ECSS-E-40B Draft
28 July 2000

EUROPEAN COOPERATION
ECSS
FOR SPACE STANDARDIZATION

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)

Foreword **3**

1 Scope **7**

2 References **9**

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

3.1 Terms and definitions 11
3.2 Abbreviated terms 14

4 Space system software engineering **17**

4.1 Introduction 17
4.2 Space system software engineering processes 18
4.3 Organization of this standard 23
4.4 Relation to ECSS-M and ECSS-Q standards 24
4.5 Tailoring of this standard 26

5 General requirements **29**

5.1 Introduction 29
5.2 System engineering processes related to software 29
5.3 Software management process 34
5.4 Software requirements engineering process 37
5.5 Software design engineering process 40
5.6 Software validation and acceptance process 43
5.7 Software operations engineering process 46
5.8 Software maintenance process 48
5.9 Software verification and validation (supporting) processes 52

6

Special requirements **61**

6.1 Introduction 61
6.2 Space segment software 61
6.3 Ground segment software 68

6.4 Software reuse 68
6.5 Man-machine interfaces 69
6.6 Critical software 70

Annex A (normative)

Software documentation 71

A.1 Introduction 71
A.2 The Requirements Baseline (RB) 72
A.3 Technical Specification (TS) 73
A.4 Design Justification File (DJF) 74
A.5 Design Definition File (DDF) 77
A.6 System level documentation 79
A.7 Contribution to management documentation 79

Annex B (informative)

References to other ECSS Standards 81

Annex C (informative)

Tailoring Guidelines 83

C.1 Introduction 83
C.2 Tailoring templates 85

Figures

Figure 1: Life cycle processes in ECSS standards 18
Figure 2: The recursive customer - supplier model 19
Figure 3: Overview of the software development processes 20
Figure 4: Process constraints 21
Figure 5: Accomodation of different software life cycles 21
Figure 6: Structure of this standard 24
Figure A-1: Overview of software engineering documents 71

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 the customers to formulate their requirements and suppliers to prepare their response and to implement 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.

This standard also applies to the development of non–deliverable software which affects the quality of the deliverable product.

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 are not part of the space system development. Such software products will, however, often be part of a supporting infrastructure for space systems.

When viewed from the perspective of a specific project context, the requirements defined in this Standard should be tailored to match the genuine requirements of a particular profile and circumstances of a project.

NOTE Tailoring is a process by which individual requirements or specifications, standards and related documents are evaluated and made applicable to a specific project, either by deletion, addition or modification.

2 References

This ECSS Standard incorporates by dated or undated reference, provisions from other publications. These 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–00–02	Selection and Tailoring Process
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
IEEE 612.10–1990	IEEE Standard Glossary of software engineering terminology IEEE Std 612.10 1990
IEEE 1062–1993	IEEE Standard Recommended practices for software acquisition

These are the level 3 documents referenced by this standard:

ECSS–E–40–01	Space engineering – Space segment software .

ECSS–E–40–03	Space engineering – Ground segment software.
ECSS–E–40–04	Space engineering – Software life cycles
ECSS–E–40–DRD	Space engineering – Software Document Requirement Definitions

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 Acceptance Testing

The test of a system or functional unit usually performed by the customer on his premises after installation with the participation of the supplier to ensure that the contractual requirements are met [ISO 2382].

3.1.2 (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.

3.1.3 Assessment

An action of applying specific documented assessment criteria to a specific software module, package, or product for the purpose of determining acceptance or release of the software module, package or product [ISO 9126].

3.1.4 Configurable code

Code that can be configured to be used by the user or by the developer.

3.1.5 Critical Software

Software supporting a safety or dependability critical function that if incorrect or inadvertently executed would result in catastrophic or critical consequences. (For the definition of catastrophic and critical see ECSS–Q–40 and ECSS–Q–30).

3.1.6 Deactivated code

code that, although incorporated through correct design and coding is not intended to execute in any software product configuration. Examples of this can be legacy code or code intended for future development.

3.1.7 Integration Test (IT)

a. The progressive linking and testing of programs or modules in order to ensure their proper functioning in the complete system [ISO 2382].

b. Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them [IEEE 610.12 1990].

3.1.8 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.

3.1.9 Metric

The defined measurement method and the measurement scale [ISO 9126].

3.1.10 Migration

Porting of a software product to a complete new opportunely environment

3.1.11 Portability (a Quality Characteristic)

The capability of software to be transferred from one environment to another [ISO 9126].

3.1.12 Quality Characteristics (Software)

A set of attributes of a software product by which its quality is described and evaluated. A software quality characteristic may be refined into multiple levels of subcharacteristics [ISO 9126].

3.1.13 Quality Model (Software)

The set of characteristics and the relationships between them which provides the basis for specifying quality requirements and evaluating quality [ISO 9126].

3.1.14 Regression testing (Software)

Selective retesting to detect faults introduced during modification of a system or system component, to verify that the modifications have not caused unintended adverse effects, or to verify that a modified system or system component still meets its specified requirements [IEEE 610.12 1990].

3.1.15 Reusability

The degree to which a software module or other work product can be used in more than one computer program or software system [IEEE 610.12 1990].

3.1.16 Singular input

Individual parameter stress testing.

3.1.17 Software

Refer to Software Product.

3.1.18 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 usage 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.

3.1.19 Software item

See Software product.

3.1.20 Software intensive system

A space system 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.

3.1.21 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.

3.1.22 Software Product

The set of computer programs, procedures and possibly associated documentation and data [ISO 12207].

3.1.23 Software Product Assurance

The totality of activities, standards, controls and procedures in the lifetime of a software product which establishes confidence that the delivered software product, or software affecting the quality of the delivered product, will conform adequatly to customer requirements.

3.1.24 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.

3.1.25 Stress test

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

3.1.26 Unit Test

A test of individual programs or modules in order to ensure that there are no analysis or programming errors [ISO 2382].

3.1.27 Unreachable code

Code that cannot be reached due to design or coding error.

3.1.28 Usability (a Quality Characteristic)

The capacity of the software to be understood, learned, used and liked by the user, when used under specified conditions [ISO 9126]..

3.1.29 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.

3.1.30 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 abbreviation is 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.
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
FMECA	Failure Mode Effect and Criticality Analysis
HSIA	Hardware Software Interaction Analysis
ICD	Interface Control Document
IRB	Interface Requirements Baseline
IRD	Interface Requirements Document
ISV	Independent Software Validation
ISVV	Independent Software Verification and Validation
MGT	Management
MF	Maintenance File
MOTS	Modifiable off-the-Shelf
OP	Operational Plan
ORR	Operational Readiness 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.
SPR	Software Problem Report
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 extreme cases, the space system mission itself is redesigned, implementing new space segment software after launch. Ground segment software is similarly characterized.

This standard is complemented by ECSS–Q–80B, Software Product Assurance, with product assurance aspects as defined in ECSS–Q–80B. Together the two documents either define or refer to the definition of all software relevant processes for space projects.

The coverage of all software life cycle processes by the different ECSS standards is illustrated in Figure 1.

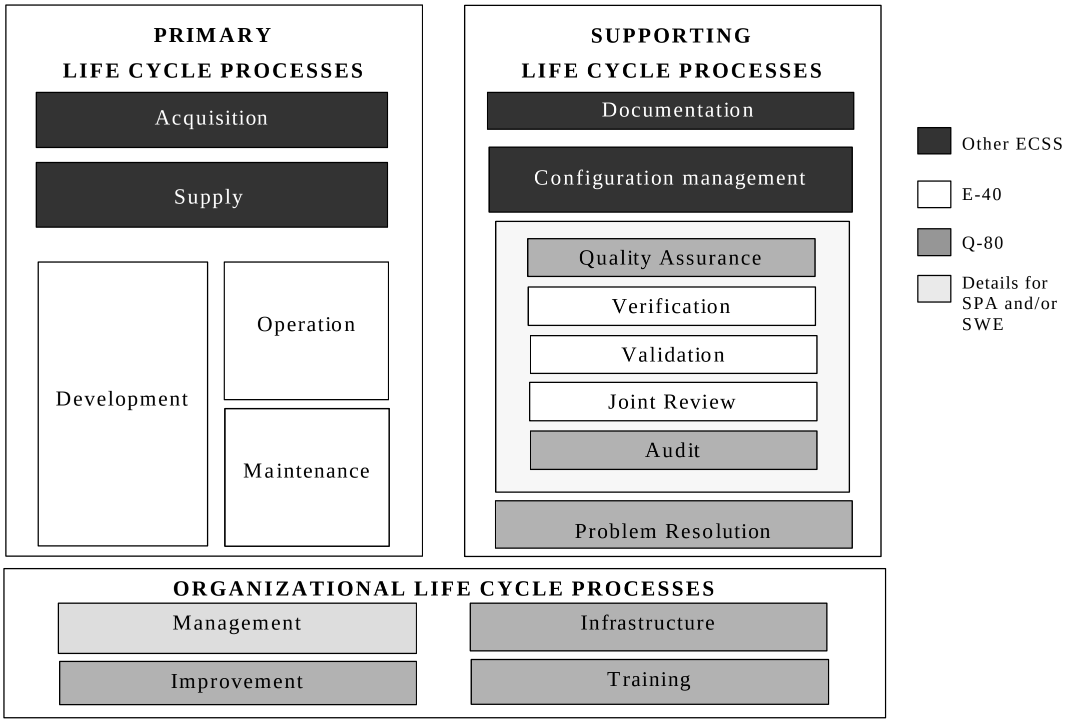

Figure 1: Life cycle processes in ECSS standards

4.2 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 2. 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.

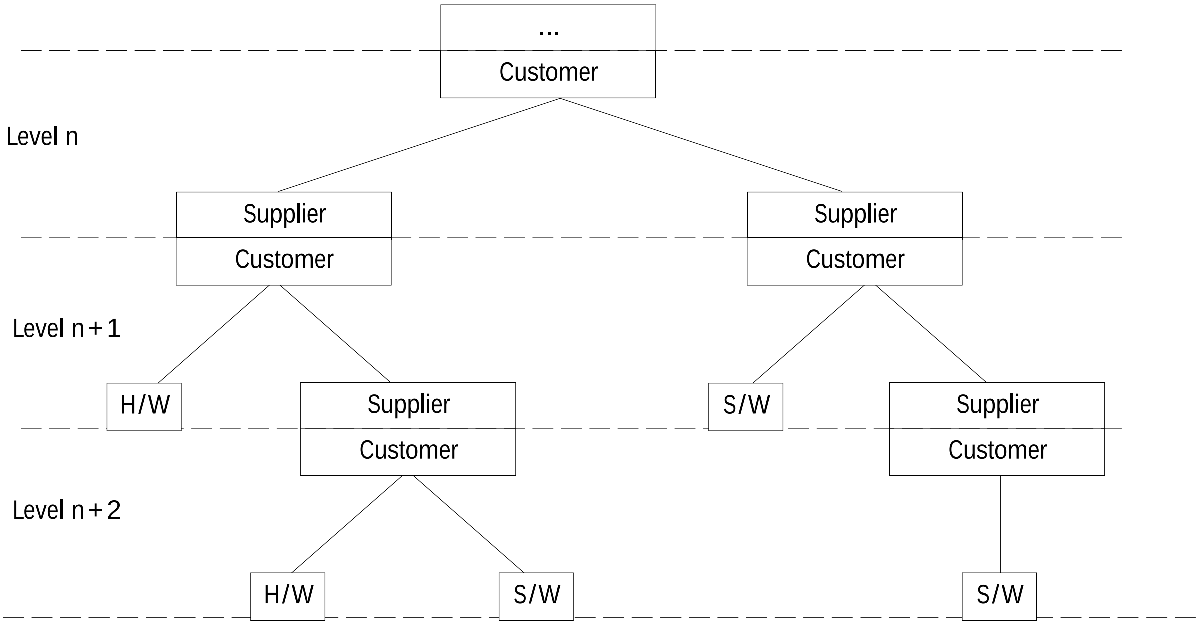

Figure 2: The recursive customer – supplier model

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. The term system in this standard shall be interpreted as system or subsystem at any decomposition level. An overview is shown in Figure 3. 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. 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.

