62A/1422/CDV

COMMITTEE DRAFT FOR VOTE (CDV)

PROJECT NUMBER:	
IEC 62304 ED2	
DATE OF CIRCULATION:	CLOSING DATE FOR VOTING:
2021-01-01	**2021-03-26**
SUPERSEDES DOCUMENTS:	
62A/1349/CDV, 62A/1383B/RVC	

IEC SC 62A : COMMON ASPECTS OF ELECTRICAL EQUIPMENT USED IN MEDICAL PRACTICE	
SECRETARIAT: United States of America	SECRETARY: Ms Hae Choe
OF INTEREST TO THE FOLLOWING COMMITTEES: TC 62,SC 62B,SC 62C,SC 62D,TC 65,TC 66,TC 76,TC 106,TC 108	PROPOSED HORIZONTAL STANDARD: ☐ Other TC/SCs are requested to indicate their interest, if any, in this CDV to the secretary.
FUNCTIONS CONCERNED: ☐ EMC ☐ ENVIRONMENT ☐ QUALITY ASSURANCE ☒ SAFETY	
☒ SUBMITTED FOR CENELEC PARALLEL VOTING **Attention IEC-CENELEC parallel voting** The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this Committee Draft for Vote (CDV) is submitted for parallel voting. The CENELEC members are invited to vote through the CENELEC online voting system.	☐ NOT SUBMITTED FOR CENELEC PARALLEL VOTING

This document is still under study and subject to change. It should not be used for reference purposes.

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

TITLE:

IEC 62304 Ed. 2: Health software - Software life cycle processes

PROPOSED STABILITY DATE: 2024

NOTE FROM TC/SC OFFICERS:

Please note that this draft is a joint project between IEC/SC 62A and ISO/TC 215 and is IEC led. During the last CDV stage, this project was approved on the IEC side but not approved in ISO and CENELEC. A task group was assigned to develop proposed resolutions to the comments and to the draft which were reviewed by the 62304 Project Team and the IEC/ISO Joint Working Group. Attached is the result of that extensive work. Some comments did not offer specific changes but provided ideas that may be better utilized during the next maintenance cycle for this document.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

HEALTH SOFTWARE –

SOFTWARE LIFE CYCLE PROCESSES

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62304 has been prepared by a joint working group of subcommittee 62A: Common aspects of electrical equipment used in medical practice, of IEC technical committee 62: Electrical equipment in medical practice,in cooperation with ISO Technical Committee 215, Health informatics. Table C.5 was prepared by ISO/IEC JTC 1/SC 7, Software and systems engineering.

It is published as a dual logo standard.

This second edition cancels and replaces the first edition published in 2006 and Amendment 1:2015. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) the scope of this document has been expanded to HEALTH SOFTWARE;

b) Clause 4 related to general requirements has been updated to assure that this document would meet the state of art of the use-environment and the way that HEALTH SOFTWARE is being used.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
62A/XX/FDIS	62A/XX/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

In this document, the following print types are used:

– requirements and definitions: roman type;

– informative material appearing outside of tables, such as notes, examples and references: smaller type. Normative text of tables is also in a smaller type;

– TERMS USED THROUGHOUT THIS DOCUMENT THAT HAVE BEEN DEFINED IN CLAUSE 3: SMALL CAPITALS.

The verbal forms used in this document conform to usage described in Clause 7 of the ISO/IEC Directives, Part 2:2018. For the purposes of this document, the verb:

– "shall" means that compliance with a requirement is mandatory for compliance with this document;

– "should" means that compliance with a requirement is recommended but is not mandatory for compliance with this document;

– "may" is used to describe a permissible way to achieve compliance with a requirement;

– "establish" means to define, document and implement.

Where this document uses the term "as appropriate" in conjunction with a required PROCESS, ACTIVITY, TASK or output, the intention is that the MANUFACTURER shall use the PROCESS, ACTIVITY, TASK or output unless the MANUFACTURER can document a justification for not so doing.

An asterisk (*) as the first character of a title or at the beginning of a paragraph indicates that there is guidance related to that item in Annex B.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

NOTE The attention of users of this document is drawn to the fact that equipment MANUFACTURERS and testing organizations may need a transitional period following publication of a new, amended or revised IEC or ISO publication in which to make products in accordance with the new requirements and to equip themselves for conducting new or revised tests. It is the recommendation of the committee that the content of this publication be adopted for mandatory implementation nationally not earlier than 3 years from the date of publication.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

181

182

INTRODUCTION

Software is becoming increasingly important in healthcare. The use of software can help contribute to more efficient and safe care of patients. Thus, HEALTH SOFTWARE needs to be developed with appropriate controls to ensure its safe, effective and secure use.

Users of software in the care environment have expanded from clinical users (nurses, technicians, dieticians, physicians, etc.) to include non-clinical users (patients, consumers, laypersons, family care givers, etc.). IEC 62304:2006 and IEC 62304:2006/AMD1:2015 focused on software life cycle activities for MEDICAL DEVICE SOFTWARE that was primarily used by clinical users. For this reason, the scope of this document has been expanded to HEALTH SOFTWARE.

As software becomes more dependent on network connectivity and integral to clinical workflows, additional considerations need to be made for SECURITY and USABILITY. HEALTH SOFTWARE is being used more commonly in the home and outside of the hospital, so it becomes even more important to develop these products with the user and use environment in mind. For these reasons, Clause 4 related to general requirements has been updated to assure that this document would meet the state of art of the use-environment and the way that HEALTH SOFTWARE is being used.

This document does not duplicate well-established requirements from standards for USABILITY and SECURITY.

Establishing the SAFETY and EFFECTIVENESS of HEALTH SOFTWARE requires knowledge of what the HEALTH SOFTWARE is intended to do and demonstration that the use of the HEALTH SOFTWARE fulfils those intentions without causing any unacceptable RISKS. To demonstrate the EFFECTIVENESS, software VALIDATION is required, which is outside of the scope of this document.

The MANUFACTURER of HEALTH SOFTWARE is responsible for determining and complying with the appropriate SAFETY, SECURITY, environmental, health, and interference protection practices. Many laws, regulations, and other rules from authorities having jurisdiction have a direct impact on the way SOFTWARE SYSTEMS are developed, tested, and maintained. From a software development perspective, MANUFACTURERS consider these laws, regulations, and other rules as inputs into the requirements that the HEALTH SOFTWARE supports. This means that the requirements of some laws or regulations can translate into specific HEALTH SOFTWARE product requirements. For example, if HEALTH SOFTWARE is going to send or share health data to a doctor, hospital, or other covered entity, it has an obligation to adhere to privacy and SECURITY rules. This can involve authentication and SECURITY mechanisms to protect patient information saved in an electronic format. Other requirements of the laws or regulations can impact the PROCESS used during the development of the HEALTH SOFTWARE product. For example, many national regulations and quality systems standards have design control requirements that translate into specific procedures to confirm that the product is designed, verified, and validated in a systematic manner and per proven software engineering practices.

This document specifies that MANUFACTURERS develop and maintain HEALTH SOFTWARE within a quality management SYSTEM (see 4.1) and a RISK MANAGEMENT SYSTEM (4.2).

This document provides a framework for a life cycle PROCESS. It defines the ACTIVITIES and TASKS necessary for the safe design, development and maintenance of HEALTH SOFTWARE. The development and maintenance life cycle ACTIVITIES are shown in Figure 1 and described in Clause 5 and Clause 6. Some incidents in healthcare delivery are related to HEALTH SOFTWARE SYSTEMS, including failures that can occur or be injected when software is modified.

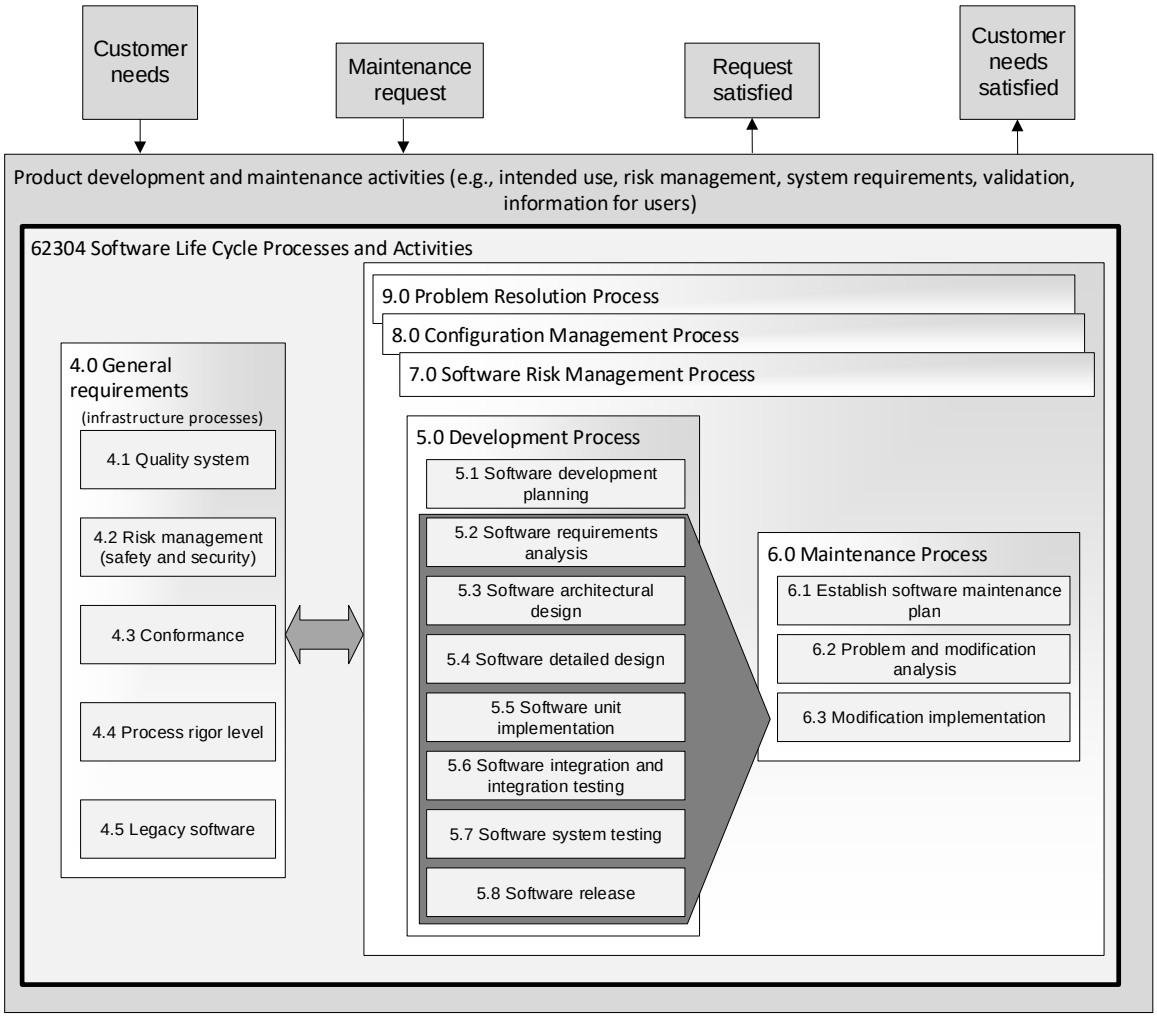


Figure 1 – Overview of software development and maintenance PROCESSES and ACTIVITIES

This document identifies two additional supporting PROCESSES considered essential for developing safe HEALTH SOFTWARE. They are the software configuration management PROCESS (Clause 8) and the software problem resolution PROCESS (Clause 9).

This document does not specify a specific organizational structure nor responsibilities within the organization of the MANUFACTURER to perform PROCESSES, ACTIVITIES, and TASKS. This document specifies planning of software development, maintenance and supporting PROCESS ACTIVITIES, and the completion of the ACTIVITIES or TASKS for conformance with this document.

This document does not prescribe the name, format, or explicit content of the documentation to be produced. This document calls for adequate evidence of required ACTIVITIES and TASKS by documentation. Regardless how content is structured and packaged, it is expected that a controlled documentation PROCESS is in place. This document does not prescribe a specific life cycle model. The users of this document are responsible for selecting a life cycle model for the software project and for mapping the PROCESSES, ACTIVITIES, and TASKS in this document onto that model.Annex A provides rationale for the clauses of this document. Annex B provides guidance on the provisions of this document.

245 **HEALTH SOFTWARE –**

246 **SOFTWARE LIFE CYCLE PROCESSES**

247

248 # 1 Scope

249 ## 1.1 * Purpose

250 This document defines the development and maintenance life cycle requirements for HEALTH
251 SOFTWARE. The set of PROCESSES, ACTIVITIES, and TASKS described in this document establishes
252 a common framework for HEALTH SOFTWARE life cycle PROCESSES.

253 ## 1.2 * Field of application

254 This document applies to the development and maintenance of HEALTH SOFTWARE by a
255 MANUFACTURER. MEDICAL DEVICE SOFTWARE is a subset of HEALTH SOFTWARE (see Figure 2).
256 Therefore, this document applies to:

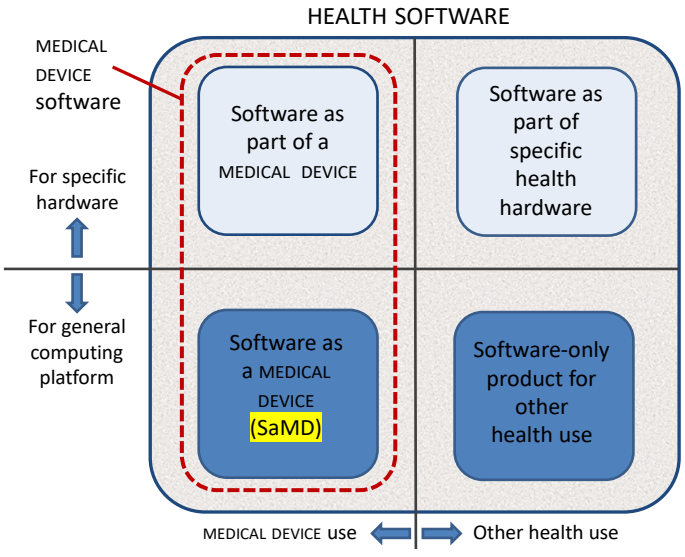
257 – software as part of a MEDICAL DEVICE;

258 – software as part of specific health hardware;

259 – software as a MEDICAL DEVICE (SaMD);

260 – software-only product for other health use.

261 Figure 2 provides a graphical representation of the health software this document applies to.



262

263 **Figure 2 – HEALTH SOFTWARE field of application**

264 NOTE 1 Examples of HEALTH SOFTWARE include the following:

265 1) software as a part of a MEDICAL DEVICE: software that is an integral part of a device such as an infusion pump or dialysis
266 machine.

267 2) software as part of specific health hardware: patient wristband printer software, healthcare scanner software, health app on
268 specific wearable hardware (i.e. watch, wristband, chestband).

3) software as a MEDICAL DEVICE (SaMD): software that is itself a MEDICAL DEVICE, such as a software application that performs diagnostic image analysis for making treatment decisions. A definition of software as a MEDICAL DEVICE is provided in [46][1].

4) software-only product for other health use: hospital information systems, electronic health records, electronic medical records, mobile applications running on devices without physiologic sensors or detectors, software as a service, i.e. software executed in an external environment, providing calculation-results that fulfil the definition of a MEDICAL DEVICE.

NOTE 2 This document can be used in the development and maintenance of HEALTH SOFTWARE. Before any type of software can be placed into service, activities are necessary before the software product is integrated into the SYSTEM. These SYSTEM activities are not covered by this document (see Figure 1), but can be found in related product standards (e.g., IEC 60601-1 [1] or IEC 82304-1 [15]). For software as a MEDICAL DEVICE (SaMD) additional guidance on ACTIVITIES at a system level (e.g. clinical EVALUATION) can be found in regulatory authority guidance documents.

This document describes PROCESSES that are intended to be applied to software which executes on a processor or which is executed by other software (for example an interpreter) which executes on a processor.

This document applies regardless of the persistent storage device(s) used to store the software (for example: hard disk, optical disk, permanent or flash memory).

This document applies regardless of the method of delivery of the software (for example: transmission by network or email, EEPROM, Smart Drive, Cloud). The method of software delivery itself is not considered HEALTH SOFTWARE.

This document does not cover the means of VALIDATION, even when the product consists entirely of software. It also does not cover software life cycle steps after release for INTENDED USE of the product, including implementation, configuration, integration (with other systems), go-live, clinical use, operations, decommissioning or disposal, other than ACTIVITIES involving maintenance of the software.

This document does not cover the VALIDATION of software tools used in the design of medical devices (e.g. computer aided design (CAD) software), software used in MEDICAL DEVICE quality systems or software for regulated processes (see ISO/TR 80002-2 [20] and AAMI TIR 36 [40]).

NOTE 3 If a product incorporates embedded software intended to be executed on a processor, the requirements of this document apply to the software, including the requirements concerning SOFTWARE OF UNKNOWN PROVENANCE (SOUP) – see 8.1.2).

Data quality and VALIDATION of emergent characteristics or functionality of artificial intelligence (AI) HEALTH SOFTWARE are not within the scope of this document.

NOTE 4 Users of this document may need to utilize other standards and technical sources to supplement this document in addressing the unique performance characteristics of their AI HEALTH SOFTWARE. As AI is a rapidly developing field and as further insights are gained, new updates may need to be incorporated into IEC 62034 via an amendment or a future new edition.

1.3 Relationship to other standards

This HEALTH SOFTWARE life cycle document is written in a way that it can be used together with referencing standards when developing and maintaining a product that includes HEALTH SOFTWARE (see Annex C).

2 * Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies.

---

1 Numbers in square brackets refer to the Bibliography.

For undated references, the latest edition of the referenced document (including any amendments) applies.

3 * Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardisation at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at https://www.iso.org/obp

3.1
ACTIVITY

set of one or more interrelated or interacting TASKS

3.2
ANOMALY

condition that deviates from expectations based on requirements specifications, design documents, standards, etc. or from someone's perceptions or experiences

[SOURCE: ISO/IEC 25051:2014, 4.1.2]

3.3
ARCHITECTURE

[SYSTEM] fundamental concepts or properties of a SYSTEM in its environment embodied in its elements, relationships, and in the principles of its design and evolution

[SOURCE: ISO/IEC/IEEE Std 24765-2017, 3.216, definition 1]

3.4
CHANGE REQUEST

documented specification of a change to be made to HEALTH SOFTWARE

3.5
CONFIGURATION ITEM

entity that can be uniquely identified at a given reference point

3.6
DELIVERABLE

required result or output (includes documentation) of an ACTIVITY or TASK

3.7
EVALUATION

systematic determination of the extent to which an entity meets its specified criteria

3.8
EFFECTIVENESS

ability to produce the intended result

[SOURCE: ISO 81001-1:—[2], 3.2.5]

3.9
HARM

injury or damage to the health of people, or damage to property or the environment

[SOURCE: ISO 81001-1:—, 3.4.5]

3.10
HAZARD

potential source of HARM

Note 1 to entry: Potential sources of HARM include breach of SECURITY and reduction of EFFECTIVENESS.

[SOURCE: ISO 81001-1:—, 3.4.6, modified – Note 1 to entry has been added.]

3.11
HAZARDOUS SITUATION

circumstance in which people, property or the environment is/are exposed to one or more HAZARD(S)

[SOURCE: ISO 81001-1:—, 3.4.7]

3.12
HEALTH SOFTWARE

SOFTWARE intended to be used specifically for managing, maintaining, or improving health of individual persons, or the delivery of care, or which has been developed for the purpose of being incorporated into a MEDICAL DEVICE

Note 1 to entry: HEALTH SOFTWARE fully includes what is considered software as a MEDICAL DEVICE.

[SOURCE: ISO 81001-1:—, 3.3.9]

3.13
INTENDED USE

use for which a product, PROCESS or service is intended according to the specifications, instructions and information provided by the MANUFACTURER

Note 1 to entry: The intended medical indication, patient population, part of the body or type of tissue iteracted with, user profile, use environment, and operating principle are typical elements of the INTENDED USE.

[SOURCE: ISO 81001-1:—, 3.2.7, modified – The second preferred term "intended purpose" has been deleted.]

---

2 Under preparation. Stage at the time of publication: ISO/DIS 81001-1:2019.

3.14
LEGACY SOFTWARE

HEALTH SOFTWARE placed on the market before the publication of this document for which the MANUFACTURER seeks conformance retrospectively

3.15
MANUFACTURER

natural or legal person with responsibility for designing, manufacturing, packaging, or labelling of HEALTH SOFTWARE product, or adapting HEALTH SOFTWARE product before it is placed on the market and/or put into service, regardless of whether these operations are carried out by that person or on that person's behalf by a third party

Note 1 to entry: Attention is drawn to the fact that the provisions of national or regional regulations can apply to the definition of MANUFACTURER.

Note 2 to entry: For a definition of "labelling", see ISO 13485:2016, 3.8.

Note 3 to entry: "Developer" or "developer organization" are commonly used terms and are synonymous with the term "MANUFACTURER" in the context of health information technology.

3.16
MEDICAL DEVICE

instrument, apparatus, implement, machine, appliance, implant, reagent for in vitro use, software, material or other similar or related article, intended by the MANUFACTURER to be used, alone or in combination, for human beings for one or more of the specific medical purpose(s) of

– diagnosis, prevention, monitoring, treatment or alleviation of disease,
– diagnosis, monitoring, treatment, alleviation of or compensation for an injury,
– investigation, replacement, modification, or support of the anatomy or of a physiological PROCESS,
– supporting or sustaining life,
– control of conception,
– disinfection of MEDICAL DEVICES,
– providing information by means of in vitro examination of specimens derived from the human body,

and which does not achieve its primary intended action by pharmacological, immunological or metabolic means, in or on the human body, but which may be assisted in its function by such means

Note 1 to entry: Products which could be considered to be MEDICAL DEVICES in some jurisdictions but not in others include:

– disinfection substances,
– aids for persons with disabilities,
– devices incorporating animal and/or human tissues,
– devices for in-vitro fertilization or assisted reproductive technologies.

Note 2 to entry: In conjunction with IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012, the term "MEDICAL DEVICE" assumes the same meaning as ME EQUIPMENT or ME SYSTEM (which are defined terms of IEC 60601-1).

[SOURCE: ISO 81001-1:—, 3.3.13, modified – Note 2 to entry has been added.]

3.17
MEDICAL DEVICE SOFTWARE

SOFTWARE SYSTEM that has been developed for the purpose of being incorporated into a MEDICAL DEVICE, or is developed to be software as a MEDICAL DEVICE (SaMD)

3.18
PROBLEM REPORT

record of actual or potential behaviour of HEALTH SOFTWARE that a user or other interested person believes to be unsafe, inappropriate for the INTENDED USE or contrary to specification

Note 1 to entry: This document does not require that every PROBLEM REPORT results in a change to HEALTH SOFTWARE. A MANUFACTURER can reject a PROBLEM REPORT as a misunderstanding, error or insignificant event with sufficient justification.

Note 2 to entry: A PROBLEM REPORT can relate to HEALTH SOFTWARE released for INTENDED USE or to HEALTH SOFTWARE that is still under development.

Note 3 to entry: This document requires the MANUFACTURER to perform extra decision making steps (see Clause 6) for a PROBLEM REPORT relating to HEALTH SOFTWARE to ensure that regulatory actions are identified and implemented.

3.19
PROCESS

set of interrelated or interacting ACTIVITIES that use inputs to deliver an intended result

Note 1 to entry: The term "ACTIVITIES" covers use of resources.

[SOURCE: ISO 81001-1:—, 3.2.10, modified – The note to entry has been added.]

3.20
REGRESSION TESTING

testing following modification to a test item or its operational environment to identify whether regression failures occur

Note 1 to entry: Sufficiency of a set of regression test cases depends on the item under test and on modifications to that item or its operational environment.

[SOURCE: ISO/IEC/IEEE 90003:2018, 3.6]

3.21
RESIDUAL RISK

RISK remaining after RISK CONTROL measures have been implemented

[SOURCE: ISO 81001-1:—, 3.4.9]

3.22
RISK

combination of the probability of occurrence of HARM and the severity of that HARM

[SOURCE: ISO 81001-1:—, 3.4.10, modified – The note to entry has been deleted.]

447 **3.23**
448 **RISK ANALYSIS**

449 systematic use of available information to identify HAZARDS and to estimate the RISK

450 [SOURCE: ISO 81001-1:—, 3.4.11]

451 **3.24**
452 **RISK CONTROL**

453 PROCESS in which decisions are made and RISKS are reduced to, or maintained within, specified levels

454 [SOURCE: ISO 81001-1:—, 3.4.13, modified – The definition has been rephrased.]

455 **3.25**
456 **RISK ESTIMATION**

457 PROCESS used to assign values to the probability of occurrence of HARM and the severity of that HARM

458 [SOURCE: ISO 81001-1:—, 3.4.14]

459 **3.26**
460 **RISK EVALUATION**

461 PROCESS of comparing the estimated RISK against given RISK criteria to determine the acceptability of
462 the RISK

463 [SOURCE: ISO 81001-1:—, 3.4.15]

464 **3.27**
465 **RISK MANAGEMENT**

466 systematic application of management policies, procedures, and practices to the TASKS of analyzing,
467 evaluating, controlling, and monitoring RISK

468 [SOURCE: ISO 81001-1:—, 3.4.160]

469 **3.28**
470 **RISK MANAGEMENT FILE**

471 set of records and other documents that are produced as a result of RISK MANAGEMENT activities

472 [SOURCE: ISO 81001-1:—, 3.4.17, modified – The words "by RISK MANAGEMENT" have been replaced
473 by "as a result of RISK MANAGEMENT activities".]

474 **3.29**
475 **SAFETY**

476 freedom from unacceptable RISK

477 [SOURCE: ISO 81001-1:—, 3.2.12]

3.30
SECURITY

CYBERSECURITY

state where information and SYSTEMS are protected from unauthorized activities, such as access, use, disclosure, disruption, modification, or destruction to a degree that the related RISKS to confidentiality, integrity, and availability are maintained at an acceptable level throughout the

[SOURCE: ISO 81001:—, 3.2.13]

3.31
SERIOUS INJURY

injury or illness that

c) is life threatening or deadly,

d) results in permanent impairment of a body function or permanent damage to a body structure, or

e) necessitates medical or surgical intervention to prevent permanent impairment of a body function or permanent damage to a body structure

Note 1 to entry: "Permanent impairment" means an irreversible impairment or damage to a body structure or function excluding trivial impairment or damage.

3.32
SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

conceptual structure spanning the life of the software from definition of its requirements to its release for INTENDED USE, which

– identifies the PROCESS, ACTIVITIES and TASKS involved in development of HEALTH SOFTWARE,

– describes the sequence of and dependency between ACTIVITIES and TASKS, and

– identifies the milestones at which the completeness of specified DELIVERABLES is verified

3.33
SOFTWARE ITEM

identifiable part of a computer program, i.e. source code, object code, control code, control data, or a collection of these items

Note 1 to entry: Three terms identify the software decomposition. The top level is the SOFTWARE SYSTEM. The lowest level that is not further decomposed is the SOFTWARE UNIT. All levels of composition, including the top and bottom levels, can be called SOFTWARE ITEMS. A SOFTWARE SYSTEM, then, is composed of one or more SOFTWARE ITEMS, and each SOFTWARE ITEM is composed of one or more SOFTWARE UNITS or decomposable SOFTWARE ITEMS. The responsibility is left to the MANUFACTURER to define the granularity of the SOFTWARE ITEMS and SOFTWARE UNITS. See Clause B.2 Guidance for Terms and Definitions and Software detailed design.

Note 2 to entry: Based on ISO/IEC/IEEE 12207:2017, 3.1.53.

3.34
SOFTWARE MAINTENANCE

modification of HEALTH SOFTWARE after release for INTENDED USE, for one or more of the following reasons:

a) corrective, as fixing faults;

b) adaptive, as adapting to new hardware or software platform;

c) perfective, as implementing new requirements;

d) preventive, as making the product more maintainable

Note 1 to entry: See also ISO/IEC 14764:2006, 3.10.

[SOURCE: IEC 82304-1:2016, 3.21, modified – In the definition, the words "HEALTH SOFTWARE PRODUCT" have been replaced by "HEALTH SOFTWARE", and reference 3.10 has been added to the note to entry.]

3.35
SOFTWARE SYSTEM

integrated collection of SOFTWARE ITEMS organized to accomplish a specific function or set of functions

3.36
SOFTWARE UNIT

SOFTWARE ITEM that is not subdivided into other items

Note 1 to entry: The granularity of SOFTWARE UNITS is defined by the MANUFACTURER (see Clause B.2 Guidance for Terms and definitions and Software detailed design0).

3.37
SOFTWARE OF UNKNOWN PROVENANCE

SOUP

SOFTWARE ITEM that is already developed and generally available and that has not been developed for the purpose of being incorporated into the product or SOFTWARE ITEM previously developed for which adequate records of the development PROCESSES are not available

Note 1 to entry: A HEALTH SOFTWARE SYSTEM is not SOUP.

Note 2 to entry: SOUP is also called "off-the-shelf software" when it has not been developed for the purpose of being incorporated into the product.

