IEC 62304

Edition 1.1 2015-06

CONSOLIDATED VERSION

colour inside

Medical device software – Software life cycle processes

IEC 62304:2006-05+AMD1:2015-06 CSV(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

IEC 62304

Edition 1.1 2015-06

CONSOLIDATED VERSION

colour inside

Medical device software – Software life cycle processes

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 11.040

ISBN 978-2-8322-2765-7

Warning! Make sure that you obtained this publication from an authorized distributor.

IEC 62304

Edition 1.1 2015-06

REDLINE VERSION

colour inside

Medical device software – Software life cycle processes

IEC 62304:2006-05+AMD1:2015-06 CSV(en)

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

MEDICAL DEVICE 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.

This Consolidated version of IEC 62304 bears the edition number 1.1. It consists of the first edition (2006-05) [documents 62A/523/FDIS and 62A/528/RVD] and its amendment 1 (2015-06) [documents 62A/1007/FDIS and 62A/1014/RVD]. The technical content is identical to the base edition and its amendment.

In this Redline version, a vertical line in the margin shows where the technical content is modified by amendment 1. Additions and deletions are displayed in red, with deletions being struck through. A separate Final version with all changes accepted is available in this publication.

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 and ISO Technical Committee 210, Quality management and corresponding general aspects for MEDICAL DEVICES. Table C.5 was prepared by ISO/IEC JTC 1/SC 7, Software and system engineering.

It is published as a dual logo standard.

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

In this standard the following print types are used:

- requirements and definitions: in roman type;
- informative material appearing outside of tables, such as notes, examples and references: in smaller type. Normative text of tables is also in a smaller type;
- terms used throughout this standard that have been defined in Clause 3 and also given in the index: in small capitals.

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 the base publication and its amendment will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

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

NOTE The attention of National Committees 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.

INTRODUCTION

Software is often an integral part of MEDICAL DEVICE technology. Establishing the SAFETY and effectiveness of a MEDICAL DEVICE containing software requires knowledge of what the software is intended to do and demonstration that the use of the software fulfils those intentions without causing any unacceptable RISKS.

This standard provides a framework of life cycle PROCESSES with ACTIVITIES and TASKS necessary for the safe design and maintenance of MEDICAL DEVICE SOFTWARE. This standard provides requirements for each life cycle PROCESS. Each life cycle PROCESS ~~is further divided into~~ consists of a set of ACTIVITIES, with most ACTIVITIES ~~further divided into~~ consisting of a set of TASKS.

As a basic foundation it is assumed that MEDICAL DEVICE SOFTWARE is developed and maintained within a quality management system (see 4.1) and a RISK MANAGEMENT system (see 4.2). The RISK MANAGEMENT PROCESS is already very well addressed by the International Standard ISO 14971. Therefore IEC 62304 makes use of this advantage simply by a normative reference to ISO 14971. Some minor additional RISK MANAGEMENT requirements are needed for software, especially in the area of identification of contributing software factors related to HAZARDS. These requirements are summarized and captured in Clause 7 as the software RISK MANAGEMENT PROCESS.

Whether software is a contributing factor to a ~~HAZARD~~ HAZARDOUS SITUATION is determined during the HAZARD identification ACTIVITY of the RISK MANAGEMENT PROCESS. ~~HAZARDS~~ HAZARDOUS SITUATIONS that could be indirectly caused by software (for example, by providing misleading information that could cause inappropriate treatment to be administered) need to be considered when determining whether software is a contributing factor. The decision to use software to control RISK is made during the RISK CONTROL ACTIVITY of the RISK MANAGEMENT PROCESS. The software RISK MANAGEMENT PROCESS required in this standard has to be embedded in the device RISK MANAGEMENT PROCESS according to ISO 14971.

The software development PROCESS consists of a number of ACTIVITIES. These ACTIVITIES are shown in Figure 1 and described in Clause 5. Because many incidents in the field are related to service or maintenance of MEDICAL DEVICE SYSTEMS including inappropriate software updates and upgrades, the software maintenance PROCESS is considered to be as important as the software development PROCESS. The software maintenance PROCESS is very similar to the software development PROCESS. It is shown in Figure 2 and described in Clause 6.