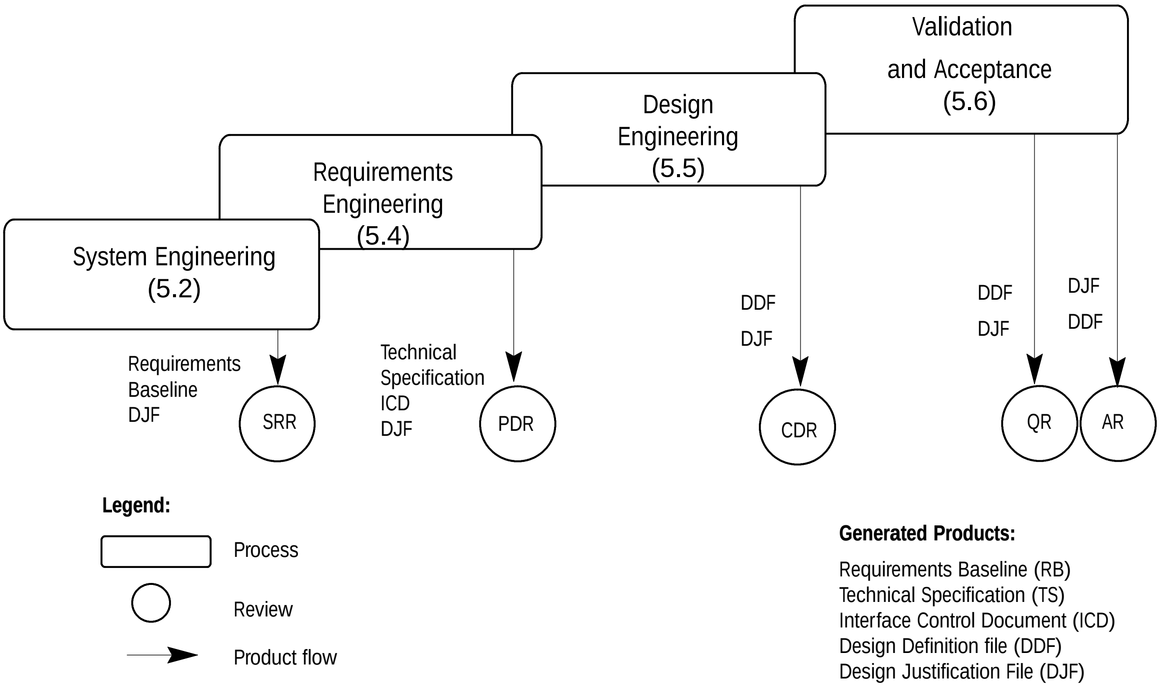

Figure 3: Overview of the software development processes

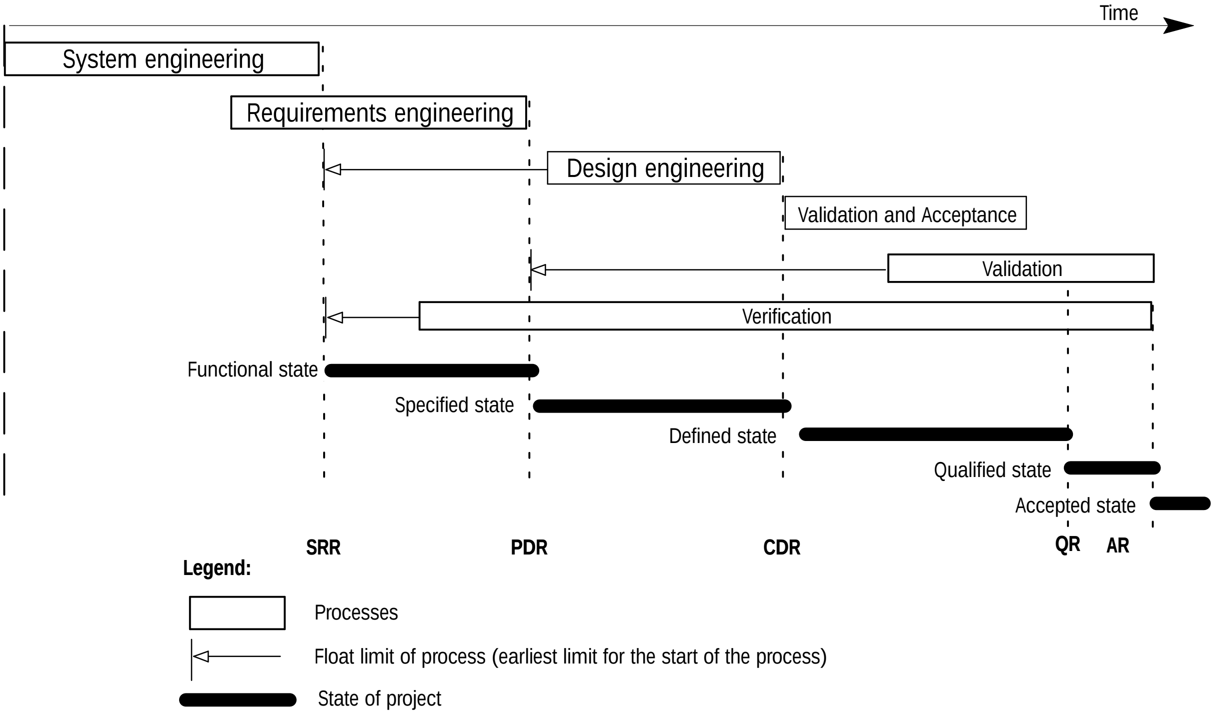

Figure 4: Process constraints

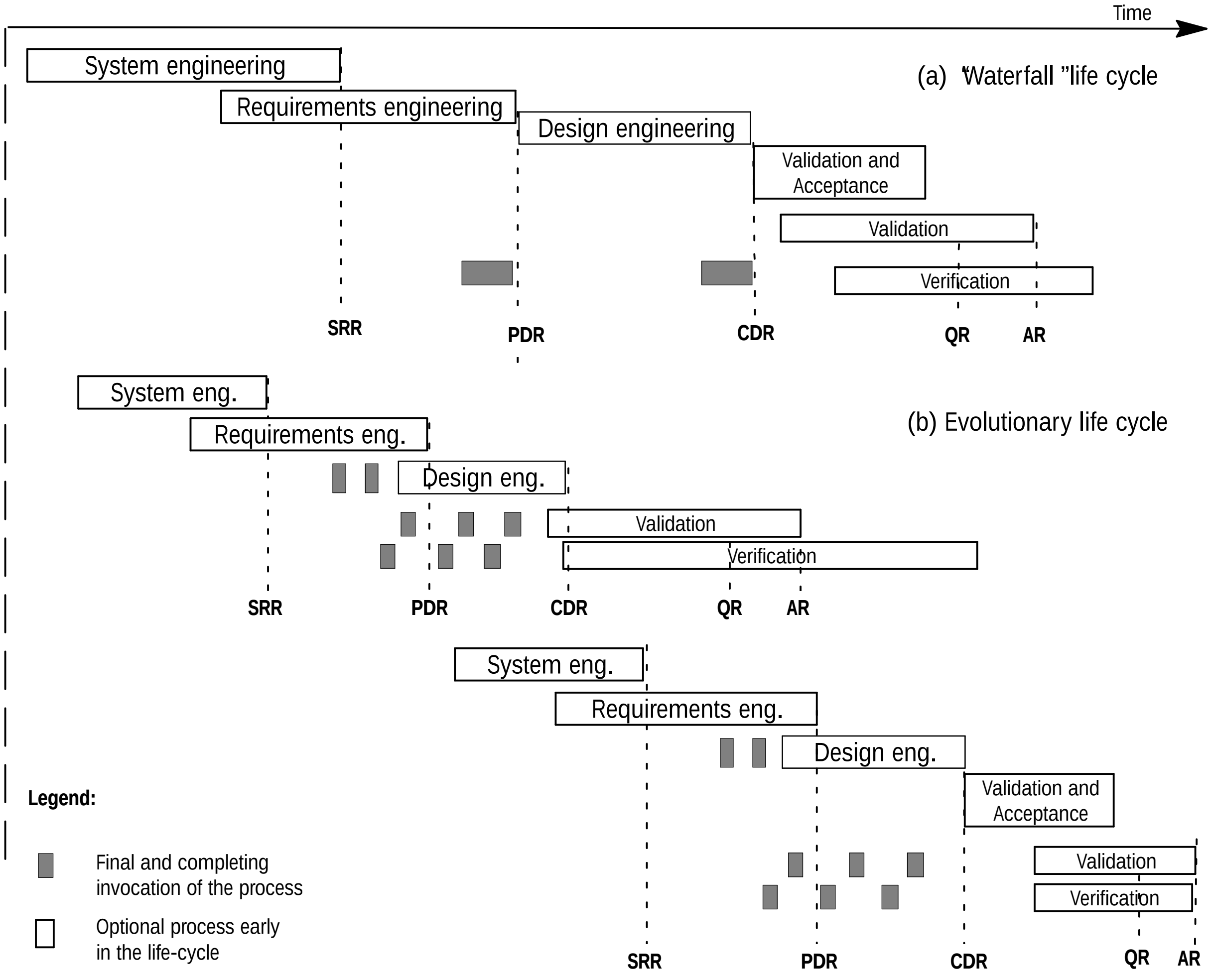

Figure 5: Accomodation of different software life cycles

4.2.1 Software requirements engineering process

The System Engineering processes produce the information required for input to 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".

4.2.2 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.

4.2.3 Validation and acceptance process

The validation and acceptance process may start after the CDR and when the validation with respect to TS is complete.

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

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".

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

General requirements for Operations are found in ECSS–E–70 Ground Systems and Operations.

4.2.5 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.

4.2.6 Software verification and validation (supporting) 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.

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 shall 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 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.

The organisation of the the general requirements of this standard is reflected in detail in the following chart:

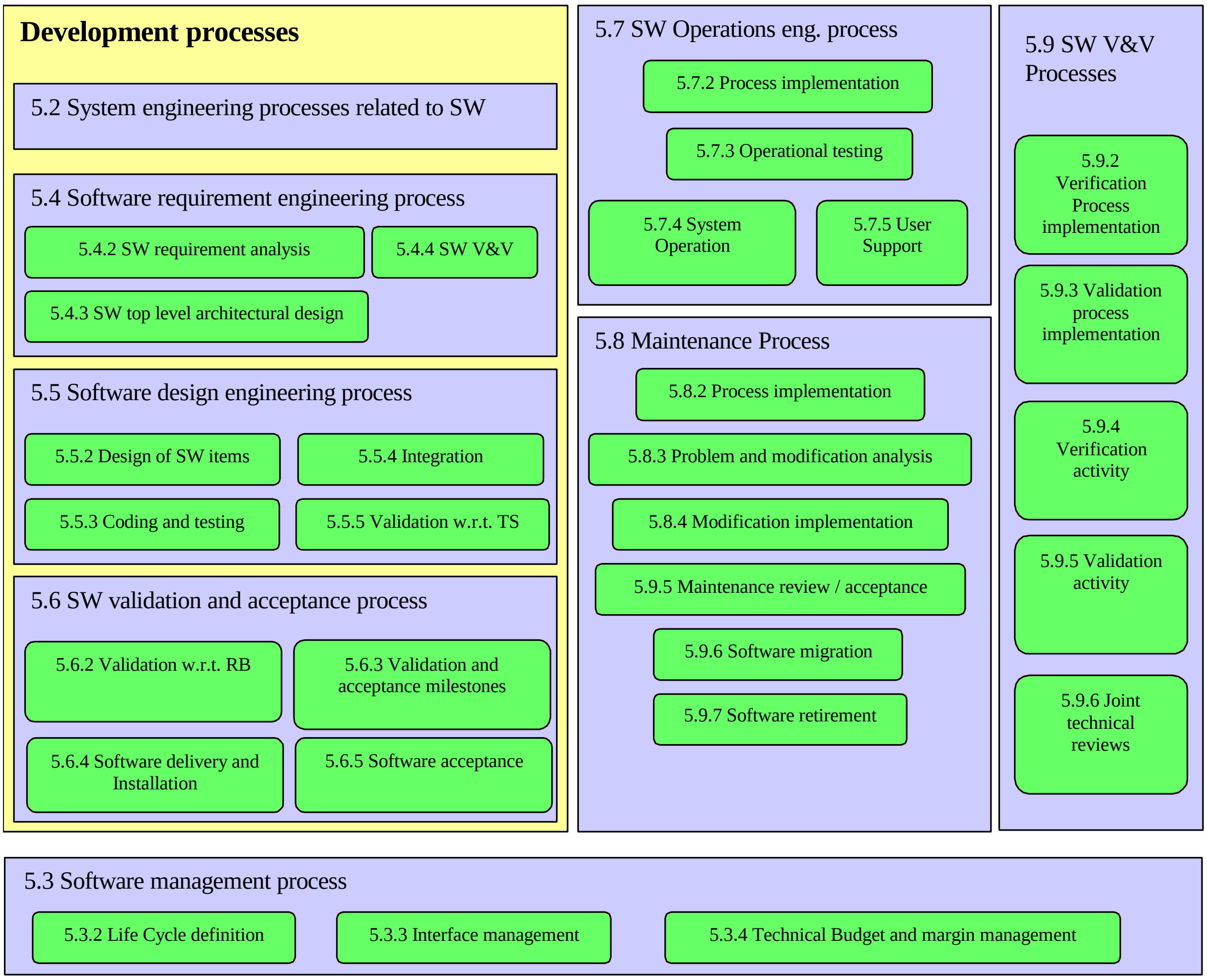
Development processes
5.2 System engineering processes related to SW
5.4 Software requirement engineering process
5.4.2 SW requirement analysis
5.4.4 SW V&V
5.4.3 SW top level architectural design
5.5 Software design engineering process
5.5.2 Design of SW items
5.5.4 Integration
5.5.3 Coding and testing
5.5.5 Validation w.r.t. TS
5.6 SW validation and acceptance process
5.6.2 Validation w.r.t. RB
5.6.3 Validation and acceptance milestones
5.6.4 Software delivery and Installation
5.6.5 Software acceptance
5.7 SW Operations eng. process
5.7.2 Process implementation
5.7.3 Operational testing
5.7.4 System Operation
5.7.5 User Support
5.8 Maintenance Process
5.8.2 Process implementation
5.8.3 Problem and modification analysis
5.8.4 Modification implementation
5.9.5 Maintenance review / acceptance
5.9.6 Software migration
5.9.7 Software retirement
5.9 SW V&V Processes
5.9.2 Verification Process implementation
5.9.3 Validation process implementation
5.9.4 Verification activity
5.9.5 Validation activity
5.9.6 Joint technical reviews
5.3 Software management process
5.3.2 Life Cycle definition
5.3.3 Interface management
5.3.4 Technical Budget and margin management

