integrated with the software development documentation, or controlled and developed as part of a system documentation tree. The outputs are identified and grouped below. The system level documentation tree defines how the documents shall be included.

Requirement	**Operational documentation**
5.7.3.1	Operational plan – plan and standards [OP;ORR]
5.7.3.2	Operational plan – procedures for problem handling [OP; ORR]
5.7.3.3	Operational plan – operational testing specifications [OP; ORR]
Requirement	**Maintenance**
5.8.2.1	Maintenance plan – plans and procedures [MP; System]
5.8.2.2	Maintenance plan – problem reporting and handling [MP; System]
5.8.3.4	Change justification file – problem analysis report [CJF]
5.8.4.1	Change justification file – modification identification [CJF; System]
5.8.5	Change justification file – baseline for changes [CJF]
Requirement	**Migration**
5.8.6.2	Migration plan
5.8.6.3	Migration justification file
Requirement	**Retirement**
ECSS-M-30	(Refer to disposal activities described there)

(This page is intentionally left blank)

Annex B (informative)

Requirement cross references

Reference	Subclause
ECSS-M-40A, 5.2.	5.3.3
ECSS-Q-80A - 3.1. f	5.3.2.5
ECSS-Q-80A - 3.1.a	5.3.2.1
ECSS-Q-80A - 3.1.r	5.3.2.2
ECSS-Q-80A - 3.1.r	5.3.2.8
ISO/IEC 12207 5.1.2.2	4.6
ISO/IEC 12207: 5.2.4.2	5.3.2.1
ISO/IEC 12207: 5.3.10.1	5.2.5
ISO/IEC 12207: 5.3.13.2	5.6.7.7
ISO/IEC 12207: 5.3.13.3	5.6.7.8
ISO/IEC 12207: 5.3.4.1	5.4.2.1
ISO/IEC 12207: 5.3.4.2	5.4.2.3
ISO/IEC 12207: 5.3.5	5.4.3
ISO/IEC 12207: 5.3.5.1	5.4.3.1
ISO/IEC 12207: 5.3.5.2	5.4.3.2
ISO/IEC 12207: 5.3.5.4	5.4.3.3
ISO/IEC 12207: 5.3.5.5	5.4.3.4
ISO/IEC 12207: 5.3.5.6	5.4.3.5
ISO/IEC 12207: 5.3.6.1	5.5.2.1
ISO/IEC 12207: 5.3.6.2	5.5.2.2
ISO/IEC 12207: 5.3.6.4	5.5.2.3
ISO/IEC 12207: 5.3.6.5	5.5.2.4
ISO/IEC 12207: 5.3.6.6	5.5.2.5
ISO/IEC 12207: 5.3.6.7	5.5.2.6
ISO/IEC 12207: 5.3.7.1	5.5.3.1
ISO/IEC 12207: 5.3.7.2	5.5.3.2
ISO/IEC 12207: 5.3.7.3	5.5.3.3
ISO/IEC 12207: 5.3.7.4	5.5.3.4
ISO/IEC 12207: 5.3.7.5	5.5.3.5
ISO/IEC 12207: 5.3.8.1	5.5.4.1
ISO/IEC 12207: 5.3.8.2	5.5.4.2

