SLOVENSKI STANDARD
kSIST-TP FprCEN/CLC/TR 17602-80-03:2021

01-julij-2021

Zagotavljanje kakovosti proizvodov v vesoljski tehniki - Zanesljivost in varnost programske opreme

Space product assurance - Software dependability and safety

Raumfahrt-Produktsicherung - Software-Zuverlässigkeit und -Sicherheit

Assurance produit des projets spatiaux - Sûreté de fonctionnement et sécurité des logiciels

Ta slovenski standard je istoveten z: FprCEN/CLC/TR 17602-80-03

ICS:

35.240.99	Uporabniške rešitve IT na drugih področjih	IT applications in other fields
49.140	Vesoljski sistemi in operacije	Space systems and operations

kSIST-TP FprCEN/CLC/TR 17602-80-03:2021 **en,fr,de**

TECHNICAL REPORT

RAPPORT TECHNIQUE

TECHNISCHER BERICHT

FINAL DRAFT FprCEN/CLC/TR 17602-80-03

May 2021

ICS 49.140; 35.240.99

English version

Space product assurance - Software dependability and safety

Assurance produit des projets spatiaux - Sûreté de fonctionnement et sécurité des logiciels

Raumfahrt-Produktsicherung - Software-Zuverlässigkeit und -Sicherheit

This draft Technical Report is submitted to CEN members for Vote. It has been drawn up by the Technical Committee CEN/CLC/JTC 5.

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

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Warning : This document is not a Technical Report. It is distributed for review and comments. It is subject to change without notice and shall not be referred to as a Technical Report.

CEN-CENELEC Management Centre:
Rue de la Science 23, B-1040 Brussels

Ref. No. FprCEN/CLC/TR 17602-80-03:2021 E

Table of contents

Figures

European Foreword

This document (FprCEN/CLC/TR 17602-80-03:2021) has been prepared by Technical Committee CEN/CLC/JTC 5 “Space”, the secretariat of which is held by DIN.

It is highlighted that this technical report does not contain any requirement but only collection of data or descriptions and guidelines about how to organize and perform the work in support of EN 16602-80.

This Technical report (FprCEN/CLC/TR 17602-80-03:2021) originates from ECSS-Q-HB-80-03A Rev.1.

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 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 TR covering the same scope but with a wider domain of applicability (e.g.: aerospace).

This document is currently submitted to the CEN CONSULTATION.

Introduction

Dependability and safety are issues of paramount importance in the development and operations of space systems. The contribution of software to system dependability and safety is a key factor, especially in view of the growing complexity of the software used in space critical applications, together with the increasing cost and schedule constraints. Hence, the need for more dependable and safe software has led to the publication of this Handbook, meant to provide guidelines on the implementation of the software dependability and safety requirements defined in ECSS-Q-ST-80C and on the application of some methods and techniques for software dependability and safety.

Analyses and activities aiming at assessing and ensuring the system dependability and safety are carried out since the early stages of the development, and software needs to be properly addressed by these system-level activities. Hardware and software products are classified based on their criticality, in order to focus engineering and product assurance activities on the most critical items. At later stages, the inherent complexity of software calls for application of specific methods and techniques, aiming at refining the software criticality classification and supporting the implementation and verification of measures for critical software handling.

This handbook provides an overall description of the entire software dependability and safety workflow, considering the different activities at system and software level, the lifecycle phases and the customer-supplier relationships, with reference to the dependability and safety requirements defined in ECSS-Q-ST-80C. Some individual software RAMS techniques are also presented. They have been selected from the list of methods and techniques mentioned in different national and international standards and literature, from which a choice has been made based on their relevance to the requirements defined in the ECSS Standards.

1 Scope

This Handbook provides guidance on the application of the dependability and safety requirements relevant to software defined in ECSS-Q-ST-80C.

This Handbook provides support for the selection and application of software dependability and safety methods and techniques that can be used in the development of software-intensive space systems.

This Handbook covers all of the different kinds of software for which ECSS-Q-ST-80C is applicable. Although the overall software dependability and safety workflow description is mainly targeted to the development of spacecraft, the described approach can be adapted to projects of different nature (e.g. launchers, ground systems).

The methods and techniques described in the scope of this Handbook are mainly focused on assessment aspects, though specific development and implementation techniques for dependability and safety (e.g. software failure propagation prevention, defensive programming) are addressed.

2 References

For each document or Standard listed, a *mnemonic* (used to refer to that source throughout this document) is proposed in the left column, and then the *complete reference* is provided in the right one.