3.38
SYSTEM

combination of interacting elements organized to achieve one or more stated purposes

Note 1 to entry: A SYSTEM can include the associated equipment, facilities, material, software, firmware, technical documentation, services and personnel required for operations and support to the degree necessary for use in its intended environment.

[SOURCE: ISO 81001-1:—, 3.3.17, modified – Note 1 to entry has been added.]

3.39
TASK

single piece of work to be done

3.40
TRACEABILITY

degree to which a relationship can be established between two or more DELIVERABLES of the development PROCESS, especially DELIVERABLES having a predecessor-successor or master-subordinate relationship to one another

Note 1 to entry: Requirements, ARCHITECTURE, RISK CONTROL measures, are examples of DELIVERABLES of the development PROCESS.

[SOURCE: ISO/IEC/IEEE 24765:2017, 3.4350, Definition 1, modified – In the definition, the word "products" has been replaced by "DELIVERABLES", and Note 1 to entry has been added.]

3.41
USABILITY

characteristic of the user interface that facilitates use and thereby establishes EFFECTIVENESS, efficiency and user satisfaction in the INTENDED USE environment

[SOURCE: ISO 81001-1:—, 3.2.15, modified – Note 1 to entry has been deleted.]

3.42
VALIDATION

confirmation, through the provision of objective evidence, that the requirements for a specific

INTENDED USE or application have been fulfilled.

Note 1 to entry: The objective evidence needed for a VALIDATION is the result of a test or other form of determination such as performing alternative calculations or reviewing documents.

Note 2 to entry: The word “validated” is used to designate the corresponding status.

Note 3 to entry: The use conditions for VALIDATION can be real or simulated.

[SOURCE: ISO 9000:2015, 3.8.13]

3.43
VERIFICATION

confirmation, through provision of objective evidence, that specified requirements have been fulfilled

Note 1 to entry: Besides testing, objective evidence can also be provided by other means such as: calculations, inspections or document reviews.

Note 2 to entry: The activities carried out for VERIFICATION are sometimes called a qualification PROCESS.

Note 3 to entry: The word "verified" is used to designate the corresponding status.

Note 4 to entry: In design and development, VERIFICATION concerns the PROCESS of examining the result of a given ACTIVITY to determine conformity with the stated requirement for that ACTIVITY.

[SOURCE: ISO 81001-1:—, 3.2.16, modified – Note 1 to entry has been rephrased, and Note 4 to entry has been added.]

3.44
VERSION

identified instance of a CONFIGURATION ITEM

Note 1 to entry: Modification to a VERSION of HEALTH SOFTWARE, resulting in a new VERSION, requires software configuration management action.

4 * General requirements

4.1 * Quality management

The MANUFACTURER of HEALTH SOFTWARE shall demonstrate the ability to provide HEALTH SOFTWARE that consistently meets requirements and meets applicable regulatory requirements.

NOTE 1 Demonstration of this ability can be by the use of a quality management SYSTEM that complies with:

– ISO 13485 [16] (see Table C.2 and Annex D); or

– a national quality management SYSTEM standard; or

– a quality management SYSTEM required by national regulation.

NOTE 2 Guidance for applying quality management SYSTEM requirements to software can be found in ISO/IEC 90003 [38].

4.2 * RISK MANAGEMENT

The MANUFACTURER of HEALTH SOFTWARE shall establish and maintain the following.

a) A PROCESS for managing RISKS, primarily to the patient, but also to the operator, other persons, property, and the environment. This PROCESS shall provide methods for identifying HAZARDS, performing RISK ESTIMATION and RISK EVALUATION, controlling identified RISKS, and monitoring the EFFECTIVENESS of the RISK CONTROL measures, taking the INTENDED USE of the HEALTH SOFTWARE into account.

b) As appropriate, a PROCESS for managing RISKS associated with SECURITY. This PROCESS shall provide methods for identifying vulnerabilities, estimating and evaluating the associated threats, controlling these threats, and monitoring the EFFECTIVENESS of the RISK CONTROL (SECURITY) measures, taking the INTENDED USE of the HEALTH SOFTWARE into account.

These PROCESSES may be combined into a single PROCESS.

NOTE See also 5.2.2 e), 7.1.2 h), and 7.2.2.

4.3 Conformance

Conformance with this document is defined as implementing all of the PROCESSES, ACTIVITIES, and TASKS identified in this document in accordance with the software process rigor level.

NOTE 1 The software process rigor levels assigned to each requirement are identified in the normative text following the requirement.

Conformance is determined by inspection of all documentation required by this document including the RISK MANAGEMENT FILE, and assessment of the PROCESSES, ACTIVITIES and TASKS required for the software process rigor level.

Conformance of LEGACY SOFTWARE can be demonstrated by implementing 4.5.

NOTE 2 This assessment could be carried out by internal or external audit.

NOTE 3 The assessment allows for flexibility in the methods of implementing the PROCESSES and performing the ACTIVITIES and TASKS.

Where any requirements contain "as appropriate" and were not performed, documentation for the justification is necessary for this assessment.

4.4 Software process rigor level

4.4.1 * Purpose of software process rigor level

The purpose of the software process rigor level is to determine the required rigor of the software PROCESSES prior to their start. The software process rigor level is determined by EVALUATION of the RISK and severity of HARM related to potential SOFTWARE SYSTEM failures and considering specified RISK CONTROLS external to the SOFTWARE SYSTEM (see Figure 3).

The ACTIVITY of determining the software process rigor level begins prior to software development ACTIVITIES (Clause 5) and is executed prior to RISK CONTROL implementation in the SOFTWARE SYSTEM.

For the purpose of determination of software process rigor level, the software failure shall be treated as occurring with 100% probability.

NOTE 1 Other standards can require additional levels of rigor for the development PROCESS.

NOTE 2 See also B.2 Guidance on Software process rigor level **Error! Reference source not found.**.

RISK ANALYSIS is updated over the life of the product, and the software process rigor level should be re-evaluated whenever there is a change in RISK or severity of HARM, associated with new or existing potential SOFTWARE SYSTEM failures, or when new RISK CONTROL measures external to the SOFTWARE are defined. The re-EVALUATION can result in a change to the software process rigor level as defined in 4.4.3. See also 5.2.4.

4.4.2 Performing initial RISK ANALYSIS to detemine process rigor level

4.4.2.1 Define how the HEALTH SOFTWARE is intended to be used and the INTENDED USE environment

The MANUFACTURE shall determine and document how the HEALTH SOFTWARE is intended to be used and the INTENDED USE environment. The MANUFACTURER should consider, at a minimum, the following as appropriate:

a) platforms and platform versions with which the HEALTH SOFTWARE is intended to work;

b) users of the HEALTH SOFTWARE;

c) HEALTH SOFTWARE's INTENDED USE;

d) problem(s) the HEALTH SOFTWARE is trying to solve;

e) health and wellness outcomes that can be achieved, if any.

f) kinds of information the HEALTH SOFTWARE is handling and the other systems it is intended to interface and communicate with;

g) scenarios describing typical use of the HEALTH SOFTWARE;

h) explicit limitations relating to the requirements or use of the HEALTH SOFTWARE; and

i) support and sustainability for the anticipated life of the HEALTH SOFTWARE;

4.4.2.2 Determine potential software failures which can contribute to a HAZARDOUS SITUATION

The MANUFACTURER shall determine and document potential software failures which can contribute to a HAZARDOUS SITUATION such as:

a) known and foreseeable HAZARDOUS SITUATIONS to patients, operator, or environment;

b) both normal and abnormal operating conditions and usage scenarios;

c) Use errors that could lead to patient or operator injury, or HARM to the environment;

d) logic errors that could compromise data quality and decision support, leading to incorrect care decisions or delays;End-to-end clinical process, including functionality and how that functionality is used in patient care contexts;

e) Inter and intra product messaging/interoperability;

f) Data / Information Security;

g) Known and foreseeable SECURITY/Cybersecurity vulnerabilities;

h) Software defects;

i) Failure or unexpected results from SOUP;

4.4.2.3 For each identified HAZARDOUS SITUATION determine

For each identified HAZARDOUS SITUATION the MANUFACTURER shall determine and document:

a) the severity associated with the HAZARD;

b) the causes of the HAZARD;

c) the resulting RISK; and

d) for each identified HAZARD that results in a HAZARDOUS SITUATION, identify if the HAZARDOUS SITUATION may result in injury, death, or HARM to the environment.

NOTE 1 Some examples of HAZARD identification techniques include fault tree analysis, failure mode and effects analysis (FMEA);

4.4.3 * Assigning software process rigor level

a) A software process rigor level for each SOFTWARE SYSTEM shall be determined prior to start of software development ACTIVITIES. Level C requirements shall apply until a software process rigor level is assigned.

NOTE 1 In this document, the software process rigor levels for which a specific requirement applies are identified following the requirement in the form [Level . . .].

b) The MANUFACTURER may assign a lower software process rigor level (A or B) to the SOFTWARE SYSTEM based on the RISK resulting from a HAZARDOUS SITUATION to which the SOFTWARE SYSTEM can contribute in a worst-case-scenario (see 4.4.2) as indicated in Figure 3.

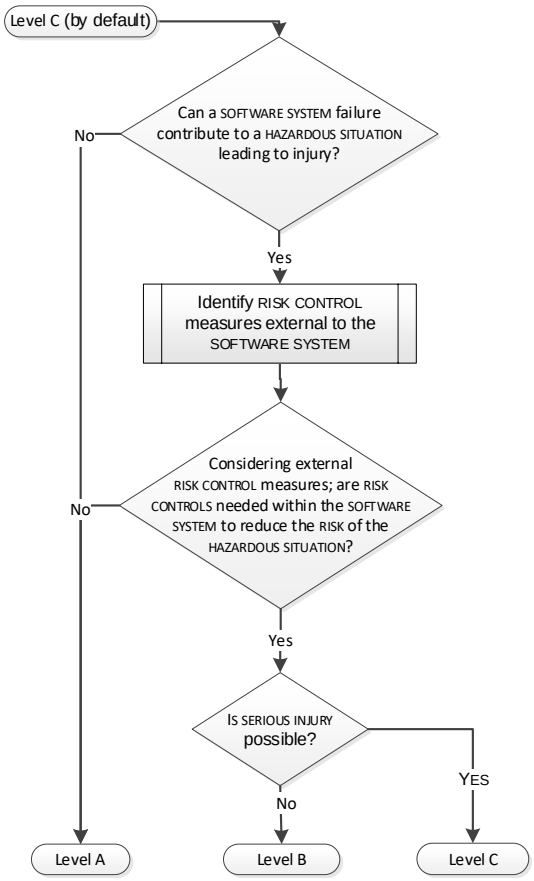


Figure 3 – Assigning software process rigor level

The SOFTWARE SYSTEM is assigned software process rigor level A if:

– a SOFTWARE SYSTEM failure cannot contribute to a HAZARDOUS SITUATION leading to injury or death.

Identify RISK CONTROL measures external to the SOFTWARE SYSTEM.

The SOFTWARE SYSTEM is assigned software process rigor level A if:

– Considering external RISK CONTROL measures, no RISK CONTROLS are needed within the SOFTWARE SYSTEM to reduce the risk of the HAZARDOUS SITUATIONHAZARDOUS SITUATION.

The SOFTWARE SYSTEM is assigned software process rigor level B if:

– a SOFTWARE SYSTEM failure can contribute to a HAZARDOUS SITUATION which results in non-SERIOUS INJURY.

Otherwise, software process rigor level C shall be assigned.

For a SOFTWARE SYSTEM initially determined to be software process rigor level B or C, the MANUFACTURER may implement additional RISK CONTROL measures external to the SOFTWARE SYSTEM (including revising the SYSTEM ARCHITECTURE containing the SOFTWARE SYSTEM) and subsequently assign a new software process rigor level to the SOFTWARE SYSTEM.

NOTE 3 A SOFTWARE SYSTEM failure could occur at the SOFTWARE SYSTEM or SOFTWARE ITEM level. Failures could be software defects, but could also include requirements failures, or failures in human factors.

NOTE 4 External RISK CONTROL measures can reduce the probability that a software failure will cause HARM and/or the severity of HARM.

NOTE 5 Software implementing a RISK CONTROL measure can fail; such a failure can contribute to a HAZARDOUS SITUATION and lead to HARM, including the HARM the RISK control measure was intended to prevent (see 7.2.2 b)).

NOTE 6 External RISK CONTROL measures can be hardware, an independent SOFTWARE SYSTEM, health care procedures, or other means to minimize that software can contribute to a HAZARDOUS SITUATION.

c) The MANUFACTURER shall document the software process rigor level assigned to each SOFTWARE SYSTEM in the RISK MANAGEMENT FILE.

d) When a SOFTWARE SYSTEM is decomposed into SOFTWARE ITEMS, and when a SOFTWARE ITEM is decomposed into further SOFTWARE ITEMS, such SOFTWARE ITEMS shall inherit the software process rigor level of the original SOFTWARE ITEM (or SOFTWARE SYSTEM) unless the MANUFACTURER documents a rationale for classification into a different software process rigor level (software process rigor level assigned according to 4.4.3 b), replacing "SOFTWARE SYSTEM" by "SOFTWARE ITEM"). Such a rationale shall explain how the new SOFTWARE ITEMS are segregated so that they can be classified separately.

e) The MANUFACTURER shall document the software process rigor level of each SOFTWARE ITEM if that process rigor level is different from the process rigor level of the SOFTWARE ITEM from which it was created by decomposition.

f) The MANUFACTURER, when applying this document to a group of SOFTWARE ITEMS, shall use the PROCESSES and TASKS which are required by the process rigor level of the highest-classified SOFTWARE ITEM in the group, unless the MANUFACTURER documents in the RISK MANAGEMENT FILE a rationale for using a lower classification.

4.5 * LEGACY SOFTWARE

4.5.1 General

The MANUFACTURER of LEGACY SOFTWARE may consider this software conformant to this document upon meeting the requirements given in 4.1, 4.2, 4.4, and 4.5.

4.5.2 RISK MANAGEMENT ACTIVITIES

RISK MANAGEMENT ACTIVITIES shall also consider SECURITY and USABILITY.

The MANUFACTURER shall:

a) assess any feedback from internal or external sources, including post-production information, about LEGACY SOFTWARE regarding incidents and/or near incidents, both from inside its own organization and/or from users, and

b) perform RISK MANAGEMENT ACTIVITIES associated with continued use of the LEGACY SOFTWARE, considering the following aspects:

- integration of the LEGACY SOFTWARE in the overall ARCHITECTURE;
- continuing validity of RISK CONTROL measures, implemented as part of the LEGACY SOFTWARE;
- identification of HAZARDOUS SITUATIONS associated with the continued use of the LEGACY SOFTWARE;
- identification of potential causes of the LEGACY SOFTWARE contributing to a HAZARDOUS SITUATION; and
- definition of RISK CONTROL measures for each potential cause of the LEGACY SOFTWARE contributing to a HAZARDOUS SITUATION.

NOTE Incidents are events that cause, or have the potential to cause, unexpected or unwanted effects involving SAFETY.

4.5.3 Gap analysis

Based on the software process rigor level of the LEGACY SOFTWARE (see 4.4.3), the MANUFACTURER shall perform a gap analysis of available DELIVERABLES against those required according to 5.2, 5.3, 5.7, and Clause 7.

a) The MANUFACTURER shall assess the continuing validity of available DELIVERABLES.

b) Where gaps are identified, the MANUFACTURER shall EVALUATE the potential reduction in RISK resulting from the generation of the missing DELIVERABLES and associated ACTIVITIES.

c) Based on this EVALUATION, the MANUFACTURER shall determine the DELIVERABLES to be created and associated ACTIVITIES to be performed. The minimum DELIVERABLE shall be software requirements (see 5.2.2), including software requirements for RISK CONTROL measures (see 5.2.3), and SOFTWARE SYSTEM test records (see 5.7.5).

NOTE Such gap analysis can assure that RISK CONTROL measures, implemented in LEGACY SOFTWARE, are included in the software requirements.

4.5.4 Gap closure activities

a) The MANUFACTURER shall establish and execute a plan to generate the identified DELIVERABLES. Where available, objective evidence can be used to generate required DELIVERABLES without performing ACTIVITIES required by 5.2, 5.3, 5.7, and Clause 7.

NOTE A plan on how to address the identified gaps can be included in a SOFTWARE MAINTENANCE plan (see 6.1).

b) The plan shall address the use of the problem resolution PROCESS for handling problems detected in the LEGACY SOFTWARE and DELIVERABLES in accordance with Clause 9.

c) Changes to the LEGACY SOFTWARE shall be performed in accordance with this document.

4.5.5 Rationale for use of LEGACY SOFTWARE

The MANUFACTURER shall document the VERSION of the LEGACY SOFTWARE together with a rationale for the continued use of the LEGACY SOFTWARE based on the outputs of 4.5.

NOTE Fulfilling 4.5 enables further use of LEGACY SOFTWARE in accordance with this document.

5 Software development PROCESS

5.1 * Software development planning

5.1.1 Software development plan

The MANUFACTURER shall establish a software development plan (or plans) for conducting the ACTIVITIES of the software development PROCESS appropriate to the scope, magnitude, and software process rigor levels of the SOFTWARE SYSTEM to be developed. The SOFTWARE DEVELOPMENT LIFE CYCLE MODEL shall either be fully defined or be referenced in the plan (or plans). The plan shall address the following:

a) the PROCESSES to be used in the development of the SOFTWARE SYSTEM;

NOTE 1 The software development plan can reference existing PROCESSES or define new ones.

b) the DELIVERABLES (includes documentation) of the ACTIVITIES and TASKS;

c) TRACEABILITY between SYSTEM requirements, software requirements, SOFTWARE SYSTEM test, and RISK CONTROL measures implemented in software;

d) software configuration and change management, including SOUP CONFIGURATION ITEMS and software used to support development; and

e) software problem resolution for handling problems detected in HEALTH SOFTWARE, DELIVERABLES and ACTIVITIES at each stage of the life cycle.

f) If appropriate, an algorithm change protocol for the delineation of data and procedures to be followed so that algorithm modifications meet intent and remain safe and effective after the modification. Components of an algorithm change protocol to consider include data management, algorithm re-training, algorithm performance evaluation, and algorithm update procedures.

[Level A, B, C]

NOTE 2 The SOFTWARE DEVELOPMENT LIFE CYCLE MODEL can identify different elements (PROCESSES, ACTIVITIES, TASKS and DELIVERABLES) for different SOFTWARE ITEMS according to the software process rigor level of each SOFTWARE ITEM of the SOFTWARE SYSTEM.

NOTE 3 These ACTIVITIES and TASKS can overlap or interact and can be performed iteratively or recursively. It is not the intent of this document to prescribe a specific life cycle model.

NOTE 4 Other PROCESSES are described in this document separately from the development PROCESS. This does not imply that the other PROCESSES are to be implemented as separate ACTIVITIES and TASKS. The ACTIVITIES and TASKS of the other PROCESSES can be integrated into the development PROCESS.

NOTE 5 The software development plan can be integrated in an overall SYSTEM development plan.

5.1.2 Software development plan updates

The MANUFACTURER shall update the plan as development proceeds as appropriate. [Level A, B, C]

5.1.3 Software development plan reference to SYSTEM design and development

a) The MANUFACTURER shall reference SYSTEM requirements in the software development plan as inputs for software development.

b) The MANUFACTURER shall include or reference procedures for coordinating the software development with the SYSTEM development, necessary to satisfy 4.1 (such as SYSTEM integration, VERIFICATION, and VALIDATION) in the software development plan.

[Level A, B, C]

NOTE There might not be a difference between SOFTWARE SYSTEM requirements and SYSTEM requirements if the SOFTWARE SYSTEM is a stand-alone SYSTEM (software-only product).

5.1.4 Software development standards, methods and tools planning

The MANUFACTURER shall include or reference in the software development plan the following items associated with the development of SOFTWARE ITEMS:

a) standards;

b) methods; and

c) tools.

[Level C]

5.1.5 Software integration and integration testing planning

The MANUFACTURER shall include or reference in the software development plan a plan to integrate the SOFTWARE ITEMS (including SOUP) and perform testing during integration.

[Level B, C]

NOTE 1 It is acceptable to combine integration testing and SOFTWARE SYSTEM testing into a single plan and set of ACTIVITIES.

NOTE 2 See 5.6.

5.1.6 Software VERIFICATION planning

The MANUFACTURER shall include or reference in the software development plan the following VERIFICATION information:

a) DELIVERABLES requiring VERIFICATION;

b) the required VERIFICATION TASKS for each life cycle ACTIVITY;

c) milestones at which the DELIVERABLES are VERIFIED; and

d) the acceptance criteria for VERIFICATION of the DELIVERABLES.

[Level A, B, C]

5.1.7 Software RISK MANAGEMENT planning

The MANUFACTURER shall include or reference in the software development plan a plan to conduct the ACTIVITIES and TASKS of the software RISK MANAGEMENT PROCESS, including the management of RISKS relating to SOUP. [Level A, B, C]

NOTE See Clause 7.

5.1.8 Documentation planning

The MANUFACTURER shall include or reference in the software development plan information about the documents to be produced during the software development life cycle. For each identified document or type of document, the following information shall be included or referenced:

a) title, name or naming convention;

b) purpose; and

c) procedures and responsibilities for development, review, approval and modification.

[Level A, B, C]

NOTE See Clause 8 for consideration of configuration management of documentation.

5.1.9 Software configuration management planning

The MANUFACTURER shall include or reference software configuration management information in the software development plan. The software configuration management information shall include or reference:

a) the classes, types, categories or lists of items to be controlled;

b) the software configuration management ACTIVITIES and TASKS;

c) the organization(s) responsible for performing software configuration management ACTIVITIES;

d) their relationship with other organizations, such as software development or maintenance;

e) when the items are to be placed under configuration control;

f) when the problem resolution PROCESS is to be used; and

g) when the formal change control PROCESS will be used.

[Level A, B, C]

NOTE See Clause 8.

5.1.10 Supporting items to be controlled

The items to be controlled shall include tools, items or settings used to develop HEALTH SOFTWARE, which could impact HEALTH SOFTWARE. [Level B, C]

NOTE 1 Examples of such items include compiler/assembler VERSIONS, make files, batch files, and specific environment settings.

NOTE 2 See Clause 8.

5.1.11 Software CONFIGURATION ITEM control before VERIFICATION

The MANUFACTURER shall plan to place CONFIGURATION ITEMS under configuration management control before they are VERIFIED. [Level B, C]

5.2 * Software requirements analysis

5.2.1 Define and document software requirements from SYSTEM requirements

The MANUFACTURER shall define and document software requirements from the SYSTEM level requirements for each SOFTWARE SYSTEM of the product. [Level A, B, C]

NOTE There might not be a difference between SOFTWARE SYSTEM requirements and SYSTEM requirements if the SOFTWARE SYSTEM is a stand-alone SYSTEM (software-only product).

5.2.2 Software requirements content

As appropriate to HEALTH SOFTWARE, the MANUFACTURER shall include in the software requirements:

a) functional and capability requirements, constraints (restrictions for the software design);

NOTE 1 Examples include

- performance (e.g. time to PROCESS transactions, data throughput rate, minimum time to detect and PROCESS an ACTIVITY),
- load (e.g. number of concurrent users, number of concurrent transactions), physical characteristics (e.g. code language, platform, operating system),
- computing environment (e.g. hardware, memory size, processing unit, time zone, network infrastructure) under which the software is to perform, and
- need for compatibility with upgrades or multiple SOUP or other device VERSIONS.

b) SOFTWARE SYSTEM inputs and outputs;

NOTE 2 Examples include

- data characteristics (e.g. numerical, alpha-numeric, format),
- ranges,
- limits, and
- defaults.

c) interfaces between the SOFTWARE SYSTEM and other SYSTEMS;

d) software-driven alarms, warnings, and operator messages;

e) SECURITY capabilities that might be considered in the software and resolved into SECURITY requirements;

NOTE 3 Examples include

- those related to the compromise of sensitive information or information related to SAFETY,
- authentication,
- authorization,

– audit trail,
– communication integrity, and
– malware protection.

NOTE 4 Information regarding SECURITY capabilities can be found in IEC TR 80001-2-8 [13]. Additionally, information regarding SECURITY requirements can be found in ISO/IEC 27002 [35] and ISO 27799 [19].

f) user interface requirements implemented by software;

NOTE 5 Examples include those related to
– support for manual operations,
– human-equipment interactions,
– constraints on personnel, and
– areas needing concentrated human attention.

NOTE 6 Information regarding user interface requirements can be found in IEC 62366-1 [4].

g) data definition and database requirements;

h) deployment, installation and acceptance requirements of the delivered HEALTH SOFTWARE at the operation and maintenance site or sites;

i) requirements related to methods of operation and maintenance;

j) requirements related to IT-network aspects;

NOTE 7 Examples include those related to
– networked alarms, warnings, and operator messages,
– network protocols, and
– handling of unavailability of network services.

k) user maintenance requirements; and

NOTE 8 Examples include those related to
– backing up health data, and
– installation of updates;

l) regulatory requirements impacting functionality or characteristics of the software.

NOTE 9 Examples include
– localization,
– labelling, and
– data privacy.

[Level A, B, C]

NOTE 10 The requirements in a) through l) can overlap.

NOTE 11 All of these requirements might not be available at the beginning of the software development.

NOTE 12 Among others, ISO/IEC 25010 [32] provides information on quality characteristics that can be useful in defining software requirements.

5.2.3 Include RISK CONTROL measures in software requirements

The MANUFACTURER shall include RISK CONTROL measures to be implemented in software in the requirements as appropriate to the HEALTH SOFTWARE. [Level B, C]

NOTE 1 These requirements might not be available at the beginning of the software development and can change as the software is designed and RISK CONTROL measures are further defined.

NOTE 2 These requirements can overlap with requirements according to 5.2.2.

NOTE 3 See also 7.2.2 for more details.

5.2.4 Re-EVALUATE SYSTEM RISK ANALYSIS

The MANUFACTURER shall re-EVALUATE the SYSTEM RISK ANALYSIS when software requirements are established and update it as appropriate. [Level A, B, C]

5.2.5 Update requirements

The MANUFACTURER shall ensure that existing requirements, including SYSTEM requirements, are re-EVALUATED and updated as appropriate as a result of the software requirements analysis ACTIVITY. [Level A, B, C]

5.2.6 Verify software requirements

The MANUFACTURER shall verify and document that the software requirements

a) implement SYSTEM requirements including those relating to RISK CONTROL,

b) do not contradict one another,

c) are expressed in terms that avoid ambiguity,

d) are stated in terms that permit establishment of test criteria and performance of tests,

e) can be uniquely identified, and

f) are traceable to SYSTEM requirements or other source.

[Level A, B, C]

NOTE 1 This document does not have a requirement for the use of a formal specification language.

NOTE 2 See Clause B.2.

5.3 * Software ARCHITECTURAL design

5.3.1 Transform software requirements into an ARCHITECTURE

The MANUFACTURER shall transform the requirements for HEALTH SOFTWARE into a documented ARCHITECTURE that describes the software's structure and identifies the SOFTWARE ITEMS. [Level B, C]

5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS

The MANUFACTURER shall develop and document the interfaces between the SOFTWARE ITEMS and the components external to the SOFTWARE ITEMS (both software and hardware), and between the SOFTWARE ITEMS. [Level B, C]

5.3.3 Specify functional and performance requirements of SOUP item

If a SOFTWARE ITEM is identified as SOUP, the MANUFACTURER shall specify functional and performance requirements for the SOUP item that are necessary for its proper integration into the HEALTH SOFTWARE. [Level B, C]

5.3.4 Specify SYSTEM hardware and software required by SOUP item

If a SOFTWARE ITEM is identified as SOUP, the MANUFACTURER shall specify the SYSTEM hardware and software necessary to support the proper operation of the SOUP item. [Level B, C]

NOTE Examples include processor type and speed, memory type and size, SYSTEM software type, communication and display software requirements.

5.3.5 Identify segregation necessary for RISK CONTROL

The MANUFACTURER shall identify any segregation between SOFTWARE ITEMS that is necessary for RISK CONTROL, and state how to ensure that such segregation is effective. [Level C]

NOTE An example of segregation is to have SOFTWARE ITEMS execute on different processors. The EFFECTIVENESS of the segregation can be ensured by having no shared resources between the processors. Other means of segregation can be applied when EFFECTIVENESS can be ensured by the software ARCHITECTURE design (see Clause B.2 Guidance Software process rigor level and Software ARCHITECTURAL design).

5.3.6 Verify software ARCHITECTURE

The MANUFACTURER shall verify and document that

a) the ARCHITECTURE of the software implements SYSTEM and software requirements including those relating to RISK CONTROL,

b) the software ARCHITECTURE is able to support interfaces between SOFTWARE ITEMS and between SOFTWARE ITEMS and hardware, and

c) the software and SYSTEM ARCHITECTURE supports proper operation of any SOUP items.

[Level B, C]

NOTE A TRACEABILITY analysis of ARCHITECTURE to software requirements can be used to satisfy requirement a).

5.4 * Software detailed design

5.4.1 Subdivide software into SOFTWARE UNITS

The MANUFACTURER shall subdivide the software until it is represented by SOFTWARE UNITS. [Level B, C]

NOTE Some SOFTWARE SYSTEMS are not divided further.