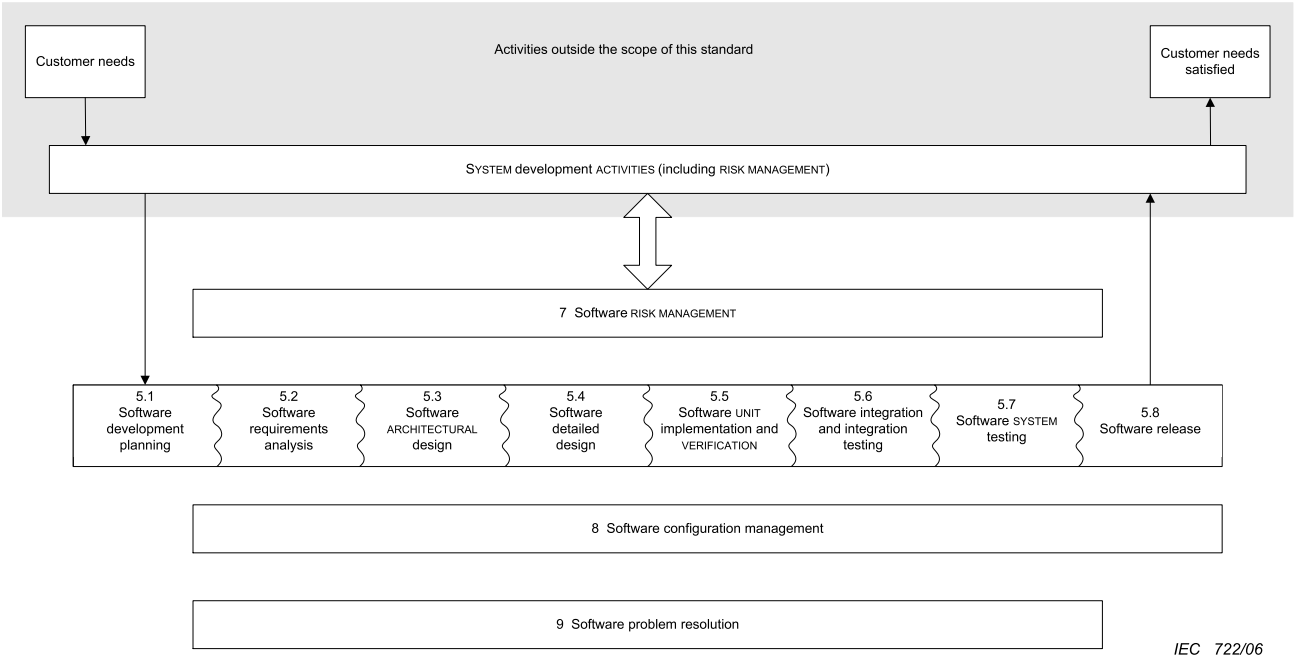


Figure 1 – Overview of software development PROCESSES and ACTIVITIES

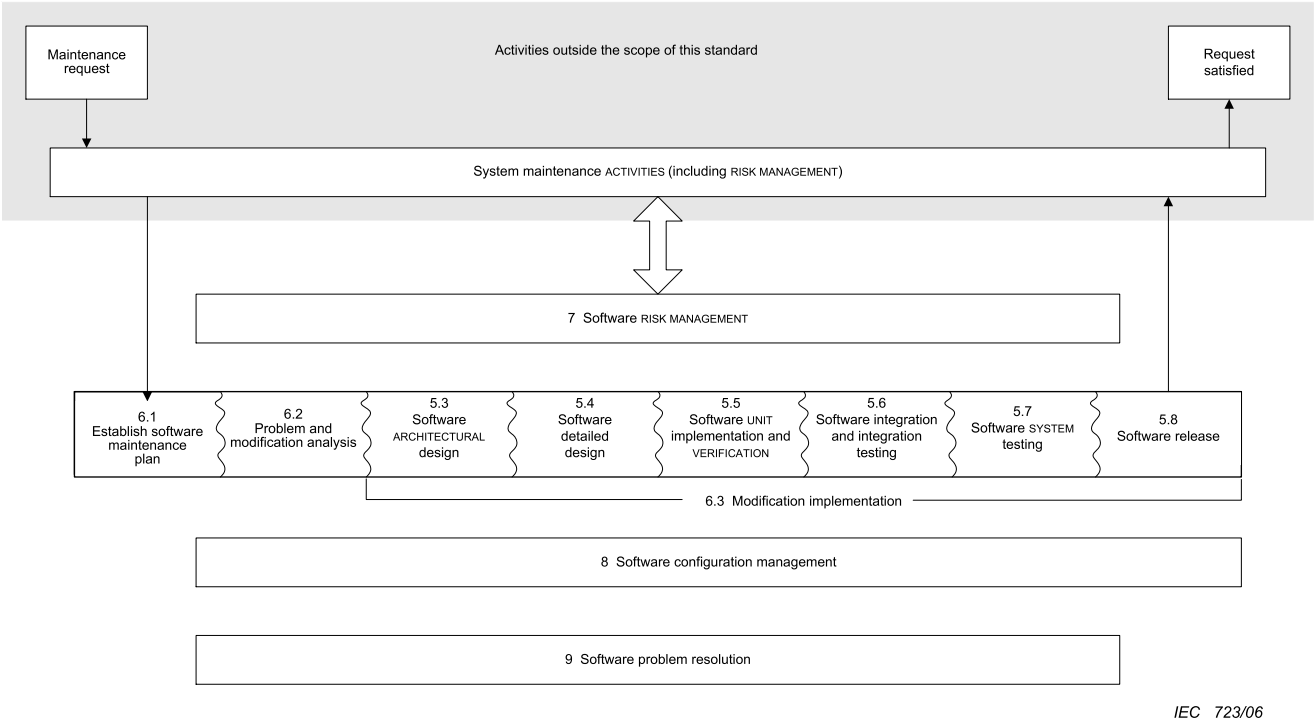


Figure 2 – Overview of software maintenance PROCESSES and ACTIVITIES

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

Amendment 1 updates the standard to add requirements to deal with LEGACY SOFTWARE, where the software design is prior to the existence of the current version, to assist manufacturers who must show compliance to the standard to meet European Directives. Software safety

classification changes include clarification of requirements and updating of the software safety classification to include a risk-based approach.

This standard does not specify an organizational structure for the MANUFACTURER or which part of the organization is to perform which PROCESS, ACTIVITY, or TASK. This standard requires only that the PROCESS, ACTIVITY, or TASK be completed to establish compliance with this standard.

This standard does not prescribe the name, format, or explicit content of the documentation to be produced. This standard requires documentation of TASKS, but the decision of how to package this documentation is left to the user of the standard.

This standard does not prescribe a specific life cycle model. The users of this standard are responsible for selecting a life cycle model for the software project and for mapping the PROCESSES, ACTIVITIES, and TASKS in this standard onto that model.

Annex A provides rationale for the clauses of this standard. Annex B provides guidance on the provisions of this standard.

For the purposes of this standard:

- “shall” means that compliance with a requirement is mandatory for compliance with this standard;
- “should” means that compliance with a requirement is recommended but is not mandatory for compliance with this standard;
- “may” is used to describe a permissible way to achieve compliance with a requirement;
- “establish” means to define, document, and implement; and
- where this standard 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.

INTRODUCTION to Amendment 1

The first edition of IEC 62304 was published in 2006. This amendment is intended to add requirements to deal with LEGACY SOFTWARE, where the software design is prior to the existence of the current version, to assist manufacturers who must show compliance to the standard to meet European Directives. Software safety classification changes needed for this amendment include clarification of requirements and updating of the software safety classification to include a risk-based approach. Work is continuing in parallel to develop the second edition of IEC 62304.

MEDICAL DEVICE SOFTWARE – SOFTWARE LIFE CYCLE PROCESSES

1 Scope

1.1 * Purpose

This standard defines the life cycle requirements for MEDICAL DEVICE SOFTWARE. The set of PROCESSES, ACTIVITIES, and TASKS described in this standard establishes a common framework for MEDICAL DEVICE SOFTWARE life cycle PROCESSES.

1.2 * Field of application

~~This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE.~~

This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE when software is itself a MEDICAL DEVICE or when software is an embedded or integral part of the final MEDICAL DEVICE.

NOTE 1 This standard can be used in the development and maintenance of software that is itself a medical device. However, additional development activities are needed at the system level before this type of software can be placed into service. These system activities are not covered by this standard, but can be found in IEC 82304-1[1] [22].

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

This standard applies regardless of the method of delivery of the software (for example: transmission by network or email, optical disk, flash memory or EEPROM). The method of software delivery itself is not considered MEDICAL DEVICE SOFTWARE.

This standard does not cover validation and final release of the MEDICAL DEVICE, even when the MEDICAL DEVICE consists entirely of software.

NOTE 2 If a medical device incorporates embedded software intended to be executed on a processor, the requirements of this standard apply to the software, including the requirements concerning software of unknown provenance (see 8.1.2).

NOTE 3 Validation and other development activities are needed at the system level before the software and medical device can be placed into service. These system activities are not covered by this standard, but can be found in related product standards (e.g., IEC 60601-1, IEC 82304-1, etc.).

1.3 Relationship to other standards

This MEDICAL DEVICE SOFTWARE life cycle standard is to be used together with other appropriate standards when developing a MEDICAL DEVICE. Annex C shows the relationship between this standard and other relevant standards.

1.4 Compliance

Compliance with this standard is defined as implementing all of the PROCESSES, ACTIVITIES, and TASKS identified in this standard in accordance with the software safety class.

---

[1] In preparation.

NOTE The software safety classes assigned to each requirement are identified in the normative text following the requirement.

Compliance is determined by inspection of all documentation required by this standard including the RISK MANAGEMENT FILE, and assessment of the PROCESSES, ACTIVITIES and TASKS required for the software safety class. ~~See Annex D.~~

NOTE 1 This assessment could be carried out by internal or external audit.