Figure 6: Structure of this standard

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–80B, which is the entry level document of the ECSS–Q series (Product assurance) for software projects.

This standard covers all aspects of space software product assurance including the implementation aspects of the software product assurance process, and both software process and product–related assurance activities.

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

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 4.2.

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). Therefore, the software configuration management may easily be automated.

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.

4.4.3 Engineering

4.4.3.1 ECSS-E-00: Policy and principles

This standard, which is informative in nature, contains the basic rules and over–all principles to be applied to all engineering activities during performance of a space project. It addresses the establishment, based on customer needs, of mission objectives, requirements, and specifications for space systems, and the design, definition, production, verification, operation, and eventual disposal of the systems themselves. It defines the scope and interfaces of these activities relative to the domains of management and product assurance which are addressed in the Management (– M) and Product Assurance (– Q) branches of the ECSS system, and explains how they may apply in different ways depending on the type of space sys–tem concerned.

4.4.3.2 ECSS-E-10: System Engineering

This standard is intended to guide the development of Systems (including hardware, software, man–in–the–loop, facilities and services) for space applications. It specifies implementation requirements for the responsible System Engineering organisation consistent with the assumption that the System Engineering process defined in Standard ECSS–E–10–01 is to be applied.

4.4.3.3 ECSS-E-70: Ground Segment

This Standard provides a high level description of all ground segment elements, the domain specific aspects of the associated engineering processes and defines related guidelines and requirements.

4.5 Tailoring of this standard

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

This standard is intended to be applied to the customer/supplier interfaces for a specific contract, in its tailored form.

The purpose of this clause is to give to the customer general guidance on tailoring the standard for a specific contract or project (see clause 1 and ECSS–M–00).

Tailoring may be done by deleting requirements, limiting applicability to a specific part of the system or modifying outputs or proofs to be provided, refining or specifying existing requirements or in exceptional cases adding new requirements.

Tailoring should include the definition of project–specific standards and procedures as appropriate.

There are several drivers for tailoring, i.e. software development constraints, dependability and safety aspects and commercial parameters.

Tailoring for software development constraints should take account of technical, operational and management factors.

Technical factors include:

- Novelty of the domain of application
- Complexity of the software and the system
- Criticality level
- Size of the software
- Reusability required of the software being developed
- Interface to system development projects
- Degree of use of COTS or existing software
- Maturity of the COTS and completeness or stability of the user requirements.

The operational factors comprise:

- Type of application (platform, payload, experiment)
- Number of potential users of the software
- Criticality of the software as measured by the consequences of its failure
- Expected lifetime of the software
- Number of sites where the software is used
- Operation, maintenance, migration and retirement constraints

Management factors:

- Amount of time and effort required to develop the software
- Budget requirements for implementing and operating the software
- Accepted risk level for the project
- Type of lifecycle
- Schedule requirements for delivering the software
- Number of people required to develop, operate and maintain the software
- Complexity of the organisation
- Experience of the supplier

(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 4.2).

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;
- software maintenance.

System level documentation is a prerequisite to the requirements engineering of the software. 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. *interface requirements [IRD(RB); SRR];*

c. *design constraints and verification and validation requirements [RB; SRR];*

d. *identification of lower level software engineering standards that will be applied [RB; RB] (see Q-80 clauses 6.3.2 and 6.3.3).*

5.2.2.2

System criticality analysis and critical functions analysis shall be performed for the system (in accordance with ECSS–Q–30).

EXPECTED OUTPUT: *Overall safety and reliability requirements of the software to be produced [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 manual operations 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(RB); 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 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 that 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].*

5.2.4.4

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.

5.2.4.5

The customer shall conduct a System Requirements Review (SRR) in accordance with subclause 5.3.2.6.

EXPECTED OUTPUT: *SRR milestone report [RB;SRR]*

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).

NOTE Space segment software is in general integrated with highly specialised processors and electrical equipment. The IRD 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: *System level interface requirements [IRD(RB); 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(RB); 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(RB); 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(RB); SRR].*

5.2.5.7

The customer shall plan the support from the software supplier in order to integrate the software at system level.

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

EXPECTED OUTPUT: *System level integration support requirements [MGT; 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

The customer shall establish system requirements for the operation of software products. The supplier’s response shall be agreed with the customer in the System Requirements Review (SRR), intended to release the operational plans for execution as established in subclause 5.7.

EXPECTED OUTPUT: *software operations requirements [RB; SRR];*

5.2.7 Software maintenance

The customer shall establish system requirements for the maintenance of software products. The supplier ś response shall be agreed with the customer in the System Requirements Review (SRR), intended to release the maintenance plans for execution as established in subclause 5.8.

EXPECTED OUTPUT: *software maintenance requirements [RB; SRR];*

5.3 Software management process

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. Management plans shall be produced in relation with the following activities:

- development;
- configuration and documentation management;
- verification and validation;
- maintenance;
- quality assurance on process and product.

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 5.9.6.

5.3.2 Software life cycle

5.3.2.1

To assure effective phasing and planning, the software development life cycle shall be broken into phases, each having its associated milestones. Detailed software lifecycle requirements are found in the level 3 standard ECSS-E-40-04.

5.3.2.2

The software supplier shall define and follow a software development life cycle in accordance with subclause 4.2, 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, operations and maintenance processes (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 4.2 and ECSS-M-30 shall be used.

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

5.3.2.3

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

EXPECTED OUTPUT: *Review Plan/Milestones (included in the software development plan) [TS;SRR]*

5.3.2.4

The output for each phase shall 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 are explicitly listed in the software life cycle definition.

5.3.2.5

The interface between development and maintenance (e.g. documents to be produced, tools to be kept for 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].*

5.3.2.6

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: *Customer approval of requirements baseline [RB; SRR]*

5.3.2.7

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

AIM: To establish the technical specification 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(TS); PDR];
d. top-level design trade-offs [DJF; PDR].

5.3.2.8

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(TS), DJF; PDR].*

5.3.2.9

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 software validation activities with respect to the Technical Specification and their relevant products (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 testsand results of validation with respect to the technical specifications) [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].

e. customer approval of the validation with respect to TS report [DJF;CDR]

5.3.2.10

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.

5.3.2.11

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].*

5.3.2.12

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].*

5.3.3 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 managemen t requirements [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;
- build an implementation–independent model of 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)*

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(TS); PDR];

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

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

e. 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];

f. data definition and database requirements [TS; PDR];

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

5.4.2.2

The supplier shall build an implementation– independent model of Software items in order to analyse and document software requirements.

EXPECTED OUTPUT: *Software logical model [TS; PDR]*

5.4.2.3

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

5.4.2.4

The supplier shall evaluate the software requirements invoking clause 5.9.4.1

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: *software architectural design [DDF; PDR];*

5.4.3.2

The design description shall as a minimum cover hierarchy, dependency and interfaces for the software components.

5.4.3.3

The design description shall document the process, data and control aspects of the product.

5.4.3.4

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(TS); PDR];*

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

5.4.3.5

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

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

5.4.3.6

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

EXPECTED OUTPUT: *Software integration test plan (preliminary) [DJF; PDR].*

5.4.3.7

The supplier shall evaluate the architecture of the software item and the interface designs invoking clause 5.9.4.2 .

EXPECTED OUTPUT:

a. *Architecture and interface verification report [DJF; PDR].*

b. *Top-Level Architectural Design to Requirements Traceability Matrices [DJF;PDR]*

5.4.3.8

The supplier shall conduct a Preliminary Design Review (PDR) in accordance with subclause 5.3.2.8. 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 5.9.2);

b. validation process implementation (subclause 5.9.3).

EXPECTED OUTPUT: *a. software verification plan -independence, criticality and effort [DJF; PDR];*

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

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

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

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

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

g. software validation plan - organization [DJF; 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;
- validation with respect to the technical specification.

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 interfaces design (update) [ICD(TS); CDR];*

b. internal interfaces design (update) [DDF; 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 unit test plan [DJF; CDR].*

5.5.2.5

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

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

5.5.2.6

The supplier shall evaluate the software design and test requirements invokingclause 5.9.4.3)

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

5.5.3.2

The supplier shall test each software unit ensuring that it satisfies its requirements. The test results shall be documented.

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

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

5.5.3.5

The supplier shall evaluate software code and test results, invoking clause 5.9.4.4):

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.

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.

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 test 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 invoking clause 5.9.4.5 and 5.9.4.6).

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

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

b. test completeness and code conformance report [DJF; CDR];

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

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

5.5.5 Validation with respect to the technical specification

This activity consists of the following tasks:

- validation with respect to the Technical Specification;
- conducting a joint review (CDR).

5.5.5.1

The supplier shall evaluate the software with respect to the Technical Specification. The validation process (5.9.5) should be invoked.

EXPECTED OUTPUT: *Validation with respect to the Technical Specification testing report [DJF;CDR]*

5.5.5.2

The supplier shall conduct a Critical Design Review (CDR) in accordance with subclause 5.3.2.9. All outputs required for CDR shall be prepared and verified by the process "verification of software documentation" (subclause 5.9.4.6) 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.

Every problem detected during the review shall be subject of a problem resolution process (invoking clause 5.9.5.5).

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

5.6 Software validation and acceptance process

5.6.1 Introduction

This process consists of the following activities:

- Validation with respect to the requirements baseline;
- Validation and acceptance milestones;
- Software delivery and installation;
- Software acceptance.

5.6.2 Validation with respect to the requirements baseline

The customer shall evaluate the software with respect to the Requirements Baseline. The validation process (5.9.5) should be invoked.This validation shall be performed not later than the Acceptance Review.

EXPECTED OUTPUT: *Validation with respect to requirements baseline testing report [DJF;QR/AR]*

5.6.3 Validation and acceptance milestones

This activity consists of the following tasks:

- Qualification review;
- Acceptance review.

5.6.3.1

The Qualification Review (QR) shall be conducted in accordance with subclause 5.3.2.11.

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. QR milestone report [DJF; QR].

5.6.3.2

The Acceptance Review (AR) shall be conducted in accordance with subclause 5.3.2.12. 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. “Support to acceptance reviews and testing” (subclause 5.6.5.4@@@@), is invoked for this.

EXPECTED OUTPUT:

a. final software acceptance data package [DJF; AR];

b. acceptance testing documentation [DJF; AR];

c. AR milestone report [DJF; AR];

d. software release documentation [DDF; AR];

e. software delivery on specified data medium [DDF; AR].

5.6.4 Software delivery and installation

This activity consists of the following tasks:

- Preparation and update of the deliverable software product;
- Training and support;
- Installation planning;
- Installation report.

5.6.4.1

The supplier shall:

a. Prepare and update the deliverable software product as established 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 [DDF;QR];

b. software release documentation [DDF;QR];

c. software acceptance data package [DJF;QR].

5.6.4.2

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

EXPECTED OUTPUT: *Training material [TS; QR]*

5.6.4.3

The supplier shall develop a plan to install the software product in the target environment.

