

BS EN 16603-40:2014



BSI Standards Publication

Space engineering

Part 40: Software

bsi.

...making excellence a habit.
AVA Documents™
Document Sharing Hub

National foreword

This British Standard is the UK implementation of EN 16603-40:2014. It supersedes BS EN 14160:2002 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee ACE/68, Space systems and operations.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014. Published by BSI Standards Limited 2014

ISBN 978 0 580 83979 5

ICS 49.140

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2014.

Amendments issued since publication

Date	Text affected
------	---------------

EUROPEAN STANDARD
NORME EUROPÉENNE
EUROPÄISCHE NORM

EN 16603-40

August 2014

ICS 49.140

Supersedes EN 14160:2001

English version

Space engineering - Part 40: Software

Ingénierie spatiale - Partie 40: Logiciel

Raumfahrttechnik - Teil 40: Software

This European Standard was approved by CEN on 10 February 2014.

CEN and CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN and CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN and CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN and CENELEC members are the national standards bodies and national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



**CEN-CENELEC Management Centre:
Avenue Marnix 17, B-1000 Brussels**

Table of contents

Foreword	7
Introduction.....	8
1 Scope.....	9
2 Normative references	10
3 Terms, definitions and abbreviated terms.....	11
3.1 Terms for other standards.....	11
3.2 Terms specific to the present standard	11
3.3 Abbreviated terms.....	17
4 Space system software product assurance principles.....	19
4.1 Introduction.....	19
4.2 Overview of space system software engineering processes	20
4.2.1 General	20
4.2.2 Software related system requirements process.....	23
4.2.3 Software management process.....	23
4.2.4 Software requirements and architecture engineering process	24
4.2.5 Software design and implementation engineering process.....	24
4.2.6 Software validation process	25
4.2.7 Software delivery and acceptance process	25
4.2.8 Software verification process	25
4.2.9 Software operation process.....	26
4.2.10 Software maintenance process	27
4.3 Organization of this Standard	27
4.4 Tailoring of this Standard	29
5 Requirements.....	30
5.1 Introduction.....	30
5.2 Software related system requirement process	31
5.2.1 Overview	31
5.2.2 Software related system requirements analysis	31
5.2.3 Software related system verification.....	32

5.2.4	Software related system integration and control	33
5.2.5	System requirements review	34
5.3	Software management process	34
5.3.1	Overview	34
5.3.2	Software life cycle management.....	35
5.3.3	Joint review process.....	36
5.3.4	Software project reviews description	38
5.3.5	Software technical reviews description.....	39
5.3.6	Review phasing.....	40
5.3.7	Interface management	40
5.3.8	Technical budget and margin management.....	41
5.3.9	Compliance to this Standard	42
5.4	Software requirements and architecture engineering process.....	42
5.4.1	Overview	42
5.4.2	Software requirements analysis	43
5.4.3	Software architectural design	44
5.4.4	Conducting a preliminary design review	46
5.5	Software design and implementation engineering process	46
5.5.1	Overview	46
5.5.2	Design of software items.....	46
5.5.3	Coding and testing	49
5.5.4	Integration	50
5.6	Software validation process	50
5.6.1	Overview	50
5.6.2	Validation process implementation.....	51
5.6.3	Validation activities with respect to the technical specification.....	51
5.6.4	Validation activities with respect to the requirements baseline	53
5.7	Software delivery and acceptance process	54
5.7.1	Overview	54
5.7.2	Software delivery and installation	54
5.7.3	Software acceptance.....	55
5.8	Software verification process	56
5.8.1	Overview	56
5.8.2	Verification process implementation.....	56
5.8.3	Verification activities.....	57
5.9	Software operation process	65
5.9.1	Overview	65

5.9.2	Process implementation	65
5.9.3	Operational testing	66
5.9.4	Software operation support	66
5.9.5	User support	67
5.10	Software maintenance process	68
5.10.1	Overview	68
5.10.2	Process implementation	68
5.10.3	Problem and modification analysis	69
5.10.4	Modification implementation	70
5.10.5	Conducting maintenance reviews	70
5.10.6	Software migration	71
5.10.7	Software retirement	72
Annex A	(informative) Software documentation	74
Annex B	(normative) Software system specification (SSS) - DRD	80
Annex C	(normative) Software interface requirements document (IRD) - DRD	88
Annex D	(normative) Software requirements specification (SRS) - DRD	91
Annex E	(normative) Interface Control Document (ICD) - DRD	98
Annex F	(normative) Software design document (SDD) - DRD	102
Annex G	(normative) Software release document (SReID) - DRD	112
Annex H	(normative) Software User Manual (SUM) - DRD	114
Annex I	(normative) Software verification plan (SVerP) - DRD	119
Annex J	(normative) Software validation plan (SVaIP) - DRD	124
Annex K	(normative) Software [unit/integration] test plan (SUITP) - DRD	129
Annex L	(normative) Software validation specification (SVS) - DRD	137
Annex M	(normative) Software verification report (SVR) - DRD	144
Annex N	(normative) Software reuse file (SRF) - DRD	151
Annex O	(normative) Software development plan (SDP) - DRD	155
Annex P	(normative) Software review plan (SRevP) - DRD	161
Annex Q	(informative) Document organization and contents at each milestones	170

Annex R (normative) Tailoring of this Standard based on software criticality	188
Annex S (informative) General Tailoring	199
Bibliography	203

Figures

Figure 4-1: Software related processes in ECSS Standards	20
Figure 4-2: Overview of the software life cycle process	22
Figure 4-3: Structure of this Standard	28
Figure A-1 : Overview of software documents	74

Tables

Table A-1 : ECSS-E-ST-40 and ECSS-Q-ST-80 Document requirements list (DRL)	75
Table B-1 : SSS traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	80
Table C-1 : IRD traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	88
Table D-1 : SRS traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	91
Table E-1 : ICD traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	98
Table F-1 : SDD traceability to ECSS-E-ST-40 Part 1 and ECSS-Q-ST-80 clauses	102
Table G-1 : SReID traceability to ECSS-E-ST-40 and ECSS-QST--80 clauses	112
Table H-1 : SUM traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	114
Table I-1 : SVerP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	119
Table J-1 : SValP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	124
Table K-1 : SUITP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	129
Table L-1 : SVS traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	137
Table M-1 : SVR traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	144
Table N-1 : SRF traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	151
Table O-1 : SDP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	155
Table P-1 : SRevP traceability to ECSS-E-ST-40 and ECSS-Q-ST-80 clauses	161
Table Q-1 : Documents content at milestone SRR	170
Table Q-2 : Documents content at milestone PDR/SWRR	173
Table Q-3 : Documents content at milestone PDR (in addition to PDR/SWRR)	174
Table Q-4 : Documents content at milestone TRR	177
Table Q-5 : Documents content at milestone TRB	177
Table Q-6 : Documents content at milestone CDR/DDR	177
Table Q-7 : Documents content at milestone CDR (in addition to CRD/DDR)	179
Table Q-8 : Documents content at milestone QR	181
Table Q-9 : Documents content at milestone AR	183

Table Q-10 : Documents content at milestone ORR	185
Table Q-11 : Documents content of documents with no explicit review.....	186
Table R-1 : Criticality applicability	189

Foreword

This document (EN 16603-40:2014) has been prepared by Technical Committee CEN/CLC/TC 5 "Space", the secretariat of which is held by DIN.

This standard (EN 16603-40:2014) originates from ECSS-E-ST-40C.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by February 2015, and conflicting national standards shall be withdrawn at the latest by February 2015.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

This document supersedes EN 14160:2001.

This document has been prepared under a mandate given to CEN by the European Commission and the European Free Trade Association.

This document has been developed to cover specifically space systems and has therefore precedence over any EN covering the same scope but with a wider domain of applicability (e.g. : aerospace).

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Introduction

This Standard defines the principles and requirements applicable to space software engineering. ECSS-Q-ST-80 defines the principles and requirements applicable to space software product assurance.

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

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 the extent defined by the tailoring process, 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, transfer, operations and maintenance.

It defines the scope of the space software engineering processes 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 processes.

This Standard reflects the specific methods used in space system developments, and the requirements for the software engineering processes 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 responses and to implement the work.

This Standard is not intended to replace textbook material on computer science or technology, and such material is 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”. This Standard also applies to the development of non-deliverable software that affects the quality of the deliverable product.

This Standard may be tailored for the specific characteristics and constraints of a space project in conformance with ECSS-S-ST-00.

2

Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECSS Standard. For dated references, subsequent amendments to, or revision of any of these publications do not apply. However, parties to agreements based on this ECSS Standard are encouraged to investigate the possibility of applying the more recent editions of the normative documents indicated below. For undated references, the latest edition of the publication referred to applies.