NOTE 2 Although the specified PROCESSES, ACTIVITIES, and TASKS are performed, flexibility exists in the methods of implementing these PROCESSES and performing these ACTIVITIES and TASKS.

NOTE 3 Where any requirements contain "as appropriate" and were not performed, documentation for the justification is necessary for this assessment.

NOTE 4 The term "conformance" is used in ISO/IEC 12207 where the term "compliance" is used in this standard.

NOTE 5 For compliance of LEGACY SOFTWARE see 4.4.

2 * Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 14971, *Medical devices – Application of risk management to medical devices*.

3 * Terms and definitions

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

3.1
ACTIVITY
a set of one or more interrelated or interacting TASKS

3.2
ANOMALY
any condition that deviates from the expected based on requirements specifications, design documents, standards, etc. or from someone's perceptions or experiences. ANOMALIES may be found during, but not limited to, the review, test, analysis, compilation, or use of MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~ or applicable documentation

NOTE Based on ~~[~~IEEE 1044:1993, definition 3.1~~]~~.

3.3
ARCHITECTURE
organizational structure of a SYSTEM or component

[IEEE 610.12:1990]

3.4
CHANGE REQUEST
a documented specification of a change to be made to a MEDICAL DEVICE SOFTWARE ~~PRODUCT~~

3.5
CONFIGURATION ITEM
entity that can be uniquely identified at a given reference point

NOTE Based on ISO/IEC 12207:~~1995~~ 2008, ~~3.6~~ 4,7.

3.6
DELIVERABLE
required result or output (includes documentation) of an ACTIVITY or TASK

3.7
EVALUATION
a systematic determination of the extent to which an entity meets its specified criteria

[ISO/IEC 12207:~~1995~~ 2008,~~ 3.9~~ 4.12]

3.8
HARM
physical injury, damage, or both to the health of people or damage to property or the environment

[~~ISO/IEC Guide 51:1999, definition 3.3~~ ISO 14971:2007, 2.2]

3.9
HAZARD
potential source of HARM

[~~ISO/IEC Guide 51:1999, definition 3.5~~ ISO 14971:2007, 2.3]

3.10
MANUFACTURER
natural or legal person with responsibility for designing, manufacturing, packaging, or labelling a MEDICAL DEVICE; assembling a SYSTEM; or adapting a MEDICAL DEVICE before it is placed on the market and/or put into service, regardless of whether these operations are carried out by that person or by a third party on that person's behalf

NOTE 1 Attention is drawn to the fact that the provisions of national or regional regulations can apply to the definition of manufacturer.

NOTE 2 For a definition of labelling, see ISO 13485:2003, definition 3.6.

[ISO 14971:~~2000~~ 2007,~~ 2.6~~ 2,8]

3.11
MEDICAL DEVICE
any instrument, apparatus, implement, machine, appliance, implant, in vitro reagent or calibrator, 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 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 for medical purposes by means of in vitro examination of specimens derived from the human body,

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

NOTE 1 This definition has been developed by the Global Harmonization Task Force (GHTF). See bibliographic reference [15] (in ISO 13485:2003).

[ISO 13485:2003, definition 3.7]

NOTE 2 Some differences can occur in the definitions used in regulations of each country.

NOTE 3 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).

3.12
MEDICAL DEVICE SOFTWARE
SOFTWARE SYSTEM that has been developed for the purpose of being incorporated into the MEDICAL DEVICE being developed or that is intended for use as a MEDICAL DEVICE ~~in its own right~~

NOTE This includes a MEDICAL DEVICE software product, which then is a MEDICAL DEVICE in its own right.

3.13
PROBLEM REPORT
a record of actual or potential behaviour of a MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ that a user or other interested person believes to be unsafe, inappropriate for the intended use or contrary to specification

NOTE 1 This standard does not require that every PROBLEM REPORT results in a change to the MEDICAL DEVICE SOFTWARE ~~PRODUCT~~. A MANUFACTURER can reject a PROBLEM REPORT as a misunderstanding, error or insignificant event.

NOTE 2 A PROBLEM REPORT can relate to a released MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ or to a MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ that is still under development.

NOTE 3 This standard requires the MANUFACTURER to perform extra decision making steps (see Clause 6) for a PROBLEM REPORT relating to a released product to ensure that regulatory actions are identified and implemented.

3.14
PROCESS
a set of interrelated or interacting ACTIVITIES that transform inputs into outputs

[ISO 9000:2000, definition 3.4.1]

NOTE The term "ACTIVITIES" covers use of resources.

3.15
REGRESSION TESTING
the testing required to determine that a change to a SYSTEM component has not adversely affected functionality, reliability or performance and has not introduced additional defects

[ISO/IEC 90003:2004, definition 3.11]

3.16
RISK
combination of the probability of occurrence of HARM and the severity of that HARM

[~~ISO/IEC Guide 51:1999 definition 3.2~~ ISO 14971:2007, 2.16]

3.17
RISK ANALYSIS
systematic use of available information to identify HAZARDS and to estimate the RISK

[~~ISO/IEC Guide 51:1999 definition 3.10~~ ISO 14971:2007, 2.17]

3.18
RISK CONTROL
PROCESS in which decisions are made and RISKS are reduced to, or maintained within, specified levels

[ISO 14971:~~2000~~ 2007, ~~2.16, modified~~ 2.19]

3.19
RISK MANAGEMENT
systematic application of management policies, procedures, and practices to the TASKS of analyzing, evaluating, and controlling RISK

[ISO 14971:~~2000~~ 2007, ~~2.18~~ 2.22, modified – The phrase "and monitoring" has been removed]

3.20
RISK MANAGEMENT FILE
set of records and other documents, not necessarily contiguous, that are produced by a RISK MANAGEMENT PROCESS

[ISO 14971:~~2000~~ 2007,~~ 2.19~~ 2.23]

3.21
SAFETY
freedom from unacceptable RISK

[~~ISO/IEC Guide 51:1999 definition 3.1~~ ISO 14971:2007, 2.24]

3.22
SECURITY
protection of information and data so that unauthorized~~ people~~ persons or systems cannot read or modify them~~ and so that~~ an authorized persons or systems are not denied access to them

NOTE Based on ~~[~~ISO/IEC 12207:~~1995~~ 2008,~~ 3.25~~ 4.39~~]~~.

3.23
SERIOUS INJURY
injury or illness that~~ directly or indirectly~~:

a) is life threatening,

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

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

NOTE Permanent impairment means an irreversible impairment or damage to a body structure or function excluding trivial impairment or damage.

3.24
SOFTWARE DEVELOPMENT LIFE CYCLE MODEL
conceptual structure spanning the life of the software from definition of its requirements to its release~~ for manufacturing~~, which:

– identifies the PROCESS, ACTIVITIES and TASKS involved in development of~~ a~~ MEDICAL DEVICE SOFTWARE~~ PRODUCT~~,

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

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

NOTE Based on ISO/IEC 12207:1995, definition 3.11

3.25
SOFTWARE ITEM
any identifiable part of a computer program, i.e., source code, object code, control code, control data, or a collection of these items

NOTE 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 provide the~~ definition and~~ granularity of the SOFTWARE ITEMS and SOFTWARE UNITS.