5.4.2 Develop detailed design for each SOFTWARE UNIT

The MANUFACTURER shall document a design with enough detail to allow correct implementation of each SOFTWARE UNIT. [Level C]

5.4.3 Develop detailed design for interfaces

The MANUFACTURER shall document a design for any interfaces between the SOFTWARE UNIT and external components (hardware or software), as well as any interfaces between SOFTWARE UNITS, detailed enough to implement each SOFTWARE UNIT and its interfaces correctly. [Level C]

5.4.4 Verify detailed design

The MANUFACTURER shall verify and document that the software detailed design

a) implements the software ARCHITECTURE, and

b) is free from contradiction with the software ARCHITECTURE.

[Level C]

NOTE 1 It is acceptable to use a TRACEABILITY analysis of ARCHITECTURE to software detailed design to satisfy requirement a).

NOTE 2 An example to satisfy requirement b) is a documented peer review to verify if segregation necessary for RISK CONTROL is not contradicted by the detailed design.

5.5 * SOFTWARE UNIT implementation

5.5.1 Implement each SOFTWARE UNIT

The MANUFACTURER shall implement each SOFTWARE UNIT. [Level A, B, C]

5.5.2 Establish SOFTWARE UNIT VERIFICATION PROCESS

The MANUFACTURER shall establish strategies, methods and procedures for verifying the SOFTWARE UNITS. Where VERIFICATION is done by testing, the test procedures shall be EVALUATED for adequacy. [Level B, C]

NOTE VERIFICATION can include static VERIFICATION (e.g., static code analysis, inspection) as well as dynamic VERIFICATION (such as testing and experimentation).

5.5.3 SOFTWARE UNIT acceptance criteria

The MANUFACTURER shall establish acceptance criteria for SOFTWARE UNITS prior to integration into larger SOFTWARE ITEMS as appropriate and ensure that SOFTWARE UNITS meet acceptance criteria. [Level B, C]

NOTE Examples of acceptance criteria are:

– Does the software code implement requirements including RISK CONTROL measures?

– Is the software code free from inconsistencies with the interface design of the SOFTWARE UNIT?

– Does the software code conform to programming procedures or coding standards?

5.5.4 Additional SOFTWARE UNIT acceptance criteria

When present in the design, the MANUFACTURER shall include additional acceptance criteria as appropriate for:

a) proper event sequence;

b) data and control flow;

c) planned resource allocation;

d) fault handling (error definition, isolation, and recovery);

e) initialisation of variables;

f) self-diagnostics;

g) memory management and memory overflows; and

h) boundary conditions.

[Level C]

5.5.5 SOFTWARE UNIT VERIFICATION

The MANUFACTURER shall perform the SOFTWARE UNIT VERIFICATION and document the results. [Level B, C]

5.6 * Software integration and integration testing

5.6.1 Integrate SOFTWARE UNITS

The MANUFACTURER shall integrate the SOFTWARE UNITS in accordance with the integration plan (see 5.1.5). [Level B, C]

5.6.2 Verify software integration

The MANUFACTURER shall verify that the SOFTWARE UNITS have been integrated into SOFTWARE ITEMS and/or the SOFTWARE SYSTEM in accordance with the integration plan (see 5.1.5) and retain records of the evidence of such VERIFICATION.

[Level B, C]

NOTE The aim of this VERIFICATION is only to ensure that the integration has been done according to the plan. This VERIFICATION is most likely implemented by some form of inspection.

5.6.3 Software integration testing

The MANUFACTURER shall test the integrated SOFTWARE ITEMS in accordance with the integration plan (see 5.1.5) and document the results. [Level B, C]

5.6.4 Software integration testing content

The MANUFACTURER shall address, in the software integration testing, whether the integrated SOFTWARE ITEM performs as intended.

[Level B, C]

NOTE 1 Examples to be considered are

– the required functionality of the software;

– implementation of RISK CONTROL measures;

– specified timing and other behaviour;

– specified functioning of internal and external interfaces; and

– testing under abnormal conditions including foreseeable misuse.

NOTE 2 It is acceptable to combine integration testing and SOFTWARE SYSTEM testing into a single plan and set of ACTIVITIES.

5.6.5 EVALUATE software integration test procedures

The MANUFACTURER shall EVALUATE the integration test procedures for adequacy. [Level B, C]

5.6.6 Conduct regression tests

When SOFTWARE ITEMS are integrated, the MANUFACTURER shall conduct REGRESSION TESTING appropriate to demonstrate that defects have not been introduced into previously integrated software. [Level B, C]

5.6.7 Integration test record contents

The MANUFACTURER shall

a) document the test result (pass/fail and a list of anomalies);

b) retain sufficient records to permit the test to be repeated; and

c) document the identity of the person(s) responsible for executing the test as well as recording and review and approval of the test results.

[Level B, C]

NOTE Requirement b) could be implemented by retaining, for example,

– test case specifications showing required actions and expected results,

– records of the equipment, and

– records of the test environment (including software tools) used for test.

5.6.8 Use software problem resolution PROCESS

The MANUFACTURER shall enter ANOMALIES found during software integration and integration testing into a software problem resolution PROCESS. [Level B, C]

NOTE See Clause 9.

5.7 * SOFTWARE SYSTEM testing

5.7.1 Establish tests for software requirements

The MANUFACTURER shall establish and perform a set of tests, expressed as input stimuli, expected outcomes, pass/fail criteria and procedures, for conducting SOFTWARE SYSTEM testing, such that all software requirements are covered. [Level A, B, C]

NOTE 1 It is acceptable to combine integration testing and SOFTWARE SYSTEM testing into a single plan and set of ACTIVITIES. It is also acceptable to test software requirements in earlier phases.

NOTE 2 Not only separate tests for each requirement, but also tests of combinations of requirements can be performed, especially if dependencies between requirements exist.

5.7.2 Use software problem resolution PROCESS

The MANUFACTURER shall enter ANOMALIES found during SOFTWARE SYSTEM testing into a software problem resolution PROCESS. [Level A, B, C]

5.7.3 Retest after changes

When changes are made to the HEALTH SOFTWARE during SOFTWARE SYSTEM testing, the MANUFACTURER shall:

a) repeat tests, perform modified tests or perform additional tests, as appropriate, to verify the EFFECTIVENESS of the change in correcting the problem;

b) conduct VERIFICATION appropriate to demonstrate that unintended side effects have not been introduced; and

c) perform relevant RISK MANAGEMENT ACTIVITIES as defined in 7.4.

[Level A, B, C]

5.7.4 Evaluate SOFTWARE SYSTEM testing

The MANUFACTURER shall EVALUATE the adequacy of VERIFICATION strategies and test procedures.

The MANUFACTURER shall verify that:

a) all software requirements have been tested or otherwise VERIFIED;

b) the TRACEABILITY between software requirements and tests or other VERIFICATION is recorded; and

c) test results meet the required pass/fail criteria.

[Level A, B, C]

5.7.5 SOFTWARE SYSTEM test record contents

In order to support the repeatability of tests, the MANUFACTURER shall document:

a) a reference to test case procedures showing required actions and expected results;

b) the test result (pass/fail and a list of ANOMALIES);

c) the VERSION of software tested;

d) relevant hardware and software test configurations;

e) relevant test tools;

f) the date of the test;

g) the identity of the person(s) responsible for executing the test and recording and review and approval of the test results; and

h) sufficient records to permit the test to be repeated.

[Level A, B, C]

NOTE Requirement h) could be implemented by retaining, for example,

– test case specifications showing required actions and expected results,

– records of the equipment, and

– records of the test environment (including software tools) used for test.

5.8 * Software release

5.8.1 Ensure software VERIFICATION is complete

The MANUFACTURER shall ensure that all software VERIFICATION ACTIVITIES have been completed and the results have been EVALUATED before the software is released for INTENDED USE. [Level A, B, C]

5.8.2 Document known residual ANOMALIES

The MANUFACTURER shall document all known residual ANOMALIES. [Level A, B, C]

5.8.3 EVALUATE known residual ANOMALIES

The MANUFACTURER shall ensure that all known residual ANOMALIES have been EVALUATED to ensure that they do not contribute to an unacceptable RISK. [Level B, C]

5.8.4 Document released VERSIONS

The MANUFACTURER shall document the VERSION of the HEALTH SOFTWARE that is being released for INTENDED USE. [Level A, B, C]

5.8.5 Document how released software was created

The MANUFACTURER shall document the procedure and environment used to create the software released for INTENDED USE. [Level B, C]

5.8.6 Ensure ACTIVITIES and TASKS are complete

The MANUFACTURER shall ensure that all software development plan (or maintenance plan) ACTIVITIES and TASKS are complete along with the associated documentation. [Level B, C]

NOTE See 5.1.3 b).

5.8.7 Archive software

The MANUFACTURER shall archive:

a) HEALTH SOFTWARE and CONFIGURATION ITEMS; and

b) the documentation identified in 5.1.8.

The archival period shall be as specified by relevant regulatory requirements or as defined by the MANUFACTURER, whichever is the longer of the two. [Level A, B, C]

5.8.8 Assure reliable delivery of released software

The MANUFACTURER shall establish procedures to ensure that HEALTH SOFTWARE released for INTENDED USE can be reliably delivered to the point of use without corruption or unauthorised change. This includes delivery via the internet. These procedures shall address the production and handling of HEALTH SOFTWARE.

[Level A, B, C]

6 SOFTWARE MAINTENANCE PROCESS

6.1 * Establish SOFTWARE MAINTENANCE plan

The MANUFACTURER shall establish a SOFTWARE MAINTENANCE plan (or plans) for conducting the ACTIVITIES and TASKS of the maintenance PROCESS.

The plan shall address the following:

a) procedures for addressing feedback arising after the release for INTENDED USE of the HEALTH SOFTWARE, including:
   - receiving;
   - documenting;
   - evaluating;
   - resolving; and
   - tracking;

NOTE 1 Including feedback on cybersecurity vulnerabilities. The cybersecurity vulnerabilities apply to all of HEALTH SOFTWARE and not just SOUP.

b) criteria for determining whether feedback indicates a problem;

c) use of the software RISK MANAGEMENT PROCESS (Clause 7);

d) use of the software problem resolution PROCESS (Clause 9) for analysing and resolving problems arising after release for INTENDED USE of HEALTH SOFTWARE;

e) use of the software configuration management PROCESS (Clause 8) for managing modifications to the SOFTWARE SYSTEM released for INTENDED USE;

f) with regard to SOUP, procedures to EVALUATE and implement:
   - upgrades;
   - bug fixes;
   - patches; and
   - obsolescence;

g) a software retirement/decommissioning strategy, as appropriate, including:
   - data management and information security; and
   - termination of software access.

NOTE 2 See also 7.1.3.

[Level A, B, C]

6.2 * Problem and modification analysis

6.2.1 Document and EVALUATE feedback

6.2.1.1 Monitor feedback

The MANUFACTURER shall monitor feedback on HEALTH SOFTWARE released for INTENDED USE. [Level A, B, C]

6.2.1.2 Document and EVALUATE feedback

Feedback shall be documented and EVALUATED to determine whether a problem exists in a HEALTH SOFTWARE released for INTENDED USE. Any such problem shall be recorded as a PROBLEM REPORT (see Clause 9). PROBLEM REPORTS shall include actual or potential adverse events, and deviations from specifications. [Level A, B, C]

6.2.1.3 Evaluate PROBLEM REPORT'S effects on SAFETY

Each PROBLEM REPORT shall be EVALUATED to determine how it affects the SAFETY of HEALTH SOFTWARE released for INTENDED USE (see 9.2) and whether a change to that software is needed to address the problem. [Level A, B, C]

6.2.2 Use software problem resolution PROCESS

The MANUFACTURER shall use the software problem resolution PROCESS (see Clause 9) to address PROBLEM REPORTS. [Level A, B, C]

NOTE A problem could show that a SOFTWARE SYSTEM or SOFTWARE ITEM has not been placed in the correct software process rigor level. The problem resolution PROCESS can suggest changes of the software process rigor level. When the PROCESS has been completed, any change of process rigor level in the SOFTWARE SYSTEM or its SOFTWARE ITEMS is made known and documented.

6.2.3 Analyse CHANGE REQUESTS

In addition to the analysis required by Clause 9, the MANUFACTURER shall analyse each CHANGE REQUEST for its effect on the HEALTH SOFTWARE released for INTENDED USE, and SYSTEMS with which it interfaces. [Level A, B, C]

6.2.4 CHANGE REQUEST approval

The MANUFACTURER shall EVALUATE and approve CHANGE REQUESTS which modify HEALTH SOFTWARE released for INTENDED USE. [Level A, B, C]

6.2.5 Communicate to users

The MANUFACTURER shall identify the approved CHANGE REQUESTS that affect HEALTH SOFTWARE released for INTENDED USE.

The MANUFACTURER shall inform users about:

a) any problem in HEALTH SOFTWARE released for INTENDED USE and the consequences of continued unchanged use; and

b) the nature of any available changes to HEALTH SOFTWARE released for INTENDED USE and how to obtain and install the changes.

[Level A, B, C]

6.3 * Modification implementation

6.3.1 Use established PROCESS to implement modification

The MANUFACTURER shall identify and perform any Clause 5 ACTIVITIES that need to be applied as a result of the change (see 8.2). [Level A, B, C]

NOTE For requirements relating to RISK MANAGEMENT of software changes, see 7.4.

6.3.2 Re-release modified SOFTWARE SYSTEM

The MANUFACTURER shall release modifications according to 5.8. [Level A, B, C]

NOTE Modifications can be released as part of a full re-release of a SOFTWARE SYSTEM or as a modification kit comprising changed SOFTWARE ITEMS and the necessary tools to install the changes as modifications to an existing SOFTWARE SYSTEM.

7 * Software RISK MANAGEMENT PROCESS

7.1 * Analysis of software contributing to HAZARDOUS SITUATIONS

7.1.1 Identify SOFTWARE ITEMS that could contribute to a HAZARDOUS SITUATION

The MANUFACTURER shall identify SOFTWARE ITEMS that could contribute to a HAZARDOUS SITUATION (see 4.2 and 4.4.2). [Level B, C]

NOTE The HAZARDOUS SITUATION could be the direct result of software failure or the result of the failure of a RISK CONTROL measure that is implemented in software.

7.1.2 Identify potential causes of contribution to a HAZARDOUS SITUATION

The MANUFACTURER shall consider potential internal and external causes including, as appropriate, the SOFTWARE ITEM(S) that could contribute to a HAZARDOUS SITUATION (see 4.4.2).

The MANUFACTURER shall consider the following minimum list of potential causes, as appropriate:

a) incorrect or incomplete specification of functionality;

b) software defects in the identified SOFTWARE ITEM functionality;

c) failure or unexpected results from SOUP;

d) failures external to the HEALTH SOFTWARE that could result in unpredictable HEALTH SOFTWARE operation, including:

   – hardware and software interface failures,

   – hardware failures, or

   – software failures external to the HEALTH SOFTWARE;

e) defects that can be introduced by the used software environment, including tools and libraries; defects that can be introduced by the software development environment;

   NOTE 1 For example, this can be done by review of published bug-list from vendors.

f) defects that can be based on the selected programming technology, including programming language issues;

   NOTE 2 Examples: For C++ this could be the use of dynamic memory allocation and diamond multiple inheritance. For Java it could be unpredictable timing due to garbage collection.

g) reasonably foreseeable misuse (including misuse of data);

h) cyberattacks and SECURITY breaches (e,g., denial of service, malware hacking, unauthorized access via user interface);

NOTE 3 See Annex B of IEC TR 80002-1:2009 [14] and ANSI/AAMI SW91 [39] for examples of categories of defects or causes contributing to HAZARDOUS SITUATIONS.

i) use errors, or use related risks.

[Level B, C]

7.1.3 EVALUATE published SOUP ANOMALY lists

If failure or unexpected results from SOUP is a potential cause of the SOFTWARE ITEM contributing to a HAZARDOUS SITUATION, the MANUFACTURER shall EVALUATE as a minimum any ANOMALY list published by the supplier of the SOUP item relevant to the VERSION of the SOUP item used in the SYSTEM to determine if any of the known ANOMALIES result in a sequence of events that could result in a HAZARDOUS SITUATION. [Level B, C]

7.1.4 Document potential causes

The MANUFACTURER shall document potential causes of the SOFTWARE ITEM contributing to a HAZARDOUS SITUATION. [Level B, C]

7.2 RISK CONTROL measures

7.2.1 Define RISK CONTROL measures

When RISK reduction is required, the MANUFACTURER shall define and document RISK CONTROL measures for each potential cause of the SOFTWARE ITEM that can contribute to a HAZARDOUS SITUATION in accordance with 4.2. [Level B, C]

NOTE The RISK CONTROL measures can be implemented in hardware, software, the working environment or user instruction. Furthermore, RISK CONTROL measures can include application of specific methods and activities in the software development life cycle.

7.2.2 RISK CONTROL measures implemented in software

If a RISK CONTROL measure is implemented as part of the functions of a SOFTWARE ITEM, the MANUFACTURER shall:

a) include the RISK CONTROL measure in the SOFTWARE SYSTEM requirements;

b) assign, to each SOFTWARE ITEM that contributes to the implementation of a RISK CONTROL measure, a software process rigor level based on the RISK that the RISK CONTROL measure is controlling (see 4.4.3 a)); and

c) develop the SOFTWARE ITEM in accordance with Clause 5.

[Level B, C]

7.3 VERIFICATION of RISK CONTROL measures

7.3.1 Verify RISK CONTROL measures

The implementation of each RISK CONTROL measure documented in 7.2 shall be VERIFIED, and this VERIFICATION shall be documented. The MANUFACTURER shall review the RISK CONTROL measure and determine if it could result in a new HAZARDOUS SITUATION. [Level B, C]

7.3.2 Document TRACEABILITY

The MANUFACTURER shall document TRACEABILITY:

a) from the HAZARD to the HAZARDOUS SITUATION to the SOFTWARE ITEM;

b) from the SOFTWARE ITEM to the specific software cause;

c) from the software cause to the RISK CONTROL measure; and

d) from the RISK CONTROL measure to the VERIFICATION of the RISK CONTROL measure.

[Level B, C]

7.4 RISK MANAGEMENT of software changes

7.4.1 Analyse changes to HEALTH SOFTWARE with respect to RISK

The MANUFACTURER shall analyse changes to HEALTH SOFTWARE (including SOUP) to determine whether:

a) the changes introduce additional potential causes contributing to a HAZARDOUS SITUATION; and

b) additional RISK CONTROL measures are required.

[Level A, B, C]

7.4.2 Analyse impact of software changes on existing RISK CONTROL measures

The MANUFACTURER shall analyse changes to the software, including changes to SOUP, to determine whether the software modification could affect the existing RISK CONTROL measures. [Level B, C]

7.4.3 Perform RISK MANAGEMENT ACTIVITIES based on analyses

The MANUFACTURER shall perform relevant RISK MANAGEMENT ACTIVITIES defined in **Error! Reference source not found.**, 7.2 and 7.3 based on these analyses. [Level B, C]

8 * Software configuration management PROCESS

8.1 * Configuration identification

8.1.1 Establish means to identify CONFIGURATION ITEMS

The MANUFACTURER shall establish a scheme for the unique identification of CONFIGURATION ITEMS and their VERSIONS to be controlled according to the development and configuration planning specified in 5.1. [Level A, B, C]

8.1.2 Identify SOUP

The MANUFACTURER shall document for each SOUP CONFIGURATION ITEM being used, including standard libraries:

a) the title;

b) the MANUFACTURER; and

c) the unique SOUP designator.

[Level A, B, C]

NOTE The unique SOUP designator could be, for example, a VERSION, a release date, a patch number or an upgrade designation.

8.1.3 Identify SYSTEM configuration documentation

The MANUFACTURER shall document the set of CONFIGURATION ITEMS and their VERSIONS that comprise the SOFTWARE SYSTEM configuration. [Level A, B, C]

8.2 * Change control

8.2.1 Change CONFIGURATION ITEMS

The MANUFACTURER shall change CONFIGURATION ITEMS identified to be controlled according to 8.1 only in response to an approved CHANGE REQUEST. [Level A, B, C]

NOTE 1 This requirement only means approval of a change precedes its implementation. The decision to approve a CHANGE REQUEST can be integral to the change control PROCESS or part of another PROCESS.

NOTE 2 Different acceptance PROCESSES can be used for CHANGE REQUESTS at different stages of the life cycle, as stated in plans – see 5.1.1 d) and 6.1 e).

NOTE 3 See Figure B.8.

8.2.2 Implement changes

The MANUFACTURER shall implement the change as specified in the CHANGE REQUEST. The MANUFACTURER shall identify and perform any ACTIVITY that needs to be repeated as a result of the change, including changes to the software process rigor level of SOFTWARE SYSTEMS and SOFTWARE ITEMS. [Level A, B, C]

NOTE This requirement states how the change is implemented to achieve adequate change control. It does not imply that the implementation is an integral part of the change control PROCESS. For more information see 5.1.1 d) and 6.1 e).

8.2.3 Verify changes

The MANUFACTURER shall verify the change, including repeating any VERIFICATION that has been invalidated by the change and taking into account 5.7.3 and 9.7. [Level A, B, C]

NOTE This requirement only means that changes be verified. It does not imply that VERIFICATION is an integral part of the change control PROCESS. For more information see 5.1.1 d) and 6.1 e).

8.2.4 Provide means for TRACEABILITY of change

The MANUFACTURER shall maintain records of the relationships and dependencies between:

a) CHANGE REQUEST;

b) relevant PROBLEM REPORT; and

c) approval of the CHANGE REQUEST.

[Level A, B, C]

8.3 * Configuration status accounting

The MANUFACTURER shall retain retrievable records of the history of controlled CONFIGURATION ITEMS including SYSTEM configuration. [Level A, B, C]

9 * Software problem resolution PROCESS

9.1 Prepare PROBLEM REPORTS

When the problem resolution PROCESS is activated, as defined in the software development plan (5.1.9 f)), the MANUFACTURER shall prepare a PROBLEM REPORT for each problem detected in HEALTH SOFTWARE. PROBLEM REPORTS shall include a statement of criticality (for example, effect on performance, SAFETY, or SECURITY) as well as other information that can aid in the resolution of the problem (for example, products affected, supported accessories affected).

[Level A, B, C]

NOTE Problems can be discovered before or after release for INTENDED USE, inside the MANUFACTURER'S organization or outside it.

9.2 Investigate the problem

The MANUFACTURER shall:

a) investigate the problem and, if possible, identify the causes;

b) EVALUATE the problem's relevance to SAFETY using the software RISK MANAGEMENT PROCESS (Clause 7);

c) document the outcome of the investigation and EVALUATION; and

d) create a CHANGE REQUEST(S) for actions needed to correct the problem, or document the rationale for taking no action. This means that with appropriate rationale, a problem may not have to be corrected for conformance with the software problem resolution PROCESS.

[Level A, B, C]

9.3 Advise relevant parties

The MANUFACTURER shall advise relevant parties of the existence of the problem, as appropriate.

[Level A, B, C]

NOTE Problems can be discovered before or after release, inside the MANUFACTURER'S organisation or outside it. The MANUFACTURER determines the relevant parties depending on the situation (an example before release: development teams. Examples after release for INTENDED USE include regulators, users, and marketing).

9.4 Use change control PROCESS

The MANUFACTURER shall approve and implement CHANGE REQUESTS observing the requirements of the change control PROCESS (see 8.2). [Level A, B, C]

9.5 Maintain records

The MANUFACTURER shall maintain records of PROBLEM REPORTS and their resolution, including their VERIFICATION.

The MANUFACTURER shall update the RISK MANAGEMENT FILE as appropriate.

[Level A, B, C]

9.6 Analyse problems for trends

The MANUFACTURER shall perform analysis to detect trends in PROBLEM REPORTS. [Level A, B, C]

NOTE See Clause B.2 Guidance for Software problem resolution process.

9.7 Verify software problem resolution

The MANUFACTURER shall verify resolutions to determine whether:

a) problem has been resolved and the PROBLEM REPORT has been closed;

b) adverse trends have been reversed;

c) CHANGE REQUESTS have been implemented; and

d) additional problems have been introduced (see 9.1).

[Level A, B, C]

9.8 Test documentation contents

When testing, retesting or REGRESSION TESTING SOFTWARE ITEMS and SYSTEMS following a change, the MANUFACTURER shall include in the test documentation, as appropriate:

a) a reference to test case procedures showing required actions and expected results;

b) the test result (pass/fail and a list of ANOMALIES);

c) the VERSION of software tested;

d) relevant hardware and software test configurations;

e) relevant test tools;

f) the date of the test;

g) the identity of the person(s) responsible for executing the test and recording and reviewing and approving the test results; and

h) sufficient records to permit the test to be repeated.

[Level A, B, C]

NOTE Requirement h) could be implemented by retaining, for example,

– test case specifications showing required actions and expected results,

– records of the equipment, and

– records of the test environment (including software tools) used for test.

Annex A
(informative)

Rationale for the requirements of this document

A.1 Rationale

The primary requirement of this document is that a set of PROCESSES be followed in the development and maintenance of HEALTH SOFTWARE, and that the choice of PROCESSES be appropriate to the RISKS to the patient and other people. This follows from the belief that testing of software is not sufficient to determine that it is safe in operation.

The PROCESSES required by this document fall into two categories:

– PROCESSES which are required to determine the RISKS arising from the operation of each SOFTWARE ITEM in the software;

– PROCESSES which are required to achieve an appropriately low probability of software failure for each SOFTWARE ITEM, chosen on the basis of these determined RISKS.

This document requires the first category to be performed for all HEALTH SOFTWARE and the second category to be performed for selected SOFTWARE ITEMS.

A claim of conformance with this document requires a documented RISK ANALYSIS that identifies foreseeable sequences of events that include software and that can result in a HAZARDOUS SITUATION (see 4.2 and Clause 7). HAZARDOUS SITUATIONS that can be indirectly caused by software (for example, by vulnerability exploit or by providing misleading information, either which could result in inappropriate treatment being administered) should be included in this RISK ANALYSIS.

All ACTIVITIES that are required as part of the first category of PROCESSES are identified in the normative text as "[Level A, B, C]", indicating that they are required irrespective of the classification of the software to which they apply.

ACTIVITIES are required for all process rigor levels A, B, and C for the following reasons:

– the ACTIVITY produces a plan relevant to RISK MANAGEMENT or determination of software process rigor level;

– the ACTIVITY produces an output that is an input to RISK MANAGEMENT or determination of software process rigor level;

– the ACTIVITY is a part of RISK MANAGEMENT or determination of software process rigor level;

– the ACTIVITY establishes an administration SYSTEM, documentation or record-keeping SYSTEM that supports RISK MANAGEMENT or determination of software process rigor level;

– the ACTIVITY normally takes place when the process rigor level of the related software is unknown;

– the ACTIVITY can cause a change that could invalidate the current software process rigor level of the associated software; this includes the discovery and analysis of SAFETY related problems after release for INTENDED USE.

Other PROCESSES are required only for SOFTWARE SYSTEMS or SOFTWARE ITEMS classified in software process rigor levels B or C. ACTIVITIES required as parts of these PROCESSES are identified in the normative text as "[LevelB, C]", or "[Level C]" indicating that they are required selectively depending on the classification of the software to which they apply.

ACTIVITIES are required selectively for software in process rigor levels B and C for the following reasons:

– the ACTIVITY enhances the reliability of the software by requiring more detail or more rigor in the design, testing or other VERIFICATION;

– the ACTIVITY is an administrative ACTIVITY that supports another ACTIVITY required for process rigor levels B or C;

– the ACTIVITY supports the correction of SAFETY-related problems;

– the ACTIVITY produces records of design, implementation, VERIFICATION and release of SAFETY-related software.

ACTIVITIES are required selectively for software in process rigor level C for the following reason:

– the ACTIVITY further enhances the reliability of the software by requiring more detail, or more rigour, or attention to specific issues in the design, testing or other VERIFICATION

All PROCESSES and ACTIVITIES defined in this document are considered valuable in assuring the development and maintenance of high-quality software. The omission of many of these PROCESSES and ACTIVITIES as requirements for software in process rigor level A should not imply that these PROCESSES and ACTIVITIES would not be of value or are not recommended. Their omission is intended to recognize that software that cannot cause a HAZARDOUS SITUATION can be assured of SAFETY and EFFECTIVENESS primarily through overall VALIDATION ACTIVITY during the design of a product (which is outside the scope of this document) and through some simple software life cycle controls.

A.2 Summary of requirements by process rigor level

Table A.1 summarizes which software process rigor levels are assigned to each requirement. This table is informative and only provided for convenience. Clause 4 to Clause 9 identifies the software process rigor levels for each requirement.