EN reference	Reference in text	Title
EN 16601-00-01	ECSS-S-ST-00-01	ECSS system – Glossary of terms
EN 16603-10-11	ECSS-E-ST-10-11	Space product assurance – Human factors engineering
EN 16601-10	ECSS-M-ST-10	Space project management – Project planning and implementation
EN 16601-10-01	ECSS-M-ST-10-01	Space project management – Organization and conduct of reviews
EN 16601-40	ECSS-M-ST-40	Space project management – Configuration and information management
EN 16602-80	ECSS-Q-ST-80	Space product assurance – Software product assurance

3

Terms, definitions and abbreviated terms

3.1 Terms for other standards

For the purpose of this Standard, the terms and definitions from ECSS-ST-00-01, in particular for the following terms:

acceptance test

software product

NOTE The terms and definitions are common for the ECSS-E-ST-40 and ECSS-Q-ST-80 Standards.

3.2 Terms specific to the present standard

3.2.1 automatic code generation

generation of source code with a tool from a model

3.2.2 code coverage

percentage of the software that has been executed (covered) by the test suite

3.2.3 competent assessor

person who has demonstrated the necessary skills, competencies and experience to lead a process assessment in conformance with ISO/IEC 15504

NOTE Adapted from ISO/IEC 15504:1998, Part 9.

3.2.4 condition

boolean expression not containing boolean operators

3.2.5 configurable code

code (source code or executable code) that can be tailored by setting values of parameters

NOTE This definition covers in particular classes of configurable code obtained by the following configuration means:

- configuration based on the use of a compilation directive;

- configuration based on the use of a link directive;
- configuration performed through a parameter defined in a configuration file;
- configuration performed through data defined in a database with impact on the actually executable parts of the software (e.g. parameters defining branch structures that result in the non-execution of existing parts of the code).

3.2.6 COTS, OTS, MOTS software

for the purpose of this Standard, commercial-off-the-shelf, off-the-shelf and modified-off-the-shelf software for which evidence of use is available

3.2.7 critical software

software of criticality category A, B or C

NOTE See ECSS-Q-ST-80 Table D-1 – Software criticality categories.

3.2.8 deactivated code

code that, although incorporated through correct design and coding, is intended to execute in certain software product configurations only, or in none of them

[adapted from RTCA/DO-178B]

3.2.9 decision

boolean expression composed of conditions and zero or more boolean operators that are used in a control construct.

NOTE 1 For example: “if.....then else” or the “case” statement are control construct.

NOTE 2 A decision without a boolean operator is a condition.

NOTE 3 If a condition appears more than once in a decision, each occurrence is a distinct condition.

3.2.10 decision coverage

measure of the part of the program within which every point of entry and exit is invoked at least once and every decision has taken “true” and “false” values at least once.

NOTE Decision coverage includes, by definition, statement coverage.

3.2.11 existing software

any software developed outside the business agreement to which this Standard is applicable, including software from previous developments provided by the

supplier, software from previous developments provided by the customer, COTS, OTS and MOTS software, freeware and open source software

3.2.12 integration testing

testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them

[IEEE 610.12:1990]

3.2.13 logical model

implementation-independent model of software items used to analyse and document software requirements

3.2.14 margin philosophy

rationale for margins allocated to the performance parameters and computer resources of a development, and the way to manage these margins during the execution of the project

3.2.15 metric

defined measurement method and the measurement scale

NOTE 1 Metrics can be internal or external, and direct or indirect.

NOTE 2 Metrics include methods for categorising qualitative data.

[ISO/IEC 9126-1:2001]

3.2.16 migration

porting of a software product to a new environment

3.2.17 mission products

products and services delivered by the space system

NOTE For example: Communications services, science data.

3.2.18 modified condition and decision coverage

measure of the part of the program within which every point of entry and exit has been invoked at least once, every decision in the program has taken “true” and “false” values at least once, and each condition in a decision has been shown to independently affect that decision’s outcome

NOTE A condition is shown to independently affect a decision’s outcome by varying that condition while holding fixed all other possible conditions.

3.2.19 operational

for the purpose of this Standard, related to the software operation

NOTE It is not related to the spacecraft operation.

3.2.20 portability (a quality characteristic)

capability of software to be transferred from one environment to another

3.2.21 quality characteristics (software)

set of attributes of a software product by which its quality is described and evaluated

NOTE A software quality characteristic can have multiple levels of sub-characteristics.

3.2.22 quality model (software)

set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality

[ISO/IEC 9126-1:2001]

3.2.23 real-time

pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process

[IEEE 610.12:1990]

3.2.24 regression testing (software)

selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements

[IEEE 610.12:1990]

3.2.25 reusability

degree to which a software unit or other work product can be used in more than one computer program or software system

[IEEE 610.12:1990]

3.2.26 singular input

input corresponding to a singularity of the function

3.2.27 software

see “software product” in ECSS-S-ST-00-01

3.2.28 software component

part of a software system

NOTE 1 Software component is used as a general term.

NOTE 2 Components can be assembled and decomposed to form new components. In the production activities, components are implemented as units, tasks or programs, any of which can 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 this Standard, components are not always “basic” as they can be decomposed.

3.2.29 software intensive system

space system in which the dominant part of the constituents are software elements

NOTE In such systems, subsystems consist mainly of software. For this type of system, the majority of interfaces are software-software interfaces.

3.2.30 software item

see “software product” in ECSS-S-ST-00-01

3.2.31 software observability

property of a system for which the values of status variables can be determined throughout observations of the output variables

3.2.32 software problem

condition of a software product that causes difficulty or uncertainty in the use of the software

[CMU/SEI-92-TR-022]

3.2.33 software product assurance

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, conforms to customer requirements

3.2.34 software unit

separately compilable piece of source code

NOTE In this Standard no distinction is made between a software unit and a database; both are covered by the same requirements.

3.2.35 statement coverage

measure of the part of the program within which every executable source code statement has been invoked at least once.

3.2.36 stress test

test that evaluates a system or software component at or beyond its required capabilities

3.2.37 test case

set of test inputs, execution conditions and expected results developed for a particular objective such as to exercise a particular program path or to verify compliance with a specified requirement

3.2.38 test design

documentation specifying the details of the test approach for a software feature or combination of software features and identifying associated tests

3.2.39 test procedure

detailed instructions for the set up, operation and evaluation of the results for a given test

3.2.40 test script

file containing a set of commands or instructions written in native format (computer or tool processable) in order to automate the execution of one or a combination of test procedures (and the associated evaluation of the results)

3.2.41 unit test

test of individual software unit

3.2.42 unreachable code

code that cannot be executed due to design or coding error

3.2.43 usability (a quality characteristic)

capability of the software to be understood, learned, used and liked by the user, when used under specified conditions

3.2.44 validation

<software> process to confirm that the requirements baseline functions and performances are correctly and completely implemented in the final product

3.2.45 verification

<software> process to confirm 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

3.2.46 walk-through

static analysis technique in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems

[IEEE Std 1028-1997]

3.3 Abbreviated terms

For the purpose of this Standard and of ECSS-Q-ST-80, the abbreviated terms from ECSS-S-ST-00-01 and the following apply:

For the definition of DRD acronyms see Annex A.

NOTE The abbreviated terms are common for the ECSS-E-ST-40 and ECSS-Q-ST-80 Standards.

Abbreviation	Meaning
AR	acceptance review NOTE The term SW-AR can be used for clarity to denote ARs that solely involve software products.
CDR	critical design review NOTE The term SW-CDR can be used for clarity to denote CDRs that solely involve software products.
CMMI	capability maturity model integration
COTS	commercial-off-the-shelf
CPU	central processing unit
DDF	design definition file
DDR	detailed design review
DJF	design justification file
DRD	document requirements definition
ECSS	European Cooperation for Space Standardization
eo	expected output
GS	ground segment
HMI	human machine interface
HSIA	hardware-software interaction analysis
HW	hardware
ICD	interface control document
INTRSA	international registration scheme for assessors
IRD	interface requirements document
ISO	International Organization for Standardization
ISV	independent software validation
ISVV	independent software verification and validation
MGT	management file
MF	maintenance file
MOTS	modified off-the-shelf
OBCP	on-board control procedure

OP	operational plan
ORR	operational readiness review
OTS	off-the-shelf
PAF	product assurance file
PDR	preliminary design review NOTE The term SW-PDR can be used for clarity to denote PDRs that solely involve software products.
PRR	preliminary requirement review
QR	qualification review NOTE The term SW-QR can be used for clarity to denote QRs that solely involve software products.
RB	requirements baseline
SCAMPI	standard CMMI appraisal method for process improvement
SDE	software development environment
SOS	software operation support
SPA	software product assurance
SPAMR	software product assurance milestone report
SPAP	software product assurance plan
SPR	software problem report
SRB	software review board
SRR	system requirements review NOTE The term SW-SRR can be used for clarity to denote SRRs that solely involve software products.
SW	software
SWE	software engineering
TRR	test readiness review
TS	technical specification

4

Space system software product assurance principles

4.1 Introduction

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

The context of space software engineering is the overall space system engineering process. This Standard focuses on space software engineering processes requirements and their expected outputs. This clause 4 introduces the software engineering processes in their context, and the structure of the Standard.