NOTE 2 Based on ~~[~~ISO/IEC 90003:2004, 3.14~~, modified~~ and ISO/IEC 12207:2008, 4.41~~]~~

3.26

~~**SOFTWARE PRODUCT**~~

~~set of computer programs, procedures, and possibly associated documentation and data~~

~~[ISO/IEC 12207:1995 definition 3.26]~~

Not used

3.27

SOFTWARE SYSTEM

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

3.28

SOFTWARE UNIT

SOFTWARE ITEM that is not subdivided into other items

NOTE ~~SOFTWARE UNITS can be used for the purpose of software configuration management or testing.~~ The granularity of SOFTWARE UNITS is defined by the MANUFACTURER (see B.3).

3.29

SOUP

software of unknown provenance (acronym)

SOFTWARE ITEM that is already developed and generally available and that has not been developed for the purpose of being incorporated into the MEDICAL DEVICE (also known as "off-the-shelf software") or SOFTWARE ITEM previously developed for which adequate records of the development PROCESSES are not available

NOTE A MEDICAL DEVICE SOFTWARE SYSTEM in itself cannot be claimed to be SOUP.

3.30

SYSTEM

integrated composite consisting of one or more of the PROCESSES, hardware, software, facilities, and people, that provides a capability to satisfy a stated need or objective

NOTE Based on ISO/IEC ~~[~~ISO/IEC 12207:~~1995~~ 2008, ~~3.31~~ 4.48~~]~~.

3.31

TASK

a single piece of work that needs to be done

3.32

TRACEABILITY

degree to which a relationship can be established between two or more products of the development PROCESS

[IEEE 610.12:1990]

NOTE Requirements, architecture, risk control measures, etc. are examples of deliverables of the development PROCESS.

3.33

VERIFICATION

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

NOTE 1 "Verified" is used to designate the corresponding status.

[ISO 9000:2000, definition 3.8.4]

NOTE 2 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.

3.34
VERSION
identified instance of a CONFIGURATION ITEM

NOTE 1 Modification to a VERSION of ~~a~~ MEDICAL DEVICE SOFTWARE ~~PRODUCT~~, resulting in a new VERSION, requires software configuration management action.

NOTE 2 Based on ISO/IEC 12207:~~1995~~ 2008, ~~3.37~~ 4.56.

3.35
HAZARDOUS SITUATION
circumstance in which people, property or the environment are exposed to one or more HAZARD(S)

[SOURCE: ISO 14971:2007, 2.4]

3.36
LEGACY SOFTWARE
MEDICAL DEVICE SOFTWARE which was legally placed on the market and is still marketed today but for which there is insufficient objective evidence that it was developed in compliance with the current version of this standard

3.37
RELEASE
particular VERSION of a CONFIGURATION ITEM that is made available for a specific purpose

NOTE Based on ISO/IEC 12207:2008, definition 4.35.

3.38
RESIDUAL RISK
RISK remaining after RISK CONTROL measures have been taken

NOTE 1 Adapted from ISO/IEC Guide 51:1999, definition 3.9.

NOTE 2 ISO/IEC Guide 51:1999, definition 3.9 uses the term “protective measures” rather than “RISK CONTROL measures.” However, in the context of this International Standard, “protective measures” are only one option for controlling RISK as described in 6.2 [of ISO 14971:2007].

[SOURCE: ISO 14971:2007, 2.15].

3.39
RISK ESTIMATION
PROCESS used to assign values to the probability of occurrence of HARM and the severity of that HARM

[SOURCE: ISO 14971:2007 2.20]

3.40
RISK EVALUATION
PROCESS of comparing the estimated RISK against given RISK criteria to determine the acceptability of the RISK

[SOURCE: ISO 14971:2007 2.21]

4 * General requirements

4.1 * Quality management system

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

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

- ISO 13485 [8]; 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 [15].

4.2 * RISK MANAGEMENT

The MANUFACTURER shall apply a RISK MANAGEMENT PROCESS complying with ISO 14971.

4.3 * Software safety classification

a) The MANUFACTURER shall assign to each SOFTWARE SYSTEM a software safety class (A, B, or C) according to the ~~possible effects on~~ RISK of HARM to the patient, operator, or other people resulting from a ~~HAZARD~~ HAZARDOUS SITUATION to which the SOFTWARE SYSTEM can contribute in a worst-case-scenario as indicated in Figure 3.

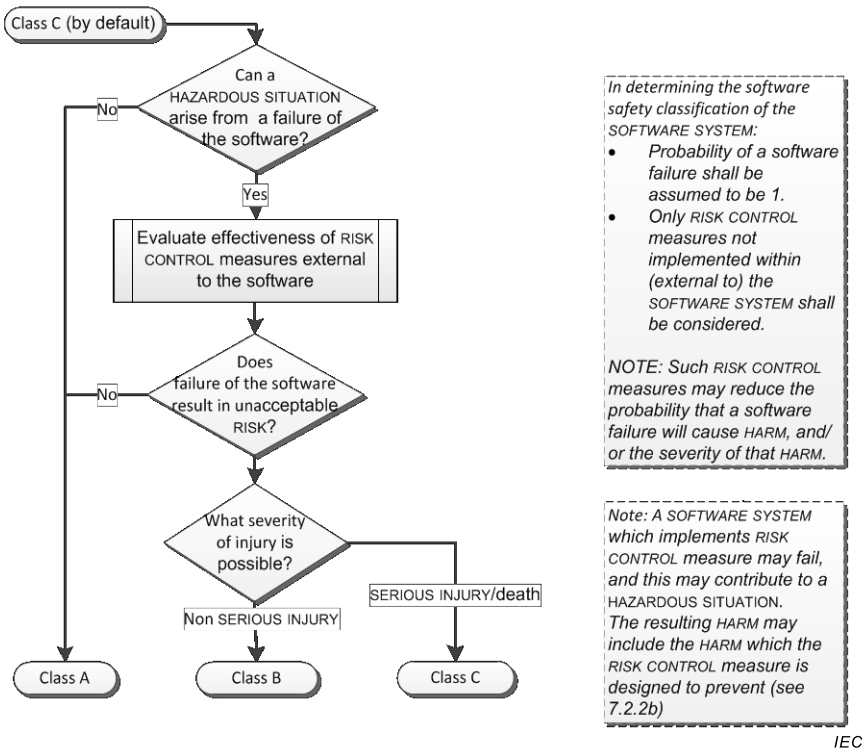


Figure 3 – Assigning software safety classification

~~The software safety classes shall initially be assigned based on severity as follows:~~

~~Class A: No injury or damage to health is possible~~

~~Class B: Non-SERIOUS INJURY is possible~~

~~Class C: Death or SERIOUS INJURY is possible~~

~~If the HAZARD could arise from a failure of the SOFTWARE SYSTEM to behave as specified, the probability of such failure shall be assumed to be 100 percent.~~

~~If the RISK of death or SERIOUS INJURY arising from a software failure is subsequently reduced to an acceptable level (as defined by ISO 14971) by a hardware RISK CONTROL measure, either by reducing the consequences of the failure or by reducing the probability of death or SERIOUS INJURY arising from that failure, the software safety classification may be reduced from C to B; and if the RISK of non-SERIOUS INJURY arising from a software failure is similarly reduced to an acceptable level by a hardware RISK CONTROL measure, the software safety classification may be reduced from B to A.~~

The SOFTWARE SYSTEM is software safety class A if:

– the SOFTWARE SYSTEM cannot contribute to a HAZARDOUS SITUATION; or

– the SOFTWARE SYSTEM can contribute to a HAZARDOUS SITUATION which does not result in unacceptable RISK after consideration of RISK CONTROL measures external to the SOFTWARE SYSTEM.