EN Reference	Reference in text	Title
EN 16602-30	[ECSS-Q-30]	ECSS-Q-ST-30C – Space product assurance - Dependability
EN 16602-30-02	[ECSS-Q-30-02]	ECSS-Q-ST-30-02C – Space product assurance – Failure modes, effects and criticality analysis (FMECA/FMEA)
EN 16602-30-09	[ECSS-Q-30-09]	ECSS-Q-ST-30-09C – Space product assurance – Availability analysis
EN 16602-40	[ECSS-Q-40]	ECSS-Q-ST-40C – Space product assurance – Safety
EN 16602-40-12	[ECSS-Q-40-12]	ECSS-Q-ST-40-12C – Space product assurance – Fault tree analysis – Adoption notice ECSS/IEC 61025
EN 16602-80	[ECSS-Q-80]	ECSS-Q-ST-80C – Space product assurance – Software product assurance
EN 16602-80-04	[ECSS-Q-80-04]	ECSS-Q-HB-80-04A – Space product assurance – Software metrication programme definition and implementation
EN 17603-40	[ECSS-E-HB-40]	ECSS-E-HB-40A – Space engineering – Software guidelines
EN 16601-00-01	[ECSS-S-ST-00-01]	ECSS-S-ST-00-01C – ECSS – Glossary of terms

3 Terms, definitions and abbreviated terms

3.1 Terms from other documents

a. For the purpose of this document, the terms and definitions from ECSS-S-ST-00-01 and ECSS-Q-ST-80 apply.

3.2 Abbreviated terms

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

Abbreviation	Meaning
LEOP	launch and early orbit phase
RAMS	reliability, availability, maintainability and safety
FMECA	Failure Modes, Effects and Criticality Analysis
(S)FMEA	(Software) Failure Modes and Effects Analysis
(S)FTA	(Software) Fault Tree Analysis
HSIA	Hardware/Software Interaction Analysis
ISVV	Independent Software verification and Validation
CPU	Central Processing Unit

4 Principles

4.1 General concepts

4.1.1 Software failures and faults

Several definitions of failure, fault and error in relation to software exist in literature (see e.g. [NASA-8719], [Laprie92]).

The most common approach is the following: a human mistake made in requirements specification, design specification or coding can result in a *fault* to be present (latent) in a software item. This hidden defect, under particular circumstances, can manifest itself as an *error* (a discrepancy between an expected value or action and the actual one) which, in turn, can lead to a *failure*, i.e. an unexpected/unintended behaviour of the system.

Different terms are used in different Standards. [ECSS-S-ST-00-01] defines "failure" as the "event resulting in an item being no longer able to perform its required function", while "fault" is the "state of an item characterized by inability to perform as required". According to [ECSS-S-ST-00-01], a fault can be the cause of a failure, but it can also be caused by a failure. [ECSS-Q-80] mentions both "software faults" and "software failures", while [ECSS-Q-40] uses the term "software errors". [NASA-8719] refers to "software hazards". [JPL D-28444] and [ED-153] use "software failures". This handbook uses the terms "software fault" and "software error" as defined above, but "software-caused failures" rather than just "software failures", for better clarity.

Software-caused failures differ from hardware-caused failures in many respects. For instance, software is not subject to wear or energy-related phenomena which are often the cause of hardware failures. This could lead to think that "software failures" do not exist as such, because the software always does what it is programmed to do. Actually, software-caused failures occur. They are due to conceptual mistakes in requirements definition, design or coding that result in faults residing in the code, ready to manifest themselves in specific conditions. Therefore, software faults are systematic, but the conditions that lead them to cause failures are extremely difficult to predict. Software-caused failures, much as hardware ones, often appear to occur randomly.

In addition, it is worth noting that software-caused failures lead sometimes to catastrophic consequences, as the space community has experienced several times in the past decades.

4.1.2 Software reliability

Software reliability, i.e. the capability of the software to perform a required function under given conditions for a given time interval, is a key issue for space applications. Software-caused failures can result in critical degradation of system performance, up to complete mission loss.

Activities meant to increase the reliability of the software encompass the whole system and software life cycle. Software reliability requirements are derived from system reliability requirements, and responded to through the implementation of specific engineering and product assurance measures.