Software is found at all levels, ranging from system functions down to firmware, including safety and mission critical functions. Therefore, a special emphasis is put in this Standard on the system-software relationship and on the verification and validation of software items.

This Standard is complemented by ECSS-Q-ST-80 Space product assurance — Software product assurance, which specifies the product assurance aspects and is the entry point for ECSS-E-ST-40 into the Q-series of standards. Requirements for space configuration and information management are in ECSS-M-ST-40, which includes the software DRD for software configuration file. Together, these standards either define or refer to the definition of all software relevant processes for space projects. ECSS-Q-ST-20 is the reference, through ECSS-Q-ST-80, for the software acquisition process, and the software management process tailors ECSS-M-ST-10 for software.

Figure 4-1 presents the different software related processes implemented by ECSS-E-ST-40, ECSS-Q-ST-80 and other ECSS Standards.

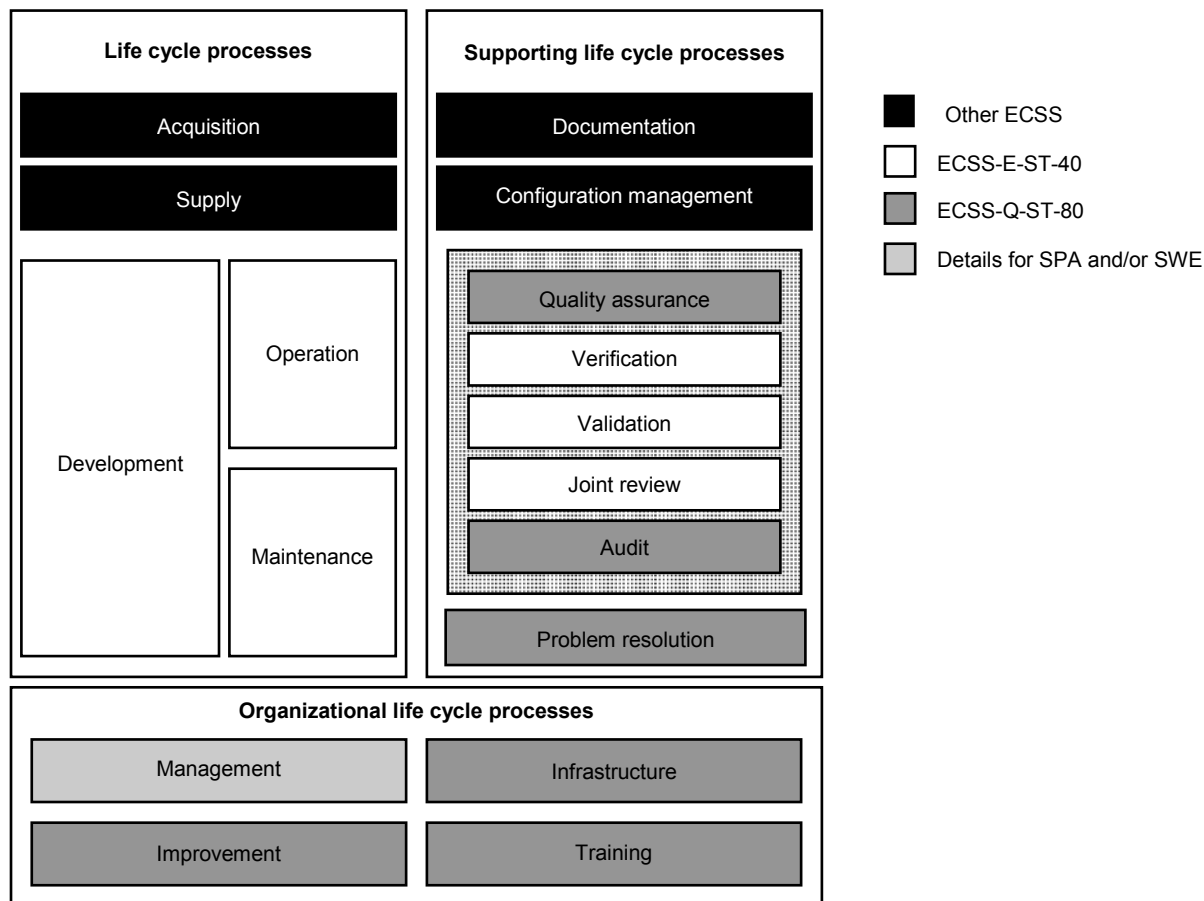


Figure 4-1: Software related processes in ECSS Standards

4.2 Overview of space system software engineering processes

4.2.1 General

In accordance with the ECSS theoretical concept, a fundamental principle of this Standard is the 'customer-supplier' relationship, assumed for all software developments. The project organization is defined in ECSS-M-ST-10. The customer is, in the general case, the procurer of two associated products: the hardware and the software for a system, subsystem, set, equipment or assembly. The concept of the "customer-supplier" relationship is applied recursively, i.e. the customer can be a supplier to a higher level in the space system. The software customer therefore has two important interfaces:

- the software customer interfaces with his software and hardware suppliers in order to adequately allocate to them functional and performance requirements, through the functional analysis.
- the software customer assumes a supplier role to interface in turn with his customer at the next higher level, ensuring that 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 can 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 specified by this process, and they provide the starting point for the software engineering.

Reviews are the main interaction points between the customer and the supplier. They also synchronize software engineering processes. The reviews relevant to the software engineering processes are the SRR, PDR, CDR, QR, AR, and ORR as defined by ECSS-M-ST-10. This Standard offers in addition the possibility to anticipate the PDR in a SWRR, and the CDR into a DDR. The SWRR and DDR are executed as a first part of the ECSS-M-ST-10 reviews, but precede them.

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.

The notion of engineering processes is fundamental to this Standard. The processes provide the means to describe the overall constraints and interfaces to the engineering processes at system level. At the same time, they provide the possibility to the supplier to implement the individual activities and tasks implied by the processes in accordance with a selected software life cycle. This Standard is a process model, and does not prescribe a particular software life cycle. Although the spacecraft reviews suggest a waterfall model, the software development plan can implement any life cycle, in particular by the use of the technical reviews, less formal than the project reviews.

When the software development is included in a complete space project, the software engineering processes have also a relationship with the project phases (0, A, B, C, D, E, F) and in particular with the system level reviews (e.g. the system-software activities are executed in phase B). For other software developments which do not rely on a particular system, or which are developed for reuse, not applicable requirements can be tailored.

A software development plan defines and instantiates in details the particular implementation of this standard in a project.

Figure 4-2 identifies the main concurrent software engineering processes and shows how and when they are synchronized by the customer/supplier reviews.