The SOFTWARE SYSTEM is software safety class B if:

– the SOFTWARE SYSTEM can contribute to a HAZARDOUS SITUATION which results in unacceptable RISK after consideration of RISK CONTROL measures external to the SOFTWARE SYSTEM and the resulting possible HARM is non-SERIOUS INJURY.

The SOFTWARE SYSTEM is software safety class C if:

– the SOFTWARE SYSTEM can contribute to a HAZARDOUS SITUATION which results in unacceptable RISK after consideration of RISK CONTROL measures external to the SOFTWARE SYSTEM and the resulting possible HARM is death or SERIOUS INJURY.

For a SOFTWARE SYSTEM initially classified as software safety class 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 safety classification to the SOFTWARE SYSTEM.

NOTE 1 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.

NOTE 2 See ISO 14971:2007 subclause 3.2, *Management Responsibilities*, for the definition of risk acceptability.

b) ~~The MANUFACTURER shall assign to each SOFTWARE SYSTEM that contributes to the implementation of a RISK CONTROL measure a software safety class based on the possible effects of the HAZARD that the RISK CONTROL measure is controlling.~~
Not used.

c) The MANUFACTURER shall document the software safety class 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 safety classification of the original SOFTWARE ITEM (or SOFTWARE SYSTEM) unless the MANUFACTURER documents a rationale for classification into a different software safety class (software safety classes assigned according to 4.3 a) replacing "SOFTWARE SYSTEM" with "SOFTWARE ITEM"). Such a rationale shall explain how the new SOFTWARE ITEMS are segregated so that they may be classified separately.

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

f) For compliance with this standard, ~~wherever a PROCESS is required for SOFTWARE ITEMS of a specific classification and the PROCESS is necessarily applied~~ when applying this standard to a group of SOFTWARE ITEMS, the MANUFACTURER shall use the PROCESSES and TASKS which are required by the classification 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.

g) For each SOFTWARE SYSTEM, until a software safety class is assigned, Class C requirements shall apply.

NOTE In the ~~requirements~~ clauses and subclauses that follow, the software safety classes ~~that the requirement must be performed~~ for which a specific requirement applies are identified following the requirement in the form [Class . . .].

4.4 * LEGACY SOFTWARE

4.4.1 General

As an alternative to applying Clauses 5 through 9 of this standard, compliance of LEGACY SOFTWARE may be demonstrated as indicated in 4.4.2 to 4.4.5.

4.4.2 RISK MANAGEMENT ACTIVITIES

In accordance with 4.2 of this standard, the MANUFACTURER shall:

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

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 MEDICAL DEVICE 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;
- definition of RISK CONTROL measures for each potential cause of the LEGACY SOFTWARE contributing to a HAZARDOUS SITUATION.

4.4.3 Gap analysis

Based on the software safety class of the LEGACY SOFTWARE (see 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 SYSTEM test records (see 5.7.5).

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

4.4.4 Gap closure activities

a) The MANUFACTURER shall establish and execute a plan to generate the identified DELIVERABLES. Where available, objective evidence may 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 Clause 6.

4.4.5 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 4.4.

NOTE Fulfilling 4.4 enables further use of LEGACY SOFTWARE in accordance with IEC 62304.

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 safety classifications 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 (see Note 4);

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 the MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~, DELIVERABLES and ACTIVITIES at each stage of the life cycle.

[Class A, B, C]

NOTE 1 The SOFTWARE DEVELOPMENT LIFE CYCLE MODEL can identify different elements (PROCESSES, ACTIVITIES, TASKS and DELIVERABLES) for different SOFTWARE ITEMS according to the software safety classification of each SOFTWARE ITEM of the SOFTWARE SYSTEM.

NOTE 2 These ACTIVITIES and TASKS can overlap or interact and can be performed iteratively or recursively. It is not the intent to imply that a specific life cycle model should be used.

NOTE 3 Other PROCESSES are described in this standard separately from the development PROCESS. This does not imply that they must be implemented as separate ACTIVITIES and TASKS. The ACTIVITIES and TASKS of the other PROCESSES can be integrated into the development PROCESS.

NOTE 4 The software development plan can reference existing PROCESSES or define new ones.

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

5.1.2 Keep software development plan updated

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

5.1.3 Software development plan reference to SYSTEM design and development

a) As inputs for software development, SYSTEM requirements shall be referenced in the software development plan by the MANUFACTURER.

b) In the software development plan, the MANUFACTURER shall include or reference ~~in the software development plan~~ procedures for coordinating the software development ~~and the design and development validation~~ with the system development necessary to satisfy 4.1 (such as system integration, verification, and validation).

[Class 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 device).

5.1.4 Software development standards, methods and tools planning

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

a) standards,

b) methods, and

c) tools

associated with the development of SOFTWARE ITEMS of class C. [Class 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. [Class 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.

[Class 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. [Class 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;

~~c) intended audience of document; and~~

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

[Class 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 ~~and~~ ACTIVITIES;

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

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

f) when the problem resolution PROCESS is to be used.

[Class 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 the MEDICAL DEVICE SOFTWARE, which could impact the MEDICAL DEVICE SOFTWARE. [Class 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 ~~documented~~ configuration management control before they are VERIFIED. [Class B, C]

5.1.12 Identification and avoidance of common software defects

The MANUFACTURER shall include or reference in the software development plan a procedure for:

a) identifying categories of defects that may be introduced based on the selected programming technology that are relevant to their SOFTWARE SYSTEM; and

b) documenting evidence that demonstrates that these defects do not contribute to unacceptable RISK.

NOTE See Annex B of IEC TR 80002-1:2009 for examples of categories of defects or causes contributing to HAZARDOUS SITUATIONS.

[Class B, C]

5.2 * Software requirements analysis

5.2.1 Define and document software requirements from SYSTEM requirements

For each SOFTWARE SYSTEM of the MEDICAL DEVICE, the MANUFACTURER shall define and document SOFTWARE SYSTEM requirements from the SYSTEM level requirements. [Class 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 device).

5.2.2 Software requirements content

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

a) functional and capability requirements;

NOTE 1 Examples include:

– performance (e.g., purpose of software, timing requirements),

– 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 requirements;

NOTE 3 Examples include:

– those related to the compromise of sensitive information,

– authentication,

– authorization,

– audit trail, and

– communication integrity,

– system security/malware protection.

f) ~~usability engineering requirements that are sensitive to human errors and training~~ user interface requirements implemented by software;

NOTE 4 Examples include those related to:

– support for manual operations,

– human-equipment interactions,

– constraints on personnel, and

– areas needing concentrated human attention.