1527 **Table A.1 – Summary of requirements by software process rigor level**

Clauses and subclauses		Level A	Level B	Level C
Clause 4	All requirements	X	X	X
5.1	5.1.1, 5.1.2, 5.1.3, 5.1.6, 5.1.7, 5.1.8, 5.1.9	X	X	X
	5.1.5, 5.1.10, 5.1.11		X	X
	5.1.4			X
5.2	5.2.1, 5.2.2, 5.2.4, 5.2.5, 5.2.6	X	X	X
	5.2.3		X	X
5.3	5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.6		X	X
	5.3.5			X
5.4	5.4.1		X	X
	5.4.2, 5.4.3, 5.4.4			X
5.5	5.5.1	X	X	X
	5.5.2, 5.5.3, 5.5.5		X	X
	5.5.4			X
5.6	All requirements		X	X
5.7	All requirements	X	X	X
5.8	5.8.1, 5.8.2, 5.8.4, 5.8.7, 5.8.8	X	X	X
	5.8.3, 5.8.5, 5.8.6		X	X
Clause 6	All requirements	X	X	X
7.1	All requirements		X	X
7.2	All requirements		X	X
7.3	All requirements		X	X
7.4	7.4.1	X	X	X
	7.4.2, 7.4.3		X	X
Clause 8	All requirements	X	X	X
Clause 9	All requirements	X	X	X

1528

Annex B
(informative)

Guidance on the provisions of this document

B.1 General

This annex contains guidance for specific clauses and subclause in this document, with clause and subclause numbers parallel to those in the body of the document. The numbering is, therefore, not consecutive.

ISO 14971 is referenced in this Annex to demonstrate concepts in Risk Management, this document does not require the use of 14971, but does require that Risk Management is conducted (Clause 4.2).

B.2 Guidance

Subclause 1.1 – Purpose

The purpose of this document is to provide a development PROCESS that will consistently produce high quality, safe HEALTH SOFTWARE. To accomplish this, this document identifies the minimum ACTIVITIES and TASKS that need to be accomplished to provide confidence that the software has been developed in a manner that is likely to produce highly reliable and safe HEALTH SOFTWARE.

Annex B provides guidance for the application of the requirements of this document. It does not add to, or otherwise change, the requirements of this document. Annex B can be used to better understand the requirements of this document.

In this document, ACTIVITIES are subclauses called out within the PROCESSES and TASKS are defined within the ACTIVITIES. For example, the ACTIVITIES defined for the software development PROCESS are software development planning, software requirements analysis, software ARCHITECTURAL design, software detailed design, SOFTWARE UNIT implementation, software integration and integration testing, SOFTWARE SYSTEM testing, and software release for INTENDED USE. The TASKS within these ACTIVITIES are the individual requirements.

This document does not require a particular SOFTWARE DEVELOPMENT LIFE CYCLE MODEL. However, conformance with this document does imply dependencies between PROCESSES, because inputs of a PROCESS are generated by another PROCESS. For example, the software process rigor level of the SOFTWARE SYSTEM should be completed after the RISK ANALYSIS PROCESS has established what HARM could arise from a SOFTWARE SYSTEM failure.

Because of such logical dependencies between PROCESSES, it is easier to describe the PROCESSES in this document in a sequence, implying a "waterfall" or "once-through" life cycle model. However, other life cycles can also be used. Some development (model) strategies as defined at ISO/IEC/IEEE 12207 [22] include the following.

– Incremental

  The “incremental development” model includes initial planning, initial requirements analysis, initial architectural definition, and initial VALIDATION, but allocates design, implementation, VERIFICATION (and sometimes delivery) activities to a series of stages, each of which provides a portion of the intended functionality. The approach provides for some flexibility to respond to inaccurate cost or schedule estimates by moving functionality to later increments.

– Spiral

The “spiral” variation on incremental developmental proposes ordering the development of functionality based on RISK, with the riskiest problems considered in the early increments. This provides some protection against cost surprises occurring late in the development cycle.

– Iterative

The “iterative development” model performs initial planning and then consists of a cyclic process of prototyping, testing, analyzing and refining the requirements and the solution. “Iterative” models repeatedly perform the life cycle PROCESSES to deliver prioritized SYSTEM functions sooner, with refined or more complex elements of the SYSTEM coming in later iterations.

– Evolutionary

The “evolutionary model” is intended to deal with incomplete knowledge of requirements. It provides for initial planning and initial ARCHITECTURE definition, but allocates requirements analysis, design, construction, VERIFICATION, VALIDATION and delivery to a series of stages. Delivered capabilities that do not meet user needs can be reworked in subsequent stages of the evolution.

– Agile

“Agile” methods actually can be applied within a variety of models. While Agile methods are common in executing an evolutionary life cycle model, they can be used in other life cycle models at various stages. What the methods have in common is an emphasis on continuous inspection and collaboration in the rapid production of working software in an environment where changes, including changes to requirements, are expected.

ISO/IEC/IEEE 24748-1 [27], ISO/IEC/IEEE 24748-2 [28], ISO/IEC/IEEE 24748-3 [29], and ISO/IEC/IEEE 24748-4 [30] provide additional detail regarding life cycle models and stages.

Whichever life cycle is chosen, it is necessary to maintain the logical dependencies between PROCESS outputs such as specifications, design documents and software. The waterfall life cycle model achieves this by delaying the start of a PROCESS until the inputs for that PROCESS are complete and approved.

Other life cycles, particularly evolutionary life cycles, permit PROCESS outputs to be produced before all the inputs for that PROCESS are available. For example, a new SOFTWARE ITEM can be specified, classified, implemented and VERIFIED before the whole software ARCHITECTURE has been finalised. Such life cycles carry the RISK that a change or development in one PROCESS output will invalidate another PROCESS output. All life cycles therefore use a comprehensive configuration management SYSTEM to ensure that all PROCESS outputs are brought to a consistent state and the dependencies maintained.

The following principles are important regardless of the software development life cycle used.

– All PROCESS outputs should be maintained in a consistent state; whenever any PROCESS output is created or changed, all related PROCESS outputs should be updated promptly to maintain their consistency with each other and to maintain all dependencies explicitly or implicitly required by this document;

– All PROCESS outputs should be available when needed as input to further work on the software.

– Before any HEALTH SOFTWARE is released for INTENDED USE, all PROCESS outputs should be consistent with each other and all dependencies between PROCESS outputs explicitly or implicitly required by this document should be observed.

Subclause 1.2 – Field of application

This document applies to the development and maintenance of HEALTH SOFTWARE; MEDICAL DEVICE SOFTWARE and software as a MEDICAL DEVICE are a subset of HEALTH SOFTWARE.

The use of this document requires the HEALTH SOFTWARE MANUFACTURER to perform RISK MANAGEMENT (see 4.2). Therefore, when the SYSTEM ARCHITECTURE includes an acquired component (this could be a purchased component or a component of unknown provenance), such as an open source electrocardiogram (ECG) algorithm, the acquired component becomes the responsibility of the MANUFACTURER and is included as part of RISK MANAGEMENT of the HEALTH SOFTWARE. It is expected that, through proper performance of HEALTH SOFTWARE RISK MANAGEMENT, the MANUFACTURER would understand the component and recognize that it includes SOUP. The MANUFACTURER using this document would invoke the software RISK MANAGEMENT PROCESS as part of the overall HEALTH SOFTWARE RISK MANAGEMENT PROCESS.

The maintenance of HEALTH SOFTWARE released for INTENDED USE applies to the post-production experience with HEALTH SOFTWARE. SOFTWARE MAINTENANCE includes the combination of all technical and administrative means, including supervision actions, to act on PROBLEM REPORTS to retain an item in, or restore it to, a state in which it can perform a required function as well as modification requests related to HEALTH SOFTWARE released for INTENDED USE. For example, this includes problem rectification, regulatory reporting, re-VALIDATION and preventive action. See ISO/IEC 14764 [23].

Clause 2 – Normative references

ISO/IEC 90003 [38] provides guidance for applying a quality management SYSTEM to software development. This guidance is not required by this document but is highly recommended.

Clause 3 – Terms and definitions

Where possible, terms have been defined using definitions from international standards.

This document chose to use three terms to describe the decomposition of a SOFTWARE SYSTEM (top level). The SOFTWARE SYSTEM can be a subsystem of a product (see IEC 60601-1 [1]), a HEALTH SOFTWARE product, or a MEDICAL DEVICE in its own right, which then becomes a software MEDICAL DEVICE (or software as a MEDICAL DEVICE). The lowest level that is not further decomposed for the purposes of testing or software configuration management is the SOFTWARE UNIT. All levels of composition, including the top and bottom levels, can be called SOFTWARE ITEMS. A SOFTWARE SYSTEM, then, is composed of one or more SOFTWARE ITEMS, and each SOFTWARE ITEM is composed of one or more SOFTWARE UNITS or decomposable SOFTWARE ITEMS. The responsibility is left to the MANUFACTURER to provide the granularity of the SOFTWARE ITEMS and SOFTWARE UNITS. Leaving these terms vague allows their application to the many different development methods and types of software used in products.

Clause 4 – General requirements

There is no known method to guarantee 100 % SAFETY for any kind of software.

There are three major principles which promote SAFETY for HEALTH SOFTWARE:

– RISK MANAGEMENT;

– quality management; and

– software engineering.

For the development and maintenance of safe HEALTH SOFTWARE, it is necessary to establish RISK MANAGEMENT as an integral part of a quality management SYSTEM as an overall framework for the application of appropriate software engineering methods and techniques. The

combination of these three concepts allows a MANUFACTURER to follow a clearly structured and consistently repeatable decision-making PROCESS to promote SAFETY for HEALTH SOFTWARE.

Many laws, regulations, and other authoritative rules have a direct effect on the way SOFTWARE SYSTEMS are developed, validated, and maintained. Therefore, these laws and regulations need to be considered during development as they can change how the HEALTH SOFTWARE is designed and developed. From a software development perspective, laws, regulations, and other authoritative rules lead to the creation of another set of requirements that the SYSTEM needs to meet (see also B.2 Guidance for Quality management).

Subclause 4.1 – Quality management

A disciplined and effective set of software PROCESSES includes organizational PROCESSES such as management, infrastructure, improvement, and training. To avoid duplication and to focus this document on software engineering, these PROCESSES have been omitted from this document. These PROCESSES are covered by a quality management SYSTEM.

ISO 13485 [16] is an International Standard that is specifically intended for applying the concepts of quality management to MEDICAL DEVICES. Conformance to ISO 13485 quality management SYSTEM requires the MANUFACTURER to identify and establish conformance with applicable regulatory requirements for the markets in which the HEALTH SOFTWARE is intended to be used.

Subclause 4.2 – RISK MANAGEMENT

Software development participates in RISK MANAGEMENT ACTIVITIES sufficiently to ensure that all reasonably foreseeable RISKS associated with HEALTH SOFTWARE are considered.

Rather than trying to define an appropriate RISK MANAGEMENT PROCESS in this document, it is required that the MANUFACTURER apply a RISK MANAGEMENT PROCESS. Specific software RISK MANAGEMENT ACTIVITIES resulting from HAZARDOUS SITUATIONS that have software as a contributing cause are identified in a supporting PROCESS described in Clause 7.

SECURITY is a broad term and it is not intended that this document conflict with existing standards [34] specifically ISO 27799 [19], or with AAMI TIR 57:2016 [41], or AAMI TIR97:2019 See also Table C.1 – Useful SECURITY standards.

The expectation within this document is to assure that SECURITY is assessed for issues related to SAFETY. This can be achieved by proper requirement and RISK MANAGEMENT on the SYSTEM level, supplying information to the SOFTWARE SYSTEM development PROCESS or within the SOFTWARE SYSTEM development itself.

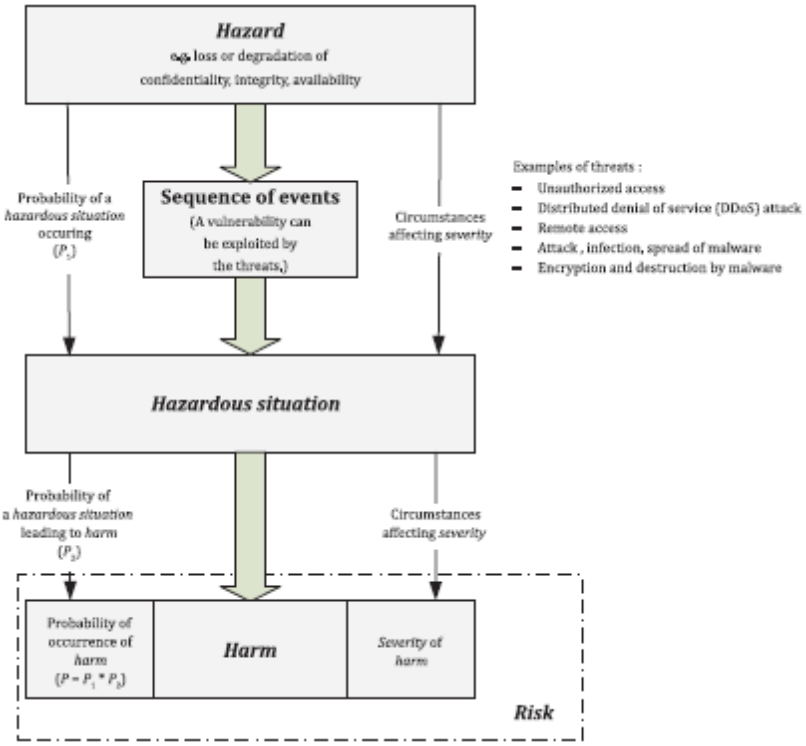
In terms of SAFETY it is believed that SECURITY can contribute with the following aspects:

1) lack of integrity, which can cause a HAZARDOUS SITUATION(S);
2) lack of availability, which can cause a HAZARDOUS SITUATION(S);
3) loss of confidentiality, which is a potential HARM on its own.

When the HEALTH SOFTWARE is networked, the MANUFACTURER needs to manage RISKS arising from the connected HEALTH SOFTWARE negatively impacting the larger IT environment with regard to compromise to confidentiality, integrity or availability of SYSTEMS or data (including privacy breach).

Figure B.1 is useful information from ISO/TR 24971:2020 [18] that can be used to relate both the IT SECURITY and ISO 14971-based terminology (aligns the concepts).

1703



1704

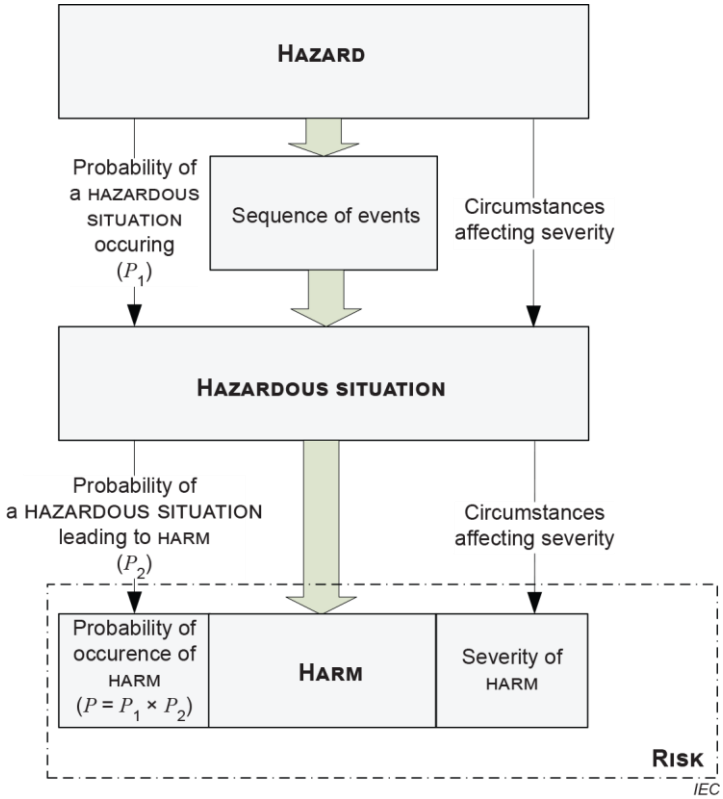
1705 SOURCE: ISO/TR 24971:2020, Figure F.1

1706 **Figure B.1 – Relation between HAZARD, HAZARDOUS SITUATION, HARM and SECURITY terminology**

1707 ## Subclause 4.4.1 – Purpose of software process rigor level

1708 The RISK associated with the use of the software serves as the input to a method for determining
1709 software process rigor level, which determines the ACTIVITIES and TASKS to be used during the
1710 development and maintenance of the software. The determination of software process rigor
1711 level ACTIVITY begins prior to software development ACTIVITIES (Clause 5); the RISK ANALYSIS at
1712 this stage is prior to RISK CONTROL implementation. The outcome of the software process rigor
1713 level determination (Clause 7.1) is used to plan development ACTIVITIES and TASKS (different
1714 requirements based on software process rigor level (A, B, or C). There can be unacceptable
1715 RISK prior to any SOFTWARE SYSTEM RISK CONTROL measures being implemented. The
1716 contribution of the HEALTH SOFTWARE to HAZARDOUS SITUATIONS and the related RISKS is used to
1717 define the software process rigor level.

1718 RISK is considered to be a combination of the severity of HARM and the probability of its
1719 occurrence. However, quantitatively estimating the probability of occurrence of a software
1720 failure is difficult due to the variability of inputs and the complex nature of software. Therefore,
1721 the probability of failure occurring should be set to 1. When assessing a sequence of events,
1722 the probability of other events not originating from software can be used for the probability of
1723 the HAZARDOUS SITUATION occurring (P_1 in Figure B.2).



NOTE 1 Depending on the complexity of the MEDICAL DEVICE, a HAZARD can lead to multiple HAZARDOUS SITUATIONS, and each HAZARDOUS SITUATION can lead to multiple HARMS.

NOTE 2 The probability of occurrence of HARM (P) can be composed of separate P_1 and P_2 values.

NOTE 3 The thin arrows represent elements of RISK ANALYSIS and the thick arrows depict how a HAZARD can lead to HARM.

SOURCE: ISO 14971:2019, Figure C.1

Figure B.2 – Pictorial example of the relationship of HAZARD, sequence of events, HAZARDOUS SITUATION, and HARM

In many cases however, it might not be possible to estimate the probability for the remaining events in the sequence, and the RISK should be EVALUATED on the basis of the nature of the HARM alone (the probability of the HAZARDOUS SITUATION occurring should be set to 1). RISK ESTIMATION in these cases should be focused on the SEVERITY of the HARM resulting from the HAZARDOUS SITUATION.

Estimates of probability of a HAZARDOUS SITUATION leading to HARM (P_2 in Figure B.2) generally require SYSTEM knowledge to distinguish between HAZARDOUS SITUATIONS where practices external to the SOFTWARE SYSTEM would be likely to prevent HARM or not.

The information gathered and analysed during the RISK MANAGEMENT PROCESS is an important source of inputs to the determination of the software process rigor level. As illustrated in Figure B.3, specific steps in the ISO 14971 flow produce a list of HAZARDOUS SITUATIONS, associated RISK CONTROL measures (RCM), HARMS, and severities. This information can be utilized in the determination of the proper determination of software process rigor level. It is not necessary to perform a separate PROCESS to collect the necessary information to classify the software. It can, and should, be done with the same information used to manage the overall RISK of the product.

RISK MANAGEMENT ACTIVITIES continue throughout the life cycle of a product. During the development PROCESS, RISKS continue to be refined and new causes can be identified. As a

result, the software process rigor level may be modified as the development progresses. The grey line at the bottom of Figure B.3 indicates the final point in the RISK MANAGEMENT PROCESS at which the determination of software process rigor level should have been finalized. See also B.2 Guidance for Software ARCHITECTURAL design for a discussion of software ARCHITECTURAL design and determination of software process rigor level.

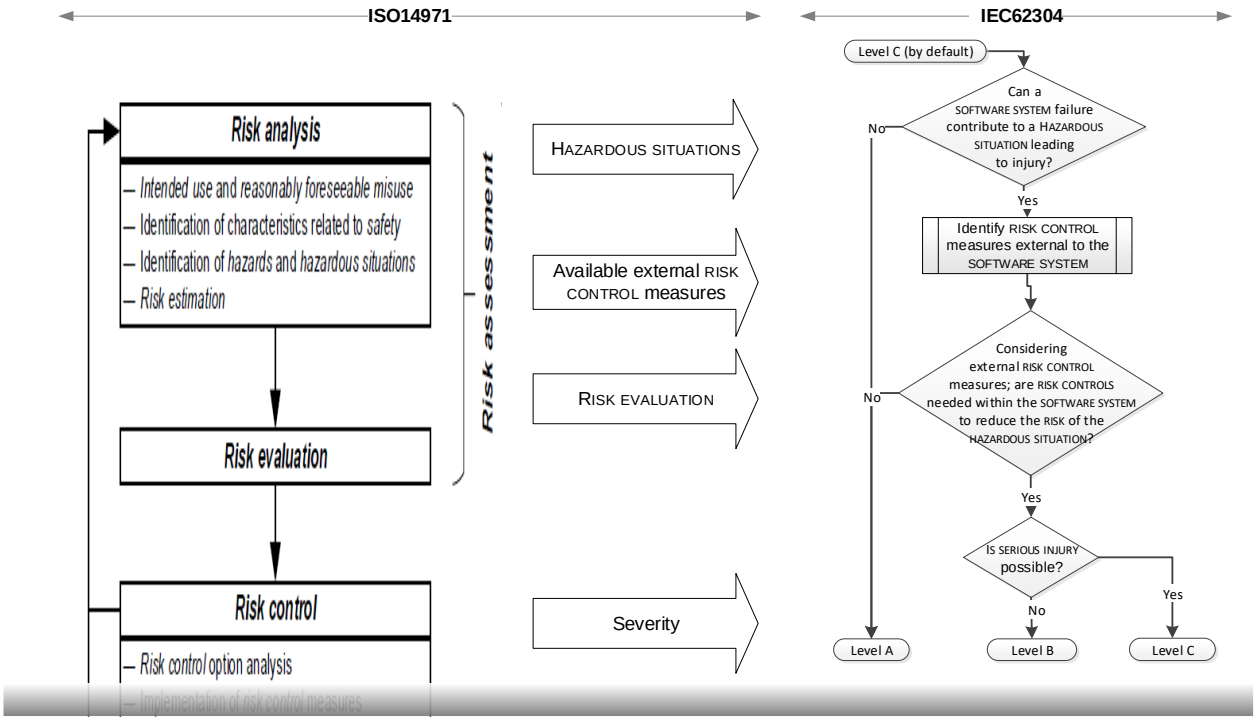


Figure B.3 – Pictorial representation of the relationship of RISK MANAGEMENT (ISO 14971:2019 Figure 1) and software process rigor level

Additional information regarding the application of RISK MANAGEMENT to software can be found in IEC TR 80002-1 [14].

Subclause 4.4.3 – Assigning software process rigor level

General

Figure B.3 shows one example for determining software process rigor level. This is not intended to be all inclusive as there are many acceptable methods to determine the software process rigor level which can be used.

Software process rigor level is determined in four steps, as illustrated in Figure B.4. The purpose of this section is to provide guidance for navigating each step of this flowchart.

There is no specific requirement to document the rationale for each step. However, for the purpose of illustration in this section, a detailed rationale has been provided to aid in explanation of the example used to determine the software process rigor level.

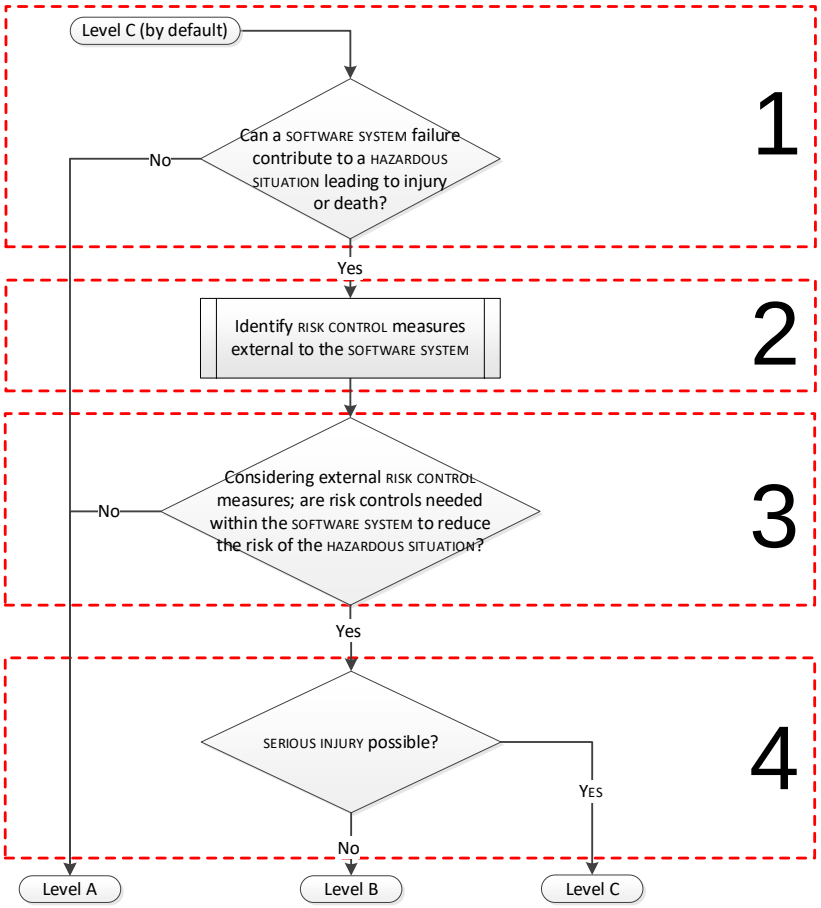


Figure B.4 – Determining software process rigor level in steps

Step 1 – "Contribute to HAZARDOUS SITUATION"

Before this EVALUATION can take place, the following inputs are needed to better understand the SOFTWARE SYSTEM:

– a list of HAZARDOUS SITUATIONS and their potential severity;

– the INTENDED USE of the SOFTWARE SYSTEM (what problem is the software intended to solve?);

– the requirements of the SOFTWARE SYSTEM.

In early stages of a project, the above inputs can be subject to frequent changes. If an input changes after the ACTIVITY of determining software process rigor level, it can necessitate a reassessment.

Table B.2 below shows an example of an assessment of HAZARDOUS SITUATIONS for this step 1.

Table B.2 – Analysis of HAZARDOUS SITUATIONS

HAZARDOUS SITUATION	Can SOFTWARE SYSTEM contribute?
Situation 1	Yes
Situation 2	Yes
Situation 3	Yes
Situation 4	No

Step 2 – "RISK CONTROL MEASURES external to the software"

After step 1, the HAZARDOUS SITUATIONS related to the SOFTWARE SYSTEM should be well-understood. The next step is to assess available external RISK CONTROL measures and their possible relation, with regards to SAFETY, to the SOFTWARE SYSTEM.

External RISK CONTROL measures are those implemented outside of the SOFTWARE SYSTEM, i.e. external to the SOFTWARE SYSTEM.

These external RISK CONTROL measures can vary and might differ in both suitability and efficacy depending on type of SYSTEM. Examples are as follows.

– A standalone SOFTWARE SYSTEM, not embedded in a device, can usually not rely on hardware RISK CONTROL measures. Such SYSTEMS often need to rely on non-hardware RISK CONTROL measures such as software solutions or procedural constraints.

– In an embedded SOFTWARE SYSTEM, a hardware RISK CONTROL measure is often a preferred RISK CONTROL measure. However, other mitigations might also be adequate.

Figure B.5 and Figure B.6 are extensions of Table B.2. They show an example of how the SOFTWARE SYSTEM can contribute to a variety of HAZARDOUS SITUATIONS and illustrate how external RISK CONTROL measures can reduce or eliminate the linkage between a SOFTWARE SYSTEM and a HAZARDOUS SITUATION.