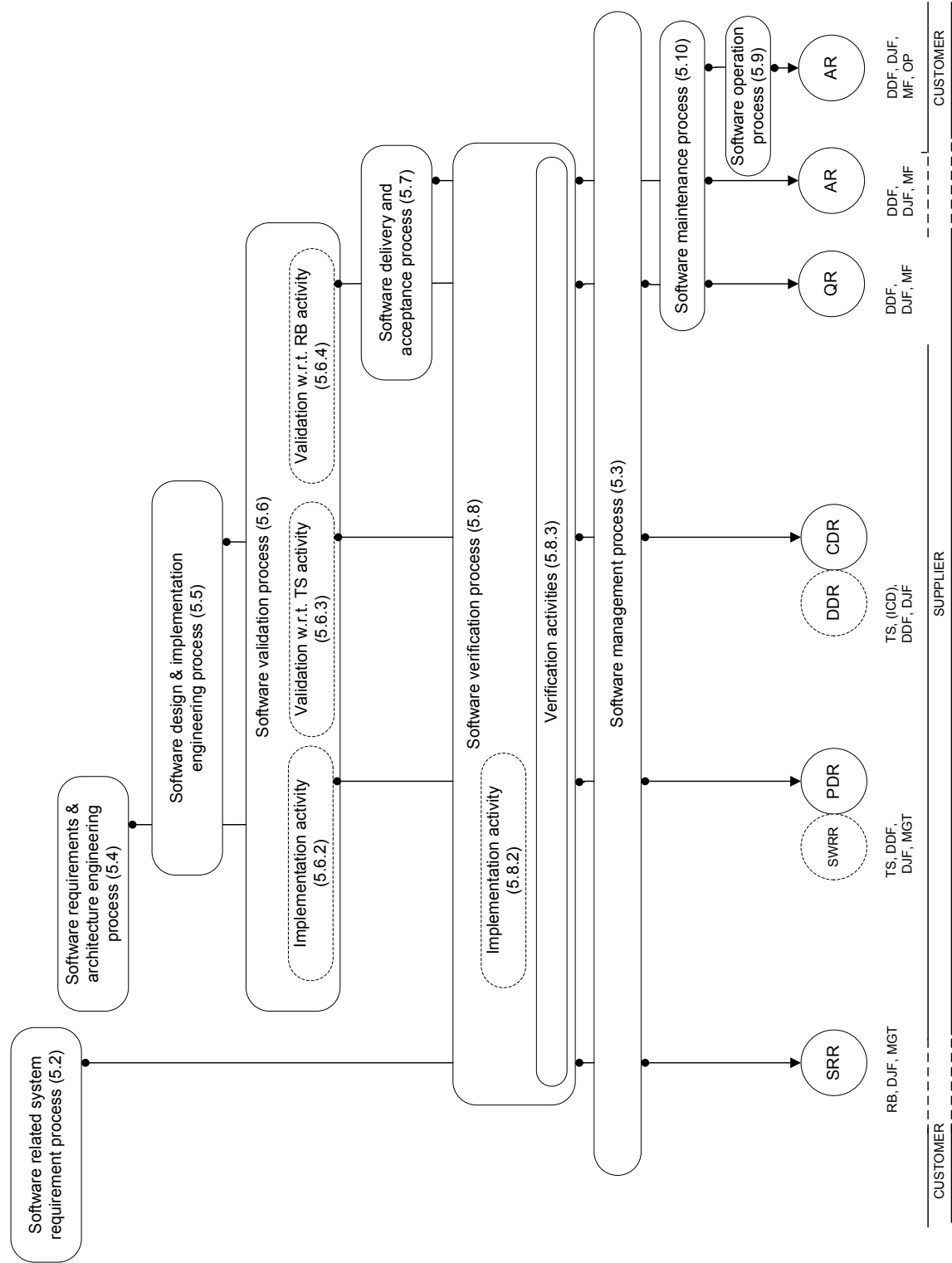


Figure 4-2: Overview of the software life cycle process

4.2.2 Software related system requirements process

The software related system requirement process produces the information for input to the system requirements review (SRR). This establishes the functional and the performance requirements baseline (including the interface requirement specification) (RB) of the software development. It constitutes the link between the space system processes, as defined in ECSS-E-ST-10 and ECSS-E-ST-70, and the software processes.

According to the recursive customer-supplier model of ECSS, at each level, the customer is responsible for the delivery of a system in which the developed software is integrated. According to the recursive system-subsystem model of ECSS-E-ST-10, he is responsible for the specification of the system requirements at lower level and, in particular, for any software comprised in the system. As such, the tasks of this process are normally performed by the customer, but can be delegated to the supplier under the responsibility of the customer.

ECSS-E-ST-10 describes the following system engineering functions:

- requirement engineering, producing the system and lower level functional and technical specifications,
- system analysis, consolidating requirements through trade-off and various analysis (mission, requirement, functional, physical, performance), producing in particular the functional and physical architecture,
- system design (producing the physical architecture) and system configuration (producing the physical system with the software),
- system verification as being conformant to requirements, and
- system engineering integration and control, including in particular system integration, its relation with production, operation, product assurance and management, and the engineering database, the interface management and the technical budget management

The system engineering functions giving input to the software are the requirement engineering, the system verification and the system engineering integration and control, and clause 5.2 is organised along this line. For software, the system analysis and design only support the system requirement engineering, the system configuration includes the software development.

According to ECSS-E-ST-10, the system engineering produces “lower level element functional specification” in phase A before the PRR, and “technical specification” of lower level elements in the preliminary system design file, baselined at SDR (system design review) in phase B. For software, this document is the requirements baseline addressed in this clause, and the software SRR is synchronized with the system SDR.

4.2.3 Software management process

This process covers the complete life cycle of the software project.

The M-branch of the ECSS standards defines how to manage the software project. This process tailors the M standards for software-specific issues, in

particular ECSS-M-ST-10. In addition, the software product assurance requirements specified in ECSS-Q-ST-80 are used for the control of space systems software projects. These requirements are not repeated here.

The main activity is the production of a software development plan including the life cycle description, activities description, milestones and outputs, the techniques to be used, and the risks identification.

The management process also includes the organization and handling of all the joint reviews, the definition of procedures for the management of the system interface, and the technical budget and margin management.

The reviews are defined and positioned in the life cycle in clause 5.3. They are called in their relevant process in clause 5.2, 5.4, 5.5, 5.6, and 5.7. The data package for each review is specified in Annex A.

Software requirements or detailed design dedicated reviews (i.e. SWRR and DDR) are defined as anticipation of the PDR and CDR respectively.

4.2.4 Software requirements and architecture engineering process

The software requirements and architecture engineering process consists of

- the elaboration of the technical specification, including the preliminary definition of the ICD (TS), which is the supplier's response to the requirements baseline including the interface requirement specification,
- the architectural design documented in a preliminary SDD

It is the duty of the supplier to involve all stakeholders in the requirement elicitation.

During this process, the result of all significant trade-offs, feasibility analyses, make-or-buy decisions and supporting technical assessments are documented in a design justification file (DJF).

The software requirements and architecture engineering process is completed by the preliminary design review (PDR).

4.2.5 Software design and implementation engineering process

One of the outputs of this process is the detailed design of the software items identified in the software product tree (see ECSS-M-ST-10). It is provided in response to the technical specification, the ICD and the preliminary DDF. All elements of the software design are 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 show 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).

As part of this process, the code, unit testing and integration testing of the software product are also produced. All elements of the testing activities are documented in the design justification file (DJF).

4.2.6 Software validation process

In this standard, the word software validation refers to software product testing against both the technical specification and the requirements baseline.

This process is intended to confirm that the technical specification and the requirements baseline functions and performances are correctly and completely implemented in the final product.

The result of this process is included in the DJF.

This process is established before the PDR to basically produce a software validation plan ("process implementation"). The plan includes the validation activity with respect to the technical specification (which is held before the CDR) and the validation activity with respect to the requirements baseline (which is held before the QR and possibly repeated at AR).

This process can include a test readiness review (TRR) to verify that all test facilities and test cases and procedures are available before each significant test campaign, and under configuration control.

The results relevant to the validation against the TS are checked at the CDR, and those against the RB are verified at the QR, using the DJF as input.

This process can be executed with varying degrees of independence. The degree of independence can range from the same person, or a different person in the same organization, to a person in a different organization, with varying degrees of separation. Where the software validation process and the software verification process 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.

4.2.7 Software delivery and acceptance process

This process prepares the software product for delivery and testing in its operational environment. It is completed with an acceptance review (AR), with the DJF as input. The acceptance review is a formal event in which the software product is evaluated in its operational environment. It is carried out after the software product is transferred to the customer and installed on an operational basis.

4.2.8 Software verification process

The software verification process is intended to confirm that adequate specifications and inputs exist for every activity and that the outputs of the activities are correct and consistent with the specifications and inputs.

This process is concurrent with all the previous processes.

The customer verifies the requirement baseline for the SRR.

For the supplier, this process is established before the PDR to basically produce a software verification plan ("process implementation"). This process includes all the verification activities of all the expected outputs of the development activities

The result of this process is included in the DJF and reported at each reviews.

This process can be executed with varying degrees of independence. The degree of independence can range from the same person, or a different person in the same organization, to a person in a different organization, with varying degrees of separation. Where the software verification process and the software validation process are executed by an organization independent of the supplier, it is called Independent Software Verification and Validation (ISVV).

4.2.9 Software operation process

The operation process can 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 are determined by the overall system requirements and applied to the software products. The operation process is not directly connected to the overall mission phase E, but is instead determined by the requirement at system level to operate the software product at a given time.

Indeed, software operation in this Standard is totally different from spacecraft or payload operations performed by ground stations on space systems. Software operation in this Standard includes mainly the helpdesk and the link between the users, the developers or maintainers, and the customer.