NOTE 5 Information regarding usability engineering requirements can be found in IEC 62366-1 [21] among others (e.g., IEC 60601-1-6 [3]).

g) data definition and database requirements;

NOTE 6 Examples include:

– form;

– fit;

– function.

h) installation and acceptance requirements of the delivered MEDICAL DEVICE SOFTWARE at the operation and maintenance site or sites;

i) requirements related to methods of operation and maintenance;

j) ~~user documentation to be developed~~ requirements related to IT-network aspects;

NOTE 9 Examples include those related to:

– networked alarms, warnings, and operator messages;

– network protocols;

– handling of unavailability of network services.

k) user maintenance requirements; and

l) regulatory requirements.

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

[Class A, B, C]

NOTE 7 All of these requirements might not be available at the beginning of the software development.

NOTE 8 Among others, ~~ISO/IEC 9126-1 [8]~~ ISO/IEC 25010 [12] provides information on quality characteristics that may be useful in defining software requirements.

5.2.3 Include RISK CONTROL measures in software requirements

The MANUFACTURER shall include RISK CONTROL measures implemented in software ~~for hardware failures and potential software defects~~ in the requirements as appropriate to the MEDICAL DEVICE SOFTWARE. [Class B, C]

NOTE 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.

5.2.4 Re-EVALUATE MEDICAL DEVICE RISK ANALYSIS

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

5.2.5 Update ~~SYSTEM~~ 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. [Class 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 ~~to determine whether the test criteria have been met~~;

e) can be uniquely identified; and

f) are traceable to SYSTEM requirements or other source.

[Class A, B, C]

NOTE This standard does not require the use of a formal specification language.

5.3 * Software ARCHITECTURAL design

5.3.1 Transform software requirements into an ARCHITECTURE

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

5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS

The MANUFACTURER shall develop and document an ARCHITECTURE for the interfaces between the SOFTWARE ITEMS and the components external to the SOFTWARE ITEMS (both software and hardware), and between the SOFTWARE ITEMS. [Class 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 intended use. [Class 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. [Class 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 ~~the~~ any segregation between SOFTWARE ITEMS that is ~~essential to~~ necessary for RISK CONTROL, and state how to ensure that ~~the~~ such segregation is effective. [Class 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 B.4.3).

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 MEDICAL DEVICE ARCHITECTURE supports proper operation of any SOUP items.

[Class 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 ~~Refine SOFTWARE ARCHITECTURE~~ Subdivide software into SOFTWARE UNITS

The MANUFACTURER shall ~~refine~~ subdvide the software ~~ARCHITECTURE~~ until it is represented by SOFTWARE UNITS. [Class B, C]

NOTE Some SOFTWARE SYSTEMS are not divided further.

5.4.2 Develop detailed design for each SOFTWARE UNIT

The MANUFACTURER shall ~~develop and~~ document a ~~detailed~~ design with enough detail to allow correct implementation of ~~for~~ each SOFTWARE UNIT ~~of the SOFTWARE ITEM~~. [Class C]

5.4.3 Develop detailed design for interfaces

The MANUFACTURER shall ~~develop and~~ document a ~~detailed~~ 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. [Class 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.

[Class C]

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

5.5 * SOFTWARE UNIT implementation ~~and verification~~

5.5.1 Implement each SOFTWARE UNIT

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

5.5.2 Establish SOFTWARE UNIT VERIFICATION PROCESS

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

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

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. [Class B, C]

NOTE Examples of acceptance criteria are:

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

– is the software code free from contradiction with the interface~~s documented in the detailed~~ 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.

[Class C]

5.5.5 SOFTWARE UNIT VERIFICATION

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

5.6.2 Verify software integration

The MANUFACTURER shall verify ~~and record the following aspects of the software integration~~ 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.

~~a) the SOFTWARE UNITS have been integrated into SOFTWARE ITEMS and the SOFTWARE SYSTEM; and~~

~~b) the hardware items, SOFTWARE ITEMS, and support for manual operations (e.g., human-equipment interface, on-line help menus, speech recognition, voice control) of the SYSTEM have been integrated into the SYSTEM.~~

[Class B, C]

NOTE This VERIFICATION is only that the ~~items have been integrated~~ integration has been done according to the plan~~, not that they perform as intended~~. This VERIFICATION is most likely implemented by some form of inspection.

5.6.3 ~~Test integrated~~ 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. [Class B, C]

5.6.4 Software integration testing content

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

[Class 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 ~~Verify~~ EVALUATE software integration test procedures

The MANUFACTURER shall EVALUATE the integration test procedures for ~~correctness~~ adequacy. [Class 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. [Class 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) identify the tester.

[Class 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;
- 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. [Class B, C]

NOTE See Clause 9.

5.7 * SOFTWARE SYSTEM testing

5.7.1 Establish tests for software requirements

a) 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. [Class 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.

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

5.7.2 Use software problem resolution PROCESS

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

5.7.3 Retest after changes

When changes are made 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 testing appropriate to demonstrate that unintended side effects have not been introduced; and

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

[Class A, B, C]

5.7.4 ~~Verify~~ Evaluate SOFTWARE SYSTEM testing

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

The MANUFACTURER shall verify that:

~~a) the VERIFICATION strategies and the test procedures used are appropriate;~~

~~b) SOFTWARE SYSTEM test procedures trace to software requirements;~~

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

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

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

[Class 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;

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

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

~~c) identify the tester.~~

c) the version of software tested;

d) relevant hardware and software test configurations;

e) relevant test tools;

f) date tested; and

g) the identity of the person responsible for executing the test and recording the test results.

[Class A, 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.~~

5.8 * Software RELEASE for utilization at a SYSTEM level

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. [Class A, B, C]

5.8.2 Document known residual ANOMALIES

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

5.8.4 Document released VERSIONS

The MANUFACTURER shall document the VERSION of the MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ that is being released. [Class A, B, C]

5.8.5 Document how released software was created

The MANUFACTURER shall document the procedure and environment used to create the released software. [Class 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 ~~all~~ the associated documentation. [Class B, C]

NOTE See 5.1.3.b).

5.8.7 Archive software

The MANUFACTURER shall archive:

a) the MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ and CONFIGURATION ITEMS; and

b) the documentation

for at least a period of time determined as the longer of: the life time of the ~~device~~ MEDICAL DEVICE SOFTWARE as defined by the MANUFACTURER or a time specified by relevant regulatory requirements. [Class A, B, C]

5.8.8 Assure ~~repeatability of software release~~ reliable delivery of released software

The MANUFACTURER shall establish procedures to ensure that the released MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ can be reliably delivered to the point of use without corruption or unauthorised change. These procedures shall address the production and handling of media containing the MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ including as appropriate:

– replication,
– media labelling,
– packaging,
– protection,
– storage, and
– delivery.