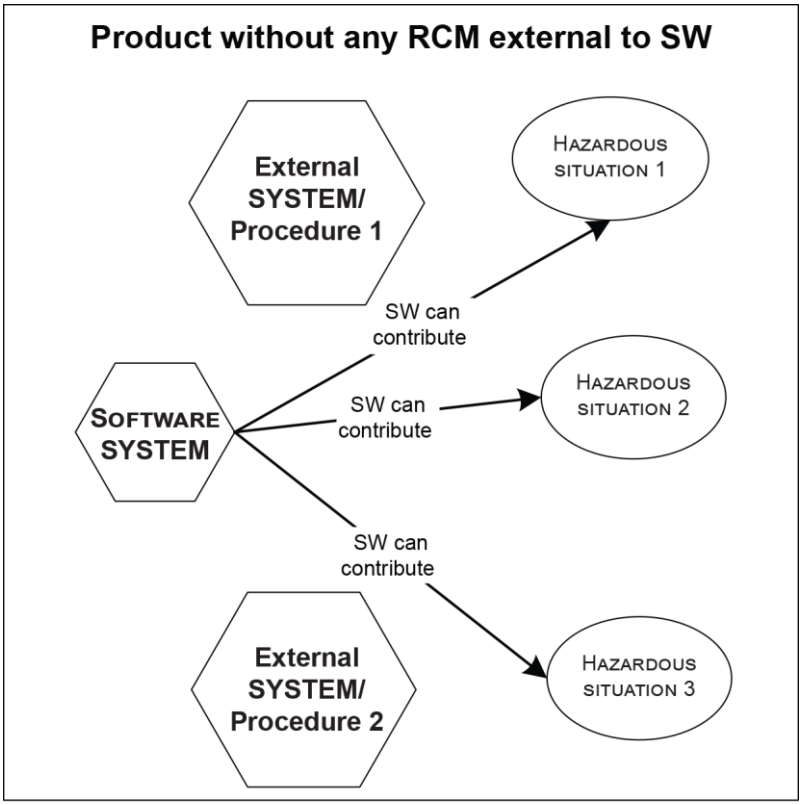
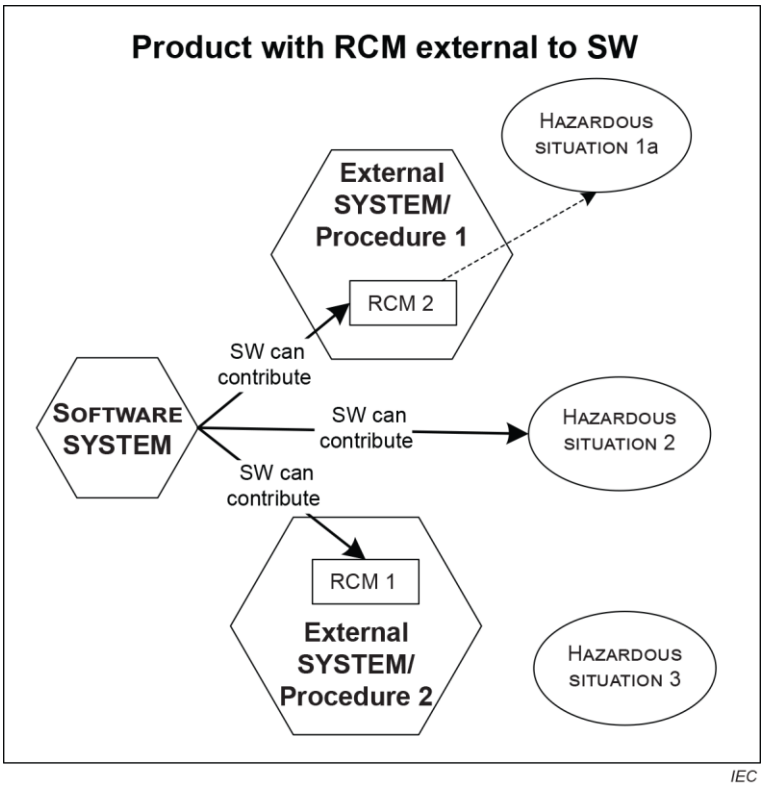


Figure B.5 – SOFTWARE SYSTEM contributing to HAZARDOUS SITUATIONS



NOTE HAZARDOUS SITUATION 1 can change to a new HAZARDOUS SITUATION, 1a, depending on what RISK CONTROL measure is introduced by RCM 2.

Figure B.6 – SOFTWARE SYSTEM contributing to HAZARDOUS SITUATIONS with RISK CONTROL measures

In this example, a RISK CONTROL measure, RCM 1, completely breaks the link between the SOFTWARE SYSTEM and HAZARDOUS SITUATION 3 (demonstrated by the arrow that ends at the RCM 1). RCM2 only reduces, but does not completely eliminate, the impact or occurrence rate of the HAZARDOUS SITUATION (HAZARDOUS SITUATION 1) and therefore the arrow becomes dashed after passing through RCM 2.

The assessment table, Table B.2, is now expanded in Table B.3 to include the associated external RISK CONTROL measure.

Table B.3 – Identification of HAZARDOUS SITUATIONS with external RISK CONTROL measure

HAZARDOUS SITUATION	Can SOFTWARE SYSTEM contribute?	External RCM?
Situation 1	Yes	RCM 2
Situation 2	Yes	Not available
Situation 3	Yes	RCM 1
Situation 4	No	Not applicable

The EFFECTIVENESS of a RISK CONTROL measure is critical to the determination of software process rigor level, but very often EFFECTIVENESS cannot be fully evaluated before the end of the product development. Depending on the outcome of the EVALUATION, this can prompt the developer to reassess the software process rigor level if the EFFECTIVENESS of the RISK CONTROL measure differs from the original estimation.

It has been asked how to assign software process rigor level for SOFTWARE SYSTEMS that utilize redundancy or diversity to achieve the desired level of SAFETY.

In both cases, one of the SOFTWARE SYSTEMS is obligated to take responsibility for the implementation of the RISK CONTROL measure (taking a higher software process rigor level based on the RISK being controlled). In the case of redundancy, the argument that each SOFTWARE SYSTEM has an external RISK CONTROL measure (pointing to the other SOFTWARE SYSTEM) and therefore each are software process rigor level A is not logical because one of the SOFTWARE SYSTEMS is obligated to take responsibility for the RISK being controlled. See Table B.4, Situation 5.

See also NOTE 5 and NOTE 6 of 4.4.3; NOTE 5 states that a SOFTWARE SYSTEM that implements a RISK CONTROL measure can fail and contribute to a HAZARDOUS SITUATION, NOTE 6 states that external RISK CONTROL measure can be an independent SOFTWARE SYSTEM. When a SOFTWARE SYSTEM implements a RISK CONTROL measure, it needs to be assigned a software process rigor level based on the RISK it is controlling (see 7.2.2 b)).

Step 3 – "–Considering external RISK CONTROL measures, are RISK CONTROLS needed within the SOFTWARE SYSTEM to reduce the risk of the HAZARDOUS SITUATION?"

Per the RISK MANAGEMENT PROCESS, it is expected that each MANUFACTURER has defined acceptable and unacceptable RISKS. These definitions are used as an input to determine the software process rigor level. It is also expected that the probability of a software failure occurring is 1 (see 4.4.3 b)).

For example, if P_2 is acceptably low it might justify an adjustment from the default software process rigor level C to A, assuming the adjustment is supported by the SYSTEM RISK assessment.

Step 3 occurs prior to implementation of RISK CONTROL measures, including software RISK CONTROL measures and application of this document. Consequently, at this stage, a benefit-RISK analysis cannot be used to support the decision in this step 3. The benefit-RISK analysis is done after RISK CONTROL measures are implemented and after EVALUATION of RESIDUAL RISK. It is important to involve people with the expertise necessary for the benefit-RISK analysis, especially those that know how the HEALTH SOFTWARE is actually used.

Step 4 – "Severity of HARM"

Based on previous steps, there should now be information available from RISK MANAGEMENT to determine the severity of HARM associated with the HAZARDOUS SITUATIONS.

As for any rule, there might be borderline scenarios in the determination of software process rigor level.

Table B.3 has been expanded to show the original software process rigor level (prior to RISK CONTROL measure) and the software process rigor level after RISK CONTROL measures, shown in Table B.4. The final software process rigor level for the entire SOFTWARE SYSTEM is also documented in Table B.4.

1864 **Table B.4 – Identification of HAZARDOUS SITUATIONS with software process rigor level**

HAZARDOUS SITUATION	Can SOFTWARE SYSTEM contribute?	Software process rigor level prior to RCM [f]	External RCM?	Software process rigor level after RCM
Situation 1	Yes	C	RCM 2	B [a]
Situation 2	Yes	B	Not available	B
Situation 3	Yes	B	RCM 1	A [b]
Situation 4	No	A	Not applicable	A
Situation 5	Yes	C	RCM 3	B or C [c]
Situation 6	Yes	B	RCM 4	B [d]
Final software process rigor level for the SOFTWARE SYSTEM				**B or C [e]**

[a] The RISK CONTROL measure RCM 2 changes the possible SEVERITY resulting from "Situation 1". This is possible if RCM2 limits the consequences of a software failure. Example: assume time-based exposure of a hazardous substance, too long exposure can result in SERIOUS INJURY. To prevent a potential over exposure, caused by software, RCM 2 is added to implement a time out function in hardware. The time out function prevents the SYSTEM from causing SERIOUS INJURY but non-SERIOUS INJURY is still possible.

[b] In this case, RCM1 is deemed to be powerful enough to actually break the relation between the SOFTWARE SYSTEM and the HAZARDOUS SITUATION. Usually, this is a result of changed requirements or improved overall product ARCHITECTURE.

[c] This scenario is not shown in Figure B.2 to Figure B.6 but it is similar to the examples provided for RCM 2. Example: a different SOFTWARE SYSTEM could be used to implement a redundant ARCHITECTURE, as a RISK CONTROL measure, for the identified HAZARDOUS SITUATION. The added SOFTWARE SYSTEM will become software process rigor level C since it inherits the severity of the RISK it is supposed to mitigate.

[d] Depending on RISK EVALUATION criteria, some RISK CONTROL measures would not pay off in terms of reduced software process rigor level but they can still improve the overall product SAFETY and make the RESIDUAL RISKS acceptable. In theory, both severity and probability of HARM can be lowered without affecting the software process rigor level. But, it still makes the SOFTWARE SYSTEM safer. This scenario is not shown in Figure B.2 to Figure B.6 but it is similar to the examples provided for RCM 2.

[e] The final software process rigor level is based on the HAZARDOUS SITUATION contributing to the RISK with highest severity.

[f] The software process rigor level prior to RISK CONTROL measure is only provided in the table to provide full overview of the ACTIVITY of determining software process rigor level.

1865

1866 Software process rigor level decomposed into SOFTWARE ITEMS

1867 If a SOFTWARE SYSTEM is decomposed into SOFTWARE ITEMS, each SOFTWARE ITEM can be
1868 assigned an individual software process rigor level; this is then carried out with the same
1869 approach as for the SOFTWARE SYSTEM itself.

1870 The breakdown of software process rigor level into SOFTWARE ITEMS has following constraints.

1871 – At least one SOFTWARE ITEM is of the same software process rigor level as the SOFTWARE SYSTEM.

1872 – A SOFTWARE ITEM can only be equal or lower than its parent SOFTWARE ITEM software process rigor
1873 level.

1874 – A SOFTWARE ITEM is properly segregated within the ARCHITECTURE of the SOFTWARE SYSTEM. A
1875 SOFTWARE ITEM that is to be classified separately from the overall SOFTWARE SYSTEM should be
1876 developed in such a way that ARCHITECTURE supports this segregation through proper planning.

1877 **Subclause 4.5 – LEGACY SOFTWARE**

Some geographies require the MANUFACTURER to demonstrate that HEALTH SOFTWARE is in conformity with the requirements of this document. Not all software that was developed prior to the publication of this document will conform to this document. Therefore, 4.5 establishes a PROCESS for the application of this document to LEGACY SOFTWARE.

A MANUFACTURER can determine that retrospective documentation of an already finished development-life cycle performed as an isolated ACTIVITY does not result in the reduction of RISK associated with the use of the product. The PROCESS results in the identification of a subset of ACTIVITIES defined in this document which does result in reduction of RISK. Some additional goals implicit in the PROCESS are:

– required ACTIVITIES and resulting documentation should rely on and make use of, wherever possible, existing documentation;

– a MANUFACTURER should utilize design and RISK MANAGEMENT expertise as effectively as possible to achieve a reduction of RISK.

In addition to a plan identifying the subset of ACTIVITIES to execute, the PROCESS also results in objective evidence which can support safe continued use of the LEGACY SOFTWARE and a summary rationale for this conclusion.

The RISKS associated with the planned continued use of the LEGACY SOFTWARE depend on the context in which the LEGACY SOFTWARE will be used to create a SOFTWARE SYSTEM. The MANUFACTURER will document all identified HAZARDS associated with the LEGACY SOFTWARE.

Subclause 4.5.2 requires a comprehensive assessment of available post-production field data obtained for the LEGACY SOFTWARE during the time it has been in production and use. Typical sources of post-production data include

– adverse events attributable to the device,

– feedback received from users of the device, and

– ANOMALIES discovered by the MANUFACTURER.

Though no consensus exists for a method of prospectively estimating quantitatively the probability of occurrence of a software failure, such information can be available for LEGACY SOFTWARE, based on the usage of such software and EVALUATION of post-production data. If it is possible in such cases to quantitatively estimate the probability of events in the sequence, a quantitative value can be used for expressing the probability of the entire sequence of events occurring. If such quantitative estimation is not possible, considering a worst case probability is appropriate, and the probability for the software failure occurring is 1.

The MANUFACTURER determination of how the LEGACY SOFTWARE will be used in the overall SYSTEM ARCHITECTURE is input to the assessment of RISK. The RISKS to be considered vary accordingly.

– When LEGACY SOFTWARE has been safely and reliably used and the MANUFACTURER wishes to continue using the LEGACY SOFTWARE, the rationale for continued use rests primarily on the assessment of RISK based on post-production records.

– When LEGACY SOFTWARE is reused to create a new SOFTWARE SYSTEM, the INTENDED USE of the LEGACY SOFTWARE might be different from its original INTENDED USE. In this case, the RISK assessment takes into account the modified set of HAZARDOUS SITUATIONS which can arise due to failures of the LEGACY SOFTWARE.

When LEGACY SOFTWARE will be changed and used within a new SOFTWARE SYSTEM, the MANUFACTURER should consider how the existing records of safe and reliable operation can be invalidated by the changes.

Per 4.5.4 c), changes to the LEGACY SOFTWARE are to be performed in accordance with this document, including assessment of impact to RISK CONTROL measures according to 7.4. In the case of LEGACY SOFTWARE, it is likely that existing RISK CONTROL measures are not fully documented, and special care should be taken to EVALUATE the potential impact of changes, utilizing available documented design records as well as expertise of individuals having knowledge of the SYSTEM.

According to 4.5, the MANUFACTURER performs a gap analysis in order to determine the available documentation including objective evidence of performed TASKS done during development of the LEGACY SOFTWARE and compared to 5.2, 5.3, 5.7, and Clause 7. Typical steps to accomplish this gap analysis include

a) identification of the LEGACY SOFTWARE, including VERSION, revision and any other means, required for clear identification,

b) EVALUATION of existing DELIVERABLES corresponding to the DELIVERABLES required by 5.2, 5.3, 5.7, and Clause 7;

c) EVALUATION of available objective evidence, documenting the previously applied SOFTWARE DEVELOPMENT LIFE CYCLE MODEL (as appropriate), and

d) EVALUATION of the adequacy of existing RISK MANAGEMENT documentation.

Taking the performed gap analysis into account, the MANUFACTURER will EVALUATE the potential reduction in RISK resulting from the generation of the missing DELIVERABLES and associated ACTIVITIES, and create a plan to perform ACTIVITIES and generate DELIVERABLES to close these gaps.

Reduction of RISK should balance the benefit of applying the software development PROCESS according to Clause 5 against the possibility that modification of the LEGACY SOFTWARE without full knowledge of its development history could introduce new defects that increase the RISK. Some of the elements of Clause 5 may be assessed to have little to no reduction of RISK when done after the fact. For example, detailed design and unit VERIFICATION reduce RISK primarily during the PROCESS of developing new software or refactoring existing software. If these objectives are not planned, performing the ACTIVITIES in isolation can create documentation but lead to no reduction in RISK.

At a minimum, the gap closure plan addresses missing SOFTWARE SYSTEM test records. If these do not exist or are not suitable to support a rationale to continue use of the LEGACY SOFTWARE, the gap closure plan should include creation of SOFTWARE SYSTEM requirements at a functional level according to 5.2 and tests according to 5.7.

The documented rationale for continued use of the LEGACY SOFTWARE builds on the available objective evidence and analysis obtained in the course of assessing the RISK and creating a gap closure plan appropriate for the context of LEGACY SOFTWARE re-use.

The rationale makes a positive case for the safe and reliable performance of the LEGACY SOFTWARE in the planned reuse context, taking into account both the post-production records available for the LEGACY SOFTWARE and the RISK CONTROL measures affected by filling PROCESS gaps.

Subclause 5.1 – Software development planning

The objective of this ACTIVITY is to plan the software development TASKS to reduce RISKS caused by software, communicate procedures and goals to members of the development team, and ensure that SYSTEM quality requirements for HEALTH SOFTWARE are met.

The software development planning ACTIVITY can document TASKS in a single plan or in multiple plans. Some MANUFACTURERS might have established policies and procedures that apply to the development of all their HEALTH SOFTWARE. In this case, the plan can simply reference the

existing policies and procedures. Some MANUFACTURERS might prepare a plan or set of plans specific to the development of each HEALTH SOFTWARE that spell out in detail specific ACTIVITIES and reference general procedures. Another possibility is that a plan or set of plans is tailored for the development of each HEALTH SOFTWARE. The planning should be specified at the level of detail necessary to carry out the development PROCESS and should be proportional to the RISK. For example, SYSTEMS or items with higher RISK would be subject to a development PROCESS with more rigour and TASKS should be spelled out in greater detail.

Planning is an iterative ACTIVITY that should be re-examined and updated as development progresses. The plan can evolve to incorporate more and better information as more is understood about the SYSTEM and the level of effort needed to develop the SYSTEM. For example, a SYSTEM's initial software process rigor level can change as a result of exercising the RISK MANAGEMENT PROCESS and development of the software ARCHITECTURE. Or it might be decided that a SOUP be incorporated into the SYSTEM. It is important that the plan(s) be updated to reflect current knowledge of the SYSTEM and the level of rigour needed for the SYSTEM or items in the SYSTEM to enable proper control over the development PROCESS.

Subclause 5.2 – Software requirements analysis

This ACTIVITY requires the MANUFACTURER to establish and verify the software requirements for HEALTH SOFTWARE. Establishing verifiable requirements is essential for determining what is to be built, for determining that the HEALTH SOFTWARE exhibits acceptable behaviour, and for demonstrating that the completed HEALTH SOFTWARE is ready for use. To demonstrate that the requirements have been implemented as desired, each requirement should be stated in such a way that objective criteria can be established to determine whether it has been implemented correctly. If the device RISK MANAGEMENT PROCESS imposes requirements on the software to control identified RISKS, these requirements are to be identified in the software requirements in such a way as to make it possible to trace the RISK CONTROL measures to the software requirements. All software requirements should be identified in such a way as to make it possible to demonstrate TRACEABILITY between the requirement and SOFTWARE SYSTEM testing. If regulatory approval in some countries requires conformance to specific regulations or international standards, this conformance requirement should be documented in the software requirements. Because the software requirements establish what is to be implemented in the software, an EVALUATION of the requirements is required before the requirements analysis ACTIVITY is complete.

An area of frequent confusion is the distinction between customer needs, design inputs, software requirements, software functional specifications, and software design specifications. Design inputs are the interpretation of customer needs into formally documented SYSTEM requirements. Software requirements are the formally documented specifications of what the software does to meet the customer needs and the design inputs. Software functional specifications are often included within the software requirements and define in detail what the software does to meet its requirements, even though many different alternatives might also meet the requirements. Software design specifications define how the software will be designed and decomposed to implement its requirements and functional specifications.

Traditionally, software requirements, functional specifications, and design specifications have been written as a set of one or more documents. It is now feasible to consider this information as data items within a common database. Each item would have one or more attributes that would define its purpose and linkage to other items in the database. This approach allows presentation and printing of different views of the information best suited for each set of intended users (e.g., marketing, MANUFACTURERS, testers, auditors) and supports TRACEABILITY to demonstrate adequate implementation and the extent to which test cases test the requirements. Tools to support this approach can be as simple as a hypertext document using HTML hyperlinks or as complex and capable as computer aided software engineering (CASE) tools and requirements analysis tools.

The SYSTEM requirements PROCESS is out of scope of this document. However, the decision to implement SYSTEM functionality with software is normally made during SYSTEM design. Some or all of the SYSTEM requirements are allocated to be implemented in software. The software

requirements analysis ACTIVITY consists of analysing the requirements allocated to software by the SYSTEM requirements PROCESS and deriving a comprehensive set of software requirements that reflect the allocated requirements.

Subclause 5.2.2 e) requires that SECURITY requirements be established for SECURITY capabilities implemented in the HEALTH SOFTWARE. IEC TR 80001-2-2 [12] lists 19 SECURITY capabilities that should be considered when developing the SECURITY requirements. IEC TR 80001-2-8 [13] provides guidance for establishing the SECURITY capabilities identified in IEC TR 80001-2-2. Both of these standards are guidance documents in the IEC 80001 [10] series of standards, for the application of RISK MANAGEMENT to IT-networks incorporating MEDICAL DEVICES. This series of standards provides roles, responsibilities, and activities necessary for RISK MANAGEMENT in this networked environment. Information regarding SECURITY requirements can also be found in ISO/IEC 27002 [35] and ISO 27799 [19].

To ensure the integrity of the SYSTEM, the MANUFACTURER should provide a mechanism for negotiating changes and clarifications to the SYSTEM requirements to correct impracticalities, inconsistencies or ambiguities in either the parent SYSTEM requirements or the software requirements.

The PROCESS of capture and analysis of SYSTEM and software requirements can be iterative. This document does not intend to require the PROCESSES to be rigidly segregated into two layers. In practice, SYSTEM ARCHITECTURE and software ARCHITECTURE are often outlined simultaneously and the SYSTEM and software requirements are subsequently documented in a layered form.

Subclause 5.3 – Software ARCHITECTURAL design

This ACTIVITY requires the MANUFACTURER to define the major structural components of the software and identify their key responsibilities, their externally visible properties, and the relationship among them. If the behaviour of a component can affect other components, that behaviour should be described in the software ARCHITECTURE. This description is especially important for behaviour that can affect components of the SYSTEM that are outside the software (see 5.3.5).

The software ARCHITECTURE should include credible strategies for segregating SOFTWARE ITEMS so that they do not interact in unsafe ways.

ARCHITECTURAL decisions are extremely important for implementing RISK CONTROL measures. Without understanding (and documenting) the behaviour of a component that can affect other components, it will be nearly impossible to show that the SYSTEM is safe.

A software ARCHITECTURE is necessary to ensure the correct implementation of the software requirements. The software ARCHITECTURE is not complete unless all software requirements can be implemented by the identified SOFTWARE ITEMS. Because the design and implementation of the software is dependent on the ARCHITECTURE, the ARCHITECTURE is VERIFIED to complete this ACTIVITY. VERIFICATION of the ARCHITECTURE is generally done by technical EVALUATION.

The software process rigor level of SOFTWARE ITEMS during the software ARCHITECTURE ACTIVITY creates a basis for the subsequent choice of software PROCESSES. The records of classification are placed under change control as part of the RISK MANAGEMENT FILE.

Many subsequent events might invalidate the classification. These include, for example,

– changes of SYSTEM specification, software specification or ARCHITECTURE,

– discovery of errors in the RISK ANALYSIS, especially unforeseen HAZARDS, and

– discovery of the infeasibility of a requirement, especially a RISK CONTROL measure.

Therefore, during all ACTIVITIES following the design of the software ARCHITECTURE, the classification of the SOFTWARE SYSTEM and SOFTWARE ITEMS should be re-EVALUATED and might need to be revised. This would trigger rework to apply additional PROCESSES to a SOFTWARE ITEM as a result of its upgrading to a higher process rigor level. The software configuration management PROCESS (Clause 8) is used to ensure that all necessary rework is identified and completed.

Figure B.7 illustrates the possible partitioning for SOFTWARE ITEMS within a SOFTWARE SYSTEM and how the software process rigor levels would be applied to the group of SOFTWARE ITEMS in the decomposition.

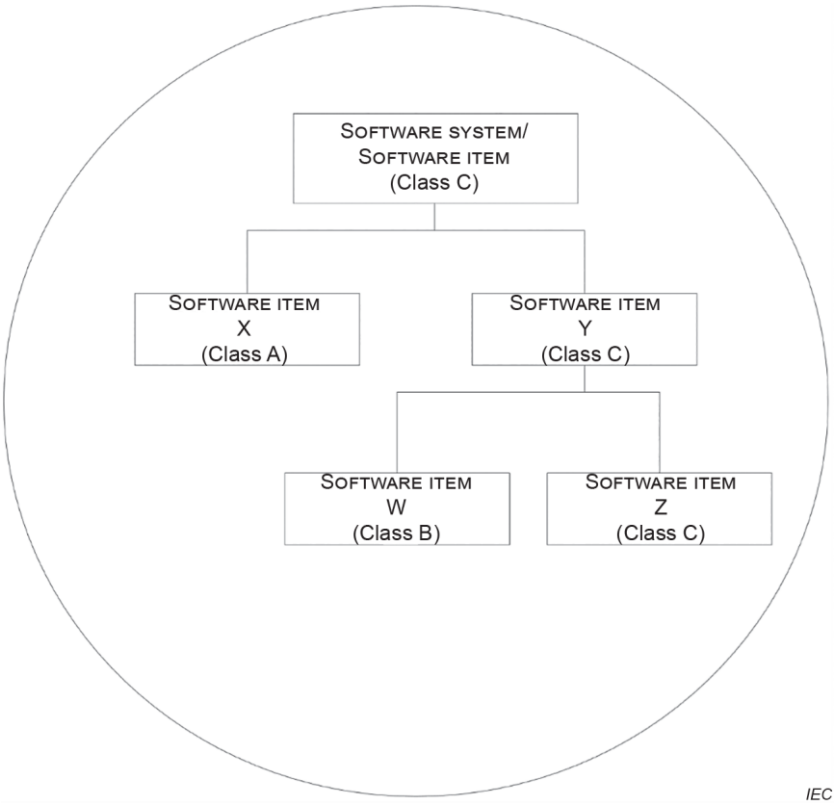


Figure B.7 – Example of partitioning of SOFTWARE ITEMS

For this example, the MANUFACTURER assigns software process rigor level C to the SOFTWARE SYSTEM.

During software ARCHITECTURE design, the MANUFACTURER has decided to partition the SYSTEM, as shown, with 4 SOFTWARE ITEMS: X, Y, W and Z. The MANUFACTURER has segregated all SOFTWARE SYSTEM elements that contribute to HAZARDOUS SITUATIONS that could result in death or SERIOUS INJURY to SOFTWARE ITEM Z, i.e. software process rigor level C. SOFTWARE ITEM Y therefore is classified as software process rigor level C, per 4.4.3 d).

All remaining SOFTWARE SYSTEM elements that contribute to HAZARDOUS SITUATIONS that could result in a non-SERIOUS INJURY are segregated to SOFTWARE ITEM W. SOFTWARE ITEM W is classified as software process rigor level B.

The SOFTWARE SYSTEM is also at a software process rigor level C per this requirement. SOFTWARE ITEM X has been classified as a software process rigor level of A. The MANUFACTURER is able to document a rationale for the segregation between SOFTWARE ITEMS X and Y, as well as SOFTWARE ITEMS W and Z, to assure the integrity of the segregation. If segregation is not possible, SOFTWARE ITEM X is classified as software process rigor level C.

Subclause 5.4 – Software detailed design

This ACTIVITY requires the MANUFACTURER to refine the SOFTWARE ITEMS and interfaces defined in the ARCHITECTURE to create SOFTWARE UNITS and their interfaces. Although SOFTWARE UNITS are often thought of as being a single function or module, this view is not always appropriate. This document has defined SOFTWARE UNIT to be a SOFTWARE ITEM that is not subdivided into smaller items. SOFTWARE UNITS can be tested separately. The MANUFACTURER should define the level of detail of the SOFTWARE UNIT. Detailed design specifies algorithms, data representations, interfaces among different SOFTWARE UNITS, and interfaces between SOFTWARE UNITS and data structures. Detailed design is also concerned with the packaging of the software product. It is necessary to define the design of the SOFTWARE UNITS and the interfaces in sufficient detail to permit its SAFETY and EFFECTIVENESS to be objectively VERIFIED where this can be ensured using other requirements or design documentation. It should be complete enough that the programmer is not required to make ad hoc design decisions. Detailed design is also concerned with the ARCHITECTURE of HEALTH SOFTWARE.

A SOFTWARE ITEM can be decomposed so that only a few of the new SOFTWARE ITEMS implement the SAFETY-related requirement of the original SOFTWARE ITEM. The remaining SOFTWARE ITEMS do not implement SAFETY-related functions and can be reclassified into a lower software process rigor level. However, the decision to do this is in itself part of the RISK MANAGEMENT PROCESS, and is documented in the RISK MANAGEMENT FILE.

It is necessary to verify the detailed design before the ACTIVITY is complete because implementation depends on detailed design. VERIFICATION of detailed design is generally done by a technical EVALUATION. Subclause 5.4.4 requires the MANUFACTURER to verify the outputs of the detailed design ACTIVITIES. The design specifies how the requirements shall be implemented. VERIFICATION of the design provides assurance that it implements the software ARCHITECTURE and is free from contradiction with the software ARCHITECTURE.

If the design contains defects, the code will not implement the requirements correctly.

When present in the design, the MANUFACTURER should verify design characteristics which the MANUFACTURER believes are important for SAFETY. Examples of these characteristics include

– implementation of the intended events, inputs, outputs, interfaces, logic flow, allocation of CPU, allocation of memory resources, error and exception definition, error and exception isolation, and error recovery;
– definition of the default state, in which all faults that can result in a HAZARDOUS SITUATION are addressed, with events and transitions;
– initialization of variables, memory management; and
– cold and warm resets, standby, and other state changes that can affect the RISK CONTROL measures.

Subclause 5.5 – SOFTWARE UNIT implementation

This ACTIVITY requires the MANUFACTURER to write and verify the code for the SOFTWARE UNITS (see Clause B.2 Guidance on Terms and definitions and Software detailed design). The detailed design shall be translated into source code. Coding represents the point where decomposition of the specifications ends and composition of the executable software begins. To consistently achieve the desirable code characteristics, coding standards should be used to specify a preferred coding style. Examples of coding standards include requirements for understandability, language usage rules or restrictions, and complexity management. The code for each unit is VERIFIED to ensure that it functions as specified by the detailed design and that it complies with the specified coding standards.

Subclause 5.5.5 requires the MANUFACTURER to verify the code. If the code does not implement the design correctly, the HEALTH SOFTWARE will not perform as intended.