The organisational entity responsible for the software operation support (SOS entity) ensures that the software remains operational for the users. Examples of SOS entities and users for different types of software are typically the following:

- for a space platform flight software:
 - the user is the spacecraft operation team when he uses the on-board software for commanding the spacecraft and receiving telemetry;
 - the SOS entity is the spacecraft operation team when he patches, dump or reboot the flight software.
- for manned mission flight software such as microgravity experiment software:
 - the user is the astronaut who manipulates the experiment through the laptop in space;
 - the SOS entity is either the astronaut or the ground centre, depending on who acts on the software for installations, reboot, patching, etc.
- for ground segment software:
 - the user are the spacecraft and ground segment operations team who uses the ground segment software to operate the spacecraft;

- the SOS entity is the ground segment maintenance team who performs the installation and the deployment of the ground software.
- for payload data ground segment software:
 - the user is the operation team who uses the ground segment software to produce, quality control and disseminate data products;
 - the SOS entity is the ground segment maintenance team who performs the installation and the deployment of the ground software.
- for a software tool e.g. a distributed database with a human machine interface:
 - the user uses the software through the human machine interface;
 - the SOS entity administrates the database by supporting user's requests, rebooting or backing up the system.

4.2.10 Software maintenance process

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 maintenance process contains the activities and tasks of the maintainer. The objective is to modify an existing software product while preserving its integrity. This process includes the migration and retirement of the software product. The process ends with the retirement of the software product.

The activities provided in clause 5.10 are specific to the maintenance process; however, the process can utilize other processes in this Standard. If the software engineering processes are utilized, the term supplier there is interpreted as maintainer.

The maintainer manages the maintenance process at the project level following the management process, which is instantiated for software in this process.

Both the documents and the reviews identified by the clauses in 5.10 are part of the general maintenance activities for the space systems, and the requirements for these reviews and documentation are part of the space system maintenance engineering requirements, covered in other ECSS Standards. The provisions of clause 5.10 produce the required software engineering inputs for this system level activities.

4.3 Organization of this Standard

This Standard contains one normative clause (clause 5) which is organized with respect to the software processes and activities breakdown described in Figure 4-3. Each process includes activities which are themselves decomposed into a list of one single or several tasks in the shape of process requirements (clauses), producing expected outputs.

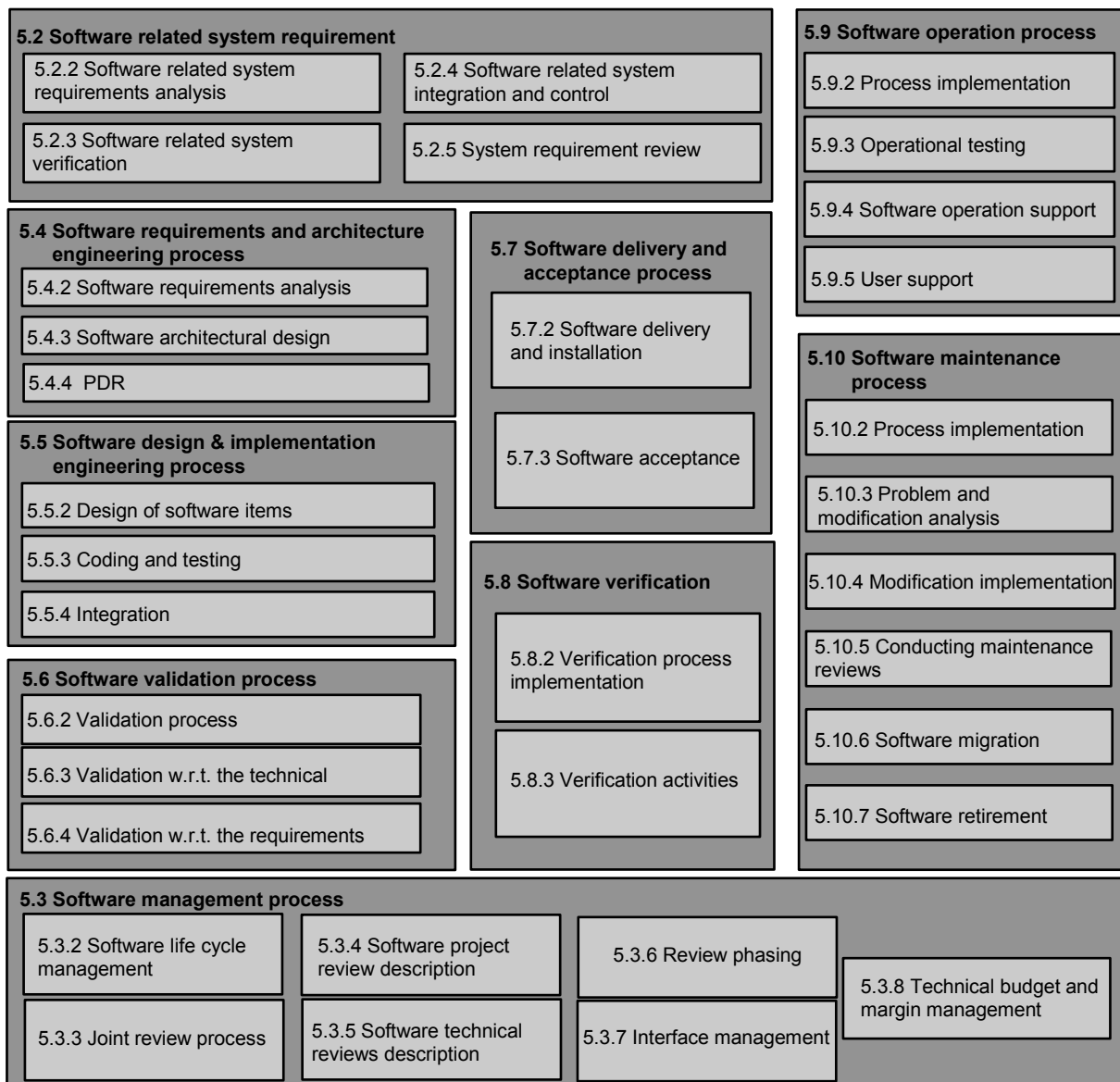


Figure 4-3: Structure of this Standard

The software documentation produced by all the expected output of this Standards is summarized in Annex A.

Annex B to Annex P specify the DRD (the document requirements) allowing to organize most of the expected output into documents.

Annex Q shows, for each reviews, the documents (and the expected outputs without DRDs) that are delivered at the review, as well as the trace to the process requirements.

Annex R is a pre-tailoring of this Standard according to the criticality levels as defined in the ECSS-Q-ST-80.

Annex S gives other tailoring guidelines.

The Bibliography gives reference to other standards and documents.

4.4 Tailoring of this Standard

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

There are several drivers for tailoring, such as dependability and safety aspects, software development constraints, product quality objectives and business objectives.

Tailoring for dependability and safety aspects is based on the selection of requirements related to the verification, validation and levels of proof demanded by the criticality of the software. Annex R contains a tailoring of this Standard based on software criticality.

Tailoring for software development constraints takes into account the special characteristics of the software being developed, and of the development environment. The type of software development (e.g. database or real-time) and the target system (e.g. embedded processor, host system, programmable device, or application specific integrated circuits) is also taken into account (see Annex S of this Standard). Specific requirements for verification, review and inspection are imposed, for example, when full validation on the target computer is not feasible or where performance goals are difficult to achieve.

Tailoring for product quality and business objectives is done by selecting requirements on quality of the products as explained in clause 7 of ECSS-Q-ST-80. In this process, the customer specifies the quality objectives for the product.

5 Requirements

5.1 Introduction

This clause 5 defines the requirements for engineering software for space systems, applicable to any space projects producing computer software.

Each requirement can be identified by a hierarchical number, made of the four digits of the clause (e.g. 1.2.3.4), followed by a letter (e.g. “a” or “b”) if the clause needs several requirements.

The text of the requirement is followed, where necessary, by further explanation of the aim (normative) or of a note (informative).

For each requirement, the associated output is given in the “expected output” section. When several outputs are expected, they are identified by a letter in italic (e.g. “a” or “b”). With each output, the destination file of the output is indicated in brackets, together with the corresponding document DRD (after a comma) and review (after a semicolon)

Example 1:

5.5.3.2 Software unit testing

- a. The supplier shall develop and document the test procedures and data for testing each software unit.

EXPECTED OUTPUT: The following outputs are expected:

- a. Software component design document and code (update) [DDF, SDD, source code; CDR];
- b. Software unit test plan (update) [DJF, SUITP; CDR].

denote the output *a* of the requirement 5.5.3.2a. contained in the SDD and the source code documents, part of the design definition file, requested for the CDR review, and the output *b* of the requirement 5.5.3.2a. contained in the SUITP document, part of the design justification file, requested for the CDR review.

Example 2:

5.10.5.2 Baseline for change

- a. Upon successful completion of the reviews, a baseline for the change shall be established.

EXPECTED OUTPUT: Baseline for changes [MF, - ; -]

denote the output of requirement 5.10.5.2a. contained in the maintenance file, while no DRD nor reviews are specified.

The list of DRDs is given in Annex A.