[Class 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:
   – receiving,
   – documenting,
   – evaluating,
   – resolving and
   – tracking

   feedback arising after release of the MEDICAL DEVICE SOFTWARE;
b) criteria for determining whether feedback is considered to be a problem;
c) use of the software RISK MANAGEMENT PROCESS;
d) use of the software problem resolution PROCESS for analysing and resolving problems arising after release of the MEDICAL DEVICE SOFTWARE;
e) use of the software configuration management PROCESS (Clause 8) for managing modifications to the existing SOFTWARE SYSTEM; and
f) procedures to EVALUATE and implement:
   – upgrades,
   – bug fixes,
   – patches and
   – obsolescence

   of SOUP.

[Class 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 ~~released~~ MEDICAL DEVICE SOFTWARE ~~PRODUCT from both inside its own organization and from users~~ released for intended use. [Class 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 released MEDICAL DEVICE SOFTWARE ~~PRODUCT~~. 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. [Class A, B, C]

6.2.1.3 Evaluate PROBLEM REPORT'S affects on SAFETY

Each PROBLEM REPORT shall be EVALUATED to determine how it affects the SAFETY of ~~a released~~ MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ released for intended use (see 9.2) and whether a change to ~~the released~~ that software ~~PRODUCT~~ is needed to address the problem. [Class 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. [Class A, B, C]

NOTE A problem could show that a SOFTWARE SYSTEM or SOFTWARE ITEM has not been placed in the correct software safety class. The problem resolution process can suggest changes of the software safety class. When ~~this ACTIVITY has been done~~ the PROCESS has been completed, any change of safety class in the SOFTWARE SYSTEM or its SOFTWARE ITEMS should be 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 organization, ~~released~~ MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~ released for intended use , and SYSTEMS with which it interfaces. [Class A, B, C]

6.2.4 CHANGE REQUEST approval

The MANUFACTURER shall EVALUATE and approve CHANGE REQUESTS which modify released MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~. [Class A, B, C]

6.2.5 Communicate to users and regulators

The MANUFACTURER shall identify the approved CHANGE REQUESTS that affect released MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~.

As required by local regulation, the MANUFACTURER shall inform users and regulators about:

a) any problem in released MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~ and the consequences of continued unchanged use; and

b) the nature of any available changes to released MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~ and how to obtain and install the changes.

[Class A, B, C]

6.3 * Modification implementation

6.3.1 Use established PROCESS to implement modification

The MANUFACTURER shall ~~use the software development PROCESS (see Clause 5) or an established maintenance PROCESS to implement the modifications~~ identify and perform any Clause 5 ACTIVITIES that need to be repeated as a result of the modification. [Class 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 ~~modified SOFTWARE SYSTEMS~~ modifications according to 5.8. [Class A, B, C]

NOTE Modifications ~~may~~ 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 identified in the MEDICAL DEVICE RISK ANALYSIS ACTIVITY of ISO 14971 (see 4.2). [Class 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 identify potential causes of the SOFTWARE ITEM identified above contributing to a hazardous situation.

The MANUFACTURER shall consider potential causes including, 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) hardware failures or other software defects that could result in unpredictable software operation; and

e) reasonably foreseeable misuse.

[Class 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 MEDICAL DEVICE to determine if any of the known ANOMALIES result in a sequence of events that could result in a hazardous situation. [Class B, C]

7.1.4 Document potential causes

The MANUFACTURER shall document in the RISK MANAGEMENT FILE potential causes of the SOFTWARE ITEM contributing to a hazardous situation (see ISO 14971). [Class B, C]

~~7.1.5 Document sequences of events~~

~~The MANUFACTURER shall document in the RISK MANAGEMENT FILE sequences of events that could result in a hazardous situation that are identified in 7.1.2. [Class B, C]~~

7.2 RISK CONTROL measures

7.2.1 Define RISK CONTROL measures

For each ~~potential cause of the software item contributing to a hazardous situation~~ case documented in the RISK MANAGEMENT FILE where a SOFTWARE ITEM could contribute to a HAZARDOUS SITUATION, the MANUFACTURER shall define and document RISK CONTROL measures in accordance with ISO 14971. [Class B, C]

NOTE The RISK CONTROL measures can be implemented in hardware, software, the working environment or user instruction.

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 requirements;

b) assign ~~a software safety class to the~~ to each SOFTWARE ITEM ~~based on the possible effects of the HAZARD~~ that contributes to the implementation of a RISK CONTROL measure a software safety class based on the RISK that the RISK CONTROL measure is controlling (see 4.3 a)); and

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

[Class B, C]

NOTE This requirement provides additional detail for RISK CONTROL requirements of ISO 14971

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. [Class B, C]

7.3.2 ~~Document any new sequences of events~~

~~If a RISK CONTROL measure is implemented as a SOFTWARE ITEM, the MANUFACTURER shall EVALUATE the RISK CONTROL measure to identify and document in the RISK MANAGEMENT FILE any new sequences of events that could result in a hazardous situation. [Class B, C]~~ Not used.

7.3.3 Document TRACEABILITY

The MANUFACTURER shall document TRACEABILITY of software HAZARDS as appropriate:

a) from 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.

[Class B, C]

NOTE See ISO 14971 – RISK MANAGEMENT report.

7.4 RISK MANAGEMENT of software changes

7.4.1 Analyse changes to MEDICAL DEVICE SOFTWARE with respect to SAFETY

The MANUFACTURER shall analyse changes to the MEDICAL DEVICE SOFTWARE (including SOUP) to determine whether:

a) additional potential causes are introduced contributing to a hazardous situation; and

b) additional software RISK CONTROL measures are required.

[Class 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 interfere with existing RISK CONTROL measures. [Class B, C]

7.4.3 Perform RISK MANAGEMENT ACTIVITIES based on analyses

The MANUFACTURER shall perform relevant RISK MANAGEMENT ACTIVITIES defined in 7.1, 7.2 and 7.3 based on these analyses. [Class 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 ~~for the project~~ according to the development and configuration planning specified in 5.1. ~~This scheme shall include other SOFTWARE PRODUCTS or entities such as SOUP and documentation.~~ [Class A, B, C]

8.1.2 Identify SOUP

For each SOUP CONFIGURATION ITEM being used, including standard libraries, the MANUFACTURER shall document:

a) the title,

b) the MANUFACTURER, and

c) the unique SOUP designator

~~of each SOUP CONFIGURATION ITEM being used~~. [Class 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. [Class A, B, C]

8.2 * Change control

8.2.1 Approve CHANGE REQUESTS

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

NOTE 1 The decision to approve a CHANGE REQUEST can be integral to the change control PROCESS or part of another PROCESS. This subclause only requires that approval of a change precede its implementation.

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 ~~e)~~ d) and 6.1 e).

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 safety classification of SOFTWARE SYSTEMS and SOFTWARE ITEMS. [Class A, B, C]

NOTE This subclause states how the change should be implemented to achieve adequate change control. It does not imply that the implementation is an integral part of the change control PROCESS. Implementation should use planned PROCESSES, see 5.1.1 e) 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. [Class A, B, C]

NOTE This subclause only requires that changes be VERIFIED. It does not imply that VERIFICATION is an integral part of the change control PROCESS. VERIFICATION should use planned PROCESSES, see 5.1.1 e) and 6.1 e).