Subclause 5.6 – Software integration and integration testing

This ACTIVITY requires the MANUFACTURER to plan and execute integration of SOFTWARE UNITS into aggregate SOFTWARE ITEMS as well as integration of SOFTWARE ITEMS into higher aggregated SOFTWARE ITEMS and to verify that the resulting SOFTWARE ITEMS behave as intended.

The approach to integration can range from non-incremental integration to any form of incremental integration. The properties of the SOFTWARE ITEM being assembled dictate the chosen method of integration.

Software integration testing focuses on the transfer of data and control across a SOFTWARE ITEM's internal and external interfaces. External interfaces are those with other software, including operating system software, and product hardware.

The rigour of integration testing and the level of detail of the documentation associated with integration testing should be commensurate with the RISK associated with the device, the device's dependence on software for potentially hazardous functions, and the role of specific SOFTWARE ITEMS in higher RISK device functions. For example, although all SOFTWARE ITEMS should be tested, items that have an effect on SAFETY should be subject to more direct, thorough, and detailed tests.

As applicable, integration testing demonstrates program behaviour at the boundaries of its input and output domains and confirms program responses to invalid, unexpected, and special inputs. The program's actions are revealed when given combinations of inputs or unexpected sequences of inputs, or when defined timing requirements are violated. The test requirements in the plan should include, as appropriate, the types of white box testing to be performed as part of integration testing.

White box testing (also known as "glass box testing", "structural testing", "clear box testing" and "open box testing") is a testing technique where explicit knowledge of the internal workings of the SOFTWARE ITEM being tested is used to select the test data. White box testing uses specific knowledge of the SOFTWARE ITEM to examine outputs. The test is accurate only if the tester knows what the SOFTWARE ITEM is supposed to do. The tester can then see if the SOFTWARE ITEM diverges from its intended goal. White box testing cannot guarantee that the complete specification has been implemented since it is focused on testing the implementation of the SOFTWARE ITEM. Black box testing (also known as "behavioural testing", "functional testing", "opaque-box testing", and "closed-box testing") is focused on testing the functional specification, and it cannot guarantee that all parts of the implementation have been tested. Thus black box testing is testing against the specification and will discover faults of omission, indicating that part of the specification has not been fulfilled. White box testing is testing against the implementation and will discover faults of commission, indicating that part of the implementation is faulty. In order to fully test HEALTH SOFTWARE, both black and white box testing might be required.

The plans and test documentation identified in 5.6 and 5.7 can be individual documents tied to specific phases of development or evolutionary prototypes. They also might be combined so a single document or set of documents covers the requirements of multiple subsections. All or portions of the documents could be incorporated into higher level project documents such as a software or project quality assurance plan or a comprehensive test plan that addresses all aspects of testing for hardware and software. In these cases, a cross reference should be created that identifies how the various project documents relate to each of the software integration TASKS.

Software integration testing can be performed in a simulated environment, on actual target hardware, or on the full SYSTEM.

Subclause 5.6.2 requires the MANUFACTURER to verify the output of the software integration ACTIVITY. The output of the software integration ACTIVITY is the integrated SOFTWARE ITEMS. It

is necessary that these integrated SOFTWARE ITEMS function properly for the entire HEALTH SOFTWARE to function correctly and safely.

Subclause 5.7 – SOFTWARE SYSTEM testing

This ACTIVITY requires the MANUFACTURER to verify the software's functionality by verifying that the requirements for the software have been successfully implemented.

SOFTWARE SYSTEM testing demonstrates that the specified functionality exists. This testing VERIFIES the functionality and performance of the program as built with respect to the requirements for the software.

SOFTWARE SYSTEM testing focuses on functional (black box) testing, although it might be desirable to use white box (see B.2 Guidance for Software integration and integration testing) methods to more efficiently accomplish certain tests, initiate stress conditions or faults, or increase code coverage of the qualification tests. The organization of testing by types and test stage is flexible, but coverage of requirements, RISK CONTROL, USABILITY, and test types (e.g. fault, installation, stress) should be demonstrated and documented.

SOFTWARE SYSTEM testing tests the integrated software and can be performed in a simulated environment, on actual target hardware, or on the full SYSTEM.

When a change is made to a SOFTWARE SYSTEM (even a small change), the degree of REGRESSION TESTING (not just the testing of the individual change) should be determined to ensure that no unintended side effects have been introduced. This REGRESSION TESTING (and the rationale for not fully repeating SOFTWARE SYSTEM testing) should be planned and documented (see B.2 Guidance for Modification implementation).

SOFTWARE SYSTEM test responsibilities can be dispersed, occurring at different locations and being conducted by different organizations. However, regardless of the distribution of TASKS, contractual relations, source of components, or development environment, the device MANUFACTURER retains ultimate responsibility for ensuring that the software functions properly for its INTENDED USE.

If ANOMALIES uncovered during testing can be repeated, but a decision has been made not to fix them, then these ANOMALIES need to be EVALUATED in relation to the RISK ANALYSIS to verify that they do not affect the SAFETY nor SECURITY of the device. The root cause and symptoms of the ANOMALIES should be understood, and the rationale for not fixing them should be documented.

Subclause 5.7.4 requires the results of the SOFTWARE SYSTEM testing be EVALUATED to ensure that the expected results were obtained.

Subclause 5.8 – Software release

The release activities described in 5.8 are required if the HEALTH SOFTWARE is released for its INTENDED USE.

This ACTIVITY requires the MANUFACTURER to document the VERSION of the HEALTH SOFTWARE being released, specify how it was created, and follow appropriate procedures for release of the software for INTENDED USE.

The MANUFACTURER should be able to show that the software that was developed using the development PROCESS is the software that is being released for INTENDED USE. The MANUFACTURER should also be able to retrieve the software and the tools used for its generation in case it is needed in the future and should store, package, and deliver the software in a manner that minimizes the software from being damaged or misused. Defined procedures

should be established to ensure that these TASKS are performed appropriately and with consistent results.

Some or all release ACTIVITIES can also be applied if the MANUFACTURER, in accordance with its configuration management PROCESS, applies a formal release step for other purposes, for example, release for integration into a larger SYSTEM, SYSTEM VERIFICATION or SYSTEM VALIDATION.

Subclause 6.1 – Establish SOFTWARE MAINTENANCE plan

The SOFTWARE MAINTENANCE PROCESS differs from the software development PROCESS in two ways.

– The MANUFACTURER is permitted to use a smaller PROCESS than the full software development PROCESS to implement rapid changes in response to urgent problems.

– In responding to software PROBLEMS REPORTS relating to released product, the MANUFACTURER not only addresses the problem but also satisfies local regulations (typically by running a pro-active surveillance scheme for collecting problem data from the field and communicating with users and regulators about the problem).

Subclause 6.1 requires these PROCESSES to be established in a maintenance plan.

This ACTIVITY requires the MANUFACTURER to create or identify procedures for implementing maintenance ACTIVITIES and TASKS. To implement corrective actions, control changes during maintenance, and manage release of revised software, the MANUFACTURER should document and resolve reported problems and requests from users, as well as manage modifications to HEALTH SOFTWARE. This PROCESS is activated when HEALTH SOFTWARE undergoes modifications to code and associated documentation because of either a problem or the need for improvement or adaptation. The objective is to modify HEALTH SOFTWARE released for INTENDED USE while preserving its integrity. This PROCESS includes migration of HEALTH SOFTWARE to environments or platforms for which it was not originally released. The ACTIVITIES provided in 6.1 are specific to the maintenance PROCESS; however, the maintenance PROCESS might use other PROCESSES in this document.

The MANUFACTURER needs to plan how the ACTIVITIES and TASKS of the maintenance PROCESS will be performed.

Subclause 6.2 – Problem and modification analysis

This ACTIVITY requires the MANUFACTURER to analyse feedback for its effect; verify reported problems; and consider, select, and obtain approval for implementing a modification option. Problems and other requests for changes can affect the performance, SAFETY, or regulatory clearance of a MEDICAL DEVICE. An analysis is necessary to determine whether any effects exist because of a PROBLEM REPORT or whether any effects will result from a modification to correct a problem or implement a request. It is especially important to verify through trace or regression analysis that the RISK CONTROL measures built into the device are not adversely changed or modified by the software change that is being implemented as part of the SOFTWARE MAINTENANCE ACTIVITY. It is also important to verify that the modified software does not cause a HAZARDOUS SITUATION or increase. The software process rigor level of a SOFTWARE ITEM might have changed if the software modification now can cause a HAZARDOUS SITUATION or mitigate a RISK.

It is important to distinguish between SOFTWARE MAINTENANCE (Clause 6) and software problem resolution (Clause 9).

The focus of the SOFTWARE MAINTENANCE PROCESS is an adequate response to feedback arising after release of HEALTH SOFTWARE for INTENDED USE. As part of a SYSTEM, the SOFTWARE MAINTENANCE PROCESS needs to ensure that

– SAFETY-related PROBLEM REPORTS are addressed and reported to appropriate regulatory authorities and affected users,

– HEALTH SOFTWARE is re-validated and re-released after modification with formal controls that ensure the rectification of the problem and the avoidance of further problems, and

– the MANUFACTURER considers what other HEALTH SOFTWARE might be affected and takes appropriate action.

The focus of software problem resolution is the operation of a comprehensive control SYSTEM that

– analyses PROBLEM REPORTS and identifies all the implications of the problem,

– decides on a number of changes and identifies all their side-effects,

– implements the changes while maintaining the consistency of the software CONFIGURATION ITEMS including the RISK MANAGEMENT FILE, and

– VERIFIES the implementation of the changes.

The SOFTWARE MAINTENANCE PROCESS uses the software problem resolution PROCESS. The SOFTWARE MAINTENANCE PROCESS handles the high-level decisions about the PROBLEM REPORT (whether a problem exists, whether it has a significant effect on SAFETY, what changes are needed and when to implement them), and uses the software problem resolution PROCESS to analyse the PROBLEM REPORT to discover all the implications and to generate possible CHANGE REQUESTS which identify all the CONFIGURATION ITEMS that need to be changed and all the VERIFICATION steps that are necessary.

Subclause 6.3 – Modification implementation

This ACTIVITY requires that the MANUFACTURER use an established PROCESS to make the modification. If a maintenance PROCESS has not been defined, the appropriate development PROCESS TASKS can be used to make the modification. The MANUFACTURER should also ensure that the modification does not cause a negative effect on other parts of the HEALTH SOFTWARE. Unless the HEALTH SOFTWARE is treated as a new development, analysis of the effect of a modification on the entire HEALTH SOFTWARE is necessary. Regression analysis and testing are employed to provide assurance that a change has not created problems elsewhere in the HEALTH SOFTWARE. Regression analysis is the determination of the impact of a change based on review of the relevant documentation (e.g. software requirements specification, software design specification, source code, test plans, test cases, test scripts, etc.) in order to identify the necessary regression tests to be run. REGRESSION TESTING is the rerunning of test cases that a program has previously executed correctly and comparing the current result to the previous result in order to detect unintended effects of a software change. A rationale should be made that justifies the amount of REGRESSION TESTING that will be performed to ensure that the portions of the HEALTH SOFTWARE not being modified still perform as they did before the modification was made.

Clause 7 – Software RISK MANAGEMENT PROCESS

Software RISK MANAGEMENT is a part of overall SYSTEM RISK MANAGEMENT and cannot be adequately addressed in isolation. This document requires the use of a RISK MANAGEMENT PROCESS. The software RISK MANAGEMENT PROCESS is included in this document for two reasons:

a) the intended audience of this document needs to understand minimum requirements for RISK CONTROL measures in their area of responsibility software;

Software RISK MANAGEMENT is a part of overall SYSTEM RISK MANAGEMENT. Plans, procedures, and documentation required for the software RISK MANAGEMENT ACTIVITIES can be a series of separate documents or a single document, or they can be integrated with the SYSTEM RISK MANAGEMENT ACTIVITIES and documentation as long as all requirements in this document are met.

Subclause 7.1 – Analysis of software contributing to HAZARDOUS SITUATIONS

It is expected that the SYSTEM RISK ANALYSIS will identify HAZARDOUS SITUATIONS and corresponding RISK CONTROL measures to reduce the probability and/or severity of those HAZARDOUS SITUATIONS to an acceptable level. It is also expected that the RISK CONTROL measures will be assigned to software functions that are expected to implement those RISK CONTROL measures.

However, it is not expected that all SYSTEM HAZARDOUS SITUATIONS can be identified until the software ARCHITECTURE has been produced. At that time, it is known how software functions will be implemented in SOFTWARE ITEMS, and the practicality of the RISK CONTROL measures assigned to SOFTWARE ITEMS can be EVALUATED. At that time, the SYSTEM RISK ANALYSIS should be revised to include

– revised HAZARDOUS SITUATIONS,

– revised RISK CONTROL measures and software requirements, and

– new HAZARDOUS SITUATIONS arising from software, for example HAZARDOUS SITUATIONS related to human factors.

The software ARCHITECTURE should include credible strategies for segregating SOFTWARE ITEMS so that they do not interact in unsafe ways.

Further guidance on ways software can contribute to a HAZARDOUS SITUATION can be found in IEC TR 80002-1:2009 [14], **Error! Reference source not found.**, and Annex B, which has examples of categories of defects or causes contributing to HAZARDOUS SITUATIONS.

B.2 Guidance for Assigning software process rigor level also provides additional guidance on the identification of HAZARDOUS SITUATIONS.

Clause 8 – Software configuration management PROCESS

The software configuration management PROCESS is a PROCESS of applying administrative and technical procedures throughout the software life cycle to identify and define SOFTWARE ITEMS, including documentation, in a SYSTEM; control modifications and releases of the items; and document and report the status of the items and CHANGE REQUESTS. Software configuration management is necessary to recreate a SOFTWARE ITEM, to identify its constituent parts, and to provide a history of the changes that have been made to it.

Subclause 8.1 – Configuration identification

This ACTIVITY requires the MANUFACTURER to uniquely identify software CONFIGURATION ITEMS and their VERSIONS. This identification is necessary to identify the software CONFIGURATION ITEMS and the VERSIONS that are included in HEALTH SOFTWARE.

Subclause 8.2 – Change control

This ACTIVITY requires the MANUFACTURER to control changes of the software CONFIGURATION ITEMS and to document information identifying CHANGE REQUESTS and providing documentation about their disposition. This ACTIVITY is necessary to ensure that unauthorized or unintended changes are not made to the software CONFIGURATION ITEMS and to ensure that approved CHANGE REQUESTS are implemented fully and verified.

CHANGE REQUESTS can be approved by a change control board or by a manager or technical lead according to the software configuration management plan. Approved CHANGE REQUESTS are made traceable to the actual modification and VERIFICATION of the software. The requirement is that each actual change be linked to a CHANGE REQUEST and that documentation exists to

show that the CHANGE REQUEST was approved. The documentation might be change control board minutes, an approval signature, or a record in a database.

Subclause 8.3 – Configuration status accounting

This ACTIVITY requires the MANUFACTURER to maintain records of the history of the software CONFIGURATION ITEMS. This ACTIVITY is necessary to determine when and why changes were made. Access to this information is necessary to ensure that software CONFIGURATION ITEMS contain only authorized modifications.

Clause 9 – Software problem resolution PROCESS

The software problem resolution PROCESS is a PROCESS for analysing and resolving the problems (including non-conformances), whatever their nature or source, including those discovered during the execution of development, maintenance, or other PROCESSES. The objective is to provide a timely, responsible, and documented means to ensure that discovered problems are analysed and resolved and that trends are recognized. This PROCESS is sometimes called "defect tracking" in software engineering literature. It is called "problem resolution" in ISO/IEC/IEEE 12207 [22]. We have chosen to call it "software problem resolution" in this document.

This ACTIVITY requires that the MANUFACTURER use the software problem resolution PROCESS when a problem or non-conformance is identified. This ACTIVITY is necessary to ensure that discovered problems are analysed and EVALUATED for possible relevance to SAFETY.

Software development plan(s) or procedures, as required in 5.1, shall address how problems or non-conformances will be handled. This includes specifying at each stage of the life cycle the aspects of the software problem resolution PROCESS that will be formal and documented as well as when problems and nonconformities are to be entered into the software problem resolution PROCESS.

The problem resolution PROCESS has interdependency with the software configuration management PROCESS in that it interacts with controlling changes to CONFIGURATION ITEMS (see B.2 Guidance for Change control). See Figure B.8 for a diagram of the interactions of the two PROCESSES.

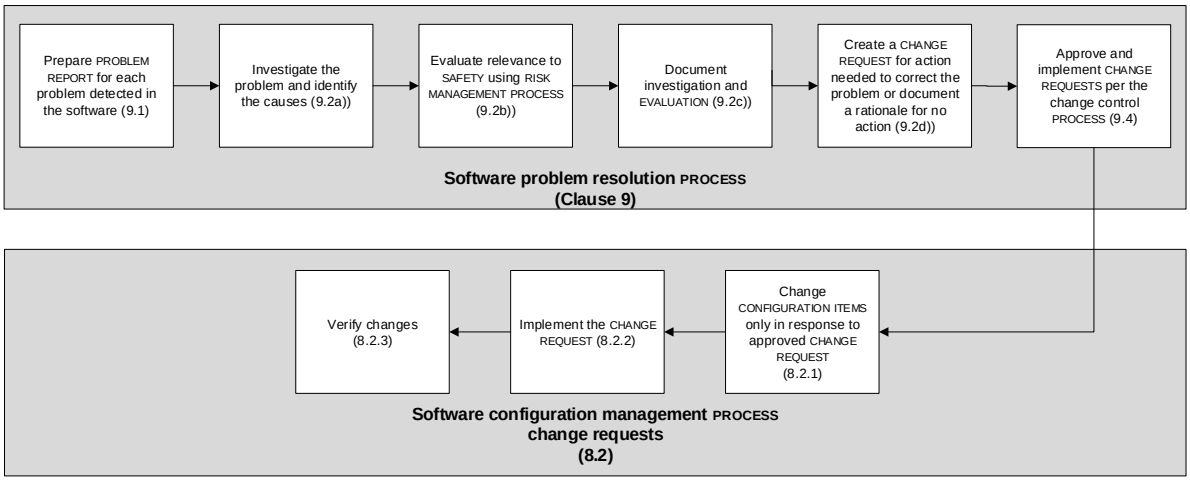


Figure B.8 – Interaction between software problem resolution and software configuration management

The maintenance PROCESS (Clause 6) also requires the use of the problem resolution PROCESS to document and evaluate feedback on the HEALTH SOFTWARE.

Another important TASK within the problem resolution PROCESS is trending quality data (9.6). This TASK is important for good product performance and SAFETY monitoring of the HEALTH SOFTWARE once it is in use. In some jurisdictions, there is a requirement for collecting quality data and trending of this data. This data can come from many sources such as service records, complaints, market surveys, and audit reports (see [9]). See Clause 6 (SOFTWARE MAINTENANCE PROCESS) and the ACTIVITIES of establishing a maintenance plan that address receiving, documenting, evaluating, resolving, and tracking feedback after the release of HEALTH SOFTWARE for INTENDED USE (6.2). The expectation is that, if the MANUFACTURER'S HEALTH SOFTWARE is not performing well, then they will trend, analyse the cause, and fix it (see 6.2.1, 6.2.2, and Clause 9).

Annex C
(informative)

Relationship to other standards

C.1 General

This document applies to the development and maintenance of HEALTH SOFTWARE and draws up the foundational principles, concepts and terms for the SAFETY, EFFECTIVENESS and SECURITY of HEALTH SOFTWARE that are being established in ISO 81001-1[21]. MEDICAL DEVICE SOFTWARE is a subset of HEALTH SOFTWARE and is considered a subsystem of the MEDICAL DEVICE or is itself a MEDICAL DEVICE. This document is to be used together with other appropriate standards when developing a HEALTH SOFTWARE or a MEDICAL DEVICE.

MEDICAL DEVICE management standards such as ISO 13485 [16] (see Clause C.2 and Annex D) and ISO 14971 (see Clause C.3) provide a management environment that lays a foundation for an organization to develop health products. SAFETY standards such as IEC 60601-1 [1] (see Clause C.4) and IEC 61010-1 [2] (see Clause C.5) give specific direction for creating safe MEDICAL DEVICES. When the software is a part of these MEDICAL DEVICES, this document provides more detailed direction on what is required to develop and maintain safe MEDICAL DEVICE SOFTWARE. Many other standards such as ISO/IEC/IEEE 12207 [22] (see Clause C.8), IEC 61508-3 [3] (see Clause C.9) and ISO/IEC 90003 [38] can be looked to as a source of methods, tools and techniques that can be used to implement the requirements in this document.

Figure C.1 shows the relationship of these standards.

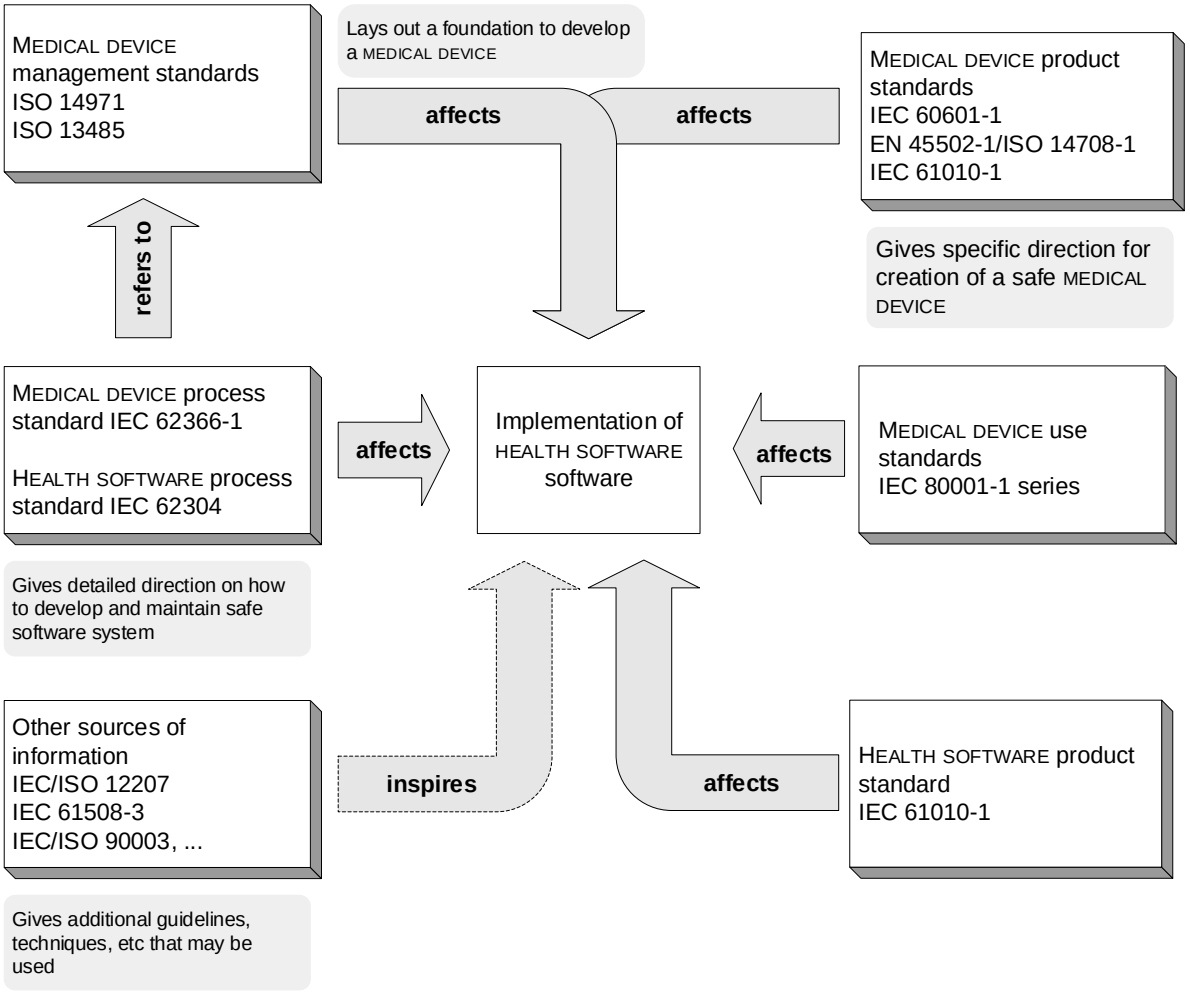


Figure C.1 – Relationship of key MEDICAL DEVICE standards to this document

Where clauses or requirements from other standards are quoted, defined terms in the quoted items are terms that are defined in the other standard, not defined terms in this document.

Poorly implemented SECURITY within HEALTH SOFTWARE can affect patient health and unintentionally expose patient data. There is an explicit relationship between SECURITY and SAFETY (as discussed in AAMI TIR 57 [41]), and this document relies on the MANUFACTURER having a SECURITY threat management PROCESS (4.2 b)) as well as requiring the MANUFACTURER to establish software requirements for the implementation of SECURITY capabilities. Table C.1 provides a list of SECURITY standards that will be helpful to the MANUFACTURER in meeting these requirements.

NOTE The selection of SECURITY standards listed in the table does not represent an exhaustive list of all potentially useful standards.

2451 **Table C.1 – Useful SECURITY standards**

	SECURITY standard	Description
1	AAMI TIR 57 [41]	Provides guidance on methods to perform information SECURITY RISK MANAGEMENT for a MEDICAL DEVICE in the context of the SAFETY RISK MANAGEMENT PROCESS required by ISO 14971. The TIR incorporates the expanded view of RISK MANAGEMENT from IEC 80001-1 by incorporating the same key properties of SAFETY, effectiveness and data and SYSTEMS SECURITY with annexes that provide PROCESS details and illustrative examples.
	AAMI TIR 97 [43]	Provides guidance on methods to perform postmarket security risk management for a medical device in the context of the Safety Risk Management process required by ISO 14971. This TIR is intended to be used in conjunction with AAMI TIR57:2016
2	IEC 80001-1 [11]	Recognizing that MEDICAL DEVICES are incorporated into IT-networks to achieve desirable benefits (for example, interoperability), defines the roles, responsibilities and activities that are necessary for RISK MANAGEMENT of IT-networks incorporating MEDICAL DEVICES to address SAFETY, effectiveness and data and SYSTEM SECURITY (the key properties). It applies to responsible organizations, MEDICAL DEVICE MANUFACTURERS and providers of other information technology for the purpose of RISK MANAGEMENT of an IT-network incorporating MEDICAL DEVICES as specified by the responsible organization.
3	IEC TR 80001-2-2 [12]	Creates a framework for the disclosure of SECURITY-related capabilities and RISKS necessary for managing the RISK in connecting MEDICAL DEVICES to IT-networks and for the SECURITY dialog that surrounds the IEC 80001-1 RISK MANAGEMENT of IT-network connection. This SECURITY report presents an informative set of common, high-level SECURITY-related capabilities useful in understanding the user needs, the type of SECURITY controls to be considered and the RISKS that lead to the controls.
4	IEC TR 80001-2-8 [13]	Provides guidance to health delivery organizations (HDOs) and MEDICAL DEVICE MANUFACTURERS (MDMs) for the application of the framework outlined in IEC TR 80001-2-2. The report presents the 19 SECURITY capabilities, their respective "requirement goal" and "user need" (identical to that in IEC TR 80001-2-2) with a corresponding list of SECURITY controls from a number of SECURITY standards.
	IEC 80001-5-1 [44]	Defines the secure Lifecycle requirements for development and Maintenance of Health Software. The set of Processes, activities, and tasks described in this document establishes a common framework for secure Health Software Lifecycle Processes.
5	ISO/IEC 15408-1 [24]	ISO/IEC 15408-1 is the introduction to the ISO/IEC 15408 series. It defines general concepts and principles of IT SECURITY EVALUATION and presents a general model of EVALUATION. ISO/IEC 15408-1 also presents constructs for expressing IT SECURITY objectives, for selecting and defining IT SECURITY requirements, and for writing high-level specifications for products and SYSTEMS. In addition, the usefulness of each part of the ISO/IEC 15408 series is described in terms of each of the target audiences.
6	ISO/IEC 15408-2 [25]	ISO/IEC 15408-2 defines the content and presentation of the SECURITY functional requirements to be assessed in a SECURITY EVALUATION using the ISO/IEC 15408 series. It contains a comprehensive catalogue of predefined SECURITY functional components that will meet most common SECURITY needs of the marketplace. These are organized using a hierarchical structure of classes, families and components, and supported by comprehensive user notes.
7	ISO/IEC 27001 [34]	Describes best practice for an information SECURITY management system (ISMS).
8	ISO/IEC 27002 [35]	Outlines guidelines for organizational information SECURITY standards and information SECURITY management practices including the selection, implementation and management of controls taking into consideration the organization's information SECURITY RISK environment(s).
9	ISO 27799 [19]	Defines guidelines to support the interpretation and implementation in health informatics of ISO/IEC 27002 and is a companion to that standard. It specifies a set of detailed controls for managing health information SECURITY and provides health information SECURITY best practice guidelines.