EXPECTED OUTPUT: *Installation plan [DJF; AR].*

5.6.4.4

The resources and information necessary to install the software product shall be determined and be available.

The supplier shall assist the customer with the set–up activities.

The supplier shall install the software product in accordance with the installation plan.

It shall be ensured that the software code and databases initialise, execute and terminate as specified in the installation plan.

The installation events and results shall be documented.

EXPECTED OUTPUT: *Installation report [DJF; AR].*

5.6.5 Software acceptance

This activity consists of the following tasks:

- Acceptance test planning;
- Execution of acceptance tests;
- Generation and installation of the executable code;
- Supplier support to customer s acceptance review;
- Evaluation of the acceptance tests.

5.6.5.1

The customer shall establish an acceptance test plan specifying the intended acceptance tests with tests suited to the target environment.

EXPECTED OUTPUT: *Acceptance test plan [DJF; AR]*

5.6.5.2

The customer shall execute the acceptance testing.

EXPECTED OUTPUT: *Acceptance test report [DJF; AR].*

5.6.5.3

The acceptance shall include generation of the executable code from configuration managed source code components and its installation on the target environment.

5.6.5.4

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, audits, software validation testing (ECSS–Q–80B subclause 6.3.4), and system validation testing (if performed). The results of the acceptance reviews and testing shall be documented.

EXPECTED OUTPUT: *Acceptance testing documentation [DJF; AR].*

5.6.5.5

The acceptance tests shall be evaluated with respect to the Requirements Baseline.

EXPECTED OUTPUT: *Traceability of Acceptance tests to Requirements Baseline [DJF;AR].*

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.

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.

5.7.2 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.

5.7.2.1

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].*

5.7.2.2

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].*

5.7.2.3

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].*

5.7.3 Operational testing

This activity consists of the following tasks:

- Perform operational testing;
- demostration of capability to meet operational requirements.

5.7.3.1

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.

5.7.3.2

The customer shall ensure that prior to the operations phase, the software has been demonstrated capable of meeting the operational requirements.

This demonstration may be part of the acceptance tests of the system.

This demonstration should be representative in terms of:

- hardware operating environment;
- situations to which the software is designed to be fault tolerant;
- system configuration;
- sequence of operations and phases;
- operator interventions.

5.7.4 System operation

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

5.7.5 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.

5.7.5.1

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

5.7.5.2

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.

5.7.5.3

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 4.2) 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 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;
- in flight modification;
- 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 [MF; QR].*

5.8.2.2

Software maintenance shall be performed using the same procedures, methods, tools and standards as used for the development.

5.8.2.3

The maintainer shall establish procedures for receiving, recording and tracking problem reports and modification requests from the operator and providing feedback to the operator. 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 [MF; QR].*

5.8.2.4

The maintainer shall implement (or establish the 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 [MF].*

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;
- document changes according to the procedures for document control and configuration management;
- 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: *Modification identification [MF].*

5.8.4.2

All changes to the software product shall be documented in accordance with the procedures for document control and configuration management.

5.8.4.3

The maintainer shall enter the software engineering process (subclause 4.2) 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 In flight modification

5.8.5.1

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 ś 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 may not always be reprogrammable in flight. In cases where reprogramming is required, the appropriate requirements shall be captured already at SRR, and the supplier ś 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 for in-flight modification capabilities [RB; SRR].*

5.8.5.2

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].*

5.8.6 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 [MF].*

5.8.7 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 operator 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.

5.8.7.1

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.

5.8.7.2

A migration plan shall be developed, documented, and executed. The planning activities shall include the operator. 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 [MF]*

5.8.7.3

The operator 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;

d. the date as of which the transition takes place.

EXPECTED OUTPUT: *Migration justification file [MF]*

5.8.7.4

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.

5.8.7.5

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.

5.8.7.6

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.

5.8.7.7

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.

5.8.8 Software retirement

The software product will be retired on the request of the customer.

This activity consists of the following tasks:

- retirement planning;
- notification of retirement to the operator;
- parallel operations of the retiring and new software product;
- to allow access to data related with the retired software product.

5.8.8.1

A retirement plan to remove active support by the operator and maintainer shall be developed and documented. The plan shall address the items listed below. The plan shall be executed.

a. Cessation of full or partial support after a certain period of time;

b. Archiving of the software product and its associated documentation;

c. Responsibility for any future residual support issues;

d. Transition to the new software product, if applicable;

e. Accessibility of archive copies of data.

EXPECTED OUTPUT: *Retirement plan [MF]*

5.8.8.2

The operator shall be given notification of the retirement plans and activities. Notifications shall include the following:

a. Description of the replacement or upgrade with its date of availability;

b. Statement of why the software product is no longer to be supported;

c. Description of other support options available, once support has been removed.

EXPECTED OUTPUT: *Retirement notification to operator [MF]*

5.8.8.3

Parallel operations of the retiring and the new software product may be conducted for smooth transition to the new system. During this period, user training shall be provided as specified in the contract.

5.8.8.4

Data used by or associated with the retired software product shall be accessible in accordance with the contract requirements for data protection and audit applicable to the data.

5.9 Software verification and validation (supporting) processes

5.9.1 Introduction

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

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).

The software verification and validation engineering processes consist of:

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

5.9.2 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;
- development and documentation of a verification plan.

5.9.2.1 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–80B subclauses 6.2.2 (software dependability and safety) and 6.2.5.14 (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: *Verification plan - criticality and effort.*

5.9.2.2 Establishment of the verification process, methods and tools

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

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 5.9.4, 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.*

5.9.2.3 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–80B subclause 6.2.5.14 (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 [MGT].*

5.9.2.4 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: *Software verification plan - organization and activities.*

5.9.3 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.

5.9.3.1 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–80B subclause 6.3.4.21.

EXPECTED OUTPUT: *Software validation plan - effort and independence.*

5.9.3.2 Establishment of a validation process

The validation process shall be established to validate the software product. Validation tasks defined in subclause 5.9.5, including associated methods, techniques, and tools for performing the tasks, shall be selected.

EXPECTED OUTPUT: *Software validation plan - methods and tools.*

5.9.3.3 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 and ECSS–Q–80B, subclause 6.3.4.21.

EXPECTED OUTPUT: *a. Appropriate element of project requirements documents dealing with project organization [MGT].*

b. independent software validation plan.

5.9.3.4 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: *Software validation plan - organization and activities.*

5.9.4 Verification activity

This activity consists of the following engineering tasks:

- verification of software requirements;
- verification of the software architectural design;
- verification of the software detailed design;
- verification of code;
- verification of software integration;
- verification of software documentation;
- evaluation of test specifications;
- verification of the software validation with respect to the Technical Specification and with respect to the Requirements Baseline;
- problem and nonconformance handling.

5.9.4.1 Verification of software requirements

The requirements shall be verified considering the criteria listed below:

a. Software requirements are traceable to system partitioning and system requirements.

b. Software Requirements are externally and internally consistent (not implying formal proof consistency).

c. Software Requirements are verifiable.

d. Feasibility of software design.

e. Feasibility of operations and maintenance.

f. 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.

5.9.4.2 Verification of the software architectural design

The architectural design shall be verified considering the criteria listed below:

a. Traceability from the requirements to the software item.

b. External Consistency with the requirements of the software item

c. Internal Consistency between the software components

d. Feasibility of producing a detailed design

e. Feasibility of operations and maintenance

f. The design is correct with respect to the requirements and the interfaces

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

h. The chosen design can be derived from requirements

i. 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 matrices;*

b. architectural design and interface verification report;

5.9.4.3 Verification of the software detailed design

The software detailed design shall be evaluated in accordance with the criteria listed below:

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. Feasibility of testing.

e. Feasibility of operation and maintenance.

f. The design is correct with respect to requirements and interfaces.

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

h. The chosen design can be derived from requirements.

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

EXPECTED OUTPUT: *a. detailed design traceability matrices;*

b. detailed design verification report;

5.9.4.4 Verification of code

The code shall be verified considering 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. The chosen 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.

e. External consistency with the requirements and design of the software item.

f. Internal consistency between software units.

g. Test coverage of units.

h. Feasibility of software integration and testing.

i. Feasibility of operation and maintenance.

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

b. software code verification report.

5.9.4.5 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. In addition, the supplier shall evaluate software integration test plan, design, code, tests, test results and operation manual, considering the criteria specified below:

a. Traceability to software architectural design.

b. External consistency with system requirements.

c. Internal consistency.

d. Interface testing coverage.

e. Requirements test coverage.

f. Conformance to expected results.

g. Feasibility of software validation testing.

h. Feasibility of operation and maintenance.

EXPECTED OUTPUT: *Software integration verification report*

5.9.4.6 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*

5.9.4.7 Evaluation of test specifications

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

EXPECTED OUTPUT: *a. traceability of the requirements baseline to the validation tests.*

b. traceability of the technical specifications to the validation tests.

5.9.4.8 Verification of software validation with respect to the technical specifications and the requirements baseline

The validation tests shall be verified considering the criteria listed below:

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: *Test results evaluation.*

5.9.4.9 Problem and nonconformance handling

Problems and nonconformances detected by the software verification effort shall be entered into the problem resolution process (ECSS–Q–80B subclause 5.3.5 and 5.3.6). 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.*

5.9.5 Validation activity

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;
- problem and nonconformance handling;
- test readiness review.

5.9.5.1 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.

The supplier shall evaluate the test specifications in accordance with subclause 5.9.4.6.

EXPECTED OUTPUT: *Software validation testing specification.*

5.9.5.2 Conducting the validation tests

The validation tests shall be conducted as specified in the output of subclause 5.9.4.1 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.*

5.9.5.3 Updating the operations manual

The supplier shall update the operations manual as necessary.

EXPECTED OUTPUT: *Operations manual (update).*

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

The supplier shall evaluate the design, code, tests, test results, and operations manual in accordance with the criteria listed below~~:~~

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.*

5.9.5.5 Problem and nonconformance handling

Problems and nonconformances detected during the validation shall be the subject of a problem resolution process (ECSS–Q–80B subclauses 5.3.5 and 5.3.6). 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.*

5.9.5.6 Test readiness review

Test readiness reviews, as established in section 5.9.6, joint technical review process, shall be held before the commencement of key test activities.

5.9.6 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 both parties, where one party (reviewing party) reviews another party (reviewed party).

This process consists of the following activities:

- support to software reviews;
- document technical reviews.

5.9.6.1 Support to software reviews

The software support to 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.

NOTE Internal reviews could be replaced by Inspections.

EXPECTED OUTPUT: *Milestone review reports.*

5.9.6.2

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.*

(This page is intentionally left blank)

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

6.2.1 Introduction

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 requirements are presented specifying which is the general activity that contains them.

6.2.2 System level integration of software: system observability

6.2.2.1

Observability data shall include the data for the system observability and shall take into account the constraints imposed by the computer such as the bandwidth allocation, the overloading of the processor, etc.

NOTE 1 The general requirement for software observability data intended to facilitate the software integration or the software trouble shooting. But space on–board software purpose is generally to access or control (in a reactive or interactive way) some hardware, whose visibility is also important.

NOTE 2 Observability requirements might go against the performance requirements (computer throughput, bandwidth, ground exploitation rate of telecommands, etc). This has to be taken into account when specifying the observability requirements (e.g. considering the need of oversampling).

EXPECTED OUTPUT: *System observability requirements [RB;SRR]*

6.2.2.2

The system team shall specify the system observability data individually. However the software observability data can only be exhaustively listed in the Technical Specification. The customer requirements shall give room for a supplier complement of the software observability data.

6.2.2.3

When specifying the observability requirements, the customer shall trade off the software visibility with the risk of activating undesirable code on–board.

NOTE The software observability requirements, considered only during the integration – and not during flight–, might result into some inactivated code.

6.2.3 System level integration of software: system database

6.2.3.1

The customer shall specify a system database to insure the consistency of common data, considering the following additional aspects:

a. Specification of the system database use for the supplier. For instance, the database may be used to produce automatically configured software (generation of tables, constant data, initial values, etc).

b. Specification of the system database size , the possible modification that can be foreseen after the Acceptance Review, and the accepted impact in terms of software maintenance.

EXPECTED OUTPUT: *System database specification (content and use) [RB;SRR]*

6.2.4 Software lifecycle: Detailed Design Review

6.2.4.1

At the end of the detailed design, the software supplier shall hold a Detailed Design Review (DDR) to which the customer shall be invited to attend.