Some requirements depend on the nature of the software. This is explicitly mentioned in the requirements. They include for example flight or ground software, presence or not of a HMI, independence or not of the V&V, real-time or not.

5.2 Software related system requirement process

5.2.1 Overview

The software related system requirement process consists of the following activities:

- software related system requirements analysis;
- software related system verification;
- software related system integration and control.

5.2.2 Software related system requirements analysis

5.2.2.1 Specification of system requirements allocated to software

- a. The customer shall derive system requirements allocated to software from an analysis of the specific intended use of the system, and from the results of the safety and dependability analysis.

EXPECTED OUTPUT: The following outputs are expected:

- a. Functions and performance system requirements allocated to software [RB, SSS; SRR];
- b. Verification and validation product requirements [RB, SSS; SRR];
- c. Software operations requirements [RB, SSS; SRR];
- d. Software maintenance requirements [RB, SSS; SRR];

- e. Requirements for in flight modification capabilities [RB, SSS; SRR];
- f. Requirements for real-time [RB, SSS; SRR];
- g. Requirements for security [RB, SSS, SRR].
- h. Quality requirements [RB, SSS, SRR]

5.2.2.2 Identification of observability requirements

- a. The customer shall specify all software observability requirements to monitor the software behaviour and to facilitate the system integration and failure investigation.

EXPECTED OUTPUT: System and software observability requirements [RB, SSS; SRR].

5.2.2.3 Specification of HMI requirements

- a. The customer shall specify HMI requirements, following the human factor engineering process specified in ECSS-E-ST-10-11.

EXPECTED OUTPUT: HMI requirements [RB, SSS; SRR].

5.2.3 Software related system verification

5.2.3.1 Verification and validation process requirements

- a. The customer shall specify the requirements needed for planning and setting up the system verification and validation process related to software.

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

5.2.3.2 System input for software validation

- a. The customer shall specify requirements for the validation of the software against the requirements baseline and technical specification, in particular mission representative data and scenarios, and operational procedures to be used.

EXPECTED OUTPUT: Validation requirements and scenario [RB, SSS; SRR].

5.2.3.3 System input for software installation and acceptance

- a. The customer shall specify requirements for the installation and acceptance of the software.

EXPECTED OUTPUT: Installation and acceptance requirements at the operational and maintenance sites [RB, SSS; SRR].

5.2.4 Software related system integration and control

5.2.4.1 Identification of software versions for software integration into the system

- a. The customer shall identify the software versions to be delivered and associate each requirement of the requirements baseline to a version.

EXPECTED OUTPUT: Association of requirements to versions [RB, SSS; SRR].

- b. The customer shall specify the content and media of the delivery.

EXPECTED OUTPUT: Delivery content and media [RB, SSS; SRR].

5.2.4.2 Supplier support to system integration

- a. The customer shall specify the support to be provided by the software supplier in order to integrate the software at system level.

NOTE For example: Raining, maintenance, configuration and test support.

EXPECTED OUTPUT: System level integration support requirements [RB, SSS; SRR].

5.2.4.3 Interface requirement specification

- a. The customer shall specify the external interfaces of the software, including the static and dynamic aspects, for nominal and degraded modes.

EXPECTED OUTPUT: External interface requirements specification [RB, IRD; SRR].

5.2.4.4 System database

- a. The customer shall specify the content of the system database for the supplier in order to ensure the consistency of common data and to define the allowed operational range of the data.

EXPECTED OUTPUT: System database content and allowed operational range [RB, SSS; SRR].

5.2.4.5 Development constraints

- a. The customer shall define specific development and design constraints on the supplier, including the use of development standards.

EXPECTED OUTPUT: Design and development constraints [RB, SSS; SRR].

5.2.4.6 On board control procedures

- a. The customer shall specify the requirements to be implemented by OBCP.

NOTE See ECSS-E-ST-70-01.

EXPECTED OUTPUT: OBCP requirements [RB, SSS; SRR].

5.2.4.7 Development of software to be reused

- a. The customer shall specify the reusability requirements that apply to the development, to enable the future reuse of the software (including models used to generate the software), or customization for mission (e.g. in a family of spacecraft or launcher).

EXPECTED OUTPUT: Requirements for 'software to be reused' [RB, SSS; SRR].

5.2.4.8 Software safety and dependability requirements

- a. The customer shall specify the software safety and dependability requirements in accordance with ECSS-Q-ST-80 clauses 5.4.4, 6.2.2 and 6.2.3, based on the results of the safety and dependability analysis performed at system level.

EXPECTED OUTPUT: Software safety and dependability requirements [RB, SSS; SRR].

5.2.4.9 Format and data medium

- a. The customer shall specify the format and the delivery medium of the exchanged data, in particular the interface and the system database.

EXPECTED OUTPUT: Format and delivery medium of exchanged data [RB, SSS; SRR].

5.2.5 System requirements review

- a. The customer shall conduct a system requirements review (SRR) in accordance with 5.3.4.1a.

5.3 Software management process

5.3.1 Overview

The management and control tasks specified in this clause are:

- software life cycle management;
- joint review process, software project reviews description, software technical reviews description, and reviews phasing;
- interface management;

- technical budget and margin management;
- compliance to this Standard.

5.3.2 Software life cycle management

5.3.2.1 Software life cycle identification

- a. The software supplier shall define and follow a software life cycle including phases, their inputs and outputs, and joint reviews, in accordance with the overall project constraints and with ECSS-M-ST-10.

EXPECTED OUTPUT: Software life cycle definition [MGT, SDP; SRR, PDR].

- b. The life cycle shall be chosen, assessing the specifics of the project technical approaches and the relevant project risks.

EXPECTED OUTPUT: Software life cycle definition [MGT, SDP; SRR, PDR].

- c. The software supplier shall define the development strategy, the software engineering standards and techniques, the software development and the software testing environment.

EXPECTED OUTPUT: Development strategy, standards, techniques, development and testing environment [MGT, SDP; PDR].

- d. The output of each phase and their status of completion, submitted as input to joint reviews, shall be specified in the software life cycle definition, including documents in complete or outline versions, and the results of verification of the outputs of the phase.

EXPECTED OUTPUT: Software life cycle definition [MGT, SDP; SRR, PDR].

5.3.2.2 Identification of interfaces between development and maintenance

- a. The interfaces between development and maintenance (e.g. documents to be handed over, tools to be kept for maintenance) shall be identified in the software life cycle.

EXPECTED OUTPUT: Identification of interface between development and maintenance [MGT, SDP; PDR].

5.3.2.3 Software procurement process implementation

- a. The supplier shall document and implement the software procurement process as specified in ECSS-Q-ST-80 clause 5.5.

EXPECTED OUTPUT: Software procurement process documentation and implementation [MGT, SDP; SRR, PDR].

5.3.2.4 Automatic code generation

- a. The autocode input models shall be reviewed together with the rest of the software specification, architecture and design.

NOTE The autocode input models are integral part of the software specification, architecture and design.

EXPECTED OUTPUT: Autocode input model review [MGT, SDP; SRR, PDR].

- b. In the case of coexisting autocode and manually written parts, the software development plan shall include the definition of a clear interface definition and resource allocation (memory, CPU) at PDR.

EXPECTED OUTPUT: Autocode interface definition and resource allocation [MGT, SDP; SRR, PDR].

- c. The input model management, the code generation process and supporting tools shall be documented in the SDP.

EXPECTED OUTPUT: Automatic code generation development process and tools [MGT, SDP; SRR, PDR].

- d. The supplier shall define in the SDP the verification and validation strategy for automatic code generation as a result of the trade off between the qualification of the code generation toolchain and the end to end validation strategy of the software item, or any combination thereof, in relation with ECSS-Q-ST-80 clause 6.2.8.

EXPECTED OUTPUT: Automatic code generation verification and validation strategy [MGT, SDP; SRR, PDR].

- e. The configuration management of the automatic code generation related elements shall be defined in the SCMP.

EXPECTED OUTPUT: Automatic code generation configuration management [MGT, SCMP; SRR, PDR].

5.3.2.5 Changes to baselines

- a. Changes to baselines shall be handled by the configuration management process described in clause 6.2.4 of ECSS-Q-ST-80.

EXPECTED OUTPUT: Changes to baselines procedures [MGT, SCMP; SRR, PDR].

5.3.3 Joint review process

5.3.3.1 Joint reviews

- a. Joint reviews shall be held to evaluate the progress and outputs of a project process or activity and provide evidence that:
1. the output are complete;
 2. the output conforms to applicable standards and specifications;

3. any changes are properly implemented and impact only those areas identified by the configuration management process;
4. the output conforms to applicable schedules;
5. the output are in such a status that the next activity can start;
6. the activity is being conducted according to the plans, schedules, standards, and guidelines laid down for the project.