SECURITY standard		Description
10	ISO/IEC 27005 [36]	Supports the general concepts specified in ISO/IEC 27001 and is designed to assist the satisfactory implementation of information SECURITY based on a RISK MANAGEMENT approach.
11	IEC TS 62443-1-1 [6]	Defines the terminology, concepts and models for industrial automation and control systems (IACS) security.
12	IEC TR 62443-3-1 [7]	Provides a current assessment of various cybersecurity tools, mitigation counter-measures, and technologies that can effectively apply to the modern electronically based IACSs.
13	IEC 62443-4-1 [8]	Specifies the PROCESS requirements for the secure development of products used in industrial automation and control systems (IACSs). The life cycle description includes SECURITY requirements definition, secure design, secure implementation (including coding guidelines), VERIFICATION and VALIDATION, defect management, patch management and product end-of-life. These requirements can be applied to new or existing PROCESSES for developing, maintaining and retiring hardware, software or firmware.
14	IEC 62443-4-2 [9]	Provides detailed technical control SYSTEM component requirements associated with the seven foundational requirements described in IEC TS 62443-1-1 including defining the requirements for control SYSTEM capability SECURITY levels and their components.
15	UL 2900-2-1 [45]	Describes the method by which the SECURITY RISK CONTROLS of healthcare SYSTEM components shall be evaluated and tested for known vulnerabilities, software weaknesses and malware while also establishing a foundational set of VERIFICATION activities intended to reduce the likelihood of exploitable weaknesses that could be vectors of zero day vulnerabilities that can affect the component.

C.2 Relationship to ISO 13485

This document requires that the MANUFACTURER employs a quality management SYSTEM. When a MANUFACTURER uses ISO 13485 [16], the requirements of this document directly relate to some of the requirements of ISO 13485 as shown in Table C.2.

Table C.2 – Relationship to ISO 13485:2016

Clauses and subclauses in this document	Related subclauses of ISO 13485:2016
5.1 Software development planning	7.3.2 Design and development planning
5.2 Software requirements analysis	7.3.3 Design and development inputs
5.3 Software ARCHITECTURAL design	
5.4 Software detailed design	
5.5 SOFTWARE UNIT implementation	
5.6 Software integration and integration testing	
5.7 SOFTWARE SYSTEM testing	7.3.4 Design and development outputs 7.3.5 Design and development review
5.8 Software release	7.3.6 Design and development VERIFICATION 7.3.7 Design and development VALIDATION
6.1 Establish SOFTWARE MAINTENANCE plan	7.3.9 Control of design and development changes
6.2 Problem and modification analysis	
6.3 Modification implementation	7.3.6 Design and development VERIFICATION 7.3.7 Design and development VALIDATION
Error! Reference source not found. Analysis of software contributing to HAZARDOUS SITUATIONS	
7.2 RISK CONTROL measures	

Clauses and subclauses in this document	Related subclauses of ISO 13485:2016
7.3 VERIFICATION of RISK CONTROL measures	7.3.6 Design and development VERIFICATION 7.3.7 Design and development VALIDATION
7.4 RISK MANAGEMENT of software changes	
8.1 Configuration identification	7.5.8 Identification 7.5.9 TRACEABILITY
8.2 Change control	7.5.8 Identification 7.5.9 TRACEABILITY
8.3 Configuration status accounting	
9 Software problem resolution PROCESS	

C.3 Relationship to ISO 14971

This document requires that the MANUFACTURER employs a risk management SYSTEM. When a MANUFACTURER uses ISO 14971, Table C.3 the requirements of this document directly relate to some of the requirements of ISO 14971 as shown in Table C.3.

Table C.3 – Relationship to ISO 14971:2019

ISO 14971:2019 clauses and subclauses		Related subclauses in this document	
4.5	RISK MANAGEMENT FILE	7.3.2	Document TRACEABILITY
5.1	RISK ANALYSIS PROCESS		
5.2	INTENDED USE and reasonably foreseeable misuse		
5.3	Identification of characteristics related to SAFETY		
5.4	Identification of HAZARDS and HAZARDOUS SITUATIONS	**Error! Reference source not found.**	Analysis of software contributing to HAZARDOUS SITUATIONS
5.5	RISK ESTIMATION	4.4	Software process rigor level
6	RISK EVALUATION		
7.1	RISK CONTROL option analysis	7.2.1	Define RISK CONTROL measures
7.2	Implementation of RISK CONTROL measures	7.2.2 7.3.1	RISK CONTROL measures implemented in software Verify RISK CONTROL measures
7.3	RESIDUAL RISK EVALUATION		
7.4	Benefit-RISK ANALYSIS		
7.5	RISKS arising from RISK CONTROL MEASURES		
7.6	Completeness of RISK CONTROL		
8	EVALUATION of overall RESIDUAL RISK		
9	RISK MANAGEMENT review		
10	Production and post-production activities	7.4	RISK MANAGEMENT of software changes

C.4 Relationship to PEMS requirements of IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012

C.4.1 General

Requirements for software are a subset of the requirements for a programmable electrical medical system (PEMS). This document identifies requirements for software which are in

addition to, but not incompatible with, the requirements of IEC 60601-1 [1] for PEMS. Because PEMS include elements that are not software, not all the requirements of IEC 60601-1 for PEMS are addressed in this document. With the publication of IEC 60601-1, this document is now a normative reference of IEC 60601-1, and conformance with Clause 14 of IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012 (and thus conformance with IEC 60601-1) requires conformance with parts of IEC 62304 (not with the whole of IEC 62304 because IEC 60601-1 does not require conformance with post-production and maintenance requirements of IEC 62304). Finally, it is important to remember that IEC 60601-1 is only used if the software is part of a PEMS and not if the software is itself a MEDICAL DEVICE.

C.4.2 Software relationship to PEMS development

By using the V-model illustrated in Figure C.2 to describe what occurs during a PEMS development, it can be seen that the requirements of this document apply at the PEMS component level, from the specification of the software requirements to the integration of the SOFTWARE ITEMS into a SOFTWARE SYSTEM. This SOFTWARE SYSTEM is a part of a programmable electrical subsystem (PESS), which is a part of a PEMS.

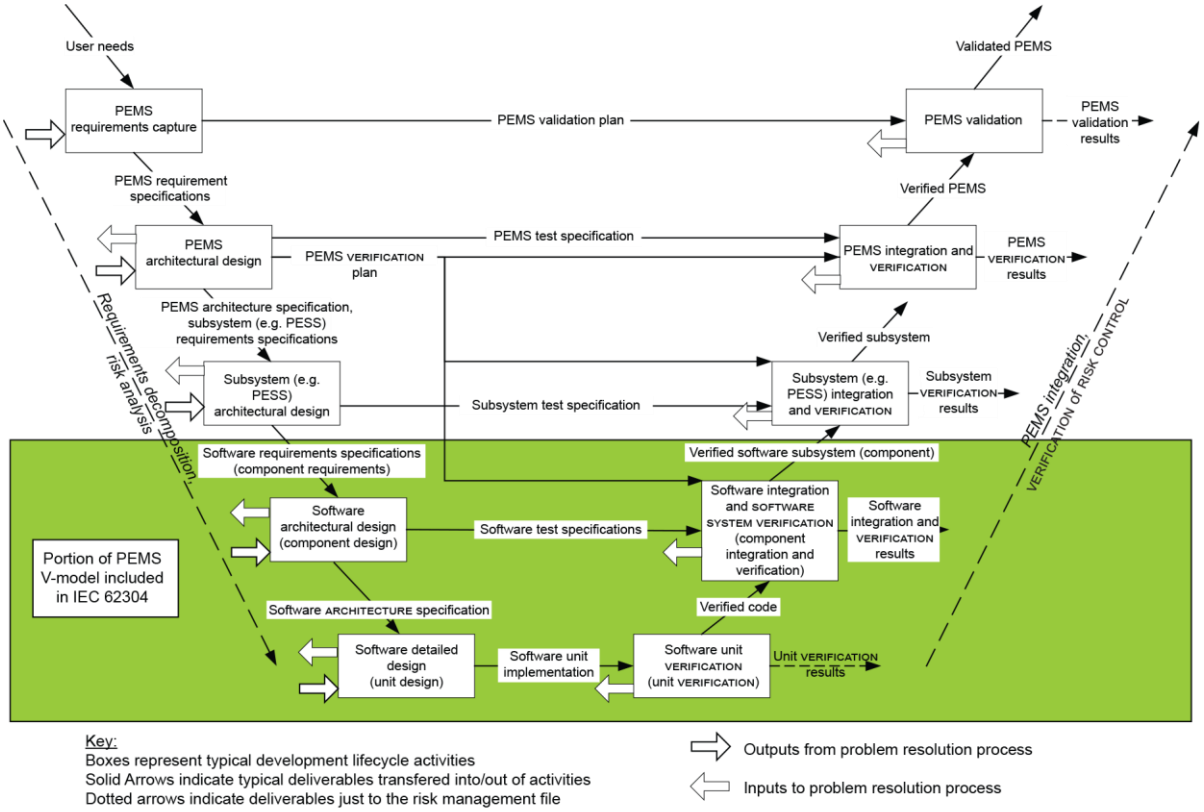


Figure C.2 – Software as part of the V-model

C.4.3 Development PROCESS

Conformance with the software development PROCESS of this document (Clause 5) requires that a software development plan be specified and then followed; it does not require that any particular life cycle model be used, but it does require that the plan include certain ACTIVITIES and have certain attributes. These requirements relate to the PEMS requirements in IEC 60601-1 [1] for development life cycle, requirement specification, ARCHITECTURE, design and implementation, and VERIFICATION. The requirements in this document provide greater detail about software development than those in IEC 60601-1.

C.4.4 Maintenance PROCESS

Conformance with the SOFTWARE MAINTENANCE PROCESS of this document (Clause 6) requires that procedures be established and followed when changes to software are made. These requirements correspond to the requirement in IEC 60601-1 for modification of a PEMS. The requirements in this document for SOFTWARE MAINTENANCE provide greater detail for SOFTWARE MAINTENANCE than the requirements for PEMS modification in IEC 60601-1 [1].

C.4.5 Other PROCESSES

The other PROCESSES in this document specify additional requirements for software beyond the similar requirements for PEMS in IEC 60601-1 [1]. In most cases, there is a general requirement for PEMS in IEC 60601-1, which the PROCESSES in this document expand upon.

The software RISK MANAGEMENT PROCESS in this document corresponds to the additional RISK MANAGEMENT requirements identified for PEMS in IEC 60601-1.

The software problem resolution PROCESS in this document corresponds to the problem resolution requirement for PEMS in IEC 60601-1.

The software configuration management PROCESS in this document specifies additional requirements that are not present for PEMS in IEC 60601-1 except for documentation.

C.4.6 Coverage of PEMS requirements in IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012

Table C.4 shows the PEMS requirements of IEC 60601-1 [1] and the corresponding requirements in this document.

Table C.4 – Relationship to IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012

PEMS requirements from IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012	Requirements of this document relating to the software subsystem of a PEMS
14.1 General The requirements in 14.2 to 14.12 (inclusive) shall apply to PEMS unless: – none of the PROGRAMMABLE ELECTRONIC SUBSYSTEMS (PESS) provides functionality necessary for BASIC SAFETY or ESSENTIAL PERFORMANCE; or – the application of RISK MANAGEMENT as described in 4.2 demonstrates that the failure of the PESS does not lead to an unacceptable RISK. The requirements in 14.13 are applicable to any PEMS intended to be incorporated into an IT-network whether or not the requirements in 14.2 to 14.12 apply. When the requirements in 14.2 to 14.13 apply, the requirements in 4.3, Clause 5, Clause 7, Clause 8 and Clause 9 of IEC 62304:2006 shall also apply to the development or modification of software for each PESS.	**4.4 Software process rigor level** The PEMS requirements 14.2 to 14.12 of IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012 apply: – when the PESS provides functionality necessary for basic SAFETY or essential performance; and – when application of RISK MANAGEMENT demonstrates that a failure of the PESS leads to unacceptable RISK. This document's requirements corresponding to software process rigor levels B and C apply. This document also includes requirements for software in a PESS that is demonstrated to have acceptable RISK, i.e. when IEC 60601-1 requirements 14.2 to 14.12 do not apply. This implies this document's software process rigor level A. The software development PROCESS required for conformance with IEC 60601-1 does not include the post-production monitoring and maintenance required by Clause 6.
14.2 Documentation The documents required by Clause 14 shall be reviewed, approved, issued and changed in accordance with a formal document control procedure.	**5.1 Software development planning** In addition to the specific requirements in the software development planning ACTIVITY, documents that are part of the RISK MANAGEMENT FILE are required to be maintained by ISO 14971. In addition, for documents that are required by the quality SYSTEM, ISO 13485 [16] requires control of the documents.

PEMS requirements from IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012	Requirements of this document relating to the software subsystem of a PEMS
14.3 Risk management plan The RISK MANAGEMENT plan required by 4.2.2 shall also include a reference to the PEMS VALIDATION plan (see 14.11).	Not specifically required. There is no specific VALIDATION plan. The PEMS VALIDATION plan is at the SYSTEM level and thus is outside the scope of this document. This document does require TRACEABILITY from HAZARD to specific software cause to RISK CONTROL measure to VERIFICATION of the RISK CONTROL measure (see 7.3).
14.4 PEMS DEVELOPMENT LIFE-CYCLE A PEMS DEVELOPMENT LIFE-CYCLE shall be documented.	**5.1 Software development planning** **5.1.1 Software development plan** The items addressed by the software development plan constitute a software development life cycle.
The PEMS DEVELOPMENT LIFE-CYCLE shall contain a set of defined milestones.	
At each milestone, the ACTIVITIES to be completed and the VERIFICATION methods to be applied to those activities shall be defined.	**5.1.6 Software VERIFICATION planning** VERIFICATION TASKS, milestones and acceptance criteria shall be planned.
Each ACTIVITY shall be defined, including its inputs and outputs.	**5.1.1 Software development plan** ACTIVITIES are defined in this document. Documentation to be produced is defined in each ACTIVITY.
Each milestone shall identify the RISK MANAGEMENT ACTIVITIES that shall be completed before that milestone.	
The PEMS DEVELOPMENT LIFE-CYCLE shall be tailored for a specific development by making plans which detail ACTIVITIES, milestones and schedules.	**5.1.1 Software development plan** This document allows the development life cycle to be documented in the development plan. This means the development plan contains a tailored development life cycle.
The PEMS DEVELOPMENT LIFE-CYCLE shall include documentation requirements.	**5.1.1 Software development plan** **5.1.8 Documentation planning**
14.5 Problem resolution Where appropriate, a documented SYSTEM for problem resolution within and between all phases and ACTIVITIES of the PEMS DEVELOPMENT LIFE-CYCLE shall be developed and maintained.	**9 Software problem resolution PROCESS**
Depending on the type of product, the problem resolution SYSTEM may: – be documented as a part of the PEMS DEVELOPMENT LIFE-CYCLE; – allow the reporting of potential or existing problems affecting BASIC SAFETY or ESSENTIAL PERFORMANCE; – include an assessment of each problem for associated RISKS; – identify the criteria that shall be met for the issue to be closed; – identify the action to be taken to resolve each problem.	**5.1.1 Software development plan** **9.1 Prepare PROBLEM REPORTS**
14.6 RISK MANAGEMENT PROCESS	**7 Software RISK MANAGEMENT PROCESS**
14.6.1 Identification of known and foreseeable HAZARDS When compiling the list of known or foreseeable HAZARDS, the MANUFACTURER shall consider those HAZARDS associated with software and hardware aspects of the PEMS including those associated with the incorporation of the PEMS into an IT-NETWORK, components of third-party origin and legacy subsystems.	Error! Reference source not found. **Analysis of software contributing to HAZARDOUS SITUATIONS** This document does not mention network/data coupling specifically.

PEMS requirements from IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012	Requirements of this document relating to the software subsystem of a PEMS
14.6.2 RISK CONTROL Suitably validated tools and PROCEDURES shall be selected and identified to implement each RISK CONTROL measure. These tools and PROCEDURES shall be appropriate to assure that each RISK CONTROL measure satisfactorily reduces the identified RISK(S).	**5.1.4 Software development standards, methods and tools planning** This document requires the identification of specific tools and methods to be used for development in general, not for each RISK CONTROL measure.
14.7 Requirements specification For the PEMS and each of its subsystems (e.g. for a PESS), there shall be a documented requirement specification.	**5.2 Software requirements analysis** This document deals only with the software subsystems of a PEMS.
The requirement specification for a SYSTEM or subsystem shall include and distinguish any ESSENTIAL PERFORMANCE and any RISK CONTROL measures implemented by that SYSTEM or subsystem.	**5.2.1 Define and document software requirements from SYSTEM requirements** **5.2.2 Software requirements content** **5.2.3 Include RISK CONTROL measures in software requirements** This document does not require that the requirements related to essential performance and RISK CONTROL measures be distinguished from other requirements, but it does require that all requirements be uniquely identified.

PEMS requirements from IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012	Requirements of this document relating to the software subsystem of a PEMS
14.8 ARCHITECTURE For the PEMS and each of its subsystems, an ARCHITECTURE shall be specified that shall satisfy the requirements specification.	**5.3 Software ARCHITECTURAL design**
Where appropriate, to reduce the RISK to an acceptable level, the ARCHITECTURE specification shall make use of: a) COMPONENTS WITH HIGH-INTEGRITY CHARACTERISTICS; b) fail-safe functions; c) redundancy; d) diversity; e) partitioning of functionality; f) defensive design, e.g. limits on potentially hazardous effects by restricting the available output power or by introducing means to limit the travel of actuators.	**5.3.5 Identify segregation necessary for RISK CONTROL** Partitioning is the only technique identified, and it is only identified because there is a requirement to state how the integrity of the partitioning is assured.
The ARCHITECTURE specification shall take into consideration: g) allocation of RISK CONTROL measures to subsystems and components of the PEMS; h) failure modes of components and their effects; i) common cause failures; j) systemic failures; k) test interval duration and diagnostic coverage; l) maintainability; m) protection from reasonably foreseeable misuse; n) the IT-NETWORK specification, if applicable.	This is not included in this document.
14.9 Design and implementation Where appropriate, the design shall be decomposed into subsystems, each having both a design and test specification.	**5.4 Software detailed design** **5.4.2 Develop detailed design for each SOFTWARE UNIT** This document does not require a test specification for detailed design.
Descriptive data regarding the design environment shall be documented.	**5.4.2 Develop detailed design for each SOFTWARE UNIT**
14.10 VERIFICATION VERIFICATION is required for all functions that implement BASIC SAFETY, ESSENTIAL PERFORMANCE or RISK CONTROL measures.	**5.1.6 Software VERIFICATION planning** VERIFICATION is required for each ACTIVITY.
A VERIFICATION plan shall be produced to show how these functions shall be verified. The plan shall include: – at which milestone(s) VERIFICATION is to be performed for each function; – the selection and documentation of VERIFICATION strategies, ACTIVITIES, techniques, and the appropriate level of independence of the personnel performing the VERIFICATION; – the selection and utilization of VERIFICATION tools; – coverage criteria for VERIFICATION.	**5.1.6 Software VERIFICATION planning** Independence of personnel is not included in this document. It is considered covered in ISO 13485.
The VERIFICATION shall be performed according to the VERIFICATION plan. The results of the VERIFICATION activities shall be documented.	VERIFICATION requirements are in most of the ACTIVITIES.
14.11 PEMS VALIDATION A PEMS VALIDATION plan shall include the VALIDATION of BASIC SAFETY and ESSENTIAL PERFORMANCE.	This document does not cover VALIDATION. PEMS VALIDATION is a SYSTEM level ACTIVITY and is outside the scope of this document.
Methods used for PEMS VALIDATION shall be documented.	This document does not cover VALIDATION. PEMS VALIDATION is a SYSTEM level ACTIVITY and is outside the scope of this document.

PEMS requirements from IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012	Requirements of this document relating to the software subsystem of a PEMS
The PEMS VALIDATION shall be performed according to the PEMS VALIDATION plan. The results of the PEMS VALIDATION activities shall be documented.	This document does not cover VALIDATION. PEMS VALIDATION is a SYSTEM level ACTIVITY and is outside the scope of this document.
The person having the overall responsibility for the PEMS VALIDATION shall be independent of the design team. The MANUFACTURER shall document the rationale for the level of independence.	This document does not cover software VALIDATION. PEMS VALIDATION is a SYSTEM level ACTIVITY and is outside the scope of this document.
No member of a design team shall be responsible for the PEMS VALIDATION of their own design.	This document does not cover VALIDATION. PEMS VALIDATION is a SYSTEM level ACTIVITY and is outside the scope of this document.
All professional relationships of the members of the PEMS VALIDATION team with members of the design team shall be documented in the RISK MANAGEMENT FILE.	This document does not cover VALIDATION. PEMS VALIDATION is a SYSTEM level ACTIVITY and is outside the scope of this document.
14.12 Modification If any or all of a design results from a modification of an earlier design, then either all of 14.12 applies as if it were a new design, or the continued validity of any previous design documentation shall be assessed under a documented modification/change PROCEDURE.	**6 SOFTWARE MAINTENANCE PROCESS** This document takes the approach that SOFTWARE MAINTENANCE should be planned and that implementation of modifications should use the software development PROCESS or an established SOFTWARE MAINTENANCE PROCESS.
When software is modified, the requirements in 4.3, Clause 5, Clause 7, Clause 8 and Clause 9 of IEC 62304:2006 shall also apply to the modification.	

PEMS requirements from IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012	Requirements of this document relating to the software subsystem of a PEMS
14.13 PEMS intended to be incorporated into an IT-NETWORK If the PEMS is intended to be incorporated into an IT-NETWORK that is not validated by the PEMS MANUFACTURER, the MANUFACTURER shall make available instructions for implementing such connection including the following: a) the purpose of the PEMS' connection to an IT-NETWORK; b) the required characteristics of the IT-NETWORK incorporating the PEMS; c) the required configuration of the IT-NETWORK incorporating the PEMS; d) the technical specifications of the network connection of the PEMS including SECURITY specifications; e) the intended information flow between the PEMS, the IT-NETWORK and other devices on the IT-NETWORK, and the intended routing through the IT-NETWORK; and NOTE 1 This can include aspects of effectiveness and data and SYSTEM SECURITY as related to BASIC SAFETY and ESSENTIAL PERFORMANCE (see also Clause H.6 of IEC 60601-1:2005 and IEC 60601-1:2005/AMD1:2012 and IEC 80001-1:2010). f) a list of the HAZARDOUS SITUATIONS resulting from a failure of the IT-NETWORK to provide the characteristics required to meet the purpose of the PEMS connection to the IT-NETWORK. NOTE 2 Connecting a PEMS to another piece of equipment for the purpose of transferring data creates a two-node IT-NETWORK. For example, connecting a PEMS to a printer creates an IT-NETWORK. If the MANUFACTURER has validated the PEMS with the printer, the resulting network would be considered within the control of the MANUFACTURER. In the ACCOMPANYING DOCUMENTS, the MANUFACTURER shall instruct the RESPONSIBLE ORGANIZATION that: – connection of the PEMS to an IT-NETWORK that includes other equipment could result in previously unidentified RISKS to PATIENTS, OPERATORS or third parties; – the RESPONSIBLE ORGANIZATION should identify, analyze, evaluate and control these RISKS; NOTE 3 IEC 80001-1:2010 provides guidance for the RESPONSIBLE ORGANIZATION to address these RISKS. – subsequent changes to the IT-NETWORK could introduce new RISKS and require additional analysis; and – changes to the IT-NETWORK include: • changes in the IT-NETWORK configuration; • connection of additional items to the IT- NETWORK; • disconnecting items from the IT-NETWORK; • update of equipment connected to the IT-NETWORK; • upgrade of equipment connected to the IT-network.	This document does not include requirements for instructions for use or accompanying documents.

C.5 Relationship to IEC 61010-1

The scope of IEC 61010-1 [2] covers electrical test and measuring equipment, electrical control equipment and electrical laboratory equipment. Only part of the laboratory equipment is used in a medical environment or as in vitro diagnostic equipment (IVD).

Due to legal regulations or normative references, IVD equipment is allocated to MEDICAL DEVICES without, however, falling within the scope of IEC 60601-1 [1]. This is attributable not only to the fact that, strictly speaking, IVD instruments are not MEDICAL DEVICES which come into direct contact with patients, but also to the fact that such products are manufactured for many different applications in various laboratories. Use as an IVD instrument or as an accessory for an IVD instrument is then rare.

If laboratory equipment is used as IVD equipment, the measured results obtained are EVALUATED in accordance with medical criteria. The application of ISO 14971 is required for RISK MANAGEMENT. If such products also contain software that can lead to a HAZARDOUS SITUATION, for example, failure caused by the software which results in an unwanted change of medical data (measuring results), IEC 62304 shall be taken into account.

IEC 61010-1:2010 [2] has a general requirement for RISK assessment in Clause 17, which is more streamlined than the full RISK MANAGEMENT requirements of ISO 14971. Applying Clause 17 of IEC 61010-1:2010 alone does not meet the required criteria for RISK MANAGEMENT of this document, which is based on full ISO 14971 RISK MANAGEMENT requirements. With this in mind, it is expected by this document that when an IVD MEDICAL DEVICE has software-related RISKS, its RISK MANAGEMENT PROCESS is performed following ISO 14971 instead of only Clause 17 of IEC 61010-1:2010. Conformance with Clause 17 of IEC 61010-1:2010 will be achieved, as detailed in the note to Clause 17 of IEC 61010-1:2010:

"NOTE One RISk assessment procedure is outlined in Annex J. Other RISK ASSESSMENT procedures are contained in ISO 14971, SEMI S10-1296, IEC 61508, ISO 14121-1, and ANSI B11.TR3. Other established procedures which implement similar steps can also be used."

The flowchart in Figure C.3 shows the application of this document with IEC 61010-1:2010, Clause 17.

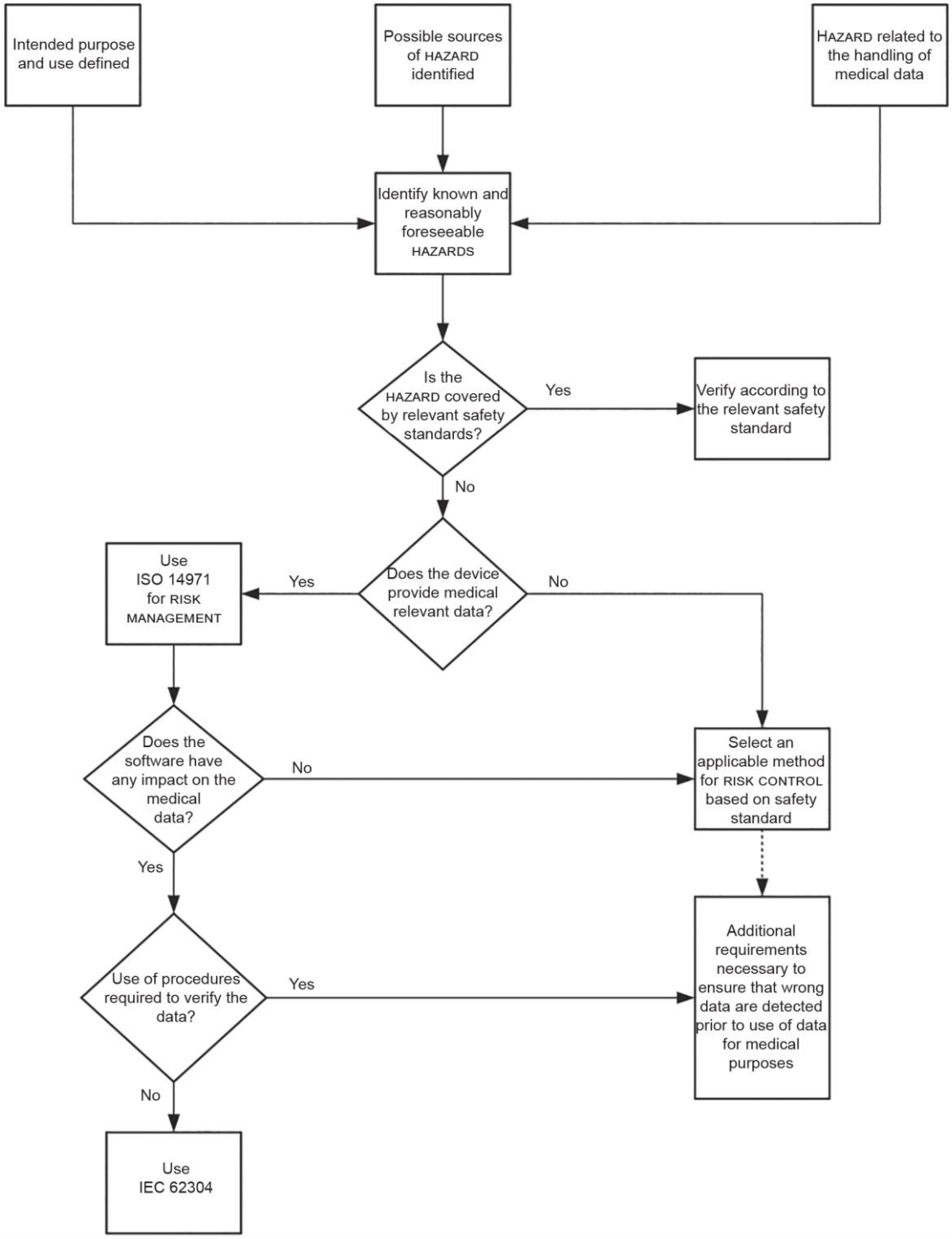


Figure C.3 – Application of IEC 62304 with IEC 61010-1

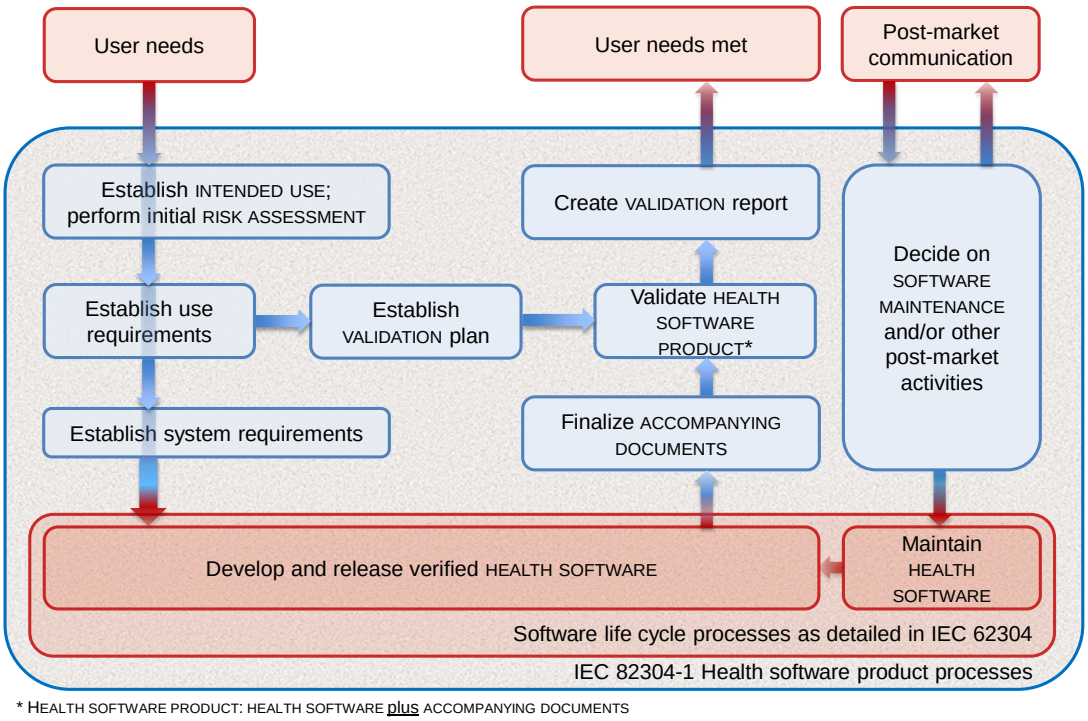
C.6 Relationship to EN 45502-1 and ISO 14708-1

The scope of both EN 45502-1 [42] and ISO 14708-1 [17] cover the general requirements for SAFETY, marking, and information to be provided by the MANUFACTURER for active implantable MEDICAL DEVICES. Both standards have normative references to IEC 62304 for the embedded software of the active implantable device and for the software of any non-implantable parts of the active implantable parts of the SYSTEM.