AIM:

- Review the detailed design.
- CPU and memory constraints are of particular importance for on board software. It is necessary to pay much attention to budget evaluation along the development. However, preliminary evaluations are performed by the PDR. The results are refined with detailed design and shall be reviewed.
- In line with the concern expressed above on the late arrivalof the requirements, the DDR could be joined if felt necessary with a Delta PDR. Its purpose will be to review the technical specification, in particular with respect to its completenes and stability before coding.

EXPECTED OUTPUT:

a. customer approval of the design definition file (architectural design, detailed design) [DDF,DJF;DDR]

b. customer approval of the design of system level interface and the system level integration plan [DDF,DJF;DDR]

c. customer approval of the margins and technical budget status [DJF;DDR]

d. customer approval of the updated technical specifications [TS;DDR]

6.2.4.2

In order to avoid that software items are subject to several reviews, the software elements defined in the general requirements for review at CDR, but actually reviewed at DDR, shall be removed from the CDR list.

6.2.5 Software requirements analysis: logical model

6.2.5.1

The supplier shall construct a logical model of the functional requirements of the future software product. The model may be the result of an iterative process with verification by the customer. The model will support the requirement capture, document and formalise the software requirements.

NOTE A logical model is a representation of the specification, independent from the implementation, possibly executable, under a formalised language. For example, a logical model can use the SADT notation, or the UML notation following e.g. the HOORA method. Or it can use the SDL notation or Lotos to make a behavioural view of the specification. It can use formal methods (like Z, B, Vdm++ Raise, etc.) to prove properties of the specification. The logical model allows in particular to verify that a specification is complete (because every element of the model could be written from an English text), and consistent (because the model compiles).

The logical model can be completed by specific feasibility analyses such as benchmarks, in order to check the technical budgets (memory size, computer throughput). In case the modelling technique allows for it, preliminary automatic code generation could be used (e.g. Data Flow Diagrams for AOCS).

The scenario that have been defined to test the logical model can be reused for the software validation and reported for that purpose into the verification plan (see 5.4.4.1)

EXPECTED OUTPUT: *Software logical model [TS;PDR]*

6.2.5.2

For space reactive software, the logical model shall include a behavioural view. Such a view can be given for example by Finite State Machines, Petri Nets, the UML sequence diagrams, Message Sequence Charts, SDL, Lotos, etc.

NOTE Space reactive software is a software that fulfills the following characteristics:

It is real–time software.

It runs remotely.

It manages closely a specific limited target computer.

It controls a potentially long living system with visibility limitation into an hostile environment.

It drives its own availability and reliability.

6.2.5.3

For interactive software, the logical model shall include a prototype of the Man Machine Interface (see 6.5.4), which will be proposed to the Customer for validation.

NOTE Space interactive software is a software that fulfills the following characteristics:

Generally it is not real–time, although there can be some real–time connection from the laptop to the environment

It is loaded into the processor from a external device (a disk, a floppy)

It doesn t run remotely.

It does not manage the hardware. It uses a general–purpose platform with extendable resources as a mean to execute.

It is generally intended to control experiments whose lifetime is in the range from hours to one year.

It has a Graphical User Interface.

As space reactive software, space interactive software runs into a hostile environment (radiation)

6.2.6 Verification of software requirements: feasibility of design and operation

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

- schedulability analysis;
- technical budget monitoring;
- behaviour model verification.

6.2.6.1

The supplier shall use an analytical model to perform a schedulability analysis and prove that the design is feasible.

EXPECTED OUTPUT: *schedulability analysis [DJF;PDR]*

6.2.6.2

The Technical Budget (memory size, CPU utilisation) shall be monitored:

a. The memory size shall be estimated for static code size, static data size and stack size. These should be estimated for each terminal object in the design. Stack size must be expressed on a per–thread basis, so it would be applied to terminal active objects.

b. The CPU utilisation shall be estimated considering the Worst Case Execution Time of each terminal active object (therefore including the call to the protected objects).

NOTE The Worst Case Execution Time of each terminal active object is multiplied by the number of times the object is executed per second. The resulting quantity is summed over all non–terminal objects. The result will be the estimated percentage processor utilisation.

EXPECTED OUTPUT: *technical budgets (update) [DJF;PDR]*

6.2.6.3

The software behaviour shall be modelled and verified by means of Formal Description Techniques (SDL, Lotos, synchronous languages, etc.) and the supporting tools. These techniques can also be used to validate the appropriateness of selected or developed communication protocol.

EXPECTED OUTPUT: *software behaviour verification [DJF;PDR]*

6.2.6.4

Modelling or simulation shall be used to prove the feasibility of the design, if no analytical model exists.

6.2.7 Software top level architectural design: static and dynamic architecture

6.2.7.1

The software top–level architectural design shall describe

a. the static architecture (decomposition into software elements such as packages and classes in UML or modules in HOOD),

b. the dynamic architecture (which involves active objects such as threads, tasks and processes),

c. the mapping between the static and the dynamic architecture (typically the relationship HOOD items and tasks),

d. the software behaviour.

EXPECTED OUTPUT: *a. software static architecture [DDF;PDR]*

b. selected analysable computational model [DDF;PDR]

c. software dynamic architecture [DDF;PDR]

d. software behaviour [DDF;PDR]

6.2.7.2

An architectural method (e.g. HOOD) shall be used to produce the static architecture including:

- Software elements, their provided and required interfaces.
- Software element relationships (use, inheritance, include...).

6.2.7.3

The dynamic architecture design shall select an analysable computational model and shall be described (e.g. using HRT–HOOD notation) accordingly.

6.2.7.4

The software architecture design shall also describe the dynamic behaviour of the software, for instance by means of description techniques based on automata and scenario. Scheduling simulations will be performed.

6.2.8 Design of software items: physical model

6.2.8.1

The software design shall produce the physical model of the software items described during the top–level architecture. The physical model includes the static design, the dynamic design, the mapping between the static and the dynamic views, and the behaviour of the software elements.

EXPECTED OUTPUT: *a. software static design [DDF;DDR | CDR]*

b. software dynamic design [DDF;DDR | CDR]

c. software elements behaviour [DDF;DDR | CDR]

d. compatibility of design methods with the computational model [DDF;DDR | CDR]

6.2.8.2

One or several design methods (e.g. HOOD, Data Flow Diagram) shall be used to produce the static design including:

- Software elements, their provided and required interfaces.
- Software element relationships (use, inheritance, include...).

6.2.8.3

The dynamic design shall be based on the computational model selected during the top–level architecture, and shall describe (using a compatible method, e.g. HRT–HOOD notation) the dynamic aspect of the physical model accordingly.

6.2.8.4

The software design shall also describe the dynamic behaviour of the software elements, for instance by means of description techniques based on automata and scenario. Scheduling simulations will be performed.

6.2.8.5

In some cases, several design methods might have to be used for different items of the same software (e.g. HOOD for Data Handling and Data Flow Diagram for Control). In this case, special care shall be dedicated to check that both methods are, from a dynamic standpoint, consistent between themselves and consistent with the selected computational model.

6.2.9 Verification of design: feasibility of operation

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

- schedulability analysis refinement;
- technical budget monitoring;
- behaviour model verification.

6.2.9.1

The schedulability analysis performed during top–level architecture shall be refined with the fresh design information.

NOTE Estimations will be more precise due to the better knowledge of the design.

EXPECTED OUTPUT: *schedulability analysis (update) [DJF;CDR]*

6.2.9.2

The Technical Budget (memory size, CPU utilisation) shall be monitored:

a. The memory size shall be refined for static code size, static data size and stack size.

b. The CPU utilisation shall be refined.

EXPECTED OUTPUT: *technical budgets (update) [DJF;CDR]*

6.2.9.3

Software behaviour shall be modelled and verified by means of Formal Description Techniques (SDL, Lotos, synchronous languages, etc.) and the supporting tools. These techniques can also be used to validate the appropriateness of selected or developed communication protocols.

EXPECTED OUTPUT: *software behaviour verification [DJF;CDR]*

6.2.10 Verification of design: feasibility of testing

6.2.10.1

The feasibility of testing evaluation shall check the following aspects:

a. Appropriate verification points have been foreseen and included in the detailed design in order to prepare the effective verification of the performance requirements.

b. Some assertions defining computational invariant properties, or temporal properties (possibly derived from the behavioural model) are added within the design.

NOTE The feasibility of testing must verify that the performance and that robustness tests are possible.

The invariant and temporal properties will be used to prepare for the robustness tests to be performed during unit testing. They will also help in the verification of safety and reliability requirements during software system testing.

EXPECTED OUTPUT: *testing feasibility report [DJF;CDR]*

6.2.11 Verification of coding and testing: feasibility of operation

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

- schedulability analysis refinement;
- technical budget update.

6.2.11.1

The schedulability analysis performed during top–level architecture shall be refined with the actual information extracted from the code.

EXPECTED OUTPUT: *schedulability analysis (update) [DJF;CDR]*

6.2.11.2

The technical budget shall be updated with the measured values and shall be compared to the margins.

EXPECTED OUTPUT: *technical budgets (update) [DJF;CDR]*

6.2.12 Evaluation of Validation: complementary system level validation

6.2.12.1

The Supplier shall identify the requirements that cannot be tested on its own environment, and shall forward to the Customer a request for having them validated on the real system.

NOTE Some of the requirements may not have been verified because the test environment used for the validation does not allow it. These requirements must be tested when the software is integrated within the system (e.g. satellite, launcher).

EXPECTED OUTPUT: *Complement of validation at system level [DJF;AR]*

6.2.13 Maintenance: long term maintenance

6.2.13.1

The maintenance plan shall take into account the spacecraft lifetime. If this lifetime goes after the expected obsolescence date of the software engineering environment, then the maintainer shall propose solution to be able to produce and upload modifications to the spacecraft up to its end of life.

NOTE Examples of such solutions are:

– To procure enough copies of the development hardware (workstations, disks, test environment, etc.) to make sure that statistically, at least one of them will still be working at the spacecraft end of life date.

– To prepare software simulators of the development hardware able to run old software development tools on new machines.

– To port the software development environment on new machines. When some tools are not running on the new machines, adopt new tools with equivalent functionality if they exist. Note that in case of compilers, the software would have to be recompiled and extensively tested again.

As these solutions are likely to involve high cost, they must be traded–off with the customer.

EXPECTED OUTPUT: *Long term maintenance solutions [MF]*

6.3 Ground segment software

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

6.4 Software reuse

6.4.1 Introduction

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" are intended to be part of the software product.

6.4.2 Developing software for intended reuse

6.4.2.1

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: *Requirements for 'design for reuse' [RB; SRR]*

6.4.2.2

The supplier shall define procedures, methods and tools for reuse, and he shall apply these to the software engineering processes to comply with the re–usability requirements for the software development.

EXPECTED OUTPUT: *Software for intended re-use - justification of methods and tools [DJF;PDR]*

6.4.2.3

An evaluation of the re–use potential of the software shall be conducted at PDR and CDR.

EXPECTED OUTPUT: *Software for intended re-use - evaluation of re-use potential [DJF;PDR,CDR]*

6.4.3 Re-use of pre-developed software

6.4.3.1

The supplier shall consider the "reuse"of already developed, commercial off-the-shelf and modifiable off-the-shelf software. 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–80B subclause 6.2.6.

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].

6.4.3.2

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 (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].*

6.4.3.3

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 Introduction

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; PDR];*

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.

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.