System reliability requirements for space applications are stated both in qualitative and quantitative terms. While consolidated reliability models exist for hardware that allow to analytically demonstrate compliance with quantitative reliability requirements, the validity of the software reliability models proposed by field engineers over the years is subject to an on-going debate within industry, academia and international standards community. One major obstacle to the usability of software reliability models is the number and nature of the assumptions to be made in order to apply the mathematical calculations on which the models are based. Those assumptions have proven to be not fully justified for the vast majority of bespoke software like space application software (see [IMECS2009]), therefore invalidating the models themselves. As a conclusion, the use of software reliability models to justify compliance with applicable reliability requirements is not advisable, hence software reliability models are not addressed in this handbook.

4.1.3 Software maintainability

Software maintainability is the capability of the software to be retained or restored to a state in which it can perform a required function, when maintenance is performed. In other words, the maintainability of the software relates to the ease with which the software can be modified and put back into operation.

Software maintainability is an issue for all space application software, but it is of extreme importance for on-board software involved in critical functions. Most of the flight software is nowadays modifiable and uploadable from ground, but in case of software faults the time needed to upgrade the software can be a major concern. An example can be a failure caused by a satellite software component that leads the satellite into safe mode. Often, the safe mode can only be maintained for a limited period of time, up to some days. This means that the maintenance team has only limited time to identify the cause of the failure, update the software, validate and upload it. It is clear that in these cases the easiness of analysing the software documentation and code, in order to quickly find the fault and correct it, is a key issue.

Software maintainability is also linked to the potential obsolescence of development and operational platforms and tools. For software with a long planned lifetime, it is important to consider from the very beginning of the development the capability to operate and modify the software at the latest stages of the operational life, possibly in different, evolved environments.

4.1.4 Software availability

Software availability is the capability of the software to be in a state to perform a required function at a given instant of time or over a given time interval. Software availability is a function of software reliability and maintainability: frequent software-caused failures and long maintenance periods reduce the probability that the software be available and operational at a certain time or for a certain period of time.

System availability depends on software availability, and this is especially true for software intensive systems, such as ground applications (like e.g. mission control systems or payload data ground systems). The dependability requirements for this kind of systems are often expressed in terms of their availability, e.g. availability of generated data.

4.1.5 Software safety

Safety is a system property. Software in itself cannot cause or prevent harm to human beings, system loss or damage to environment. However, software is a main component of the system, and therefore contributes to its safety. The contribution of software to the system safety is called software safety [NASA-8719].

It is worth recalling that reliability and safety are two different concepts, although related. Reliability is concerned with every possible software-caused failure, whereas safety is concerned only with those that can result in actual system hazards. Not all software-caused failures can raise safety problems, and not all software that functions according to its specification is safe. A boot software that fails to start up the satellite's main instrument can be judged as totally unreliable, but perfectly safe. On the other hand, since hazards are often caused by system failures, safety and dependability are linked, and [ECSS-Q-30] requires the project dependability programme to be established in conjunction with the safety programme.

[ECSS-Q-40] defines "safety-critical" the software whose "loss or degradation of its function, or its incorrect or inadvertent operation, can result in catastrophic or critical consequences". The activities that lead to the identification of safety-critical software are addressed in this handbook.

4.1.6 System level and software level

Software is an integral component of space systems. Software requirements are derived from system requirements, and the software is eventually qualified when integrated into the target system. Software is also a complex artefact that follows a dedicated life cycle and for whose development particular skills are required and specific methods and techniques are applied.

System development and software development typically occur at different engineering and management levels, also from the organizational and contractual point of view. For instance, the development of the platform application software of a satellite is often subcontracted to a different company than the Prime contractor.

This also affects software dependability and safety aspects, much like for the relation between system and subsystems/equipment. It is true that software RAMS cannot be considered in isolation from the system, and this holds in particular for software safety. However, there are safety- and dependability-related activities performed at system level by system-level engineers, considering the system as a whole, and there are other activities carried out at software level by software experts, focusing on software characteristics. The different roles, skills and organizational levels cannot be neglected or denied. Rather, a sufficient level of iterations and integration between system- and software-level dependability and safety activities at all development stages is a key to sound software development.

4.1.7 Fault prevention, removal, tolerance, and forecasting

The means to achieve software dependability and safety can be classified into four major groups of techniques: fault prevention, removal, tolerance, and forecasting.

Fault prevention aims at preventing the introduction of faults in the final product. This is mainly achieved through the application of strict rules during the requirements definition, design and production of the software (e.g. application of well-established development methods, and rigorous design and coding standards).

Fault removal is meant to detect faults and remove them from the software. Verification and validation are the principal - although not unique - means applied for fault removal.