C.7 Relationship to IEC 82304-1

The scope of IEC 82304-1 [15] is HEALTH SOFTWARE PRODUCTS which are software-only products. These products are intended to be used with computing equipment not explicitly developed for running the software. IEC 62304 has a broader scope than IEC 82304-1 since it applies to embedded software as well as software-only products (see Figure 2).

Figure C.4 shows that IEC 82304-1 calls on IEC 62304 for the software development and maintenance of the HEALTH SOFTWARE in scope of IEC 82304-1. IEC 82304-1 provides the product requirements needed for the HEALTH SOFTWARE product (intended use of the software product, system requirements, VALIDATION of the software product, and accompanying documents for the software product). OBJ



SOURCE: IEC 82304-1:2016, Figure A.2

Figure C.4 – Relationship between IEC 82304-1 and IEC 62304

C.8 Relationship to ISO/IEC/IEEE 12207

This document has been derived from the approach and concepts of ISO/IEC/IEEE 12207 [22], which defines requirements for software life cycle PROCESSES in general, i.e. not restricted to MEDICAL DEVICES.

This document differs from ISO/IEC/IEEE 12207 mainly with respect to the following. This document:

– excludes SYSTEM aspects, such as SYSTEM requirements, SYSTEM ARCHITECTURE and VALIDATION;

– omits some PROCESSES seen as duplicating ACTIVITIES documented elsewhere for MEDICAL DEVICES;

– adds the (SAFETY) RISK MANAGEMENT PROCESS and the software release PROCESS;

– incorporates the documentation and the VERIFICATION supporting PROCESSES into the development and maintenance PROCESSES;

– merges the PROCESS implementation and planning ACTIVITIES of each PROCESS into a single ACTIVITY in the development and maintenance PROCESSES;

– classifies the requirements with respect to SAFETY needs; and

– does not explicitly classify PROCESSES as primary or supporting, nor group PROCESSES as ISO/IEC/IEEE 12207 does.

Most of these changes were driven by the desire to tailor this document to the need of the MEDICAL DEVICE sector by:

– focusing on SAFETY aspects and the MEDICAL DEVICE RISK MANAGEMENT standard ISO 14971;

– selecting the appropriate PROCESSES useful in a regulated environment;

– taking into account that software development is embedded in a quality SYSTEM (which covers some of the PROCESSES and requirements of ISO/IEC/IEEE 12207); and

– lowering the level of abstraction to make it easier to use.

This document is not contradictory to ISO/IEC/IEEE 12207. ISO/IEC/IEEE 12207 can be useful as an aide in setting up a well-structured SOFTWARE DEVELOPMENT LIFE CYCLE MODEL that includes the requirements of this document.

Table C.5 shows the relationship between IEC 62304 and ISO/IEC/IEEE 12207.

2594 **Table C.5 – Relationship to ISO/IEC/IEEE 12207:2017**

IEC 62304 PROCESSES		ISO/IEC/IEEE 12207:2017	
PROCESSES	**ACTIVITY**	**PROCESSES**	**ACTIVITY**
5 Software development PROCESS	5.1 Software development planning	6.2.2 Infrastructure management PROCESS	6.2.2.3 a) Establish the infrastructure 6.2.2.3 b) Maintain the infrastructure
		6.3.1 Project planning PROCESS	6.3.1.3 a) Define the project 6.3.1.3 b) Plan project and technical management
		6.3.2 Project assessment and control PROCESS	6.3.2.3 c) Control the project
		6.3.5 Configuration management PROCESS	6.3.5.3 a) Plan configuration management
		6.3.6 Information management PROCESS	6.3.6.3 a) Prepare for information management
		6.3.8 Quality assurance PROCESS	6.3.8.3 a) Prepare for quality assurance
		6.4.2 Stakeholder needs and requirements definition PROCESS	6.4.2.3 f) Manage stakeholder needs and requirements definition
		6.4.7 Implementation PROCESS	6.4.7.3 a) Prepare for implementation
		6.4.8 Integration PROCESS	6.4.8.3 a) Prepare for integration
		6.4.9 VERIFICATION PROCESS	6.4.9.3 a) Prepare for VERIFICATION
	5.2 Software requirements analysis	6.4.3 SYSTEM/Software requirements definition PROCESS	6.4.3.3 b) Define SYSTEM/software requirements 6.4.3.3 c) Analyze SYSTEM/software requirements 6.4.3.3 d) Manage SYSTEM/software requirements
		6.4.9 VERIFICATION PROCESS	6.4.9.3 a) Perform VERIFICATION
	5.3 Software ARCHITECTURAL design	6.4.4 ARCHITECTURE definition PROCESS	6.4.4.3 d) Relate the ARCHITECTURE to design 6.4.4.3 f) Manage the selected ARCHITECTURE

IEC 62304 PROCESSES		ISO/IEC/IEEE 12207:2017	
PROCESSES	ACTIVITY	PROCESSES	ACTIVITY
	5.4 Software detailed design	6.4.5 Design definition PROCESS	6.4.5.3 b) Establish designs related to each SOFTWARE SYSTEM element 6.4.5.3 d) Manage the design
	5.5 SOFTWARE UNIT implementation	6.4.7 Implementation PROCESS	6.4.7.3 b) Perform implementation 6.4.7.3 c) Manage results of implementation
	5.6 Software integration and integration testing	6.4.8 Integration PROCESS	6.4.8.3 b) Perform integration 6.4.8.3 c) Manage results of integration
	5.7 SOFTWARE SYSTEM testing	6.4.9 VERIFICATION PROCESS	6.4.9.3 b) Perform VERIFICATION 6.4.9.3 c) Manage results of VERIFICATION
	5.8 Software release	6.4.10 Transition PROCESS 6.3.5 Configuration management PROCESS	6.4.10.3.a) Prepare for the SOFTWARE SYSTEM transition 6.3.5.3 d) Perform release control
6 SOFTWARE MAINTENANCE PROCESS	6.1 Establish SOFTWARE MAINTENANCE plan 6.2 Problem and modification analysis 6.3 Modification implementation	6.4.13 Maintenance PROCESS	6.4.13.3 a) Prepare for maintenance 6.4.13.3 b) Perform maintenance 6.4.13.3 d) Manage results of maintenance and logistics
7 Software RISK MANAGEMENT PROCESS	None	6.3.4 RISK MANAGEMENT PROCESS	This is based on ISO/IEC 16085 [26]. While there is some commonality, it does not address the specific requirements for MEDICAL DEVICE software development with regard to RISK MANAGEMENT.
8 Software configuration management PROCESS	None	6.3.5 Configuration management PROCESS 6.4.13 Maintenance PROCESS	None
9 Software problem resolution PROCESS	None	6.3.8 Quality assurance PROCESS 6.3.5 Configuration management PROCESS 6.4.13 Maintenance PROCESS	None

C.9 Relationship to IEC 61508-3

The question has been raised whether this document, being concerned with the design of SAFETY-critical software, should follow the principles of IEC 61508-3 [3]. The approach to SAFETY in this document is fundamentally different than the one in IEC 61508-3. This document

takes into account that the EFFECTIVENESS of MEDICAL DEVICES justifies RESIDUAL RISKS related to their use. The following explains the stance of this document.

IEC 61508-3 addresses 3 main issues:

1) RISK MANAGEMENT life cycle and life cycle PROCESSES;
2) definition of SAFETY integrity levels;
3) recommendation of techniques, tools and methods for software development and levels of independence of personnel responsible for performing different TASKS.

Issue 1) is covered in this document by a normative reference to ISO 14971 (the MEDICAL DEVICE sector standard for RISK MANAGEMENT). The effect of this reference is to adopt ISO 14971's approach to RISK MANAGEMENT as an integral part of the software PROCESS for HEALTH SOFTWARE.

For issue 2), this document takes a simpler approach than IEC 61508-3. The latter classifies software into four SAFETY integrity levels defined in terms of reliability objectives. The reliability objectives are identified after RISK ANALYSIS, which quantifies both the severity and the probability of HARM caused by a failure of the software.

This document simplifies issue 2) by defining the classification into three software process rigor levels based on the RISK caused by a failure. After classification, different PROCESSES are required for different software process rigor levels: the intention is to further reduce the probability (and/or the severity) of failure of the software.

Issue 3) is not addressed by this document. Readers of this document are encouraged to use IEC 61508-3 as a source for good software methods, techniques and tools, while recognising that other approaches, both present and future, can provide equally good results. This document makes no recommendation concerning independence of people responsible for one software ACTIVITY (for example VERIFICATION) from those responsible for another (for example design). In particular, this document makes no requirement for an independent SAFETY assessor, since this is a matter for ISO 14971.

C.10 Relationship to IMDRF SaMD risk categorization

This document applies to SaMD, Software as a MEDICAL DEVICE, which is a subset of HEALTH SOFTWARE. Therefore, the relationship between IMDRF SaMD RISK categorization, its implementation in some International Medical Device Regulators Forum (IMDRF) jurisdictions, and this document is important to describe.

The IMDRF is a voluntary group of MEDICAL DEVICE regulators from around the world. The objective of the IMDRF is to encourage convergence at the global level in the evolution of regulatory frameworks. There was a Software as a MEDICAL DEVICE (SaMD) working group that published 4 documents and their work is now complete. The objectives of the work group were to drive the international convergence and common understanding of SaMD and to establish a framework for regulators to incorporate converged controls into their regulatory paths or classifications.

The IMDRF SaMD WG N12 Software as a MEDICAL DEVICE: Possible Framework for Risk Categorization and Corresponding Considerations is the second document produced by the SaMD work group. The goal of the document is to provide a common approach for regulators to categorize types of SaMD based on their RISK profile and provide common expectations of controls that can be applied by each regulatory jurisdiction. Per the document, “The approach developed in this document is intended only to establish a common understanding for SaMD and can be used as reference. This document is not intended to replace or modify existing regulatory classification schemes or requirements. Further efforts are required prior to the use of this foundational approach for possible regulatory purposes.”

The document does not provide controls as such, but provides general and specific considerations for SaMD. These considerations continue to rely on the use of appropriate technical or process standards related to the development of SaMD (e.g., ISO 14971 and IEC 62304). Per the document, "This document is not intended to replace or create new RISK MANAGEMENT practices rather it uses RISK MANAGEMENT principles (e.g., principles in international standards) to identify generic RISKS for SaMD."

The document provides an approach to SaMD categorization based on factors identified in the SaMD definition statement:

a) The "significance of the information provided by the SaMD to the healthcare decision" which identifies the intended medical purpose of the SaMD. The statement should explain how the SaMD meets one or more of the purposes described in the definition of a MEDICAL DEVICE, e.g. supplying information for diagnosis, prevention, monitoring, treatment etc.

- Treat or diagnose
- Drive clinical management
- Inform clinical management

b) The "state of the healthcare situation or condition" that the SaMD is intended for.

- Critical situation or condition
- Serious situation or condition
- Non-serious situation or condition

c) Description of the SaMD's core functionality which identifies the critical features/functions of the SaMD that are essential to the intended significance of the information provided by the SaMD to the healthcare decision in the intended healthcare situation or condition. This description should include only the critical features.

State of Healthcare situation or condition	Significance of information provided by SaMD to healthcare decision		
	Treat or diagnose	Drive clinical management	Inform clinical management
Critical	IV	III	II
Serious	III	II	I
Non-serious	II	I	I

- The four categories (I, II, III, IV) are based on the levels of impact on the patient or public health where accurate information provided by the SaMD to treat or diagnose, drive or inform clinical management is vital to avoid death, long-term disability or other serious deterioration of health, mitigating public health.
- The categories are in relative significance to each other. Category IV has the highest level of impact, Category I the lowest.

As mentioned above, the IMDRF SaMD document does not provide the specific controls for the four IMDRF SaMD categories (I, II, III, IV). But, it does provide general considerations (design/development and changes) and specific considerations (socio-technical environment, technology and system environment, and information SECURITY with respect to safety). It does point to IEC 62304 as being a standard for life-cycle development of HEALTH SOFTWARE that highlights three major principles that promote SAFETY relevant to SaMD:

- RISK MANAGEMENT;
- Quality management; and
- Methodical and systematic systems engineering according to best industry practices.

The combination of these concepts allows SaMD manufacturers to follow a clearly structured and consistently repeatable decision-making process to promote SAFETY for SaMD.

IEC 62304 provides health software development and maintenance processes, ACTIVITIES and TASKS based on process rigor levels needed to ensure safe, effective, and secure use of the HEALTH SOFTWARE. The standard allows the manufacture of the HEALTH SOFTWARE to either perform all the process ACTIVITIES and TASKS (Level C) or determine a lower process rigor level based on contribution to a HAZARDOUS SITUATION.

Process level	
A – lowest level of rigor	An error cannot cause injury
B – mid-level of rigor	an error could cause non-SERIOUS INJURY
C – highest level of rigor	An error could cause SERIOUS INJURY or death

However, because the scope of IEC 62304 is broader than SaMD, the process rigor leveling focus is more than just the information provided by a SaMD to treat or diagnose, drive or inform clinical management.

The process rigor levels still could map appropriately to the IMDRF SaMD RISK categorizations in the following manners assuming the IMDRF jurisdiction implementations of the IMDRF SaMD RISK categorization at the publication time of this document. However, none of them resolve the challenge of broader scope of IEC 62304 described above.

Example 1 (straight mapping from IMDRF SaMD RISK categorization)

State of healthcare situation or condition	Significance of information provided by SaMD to healthcare decision		
	Treat or diagnose	Drive clinical management	Inform clinical management
Critical	C	C	B
Serious	C	B	A
Non-serious	B	A	A

Example 2 (mapping similar to EU MDR rule)

State of healthcare situation or condition	Significance of information provided by SaMD to healthcare decision		
	Treat or diagnose	Drive clinical management	Inform clinical management
Critical	C	C	B
Serious	C	B	B
Non-serious	B	B	A

Example 3 (mapping similar to Health Canada rule)

State of healthcare situation or condition	Significance of information provided by SaMD to healthcare decision		
	Treat or diagnose	Drive clinical management	Inform clinical management
Critical	C	C	B
Serious	B	B	A
Non-serious	A	A	A

Annex D
(informative)

Implementation

D.1 General

Annex D gives an overview of how this document can be implemented into MANUFACTURERS' PROCESSES. It also considers that other standards, like ISO 13485 [16], require adequate and comparable PROCESSES.

D.2 Quality management SYSTEM

For MANUFACTURERS of MEDICAL DEVICES, including MEDICAL DEVICE SOFTWARE in the context of this document, the establishment of a quality management system (QMS) is required in 4.1. This document does not require that the QMS necessarily has to be certified.

D.3 EVALUATE quality management PROCESSES

It is recommended to EVALUATE how well the established and documented PROCESSES of the QMS already cover the PROCESSES of the software life cycle, by means of audits, inspections, or analyses under the responsibility of the MANUFACTURER. Any identified gaps can be accommodated by extending the quality management PROCESSES or can be separately described. If the MANUFACTURER already has PROCESS descriptions available which regulate the development, VERIFICATION and VALIDATION of software, then these should also be EVALUATED to determine how well they agree with this document.

D.4 Integrating requirements of this document into the MANUFACTURER'S quality management PROCESSES

This document can be implemented by adapting or extending the PROCESSES already installed in the QMS or integrating new PROCESSES. This document does not specify how this shall be done; the MANUFACTURER is free to do this in any suitable way.

The MANUFACTURER is responsible for ensuring that the PROCESSES described in this document are suitably put into action when the HEALTH SOFTWARE is developed by original equipment MANUFACTURERS (OEM) or sub-contractors not having their own documented QMS.

D.5 Checklist for small MANUFACTURERS without a certified QMS

The MANUFACTURER should determine the highest software process rigor level (A, B or C) of the software. Table D.1 lists all ACTIVITIES described in this document. The reference to ISO 13485 should help to define the place in the QMS. Based on the required software process rigor level, the MANUFACTURER should assess each required ACTIVITY against the existing PROCESSES. If the requirement is already covered, a reference to the relevant PROCESS descriptions should be given.

If there is discrepancy, an action is needed to improve the PROCESS.

The list can also be used for an EVALUATION of the PROCESSES after the action has been performed.

2751 **Table D.1 – Checklist for small companies without a certified QMS**

ACTIVITY	Related subclause of ISO 13485:2016	Covered by existing procedure?	If yes: reference	Actions to be taken
5.1 Software development planning	7.3.2 Design and development planning	Yes/No		
5.2 Software requirements analysis	7.3.3 Design and development inputs	Yes/No		
5.3 Software ARCHITECTURAL design		Yes/No		
5.4 Software detailed design		Yes/No		
5.5 SOFTWARE UNIT implementation		Yes/No		
5.6 Software integration and integration testing		Yes/No		
5.7 SOFTWARE SYSTEM testing	7.3.4 Design and development outputs 7.3.5 Design and development review	Yes/No		
5.8 Software release	7.3.6 Design and development VERIFICATION 7.3.7 Design and development VALIDATION	Yes/No		
6.1 Establish SOFTWARE MAINTENANCE plan	7.3.9 Control of design and development changes	Yes/No		
6.2 Problem and modification analysis		Yes/No		
6.3 Modification implementation	7.3.6 Design and development VERIFICATION 7.3.7 Design and development VALIDATION	Yes/No		
Error! Reference source not found. Analysis of software contributing to HAZARDOUS SITUATIONS		Yes/No		
7.2 RISK CONTROL measures		Yes/No		
7.3 VERIFICATION of RISK CONTROL measures		Yes/No		
7.4 RISK MANAGEMENT of software changes		Yes/No		
8.1 Configuration identification	7.5.8 Identification 7.5.9 TRACEABILITY	Yes/No		
8.2 Change control	7.5.8 Identification 7.5.9 TRACEABILITY	Yes/No		
8.3 Configuration status accounting		Yes/No		
9 Software problem resolution PROCESS		Yes/No		

2752

2753

2754

Bibliography

[1] IEC 60601-1:2005, *Medical electrical equipment – Part 1: General requirements for basic safety and essential performance*
IEC 60601-1:2005/AMD1:2012

[2] IEC 61010-1:2010, *Safety requirements for electrical equipment for measurement, control, and laboratory use – Part 1: General requirements*

[3] IEC 61508-3, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements*

[4] IEC 62366-1:2015, *Medical devices – Part 1: Application of usability engineering to medical devices*

[5] IEC 62443 (all parts), *Security for industrial automation and control systems*

[6] IEC TS 62443-1-1:2009, *Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models*

[7] IEC TR 62443-3-1:2009, *Industrial communication networks – Network and system security – Part 3-1: Security technologies for industrial automation and control systems*

[8] IEC 62443-4-1:2018, *Security for industrial automation and control systems – Part 4-1: Secure product development lifecycle requirements*

[9] IEC 62443-4-2:2019, *Security for industrial automation and control systems – Part 4-2: Technical security requirements for IACS components*

[10] IEC 80001 (all parts), *Application of risk management for IT-networks incorporating medical devices*

[11] IEC 80001-1:2010, *Application of risk management for IT-networks incorporating medical devices – Part 1: Roles, responsibilities and activities*

[12] IEC TR 80001-2-2:2012, *Application of risk management for IT-networks incorporating medical devices – Part 2-2: Guidance for the disclosure and communication of medical device security needs, risks and controls*

[13] IEC TR 80001-2-8:2016, *Application of risk management for IT-networks incorporating medical devices – Part 2-8: Application guidance – Guidance on standards for establishing the security capabilities identified in IEC TR 80001-2-2*

[14] IEC TR 80002-1:2009, *Medical device software – Part 1: Guidance on the application of ISO 14971 to medical device software*

[15] IEC 82304-1:2016, *Health software – Part 1: General requirements for product safety*

[16] ISO 13485:2016, *Medical devices – Quality management systems – Requirements for regulatory purposes*

[17] ISO 14708-1:2014, *Implants for surgery – Active implantable medical devices – Part 1: General requirements for safety, marking and for information to be provided by the manufacturer*

[18] ISO/TR 24971:2020, *Medical devices – Guidance on the application of ISO 14971*

[19] ISO 27799:2016, *Health informatics – Information security management in health using ISO/IEC 27002*

[20] ISO/TR 80002-2, *Medical device software – Part 2: Validation of software for medical device quality systems*

[21] ISO 81001-1:—[3], *Health informatics – Health software and health IT systems safety, effectiveness and security - Foundational principles, concepts and terms*

[22] ISO/IEC/IEEE 12207:2017, *Systems and software engineering – Software life cycle processes*

[23] ISO/IEC 14764:2006, *Software Engineering – Software Life Cycle Processes – Maintenance*

[24] ISO/IEC 15408-1:2009, *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*

[25] ISO/IEC 15408-2:2008, *Information technology – Security techniques – Evaluation criteria for IT security – Part 2: Security functional components*

[26] ISO/IEC 16085, *Systems and software engineering – Life cycle processes – Risk management*

[27] ISO/IEC/IEEE 24748-1, *Systems and software engineering – Life cycle management – Part 1: Guidelines for life cycle management*

[28] ISO/IEC/IEEE 24748-2, *Systems and software engineering – Life cycle management – Part 2: Guidelines for the application of ISO/IEC/IEEE 15288 (System life cycle processes)*

[29] ISO/IEC/IEEE 24748-3, *Systems and software engineering – Life cycle management – Part 3: Guidelines for the application of ISO/IEC/IEEE 12207 (software life cycle processes)*

[30] ISO/IEC/IEEE 25748-4, *Systems and software engineering – Life cycle management – Part 4: Systems engineering planning*

[31] ISO/IEC 24765:2017, *Systems and software engineering – Vocabulary*

[32] ISO/IEC 25010:2011, *Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*

[33] ISO/IEC 25051:2014, *Software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing*

[34] ISO/IEC 27001:2013, *Information technology – Security techniques – Information security management systems – Requirements*

[35] ISO/IEC 27002:2013, *Information technology – Security techniques – Code of practice for information security controls*

---

[3] Under preparation. Stage at the time of publication: ISO/CDV 81001-1:2019

[36] ISO/IEC 27005:2018, *Information technology – Security techniques – Information security risk management*

[37] ISO/IEC 27034-7:2018, *Information technology – Application security – Part 7: Assurance prediction framework*

[38] ISO/IEC 90003:2018, *Software engineering – Guidelines for the application of ISO 9001:2015 to computer software*

[39] AAMI SW91:2018, *Classification of defects in health software*

[40] AAMI TIR 36*, Validation Of Software For Regulated Processes*

[41] AAMI TIR 57:2016, *Principles for medical device security – Risk management*

[42] EN 45502-1:2015, *Implants for surgery. Active implantable medical devices. General requirements for safety, marking and for information to be provided by the manufacturer*

[43] AAMI TIR 97:2019, *Principles for medical device security— Postmarket risk management for device manufacturers*

[44] IEC 80001-5-1:—[4], *Safety, security and effectiveness in the implementation and use of connected medical devices or connected health software – Part 5-1: Security activities in the product lifecycle*

[45] UL 2900-2-1:2017, *Software cybersecurity for network-connectable products, Part 2-1: Particular requirements for network connectable components of healthcare and wellness systems*

[46] IMDRF, *Software as a Medical Device: Key Definitions*. 2013 [viewed 2019-12-10]. Available at http://www.imdrf.org/docs/imdrf/final/technical/imdrf-tech-131209-samd-key-definitions-140901.pdf

_____

[4] Under preparation. Stage at the time of publication: ISO/CD 80001-5-1:2020

Annex ZA
(informative)

Relationship between this European standard and the General Safety and Performance Requirements of Regulation (EU) 2017/745 aimed to be covered

This European standard has been prepared under a Commission's standardisation request *[Full reference to the request "M/xxx"]*[5] to provide one voluntary means of conforming to the General Safety and Performance Requirements of Regulation (EU) 2017/745 of 5 April 2017 concerning MEDICAL DEVICES [OJ L 117].

Once this standard is cited in the Official Journal of the European Union under that Regulation, compliance with the normative clauses of this standard given in Table ZA.1 confers, within the limits of the scope of this standard, a presumption of conformity with the corresponding General Safety and Performance Requirements of that Regulation, and associated EFTA regulations.

NOTE 1 When a General Safety and Performance Requirement does not appear in Table ZA.1, it means that it is not addressed by this European Standard.

Table ZA.1 – Correspondence between this European standard and Annex I of Regulation (EU) 2017/745 [OJ L 117]

General Safety and Performance Requirements of Regulation (EU) 2017/745	Clause(s) / sub-clause(s) of this EN	Remarks / Notes
14.2 (d)	7.1.2 h)	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.
17.1, first sentence	4, 5, 8 and 9	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.
17.1, second sentence	7	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.
17.2	4, 5, 6, 7, 8 and 9	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.
17.3	5.2.2 a) third dash, 5.2.2 f)	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.
17.4	5.2.2 e), 5.2.2 h), 5.2.2 j)	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.
18.8	4.2 b), 4.5.2, 5.2.2 e), 7.1.2 h)	Covered in respect of the PROCESS requirements. Device-specific execution of the PROCESS is not covered.

5 Replace with the reference number and title of the relevant standardisation request.

WARNING 1: Presumption of conformity stays valid only as long as a reference to this European standard is maintained in the list published in the Official Journal of the European Union. Users of this standard should consult frequently the latest list published in the Official Journal of the European Union.

WARNING 2: Other Union legislation may be applicable to the product(s) falling within the scope of this standard.

For devices which are also machinery within the meaning of Article 2(a) of Directive 2006/42/EC on Machinery, in accordance with Article 1(12) of Regulation (EU) 2017/745, the following Table ZA.2 details the relevant Essential Health and Safety Requirements of Directive 2006/42/EC on Machinery to the extent to which they are more specific than the General Safety and Performance Requirements set out in Chapter II of Annex I of Regulation (EU) 2017/745 along with the corresponding clauses of this European Standard. Table ZA.2, however, does not imply any citation in the OJEU under the machinery directive and thus does not provide presumption of conformity for the machinery directive.

Table ZA.2 – Relevant Essential Health and Safety Requirements from Directive 2006/42/EC on machinery that are addressed by this Document (according to article 1, item 12, of Regulation (EU) 2017/745)

Essential Health and Safety Requirements of Directive 2006/42/EC	Clause(s) / sub-clause(s) of this EN	Remarks / Notes
1.2.1	7	Partly covered, only for the software of a control system.

_____