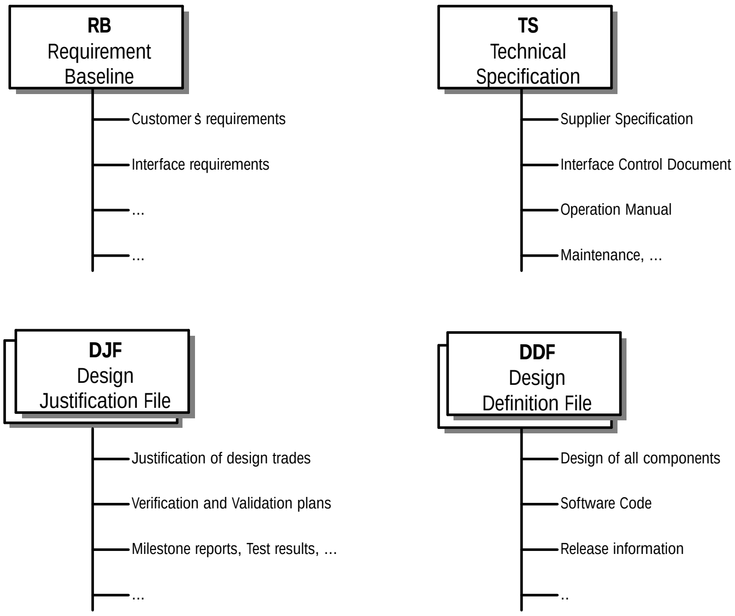

Figure A–1: Overview of software engineering documents

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–c.	Design constraints and verification and validation requirements [RB; SRR]
5.2.2.1–d.	Identification of lower level software engineering standards that will be applied [RB; SRR]
5.2.2.2	Overall safety and reliability requirements of the software to be produced [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]
5.2.4.4	SRR milestone report [RB;SRR]
5.2.5.1	Software observability requirements [RB; SRR]
5.2.5.4	Development constraints [RB; SRR]
5.2.6.1	Operations requirements [RB; SRR]
5.2.7	Maintenance requirements [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]
5.8.5.1	Requirements of in–flight modification capabilities [RB;SRR]
6.2.1.1	System observability requirements [RB;SRR]
6.2.2.1	System database specification (content and use) [RB;SRR]
6.4.1.1	Requirements for ‘design for review’[RB;SRR]
6.4.2.2	Software acquisition process for COTS and MOTS Products [RB;SRR]
6.5.2	MMI software mock–up requirements [RB;SRR]
6.5.3	MMI general requirements and guidelines [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 b.	Interface requirements [IRD(RB); SRR]
5.2.3.2–b.	Software/hardware interface requirements [IRD(RB); SRR]

5.2.5.2	System level interface requirements [IRD(RB); SRR]
5.2.5.3	System level data interfaces [IRD(RB); SRR]
5.2.5.6	System level integration preparation requirements [IRD(RB); SRR]
5.2.5.7	System level integration support requirements [IRD(RB); 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 SRR**
5.3.2.3	Review Plan / Milestones (included in the software development plan) [TS;SRR]

Requirement	**TS contents at PDR**
5.3.2.7–a.	Technical specification of the software [TS; PDR]
5.3.2.8	Customer approval of technical specification and top-level architecture [TS, DDF, ICD(TS), 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–c.	Safety specifications, including those related to methods of operation and maintenance, environmental influences, and personnel injury [TS; PDR]
5.4.2.1–d.	Security specifications, including those related to factors which might compromise sensitive information [TS; PDR]
5.4.2.1–e.	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–f.	Data definition and database requirements [TS; PDR]
5.4.2.1–g.	Installation and acceptance requirements of the delivered software product at the operation and maintenance site(s) [TS; PDR]
5.4.2.2	Software logical model [TS;PDR]
5.8.5.2	Specifications for in–flight modifications [TS;PDR]
6.2.4.1	Software logical model [TS;PDR]
6.4.2.1–a.	Specification of intended reuse [TS;PDR]
6.4.2.3	Software acquisition process implementation [TS;PDR]
6.5.4–a.	MMI specifications for software [TS;PDR]

Requirement	**TS contents at QR**
5.6.4.2	Training material [TS; QR]

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**
5.3.2.7–c.	Interface Control Document [ICD(TS); PDR]
5.3.2.8	Customer approval of technical specification and top-level architecture [TS, DDF, ICD(TS), DJF; PDR]
5.4.2.1–b.	Interfaces external to the software item [ICD(TS); PDR]
5.4.3.4–a.	Preliminary (top-level) external interfaces design [ICD(TS); PDR]

Requirement	**ICD contents at CDR**
5.5.2.2–a.	External ICDs (update) [ICD(TS); CDR]

A.3.2 Software maintenance plan

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

A.3.3 Operations manual

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

Requirement	**Operations manual contents at CDR**
5.3.2.9–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]

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(TS), DJF; PDR]
5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]
5.4.2.4–a.	Requirement traceability matrices [DJF; PDR]
5.4.2.4–b.	Requirements verification report [DJF; PDR]
5.4.3.6	Preliminary software integration test plan [DJF; PDR]

5.4.3.7–a.	Architecture and interface verification report [DJF; PDR]
5.4.3.7–b.	Top-level architectural design to requirements traceability matrices [DJF; PDR]
5.4.3.8	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]
5.4.4.1–g.	Software validation plan –organization [DJF;PDR]
6.2.6.1–a.	Schedulability analysis [DJF;PDR]
6.2.6.1–b.	Technical budgets (update) [DJF;PDR]
6.2.6.1–c.	Software behaviour verification [DJF;PDR]
6.4.1.2	Software for intended reuse – justification of methods and tools [DJF;PDR]
6.4.1.3	Software for intended re–use – evaluation of re–use potential [DJF;PDR,CDR]
6.4.2.1–b.	Justification of reuse with respect to Requirements Baseline [DJF;PDR]
6.5.4–b.	Report on evaluation of MMI specifications using a software mock–up [DJF;PDR]

Requirement	**DJF contents at CDR milestone**
5.3.2.9–b.	Customer approval of the design justification file (e.g. results of unit and integration tests) [DJF; CDR]
5.3.2.9–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]
5.5.2.4	Software unit test plan [DJF; CDR]
5.5.2.5	Software integration test plan [DJF;CDR]
5.5.2.6–a	Design verification report [DJF; CDR]
5.5.2.6–b.	Design traceability matrices [DJF; CDR]
5.5.3.1–b.	Software unit test plan (update) [DJF; CDR]
5.5.3.2–b.	Software unit test reports [DJF; CDR]
5.5.3.4	Software integration teest plan (update) [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.	Test completeness and code conformance report [DJF; CDR]
5.5.4.4–c.	Software documentation verification report [DJF; CDR]

5.5.4.4–d.	Feasibility confirmation of validation testing, operations and maintenance [DJF; CDR]
5.5.4.5	CDR milestone report [DJF; CDR]
5.5.5.1	Validation with respect to the Technical Specification testing report [DJF;CDR]
5.5.5.1	Software validation with respect to the Technical Specification testing specification [DJF;CDR] (invocation of 5.9.5.1)
5.5.5.1	Operations manual (update) [DJF;CDR] (invocation of 5.9.5.3)
5.5.5.1	Software design and test evaluation report [DJF;CDR] (invocation of 5.9.5.4)
5.5.5.2	CDR milestone report [DJF;CDR]
5.5.5.2	Software documentation verification report [DJF;CDR] (invocation of 5.9.5.4.)
5.5.5.2	Problem and nonconformance report [DJF;CDR] (invocation of 5.9.5.5)
6.2.8.2–a.	Schedulability analysis (update) [DJF;CDR]
6.2.8.2–b.	Technical budgets (update) [DJF;CDR]
6.2.8.2–c.	Software behaviour verification [DJF;CDR]
6.2.8.3	Testing feasibility report [DJF;CDR]
6.2.9.1–a.	Schedulability analysis (update) [DJF;CDR]
6.2.9.1–b.	Technical budgets (update) [DJF;CDR]
6.4.1.3	Software for intended re–use – evaluation of re–use potential [DJF;PDR,CDR]

Requirement	**DJF Contents at QR Milestone**
5.3.2.11	Customer’s approval of qualified state [DJF; QR]
5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]
5.6.2	Validation with respect to Requirements Baseline testing report [DJF; QR | AR]
5.5.5.1	Software validation with respect to the Requirements Baseline testing specification [DJF;QR | AR] (invocation of 5.9.5.1)
5.5.5.1	Operations manual (update) [DJF;QR | AR] (invocation of 5.9.5.3)
5.5.5.1	Software design and test evaluation report [DJF;QR | AR] (invocation of 5.9.5.4)
5.6.3.1–a.	Preliminary software acceptance data package [DJF; QR]
5.6.3.1–d.	Software design and test evaluation report [DJF; QR]
5.6.3.1–e.	Validation testing report [DJF; QR]
5.6.3.1–f.	Test specification evaluation [DJF; QR]
5.6.3.1–g.	QR milestone report [DJF; QR]
5.6.4.1–c.	Software acceptance data package [DJF;QR]

Requirement	**DJF contents at AR milestone**
5.3.2.12	Customer’s approval of accepted state [DJF; AR].

5.3.4.2	Margins and technical budgets status [DJF; PDR, CDR, QR, AR]
5.3.2.12	Customer ’s approval of accepted state [DJF; AR]
5.6.2	Validation with respect to Requirements Baseline testing report [DJF; QR | AR]
5.5.5.1	Software validation with respect to the Requirements Baseline testing specification [DJF;QR | AR] (invocation of 5.9.5.1)
5.5.5.1	Operations manual (update) [DJF;QR | AR] (invocation of 5.9.5.3)
5.5.5.1	Software design and test evaluation report [DJF;QR | AR] (invocation of 5.9.5.4)
5.6.3.2–a.	Final software acceptance data package [DJF; AR]
5.6.3.2–b.	Acceptance testing documentation [DJF; AR]
5.6.3.2–c.	AR milestone report [DJF; AR]
5.6.4.3	Installation plan [DJF; AR]
5.6.4.4	Installation report [DJF; AR]
5.6.5.1	Acceptance test plan [DJF; AR]
5.6.5.2	Acceptance test report [DJF; AR]
5.6.5.4	Acceptance testing documentation [DJF; AR]
5.6.5.5	Traceability of Acceptance tests to Requirements Baseline [DJF; AR]
6.2.10.1	Complement of validation at system level [DJF;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**
5.3.2.7–b.	Top-level architectural design [DDF; PDR]
5.3.2.8	Customer approval of technical specification and top-level architecture [TS, DDF, ICD(TS), DJF; PDR]
5.4.3.1	Software architectural design [DDF; PDR]
5.4.3.4–b.	Preliminary (top–level) internal interfaces design [DDF;PDR]
6.2.5.1–a.	Software static architecture [DDF;PDR]
6.2.5.1–b.	Selected analysable computational model [DDF;PDR]
6.2.5.1–b.	Software dynamic architecture [DDF;PDR]
6.2.5.1–b.	Software behaviour [DDF;PDR]

Requirement	**DDF contents at DDR**
6.2.3.1–a.	Customer approval of the design definition file (architectural design, detailed design) [DDF,DJF;DDR]
6.2.3.1–b.	Customer approval of the design system level interface and the system level integration plan [DDF,DJF;DDR]

6.2.3.1–c.	Customer approval of the margins and technical budget status [DJF;DDR]
6.2.3.1–d.	Customer approval of the updated technical specification [TS;DDR]
6.2.7.1–a.	Software static design [DDF;DDR|CDR]
6.2.7.1–b.	Software dynamic design [DDF;DDR|CDR]
6.2.7.1–c.	Software elements behaviour [DDF;DDR|CDR]
6.2.7.1–d.	Compatibility of design methods with the computational model [DDF;DDR|CDR]

Requirement	**DDF contents at CDR**
5.3.2.9–a.	Customer approval of the design definition file (e.g. architectural design and detailed design, code) [DDF; CDR]
5.3.2.9–c.	Customer approval of the design of system level interfaces and the system level integration plan [DDF, DJF; CDR]
5.5.2.1	Software components design documents [DDF; CDR]
5.5.2.2	Internal ICDs (update) [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]
6.2.7.1–a.	Software static design [DDF;DDR|CDR]
6.2.7.1–b.	Software dynamic design [DDF;DDR|CDR]
6.2.7.1–c.	Software elements behaviour [DDF;DDR|CDR]
6.2.7.1–d.	Compatibility of design methods with the computational model [DDF;DDR|CDR]

Requirement	**DDF contents at QR**
5.6.3.1–b.	Preliminary software release documentation [DDF;QR]
5.6.3.1–c.	Preliminarily software delivery on specified data medium [DDF; QR]
5.6.4.1–a.	Software delivery on specified data medium [DDF;QR]
5.6.3.1–b.	Software release documentation [DDF; QR]

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**
5.7.2.1	Operational plan – plan and standards [OP;ORR]
5.7.2.2	Operational plan – procedures for problem handling [OP; ORR]
5.7.2.3	Operational plan – operational testing specifications [OP; ORR]

Requirement	**MF contents**
5.8.2.1	Maintenance plan – plans and procedures [MF]
5.8.2.3	Maintenance plan – problem reporting and handling [MF]
5.8.3.4	Change justification file – problem analysis report [MF]
5.8.4.1	Modification identification [MF]
5.8.6	Change justification file – baseline for changes [MF]
5.8.7.2	Migration plan [MF]
5.8.7.3	Migration justification file [MF]
5.8.8.1	Retirement plan [MF]
5.8.8.2	Retirement notification to operator [MF]
6.2.11.1	Long term maintenance solutions [MF]

A.7 Contribution to management documentation

5.2.5.7	System level integration support products [MGT;SRR]
5.3.2.2	Project software development life cycle definition, included in the software project development plan [MGT;SRR]
5.9.2.3	Appropriate element of project requirements documents dealing with project organization [MGT]
5.9.3.3	Appropriate element of project requirements documents dealing with project organization [MGT]

(This page is intentionally left blank)

Annex B (informative)

References to other ECSS Standards

Referenced ECSS Standard	Page
ECSS–E–00	
ECSS–E–10	
ECSS–E–40–01	
ECSS–E–40–03	
ECSS–E–40–04	
ECSS–E–70	
ECSS–M	
ECSS–M–00	
ECSS–M–00–02	
ECSS–M–10	
ECSS–M–20	
ECSS–M–30	
ECSS–M–40	
ECSS–M–50	
ECSS–M–60	
ECSS–M–70	
ECSS–P–001	
ECSS–Q	
ECSS–Q–20	
ECSS–Q–30	
ECSS–Q–40	
ECSS–Q–80	

(This page is intentionally left blank)

Annex C (informative)

Tailoring Guidelines

C.1 Introduction

The ECSS family of standard is intended to be tailored for each individual project. The ECSS–E40 lists exhaustively the requirements that would make the project following all the best practices in space software engineering.