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

EXPECTED OUTPUT: Joint review reports [DJF, - ; SRR, PDR, CDR, QR, AR].

5.3.3.2 Software project reviews

- a. Software project reviews (i.e. joint reviews organized under the responsibility of the customer aiming at defining a customer approved technical baseline) shall be included in the software life cycle, with as a minimum SRR, PDR, CDR, QR and AR as specified in 5.3.4.

EXPECTED OUTPUT: Software project reviews included in the software life cycle definition [MGT, SDP; SRR, PDR].

- b. The review process specified in ECSS-M-ST-10-01 shall apply to all software project reviews, including the agreement on a review plan before the review process is started.

EXPECTED OUTPUT: Review Plan [MGT, SRevP; SRR, PDR].

5.3.3.3 Software technical reviews

- a. In addition to the software project reviews, software technical reviews (i.e. joint reviews organized under the responsibility of the customer or the supplier aiming at defining a technical baseline) shall be defined.

EXPECTED OUTPUT: Software technical reviews included in the software life cycle definition [MGT, SDP; SRR, PDR].

- b. The applicable technical review process shall be specified by the supplier.

EXPECTED OUTPUT: Technical reviews process [MGT; SDP; SRR, PDR].

- c. The supplier shall use the software technical reviews to implement intermediate reviews, in particular for incremental life cycle.

AIM: — this enables to cope with any alternative life cycle not necessarily waterfall.

- in the case of incremental life cycle in particular, this enables to hold formal reviews on the last increments, and to have those technical reviews in anticipation for each of the increment.

EXPECTED OUTPUT: Software technical reviews included in the software life cycle definition [MGT, SDP; SRR, PDR].

5.3.4 Software project reviews description

5.3.4.1 System requirement review

- a. After completion of the software requirements baseline specification, a system requirements review (SRR) shall take place.

AIM: Reach the approval of the software requirements baseline by all stakeholders.

EXPECTED OUTPUT: Approved requirements baseline [RB; SRR].

5.3.4.2 Preliminary design review

- a. After completion of the software requirement analysis and architectural design, and the verification and validation processes implementation, a preliminary design review (PDR) shall take place.

AIM: To review compliance of the technical specification (TS) with the requirements baseline, to review the software architecture and interfaces, to review the development, verification and validation plans.

EXPECTED OUTPUT: Approved technical specification and interface, architecture and plans [TS, DDF, DJF, MGT; PDR].

- b. In case the software requirements are baselined before the start of the architectural design, the part of the PDR addressing the software requirements specification and the interfaces specification shall be held in a separate joint review anticipating the PDR, in a software requirements review (SWRR).

AIM: e.g. in case of software intensive system or when an early baseline of the requirements is required.

EXPECTED OUTPUT: Approved technical specification and interface [TS; PDR].

5.3.4.3 Critical design review

- a. After completion of the design of software items, coding and testing, integration and validation with respect to the technical specification, a critical design review (CDR) shall take place.

AIM: —To review the design definition file, including software architectural design, detailed design, code and users manual;

- To review the design justification file, including the completeness of the software unit testing, integration and validation with respect to the technical specification.

EXPECTED OUTPUT: Approved design definition file and design justification file [DDF, DJF; CDR].

- b. In case the software detailed design is baselined before the start of the coding, the part of the CDR addressing the software detailed design, the interfaces design and the software budget shall be held in a separate joint review anticipating the CDR, in a detailed design review (DDR).

EXPECTED OUTPUT: Approved detailed design, interface design and budget [DDF, DJF; CDR].

5.3.4.4 Qualification review

- a. After completion of the software validation against the requirements baseline, and the verification activities, a qualification review (QR) shall take place.

EXPECTED OUTPUT: Qualified software product [RB, TS, DDF, DJF, MGT, MF; QR].

5.3.4.5 Acceptance review

- a. After completion of the software delivery and installation, and software acceptance, an acceptance review (AR) shall take place.

AIM: To accept the software product in the intended operational environment.

EXPECTED OUTPUT: Accepted software product [RB, TS, DDF, DJF, MGT, MF; AR].

5.3.5 Software technical reviews description

5.3.5.1 Test readiness reviews

- a. Test readiness reviews (TRR) shall be held before the beginning of test activities, as defined in the software development plan.

EXPECTED OUTPUT: Confirmation of readiness of test activities [DJF; TRR]

5.3.5.2 Test review board

- a. The test review board (TRB) shall approve test results at the end of test activities, as defined in the software development plan.

EXPECTED OUTPUT: Approved test results [DJF; TRB]

5.3.6 Review phasing

5.3.6.1 Review phasing for flight software

- a. For flight software, the phasing of the software life cycle to the system life cycle shall be chosen, assessing the following driving aspects:
 - 1. the system model philosophy (e.g. proto-flight model, versus utilization of engineering qualification model)
 - 2. the system verification and qualification approach and constraints
 - 3. the capability to baseline the system design at system CDR, by knowing enough information about software design, in particular consolidated sizing and timing budgets, consistent hardware design and software design.

EXPECTED OUTPUT: Flight software review phasing [MGT, SDP;SRR, PDR]

- b. For flight software the following software versus system level reviews synchronisation shall be planned as follows:
 - 1. the software SRR not later than the system PDR
 - 2. the software PDR between the system PDR and the system CDR
 - 3. the detailed design of the software reviewed before the system CDR e.g. in a DDR
 - 4. the software CDR before the system QR
 - 5. the software QR within system QR

EXPECTED OUTPUT: Flight software review phasing [MGT, SDP;SRR, PDR]

5.3.6.2 Review phasing for ground software

- a. For ground segment software, the software life cycle shall be chosen assessing the following constraints for the ground reviews phasing:
 - 1. the baseline of the software requirements (e.g. SWRR) is performed before the ground segment PDR ,
 - 2. the software PDR is performed before the ground segment CDR
 - 3. all the other software reviews are performed before the ground segment QR.

EXPECTED OUTPUT: Ground software review phasing [MGT, SDP;SRR, PDR]

5.3.7 Interface management

5.3.7.1 Interface management procedures

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

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

EXPECTED OUTPUT: Interface management procedures [MGT, - ; SRR];

5.3.8 Technical budget and margin management

5.3.8.1 Software technical budget and margin philosophy definition

- a. Technical budget targets and margin philosophy dedicated to the software shall be specified by the customer in the requirements baseline in order to define the limits of software budgets associated with computer and network resources (such as: CPU load, maximum memory size, deadline fulfilment, communication, archiving needs, remote access needs) and performance requirements (such as data throughput).

AIM: This allows anticipating:

- Expected changes in the requirements baseline
- Requirements on reprogramming of the system during operational use
- Required budget for temporary copies of software images
- Constraints on state transitions, especially when recovery from a faulty state is concerned
- Constraints on physical CPU type and memory (e.g. driven by radiation levels) and expected processor capacity, wait states, interfaces, caching and pipelining, etc
- Equipment, communication and performances aspects (e.g. buses, protocols, acceptable errors, bus capacity usage by other sources, etc)
- Accuracy aspects, such as conversion to/from analogue signals, and accuracy of timing signals
- Budgets for OS kernel characterisation, such as context switch latency or deadlines for tasking
- Mission and system operation characteristics and reference operational scenarios

NOTE 1 The following CPU load margin reference values are often considered: 50 % at PDR, 35 % at DDR or TRR, 25 % at CDR, QR or AR.

NOTE 2 The following memory margin reference values in RAM or EEPROM are often considered: 50 % at PDR, 40 % at DDR or TRR, 35 % at CDR and 25 % at QR or AR.

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

5.3.8.2 Technical budget and margin computation

- a. The way to compute the technical budgets and margin shall be agreed between the customer and the supplier.

EXPECTED OUTPUT: Technical budgets and margin computation [D]F, SVR; SRR, PDR]

5.3.9 Compliance to this Standard

5.3.9.1 Compliance matrix

- a. The supplier shall provide a compliance matrix documenting conformance with the individual software engineering process requirements (Clause 5) applicable to the project or business agreement, as per ECSS-S-ST-00.

EXPECTED OUTPUT: ECSS-E-ST-40 compliance matrix [MGT, SDP; SRR, PDR]

5.3.9.2 Documentation compliance

- a. The compliance to each of the individual software engineering process requirements shall make reference to the document where the expected output is placed, if it is not placed in this Standard's DRDs in annexes of this document.

EXPECTED OUTPUT: ECSS-E-ST-40 compliance matrix [MGT, SDP; SRR, PDR]

NOTE A general statement of compliance to this Standard's DRDs is acceptable.

5.4 Software requirements and architecture engineering process

5.4.1 Overview

The software requirements and architecture engineering process consists of the following activities:

- software requirements analysis;
- software architectural design.