Reference	Subclause
ISO/IEC 12207: 5.3.8.4	5.6.7.1
ISO/IEC 12207: 5.3.9.1	5.6.7.2
ISO/IEC 12207: 5.3.9.2	5.6.7.3
ISO/IEC 12207: 5.3.9.3	5.5.4.4
ISO/IEC 12207: 5.3.9.3	5.6.7.4
ISO/IEC 12207: 5.3.9.5	5.6.7.5
ISO/IEC 12207: 5.4.1	5.7.3
ISO/IEC 12207: 5.4.1.3	5.7.3.3
ISO/IEC 12207: 5.5.1.2	5.8.2.2
ISO/IEC 12207: 5.5.1.3	5.8.2.3
ISO/IEC 12207: 5.5.2.1	5.8.3.1
ISO/IEC 12207: 5.5.2.2	5.8.3.2
ISO/IEC 12207: 5.5.2.3	5.8.3.3
ISO/IEC 12207: 5.5.2.4	5.8.3.4
ISO/IEC 12207: 5.5.2.5	5.8.3.5
ISO/IEC 12207: 5.5.3.1	5.8.4.1
ISO/IEC 12207: 5.5.3.2	5.8.4.2
ISO/IEC 12207: 5.5.3.4	5.8.5
ISO/IEC 12207: 5.5.5.1	5.8.6.1
ISO/IEC 12207: 5.5.5.2	5.8.6.2
ISO/IEC 12207: 5.5.5.3	5.8.6.3
ISO/IEC 12207: 5.5.5.4	5.8.6.4
ISO/IEC 12207: 5.5.5.5	5.8.6.5
ISO/IEC 12207: 5.5.5.6	5.8.6.6
ISO/IEC 12207: 5.5.5.7	5.8.6.7
ISO/IEC 12207: 5.5.6	5.8.7
ISO/IEC 12207: 6.4.1.1	5.6.4.1
ISO/IEC 12207: 6.4.1.2	5.6.4.2
ISO/IEC 12207: 6.4.1.3	5.6.4.3
ISO/IEC 12207: 6.4.1.4	5.6.4.4
ISO/IEC 12207: 6.4.1.5	5.6.4.5
ISO/IEC 12207: 6.4.1.6	5.6.6.7
ISO/IEC 12207: 6.4.2.3	5.6.6.1
ISO/IEC 12207: 6.4.2.4	5.6.6.2
ISO/IEC 12207: 6.4.2.5	5.6.6.3
ISO/IEC 12207: 6.4.2.6	5.6.6.4
ISO/IEC 12207: 6.4.2.7	5.6.6.5
ISO/IEC 12207: 6.5.1.1	5.6.5.1
ISO/IEC 12207: 6.5.1.2	5.6.5.2
ISO/IEC 12207: 6.5.1.3	5.6.5.3
ISO/IEC 12207: 6.5.1.4	5.6.5.4
ISO/IEC 12207: 6.5.1.5	5.6.7.6
ISO/IEC 12207: 6.5.2.1	5.6.7.1
ISO/IEC 12207: 6.5.2.2	5.6.6.6
ISO/IEC 12207: 6.5.2.3	5.6.7.2
ISO/IEC 12207: 6.6.1.1 to 6.6.1.6	5.6.8
ISO/IEC 12207: 6.6.3.1	5.6.8.1

Annex C (informative)

References to other ECSS Standards

Referenced ECSS Standard	Page
ECSS-E-00	16
ECSS-E-10	20, 22, 23, 26
ECSS-E-40-01	51
ECSS-E-40-03	53
ECSS-M	21, 29
ECSS-M-00	21, 23, 40, 41, 42
ECSS-M-10	21, 46
ECSS-M-20	16, 22, 41
ECSS-M-30	17, 22, 30, 45, 46, 50
ECSS-M-40	22, 32, 46, 48
ECSS-M-50	22
ECSS-M-60	22
ECSS-M-70	22
ECSS-P-001	11
ECSS-Q	21, 27, 56
ECSS-Q-20	45
ECSS-Q-80	21, 29, 40, 41, 43, 44, 45, 51, 54, 55, 56

(This page is intentionally left blank)

Bibliography

IEEE Std. 1062-1993: Recommended Practices for Software Acquisition.

(This page is intentionally left blank)

ECSS Document Improvement Proposal

1. Document I.D.	2. Document Date	3. Document Title
ECSS-E-40A	13 April 1999	Software

4. Recommended Improvement (identify clauses, subclauses and include modified text and/or graphic, attach pages as necessary)

5. Reason for Recommendation

6. Originator of recommendation

Name:	Organization:	
Address:	Phone: Fax: E-Mail:	**7. Date of Submission:**

8. Send to ECSS Secretariat

Name:	Address:	Phone: +31-71-565-3952
W. Kriedte ESA-TOS/QR	ESTEC, Postbus 299 2200 AG Noordwijk The Netherlands	Fax: +31-71-565-6839 E-Mail: wkriedte@estec.esa.nl

Note: The originator of the submission should complete items 4, 5, 6 and 7.

This form is available as a Word and Wordperfect-Template on internet under
http://www.estec.esa.nl/ecss/improve/

(This page is intentionally left blank)