However, the resources (budget, schedule, manpower, etc) of any project are limited, and all the projects have to trade off the perfection of engineering against the available resources. The goal of the tailoring is to select, modify or add adequately requirements in order to reach the optimised ratio quality over resources. Any suppression of requirement is made actually on purpose, with full awareness of the resulting risk level increase.

C.1.1 How to tailor

The first step is therefore to understand the requested level of quality for the project, which starts with the characterisation of the project, the identification of the needed processes, and the characterisation of the product.

The ECSS–M–00–02 document (Space project management, Guidelines for selection and tailoring of ECSS standards) gives indication on the general way to tailor an ECSS standard, in particular the tailoring process and the tailoring templates. The templates are generic and deserve a more concrete description of the so–called programmatic and technical factors.

For space software, there are some examples where the full application of the E40 standard would be over killing, and the knowledge of some software project details (the tailoring factors) allows for some refinements. The main influencing factors, presented in section 4.5, are the following:

Technical factors:

- Novelty of the domain of application
- Complexity of the software and the system
- Criticality level
- Size of the software
- Reusability required of the software being developed
- Interface to system development projects

- Degree of use of COTS or existing software
- Maturity of the COTS and completeness or stability of the user requirements.

Operational factors:

- Type of application (platform, payload, experiment)
- Number of potential users of the software
- Criticality of the software as measured by the consequences of its failure
- Expected lifetime of the software
- Number of sites where the software is used
- Operation, maintenance, migration and retirement constraints

Management factors:

- Amount of time and effort required to develop the software
- Budget requirements for implementing and operating the software
- Accepted risk level for the project
- Type of lifecycle
- Schedule requirements for delivering the software
- Number of people required to develop, operate and maintain the software
- Complexity of the organisation
- Experience of the supplier

For each particular project additional factors may be used.

The tailoring can be made during a short discussion between the software engineering engineer and the software project manager. The software engineering engineer will first ask a set of questions to the project manager, in order to set–up the scope of the tailoring (i.e. the characteristics of the project that influence the selection or not of each requirements). Examples of questions are:

- Who are the Customer, the Supplier, the User, the Maintainer, and the Operator? Does the Customer intend to delegate some tasks to the Supplier?
- Where is the complexity of the project, in the requirements or in the design?
- What level of validation is necessary? Should the product be perfect at delivery, or is some room allowed for the User to participate to the tests, or is it a prototype that will be dropped later on (or reused in the next phase)?
- What level of verification is necessary? Is it necessary to verify the requirements, or the code, or the test definition, etc?
- What visibility into the design is wished? Does the project manager want to know everything on the detailed design and unit test, or does he trust the Supplier for a part of the lifecycle?
- Consequently, what are the necessary reviews to be selected into the project? Is it acceptable to merge some of them (as QR and AR, or SRR and PDR) or to waive others (as CDR or DDR)
- How much are COTS involved? Is the project an assembly of COTS products where the COTS acceptance and integration must have the emphasis?
- Is the software critical? Is it included into an hardware environment?
- How will be organised the maintenance? Is it included fully in the current contract, or is the maintenance limited to the guarantee period?

- How is produced the User Manual, is it the same as the Operation manual?

Then the requirements in section 5 and 6 will be reviewed and made or not applicable in a table.

The tailoring of E40 results in a short document including the project characteristics (as a justification for the tailoring) and the tailoring table.

Note that several E40 requirements remain mandatory for any project, e.g. the production of a minimum set of software requirements, a PDR to review them, and the production of the code.

The tables in annex propose tailoring templates for space software. They are to be considered as indication only, and each project will find its actual factors and will assess how they influence the requirements. They are available in two presentations. The first one follows the order of the requirements in E40, and could be more natural to follow during the tailoring process. The second table follows the M00–02 template order where the first column is the tailoring condition.

C.1.2 Who tailors?

The tailoring of the ECSS–E40 is implicitly a task of the Customer. When preparing the Invitation to Tender, the Customer may propose a tailored version of the standard as an indication of the level of software engineering that is required for the project. However, some tailoring factors (such as criticality, detailed design complexity) may only be known after the grant of the contract. The Supplier will also have to be part of the tailoring process and the resulting document will be baselined in the RB (at SRR). The Customer may also subcontract the tailoring to the Supplier, then review and accept the tailored version.

C.2 Tailoring templates

The following table discusses the E40 requirements considering their tailoring conditions and possibilities. The table is sorted in the order of the E40 requirements.

E40 Requirement	**Tailoring condition**	**Tailoring possibility**
5.2.2.1 system requirement specification	If the system is pure software.	The system requirement specification is reduced to user or customer requirements (the software requirement baseline).
5.2.2.1 system requirement specification	If the project is small.	The software requirement baseline may be merged with the software specification (software technical baseline).
5.2.2.2 criticality analysis	If the system is not critical.	It does not exist.
5.2.3.2 system partitioning	If the system is pure software.	It does not exist.
5.2.4.3 functional requirements for support to system and mission level validation	If the system is pure software.	It does not exist.
5.2.4.4 requirement baseline verification	If the complexity of the system is low.	The verification may be limited to the supplier reading and commenting.
5.2.4.5 SRR	If the Customer and the Supplier are the same organisational unit because the project is small enough.	The SRR may be waived in favour of the PDR.

5.2.5.1 software observability requirements	If the system is pure software.	It does not exist.
5.2.5.2 system level interface requirements	If the system is pure software.	It does not exist.
5.2.5.3 system level data interface	If the system is pure software.	It does not exist.
5.2.5.4 development constraints for system integration	If the system is pure software.	It does not exist.
5.2.5.5 system level integration support product	If the system is pure software.	It is limited to the workstation running the software.
5.2.5.6 system level integration preparation requirements	If the system is pure software.	It does not exist.
5.2.5.7 system level integration support requirements	If the system is pure software.	It does not exist.
5.2.6.1 software operations requirements		Depends on 5.7 tailoring
5.2.7 software maintenance requirements		Depends on 5.8 tailoring
5.3.2.2 life cycle definition		Mandatory
5.3.2.3 Review Plan | Milestones		Mandatory
5.3.2.4 Outputs from the milestones		Mandatory
5.3.2.5 elements of the maintenance plan	If the software is not maintained.	It does not exist.
5.3.2.6 SRR	If the Customer and the Supplier are the same organisational unit because the project is small enough.	The SRR may be waived in favour of the PDR.
5.3.2.7 technical specification	If the software is small enough and with a straightforward architecture (e.g. a set of services).	The specification and the top–level architecture may be merged.
5.3.2.8 PDR		Mandatory
5.3.2.9 CDR	If the project is small enough and if the design is not complex.	The CDR may be waived.
5.3.2.10 V&V	Tailoed according to 5.9 If the criticality is low and the budget is low.	The verification may be selective or waived.
5.3.2.10 V&V	If the software is a prototype.	The validation may be less intensive or waived (validation by the users) because integration is more intensive
5.3.2.11 QR	If both Verification and Validation have been waived.	QR is meaningless. Otherwise, QR is mandatory.

5.3.2.12 AR	If the Customer and the Supplier are the same organisational unit because the project is small enough.	The AR may be merged with the QR.
5.3.2.12 AR	If the V&V environment is the same as the operational environment.	The AR may be merged with the QR.
5.3.3 system level interface management procedures	If the system is only software.	They do not exist.
5.3.3 system level interface management procedures	If the system interface has no possibility to evolve	They may be waived.
5.3.3 system level interface management procedures	If the budget is low and the complexity is low.	They may be waived.
5.3.4.1 Technical budget and margin philosophy	If the technical budget is unlimited (memory size, computer throughput, response time).	It may be waived.
5.3.4.2 Technical budget and margin status at each milestone	If the technical budget is unlimited (memory size, computer throughput, response time).	It may be waived.
5.4.2.1 software requirements		Mandatory. Outputs are tailored as appropriate for the project needs.
5.4.2.2 software logical model	If the complexity of the system is low.	It may be waived.
5.4.2.3 requirement identification	If both Verification and Validation have been waived for all the versions of the software.	Requirement identification is not useful and it may be waived.
5.4.2.4 requirement verification	If the budget is low and the criticality is low.	It may be waived.
5.4.3.1 top–level architecture	If the software is a set of relatively independent services.	The requirement and the top–level architecture may be merged.
5.4.3.1 top–level architecture	If the design is a tree, but with a low complexity.	The top–level architecture and the detailed design may be merged.
5.4.3.2 design minimum requirements: hierarchy, dependency and interfaces		Mandatory
5.4.2.3.3 Design documentation		Mandatory
5.4.3.4 top–level interface design	If the software is a set of relatively independent services.	The requirement and the top–level interface design can be merged.
5.4.3.4 top–level interface design	If the interfaces complexity is low.	The interface top–level architecture and detailed design may be merged.

5.4.3.5 operation manual	If the software is a set of relatively independent services.	The top–level interface design and the operation manual may be merged.
5.4.3.6 integration test plan	If the software is a set of relatively independent services, or if there is little or no interface.	The plan may be waived.
5.4.3.6 integration test plan	If the complexity of the interface is low.	The plan may be delayed to 5.5.2.5.
5.4.3.7 top–level architecture verification	If the budget is low and the criticality is low.	The verification may be waived.
5.4.3.8 PDR		Mandatory
5.4.4.1 software verification plan – software validation plan	Tailored as per 5.3.2.10 V&V tailoring. If the budget is low and the criticality is low.	The verification plan may be waived.
5.4.4.1 software verification plan – software validation plan	If the software is a prototype.	The validation may be less intensive or waived (validation by the users) because integration is more intensive
5.5.2.1 components detailed design	If the design is a tree, but with a low complexity.	The top–level architecture and the detailed design can be merged.
5.5.2.2 interface detailed design	If the interfaces complexity is low.	The interface top–level architecture and detailed design can be merged.
5.5.2.3 operation manual (update)	If the budget is low.	The update may be delayed up to 5.5.4.3
5.5.2.4 software unit test plan	If the budget is low or if the criticality is low.	The plan may be waived.
5.5.2.5 software integration test plan (update)	If the software is a set of relatively independent services.	The plan may be waived.
5.5.2.6 design verification	If the budget is low and the criticality is low.	The verification may be waived.
5.5.3.1 unit code and software unit test plan		Mandatory.
5.5.3.2 software unit test	If the budget is low and the criticality is low.	The unit test reports may remain not documented.
5.5.3.2 software unit test	If the stubs needed to unit test a component are equivalent to the real components.	The unit test of this component may be replaced by the integration test of this component and its related components.
5.5.3.3 operation manual (update)	If the budget is low.	The update may be delayed up to 5.5.4.3
5.5.3.4 software integration test plan (update)	If the software is a set of relatively independent services	The plan may be waived.
5.5.3.5 code and unit test results verification	If the budget is low and the criticality is low	The verification may be waived.