5.4.2 Software requirements analysis

5.4.2.1 Establishment and documentation of software requirements

- a. The supplier shall establish and document software requirements, including the software quality requirements, as part of the technical specification.

EXPECTED OUTPUT: The following outputs are expected:

- a. Functional and performance specifications, including hardware characteristics, and environmental conditions under which the software item executes, including budgets requirements [TS, SRS; PDR];
- b. Operational, reliability, safety, maintainability, portability, configuration, delivery, adaptation and installation requirements, design constraints [TS, SRS; PDR];
- c. Software product quality requirements (see ECSS-Q-ST-80 clause 7.2) [TS, SRS; PDR];
- d. Security specifications, including those related to factors which can compromise sensitive information [TS, SRS; PDR];
- e. Human factors engineering (ergonomics including HMI usability) specifications, following the human factor engineering process specified in ECSS-E-ST-10-11 [TS, SRS; PDR];
- f. Data definition and database requirements [TS, SRS; PDR];
- g. Validation requirements [TS, SRS, ICD; PDR]
- h. Interfaces external to the software item [TS, ICD; PDR];
- i. Reuse requirements (see ECSS-Q-ST-80 clause 6.2.7) [TS, SRS; PDR].

5.4.2.2 Definition of functional and performance requirements for in flight modification

- a. When in flight modification is specified for flight software, the supplier shall perform analysis of the specific implications for the software design and validation processes and include the functional and performance requirements in the technical specification, including in case of use of automatic code generation.

EXPECTED OUTPUT: Specifications for in flight software modifications [TS, SRS; PDR].

5.4.2.3 Construction of a software logical model

- a. The supplier shall construct a logical model of the functional requirements of the software product.

EXPECTED OUTPUT: Software logical model [TS, SRS; PDR].

- b. The supplier shall use a method to support the construction of the logical model.

EXPECTED OUTPUT: Software logical model method [TS, SRS; PDR].

- c. The logical model shall include a behavioural view.

EXPECTED OUTPUT: Behavioural view in software logical model [TS, SRS; PDR].

5.4.2.4 Conducting a software requirement review

- a. The supplier shall conduct a software requirement review (SWRR) as anticipation of the PDR, in conformance with 5.3.4.2b.

5.4.3 Software architectural design

5.4.3.1 Transformation of software requirements into a software architecture

- a. The supplier shall transform the requirements for the software item into an architecture that:
 - 1. describes its top-level structure;
 - 2. identifies the software components, ensuring that all the requirements for the software item are allocated to its software components and later refined to facilitate detailed design;
 - 3. covers as a minimum hierarchy, dependency, interfaces and operational usage for the software components;
 - 4. documents the process, data and control aspects of the product;
 - 5. describes the architecture static decomposition into software elements such as packages, classes or units;
 - 6. describes the dynamic architecture, which involves the identification of active objects such as threads, tasks and processes;
 - 7. describes the software behaviour.

EXPECTED OUTPUT: Software architectural design [DDF, SDD; PDR].

5.4.3.2 Software design method

- a. The supplier shall use a method (e.g. object oriented or functional) to produce the static and dynamic architecture including:
 - 1. software elements, their interfaces, and;
 - 2. software elements relationships.

EXPECTED OUTPUT: Software architectural design method [DDF, SDD; PDR].

5.4.3.3 Selection of a computational model for real-time software

- a. The dynamic architecture design shall be described according to an analysable computational model.

EXPECTED OUTPUT: Computational model [DDF, SDD; PDR].

5.4.3.4 Description of software behaviour

- a. The software architecture design shall also describe the behaviour of the software, by means of description techniques using automata and scenarios.

EXPECTED OUTPUT: Software behaviour [DDF, SDD; PDR].

5.4.3.5 Development and documentation of the software interfaces

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

EXPECTED OUTPUT: The following outputs are expected:

- a. *Preliminary external interfaces design [TS, ICD; PDR];*
- b. *Preliminary internal interfaces design [DDF, SDD; PDR].*

5.4.3.6 Definition of methods and tools for software intended for reuse

- a. The supplier shall define procedures, methods and tools for reuse, and apply these to the software engineering processes to comply with the reusability requirements for the software development.

EXPECTED OUTPUT: Software intended for reuse - justification of methods and tools [DJF, SRF; PDR].

- b. An evaluation of the reuse potential of the software shall be performed at PDR and CDR.

EXPECTED OUTPUT: Software intended for reuse - evaluation of reuse potential [DJF, SRF; PDR, CDR].

- c. The supplier shall design the software such that mission and configuration dependant data are separated e.g. in a database.

EXPECTED OUTPUT: Software architectural design with configuration data - [DDF, SDD; PDR, CDR].

5.4.3.7 Reuse of existing software

- a. The analysis of the potential reusability of existing software components shall be performed through:
 1. identification of the reuse components and models with respect to the functional requirements baseline, and;
 2. a quality evaluation of these components, applying ECSS-Q-ST-80 clause 6.2.7.

EXPECTED OUTPUT: Justification of reuse with respect to requirements baseline [DJF, SRF; PDR].

5.4.3.8 Definition and documentation of the software integration requirements and plan

- a. The supplier shall define and document the preliminary software integration strategy in terms of responsibility and schedule, control procedures and testing approach (goals to be achieved, sequence, environment and criteria).

EXPECTED OUTPUT: Software integration strategy [DJF, SUITP; PDR].

5.4.4 Conducting a preliminary design review

- a. The supplier shall conduct a preliminary design review (PDR) in accordance with clause 5.3.4.2.

NOTE The successful completion of the review establishes a baseline for the development of the software item.

5.5 Software design and implementation engineering process

5.5.1 Overview

The software design and implementation engineering process consists of the following activities:

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

5.5.2 Design of software items

5.5.2.1 Detailed design of each software component

- a. The supplier shall develop a detailed design for each component of the software and document it.

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

- b. Each software component shall be refined into lower levels containing software units that can be coded, compiled, and tested.

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

- c. It shall be ensured that all the software requirements are allocated from the software components to software units.

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

5.5.2.2 Development and documentation of the software interfaces detailed design

- a. 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, in order to allow coding without requiring further information.

EXPECTED OUTPUT: The following outputs are expected:

- a. *External interfaces design (update) [TS, ICD; CDR];*
- b. *Internal interfaces design (update) [DDF, SDD; CDR].*

5.5.2.3 Production of the detailed design model

- a. The supplier shall produce the detailed design model of the software components defined during the software architectural design, including their static, dynamic and behavioural aspects.

EXPECTED OUTPUT: The following outputs are expected:

- a. *Software static design model [DDF, SDD; CDR];*
- b. *Software dynamic design model [DDF, SDD; CDR];*
- c. *Software behavioural design model [DDF, SDD; CDR];*

5.5.2.4 Software detail design method

- a. The supplier shall use a design method (e.g. object oriented or functional method) to produce the detailed design including:
 - 1. software units, their interfaces, and;
 - 2. software units relationships.

EXPECTED OUTPUT: Software design method [DDF, SDD; CDR]

5.5.2.5 Detailed design of real-time software

- a. The dynamic design model shall be compatible with the computational model selected during the software architectural design model.

EXPECTED OUTPUT: Real-time software dynamic design model [DDF, SDD; CDR].

- b. The supplier shall document and justify all timing and synchronization mechanisms.

EXPECTED OUTPUT: Real-time software dynamic design model [DDF, SDD; CDR].

- c. The supplier shall document and justify all the design mutual exclusion mechanisms to manage access to the shared resources.

EXPECTED OUTPUT: Real-time software dynamic design model [DDF, SDD; CDR].

- d. The supplier shall document and justify the use of dynamic allocation of resources.

EXPECTED OUTPUT: Real-time software dynamic design model [DDF, SDD; CDR].

- e. The supplier shall ensure protection against problems that can be induced by the use of dynamic allocation of resources, e.g. memory leaks.

EXPECTED OUTPUT: Real-time software dynamic design model [DDF, SDD; CDR].

5.5.2.6 Utilization of description techniques for the software behaviour

- a. The behavioural design of the software units shall be described by means of techniques using automata and scenarios.

EXPECTED OUTPUT: Software behavioural design model techniques [DDF, SDD; CDR].

5.5.2.7 Determination of design method consistency for real-time software

- a. It shall be ensured that all the methods utilized for different item of the same software are, from a dynamic stand-point, consistent among themselves and consistent with the selected computational model.

EXPECTED OUTPUT: Compatibility of real-time design methods with the computational model [DDF, SDD; CDR].

5.5.2.8 Development and documentation of the software user manual

- a. The supplier shall develop and document the software user manual.

EXPECTED OUTPUT: Software user manual [DDF, SUM; CDR].