5.5.4.1 integration test plan (update) and test code	If the software is a set of relatively independent services or there is little or no interface	The integration may be simplified or waived.
5.5.4.2 software integration test	If the budget is low and the criticality is low	The integration test reports may remain not documented.
5.5.4.3 operation manual (update)		Mandatory if there is no validation. May be delayed to 5.9.5.3 otherwise
5.5.4.4 software integration and operation manual verification	If the budget is low and the criticality is low	The verification may be waived.
5.5.5.1 validation against the TS	If the system is pure software,. If the project is small	It may be waived when TS=RB
5.5.5.2 CDR	If the budget is low and the criticality is low	The CDR may be waived.
5.6.2 validation against the RB		Mandatory. Either at QR or AR.
5.6.3.1 QR		See 5.3.2.11
5.6.3.2 AR		See 5.3.2.12
5.6.4.1 preparation of software delivery	If the budget is low	The delivery preparation may be simplified.
5.6.4.2 training material		Only if appropriate
5.6.4.3 installation plan		Only when installation is required.
5.6.4.4 installation report		Only when installation is required.
5.6.5.1 acceptance test plan		Mandatory.
5.6.5.2 acceptance test report		Mandatory.
5.6.5.3 generation of executable code for acceptance	If no automatic code is generated	It may be waived
5.6.5.4 acceptance testing documentation		Only in the terms tailored.
5.6.5.5 traceability acceptance tests–requirements baseline	If the budget is low and the criticality is low.	It may be waived.
5.7.2 operation process implementation	If the operation of the software is of low complexity.	The operation process implementation may be waived.
5.7.3 operational testing	If the operation of the software is of low complexity.	The operational testing may be waived in favor of the Acceptance Testing.
5.7.4 system operation		Mandatory
5.7.5 user support		Tailored on a case by case basis
5.8.2 maintenance process implementation	If the software is not maintained.	The maintenance process implementation may be waived.

5.8.2 maintenance process implementation	If the complexity is low and the organisation simple.	The maintenance process implementation may be waived.
5.8.3 problem and modification analysis	If the software is not maintained.	The implementation may be waived. Otherwise it is mandatory, in a more or less formal way.
5.8.4 modification implementation	If the software is not maintained.	The implementation may be waived. Otherwise it is mandatory, in a more or less formal way.
5.8.5 in–flight modifications		Only for space segment.
5.8.6 maintenance review acceptance	If the organisation is simple.	The acceptance may be waived.
5.8.7 software migration	If migration is not necessary.	The process may be waived. Otherwise, it is mandatory in a more or less formal way.
5.8.8 software retirement	If the software is not retired independently from the system.	The process may be waived. Otherwise, it is mandatory in a more or less formal way.
5.9.2 Verification process implementation	If the budget is low and the criticality is low. Note that part of the verification process implementation is the tailoring process.	The verification may be waived.
5.9.2.1 verification effort determination		Only necessary if verification is needed
5.9.2.2 verification process establishment		Only necessary if verification is needed
5.9.2.3 verification organisation		Only necessary if independent verification is needed
5.9.2.4 activities and products to verify		Only necessary if verification is needed
5.9.3.1 validation effort determination		Only necessary if validation is needed
5.9.3.2 validation process establishment		Only necessary if validation is needed
5.9.3.3 validation organisation		Only necessary if an independent validation is needed.
5.9.3.4 validation plan		Only necessary if validation is needed.
5.9.4 Verification process		Generic process, see each instance.
5.9.5.1 validation test specification	If the budget is low and the criticality is low.	The validation may be limited to a given coverage target, e.g. by validating only the nominal behavior.
5.9.5.2 validation tests	If the budget is low.	The validation report may remain non–documented.
5.9.5.3 operation manual (update)		Mandatory.

5.9.5.4 validation verification	If the budget is low and the criticality is low.	The verification may be waived.
5.9.5.5 problem and non–conformance handling		Mandatory, in a more or less formal way.
5.9.5.6 test readiness review	If the budget is low and the criticality is low.	The verification may be waived.
5.9.6.1 support to reviews		As per review tailoring
5.9.6.2 reviews definition	If the lifecycle is of type concurrent engineering or with short and numerous incremental steps	The reviews will be defined as a set of periodic meetings whose last one only includes the formal review success. The authorisation to start the next phase disappears.

The next table is sorted considering the tailoring condition.

Tailoring condition	**E40 Requirement**	**Tailoring possibility**
No condition	5.2.6.1 software operations requirements	Depends on 5.7 tailoring.
	5.2.7 software maintenance requirements	Depends on 5.8 tailoring.
	5.3.2.2 life cycle definition	Mandatory.
	5.3.2.3 Review Plan | Milestones	Mandatory.
	5.3.2.4 Outputs from the milestones	Mandatory.
	5.3.2.8 PDR	Mandatory.
	5.4.2.1 software requirements	Mandatory. Outputs are tailored as appropriate for the project needs.
	5.4.2.3.3 Design documentation	Mandatory.
	5.4.3.2 design minimum requirements: hierarchy, dependency and interfaces	Mandatory.
	5.4.3.8 PDR	Mandatory.
	5.5.3.1 unit code and software unit test plan	Mandatory.
	5.5.4.3 operation manual (update)	Mandatory if there is no validation. May be delayed to 5.9.5.3 otherwise.
	5.6.2 validation against the RB	Mandatory. Either at QR or AR.
	5.6.3.1 QR	See 5.3.2.11
	5.6.3.2 AR	See 5.3.2.12

	5.6.4.2 training material	Only if appropriate
	5.6.4.3 installation plan	Only when installation is required.
	5.6.4.4 installation report	Only when installation is required.
	5.6.5.1 acceptance test plan	Mandatory.
	5.6.5.2 acceptance test report	Mandatory.
	5.6.5.4 acceptance testing documentation	Only in the terms tailored.
	5.7.4 system operation	Mandatory.
	5.7.5 user support	Tailored on a case by case basis.
	5.8.5 in–flight modifications	Only for space segment.
	5.9.2.1 verification effort determination	Only necessary if verification is needed
	5.9.2.4 activities and products to verify	Only necessary if verification is needed
	5.9.3.1 validation effort determination	Only necessary if validation is needed
	5.9.3.2 validation process establishment	Only necessary if validation is needed
	5.9.3.3 validation organisation	Only necessary if an independent validation is needed.
	5.9.3.4 validation plan	Only necessary if validation is needed.
	5.9.4 Verification process	Generic process, see each instance.
	5.9.5.3 operation manual (update)	Mandatory.
	5.9.5.5 problem and non–conformance handling	Mandatory, in a more or less formal way.
	5.9.6.1 support to reviews	As per review tailoring
If both Verification and Validation have been waived	5.3.2.11 QR	QR is meaningless. Otherwise, QR is mandatory.
If both Verification and Validation have been waived for all the versions of the software	5.4.2.3 requirement identification	Requirement identification is not useful and it may be waived.
If migration is not necessary	5.8.7 software migration	The process may be waived. Otherwise, it is mandatory in a more or less formal way.
If no automatic code is generated	5.6.5.3 generation of executable code for acceptance	It may be waived

If the budget is low	5.5.2.3 operation manual (update)	The update may be delayed up to 5.5.4.3
	5.5.3.3 operation manual (update)	The update may be delayed up to 5.5.4.3
	5.6.4.1 preparation of software delivery	The delivery preparation may be simplified.
If the budget is low and the complexity is low	5.3.3 system level interface management procedures	They may be waived.
If the budget is low and the criticality is low	5.4.2.4 requirement verification	It may be waived.
	5.4.3.7 top–level architecture verification	The verification may be waived.
	5.5.2.6 design verification	The verification may be waived.
	5.5.3.2 software unit test	The unit test reports may remain not documented.
	5.5.3.5 code and unit test results verification	The verification may be waived.
	5.5.4.2 software integration test	The integration test reports may remain not documented.
	5.5.4.4 software integration and operation manual verification	The verification may be waived.
	5.5.5.2 CDR	The CDR may be waived.
	5.6.5.5 traceability acceptance tests–requirements baseline	It may be waived.
	5.9.5.1 validation test specification	The validation may be limited to a given coverage target, e.g. by validating only the nominal behavior.
	5.9.5.4 validation verification	The verification may be waived.
	5.9.2 Verification process implementation	The verification may be waived.
	5.5.2.4 software unit test plan	The plan may be waived.
If the complexity is low and the organisation simple	5.8.2 maintenance process implementation	The maintenance process implementation may be waived.
If the complexity of the interface is low	5.4.3.4 top–level interface design	The interface top–level architecture and detailed design may be merged.
	5.4.3.6 integration test plan	The plan may be delayed to 5.5.2.5.
	5.5.2.2 interface detailed design	The interface top–level architecture and detailed design can be merged.
If the complexity of the system is low	5.2.4.4 requirement baseline verification	The verification may be limited to the supplier reading and commenting.
	5.4.2.2 software logical model	It may be waived.

If the Customer and the Supplier are the same organisational unit because the project is small enough	5.2.4.5 SRR	The SRR may be waived in favour of the PDR.
	5.3.2.6 SRR	The SRR may be waived in favour of the PDR.
	5.3.2.12 AR	The AR may be merged with the QR.
If the design is a tree, but with a low complexity	5.4.3.1 top–level architecture	The top–level architecture and the detailed design may be merged.
	5.5.2.1 components detailed design	The top–level architecture and the detailed design can be merged.
If the lifecycle is of type concurrent engineering or with short and numerous incremental steps	5.9.6.2 reviews definition	The reviews will be defined as a set of periodic meetings whose last one only includes the formal review success. The authorisation to start the next phase disappears.
If the operation of the software is of low complexity	5.7.2 operation process implementation	The operation process implementation may be waived.
	5.7.3 operational testing	The operational testing may be waived in favor of the Acceptance Testing.
If the organisation is simple	5.8.6 maintenance review acceptance	The acceptance may be waived.
If the project is small	5.2.2.1 system requirement specification	The software requirement baseline may be merged with the software specification (software technical baseline).
	5.5.5.1 validation against the TS	It may be waived when TS=RB.
If the project is small enough and if the design is not complex	5.3.2.9 CDR	The CDR may be waived.
If the software is a prototype	5.3.2.10 V&V	The validation may be less intensive or waived (validation by the users) because integration is more intensive
	5.4.4.1 software verification plan – software validation plan	The validation may be less intensive or waived (validation by the users) because integration is more intensive

If the software is a set of relatively independent services	5.4.3.1 top–level architecture	The requirement and the top–level architecture may be merged.
	5.4.3.4 top–level interface design	The requirement and the top–level interface design can be merged.
	5.4.3.5 operation manual	The top–level interface design and the operation manual may be merged.
	5.5.2.5 software integration test plan (update)	The plan may be waived.
	5.5.3.4 software integration test plan (update)	The plan may be waived.
If the software is a set of relatively independent services or there is little or no interface	5.5.4.1 integration test plan (update) and test code	The integration may be simplified or waived.
	5.4.3.6 integration test plan	The plan may be waived.
If the software is not maintained	5.3.2.5 elements of the maintenance plan	It does not exist.
	5.8.2 maintenance process implementation	The maintenance process implementation may be waived.
	5.8.3 problem and modification analysis	The analysis may be waived. Otherwise it is mandatory, in a more or less formal way.
	5.8.4 modification implementation	The implementation may be waived. Otherwise it is mandatory, in a more or less formal way.
If the software is not retired independently from the system	5.8.8 software retirement	The process may be waived. Otherwise, it is mandatory in a more or less formal way.
If the software is small enough and with a straightforward architecture (e.g. a set of services)	5.3.2.7 technical specification	The specification and the top–level architecture may be merged.
If the stubs needed to unit test a component are equivalent to the real components	5.5.3.2 software unit test	The unit test of this component may be replaced by the integration test of this component and its related components.
If the system interface has no possibility to evolve	5.3.3 system level interface management procedures	They may be waived.
If the system is not critical	5.2.2.2 criticality analysis	It does not exist.

If the system is pure software	5.3.3 system level interface management procedures	They do not exist.
	5.2.2.1 system requirement specification	The system requirement specification is reduced to user or customer requirements (the software requirement baseline).
	5.2.3.2 system partitioning	It does not exist.
	5.2.4.2 system V&V process requirements	It does not exist.
	5.2.4.3 functional requirements for support to system and mission level validation	It does not exist.
	5.2.5.1 software observability requirements	It does not exist.
	5.2.5.2 system level interface requirements	It does not exist.
	5.2.5.3 system level data interface	It does not exist.
	5.2.5.4 development constraints for system integration	It does not exist.
	5.2.5.5 system level integration support product	It is limited to the workstation running the software.
	5.2.5.6 system level integration preparation requirements	It does not exist.
	5.2.5.7 system level integration support requirements	It does not exist.
	5.2.5.7 system level integration support requirements	It does not exist.
If the technical budget is unlimited (memory size, computer throughput, response time)	5.3.4.1 Technical budget and margin philosophy	It may be waived.
	5.3.4.2 Technical budget and margin status at each milestone	It may be waived.
If the V&V environment is the same as the operational environment	5.3.2.12 AR	The AR may be merged with the QR.
Tailored according to 5.9. If the criticality is low and the budget is low	5.3.2.10 V&V	The verification may be selective or waived.
Tailored as per 5.3.2.10 V&V tailoring. If the budget is low and the criticality is low	5.4.4.1 software verification plan – software validation plan	The verification plan may be waived.