8.2.4 Provide means for TRACEABILITY of change

The MANUFACTURER shall~~ create an audit trail whereby each~~ maintain records of the relationships and dependencies between:

a) CHANGE REQUEST;

b) relevant PROBLEM REPORT; and

c) approval of the CHANGE REQUEST

~~can be traced~~. [Class A, B, C]

8.3 * Configuration status accounting

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

9 * Software problem resolution PROCESS

9.1 Prepare PROBLEM REPORTS

The MANUFACTURER shall prepare a PROBLEM REPORT for each problem detected in~~ a~~ the MEDICAL DEVICE SOFTWARE~~ PRODUCT~~. PROBLEM REPORTS shall~~ be classified as follows:~~ include a statement of criticality (for example, effect on performance, SAFETY, or SECURITY) as well as other information that may aid in the resolution of the problem (for example, devices affected, supported accessories affected).

~~a) type;~~

~~EXAMPLE 1 corrective, preventive, or adaptive to new environment~~

~~b) scope; and~~

~~EXAMPLE 2 size of change, number of device models affected, supported accessories affected, resources involved, time to change~~

~~c) criticality.~~

~~EXAMPLE 3 effect on performance, SAFETY, or SECURITY~~

[Class A, B, C]

NOTE Problems can be discovered before or after release, 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.

[Class A, B, C]

NOTE A problem does not have to be corrected for the MANUFACTURER to comply with the software problem resolution PROCESS, provided that the problem is not relevant to SAFETY.

9.3 Advise relevant parties

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

[Class 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.

9.4 Use change control process

The MANUFACTURER shall approve and implement all CHANGE REQUESTS, observing the requirements of the change control PROCESS (see 8.2). [Class 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 ~~(see 7.4)~~. [Class A, B, C]

9.6 Analyse problems for trends

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

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 in the appropriate MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~ and ACTIVITIES; and

d) additional problems have been introduced.

[Class 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:

a) test results;

b) ANOMALIES found;

c) the VERSION of software tested;

d) relevant hardware and software test configurations;

e) relevant test tools;

f) date tested; and

g) identification of the tester.

[Class A, B, C]

Annex A
(informative)

Rationale for the requirements of this standard

Rationale for the clauses of this standard is provided in this annex.

A.1 Rationale

The primary requirement of this standard is that a set of PROCESSES be followed in the development and maintenance of MEDICAL DEVICE 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 standard 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 standard requires the first category to be performed for all MEDICAL DEVICE SOFTWARE and the second category to be performed for selected SOFTWARE ITEMS.

A claim of compliance with this standard should therefore include a documented RISK ANALYSIS that identifies foreseeable sequences of events that include software and that can result in a HAZARDOUS SITUATION (see ISO 14971). ~~HAZARDS~~ HAZARDOUS SITUATIONS that can be indirectly caused by software (for example, by providing misleading information that could cause inappropriate treatment to be 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 "[Class A, B, C]", indicating that they are required irrespective of the classification of the software to which they apply.

ACTIVITIES are required for all classes A, B, and C for the following reasons:

– the ACTIVITY produces a plan relevant to RISK MANAGEMENT or software safety classification;

– the ACTIVITY produces an output that is an input to RISK MANAGEMENT or software safety classification;

– the ACTIVITY is a part of RISK MANAGEMENT or software safety classification;

– the ACTIVITY establishes an administration system, documentation or record-keeping system that supports RISK MANAGEMENT or software safety classification;

– the ACTIVITY normally takes place when the classification of the related software is unknown;

– the ACTIVITY can cause a change that could invalidate the current software safety classification of the associated software. This includes the discovery and analysis of safety related problems after release.

Other PROCESSES are required only for SOFTWARE SYSTEMS or SOFTWARE ITEMS classified in software safety classes B or C. ACTIVITIES required as parts of these PROCESSES are identified in the normative text as "[Class B, C]", or "[Class 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 classes 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 classes 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 class C for the following reasons:

– 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

Note that all PROCESSES and ACTIVITIES defined in this standard 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 class A ~~that cannot by definition cause a HAZARD~~ 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 HAZARD can be ~~easily~~ assured of SAFETY and effectiveness primarily through overall validation ACTIVITY during the design of a MEDICAL DEVICE (which is outside the scope of this standard) and through some simple software life cycle controls.

A.2 Summary of requirements by class

Table A.1 summarizes which software safety classes are assigned to each requirement. This table is informative and only provided for convenience. The normative section identifies the software safety classes for each requirement.

Table A.1 – Summary of requirements by software safety class

Clauses and subclauses		Class A	Class B	Class 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, 5.1.12		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
	~~5.8.1, 5.8.2,~~ 5.8.3, 5.8.5, 5.8.6~~, 5.8.7, 5.8.8~~		X	X
~~6.1~~	~~6.1~~	~~X~~	~~X~~	~~X~~
~~6.2~~	~~6.2.1, 6.2.2, 6.2.4, 6.2.5~~	~~X~~	~~X~~	~~X~~
	~~6.2.3~~		~~X~~	~~X~~
~~6.3~~	~~All requirements~~	~~X~~	~~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

Annex B
(informative)

Guidance on the provisions of this standard

B.1 Scope

B.1.1 Purpose

The purpose of this standard is to provide a development PROCESS that will consistently produce high quality, safe MEDICAL DEVICE SOFTWARE. To accomplish this, the standard 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 MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~.

This annex provides guidance for the application of the requirements of this standard. It does not add to, or otherwise change, the requirements of this standard. This annex can be used to better understand the requirements of this standard.

Note that in this standard, 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 and VERIFICATION, software integration and integration testing, SOFTWARE SYSTEM testing, and software release. The TASKS within these ACTIVITIES are the individual requirements.

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

Because of such logical dependencies between processes, it is easiest to describe the processes in this standard 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 12207 [9] include (see also Table B.1):

– Waterfall. The “once-through" strategy, also called “waterfall”, consists of performing the development PROCESS a single time. Simplistically: determine customer needs, define requirements, design the SYSTEM, implement the system, test, fix and deliver.
– Incremental: The “incremental” strategy determines customer needs and defines the SYSTEM requirements, then performs the rest of the development in a sequence of builds. The first build incorporates part of the planned capabilities, the next build adds more capabilities, and so on, until the SYSTEM is complete.
– Evolutionary: The “evolutionary” strategy also develops a SYSTEM in builds but differs from the incremental strategy in acknowledging that the user need is not fully understood and all requirements cannot be defined up front. In this strategy, customer needs and SYSTEM requirements are partially defined up front, then are refined in each succeeding build.

Table B.1 – Development (model) strategies as defined in ISO/IEC 12207

Development Strategy	Define all requirements first?	Multiple development cycles?	Distribute interim software?
Waterfall (Once-through)	yes	no	no
Incremental (Preplanned product improvement)	yes	yes	maybe
Evolutionary	no	yes	yes

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 standard;
– all PROCESS outputs should be available when needed as input to further work on the software.
– before any MEDICAL DEVICE SOFTWARE is released, all PROCESS outputs should be consistent with each other and all dependencies between PROCESS outputs explicitly or implicitly required by this standard should be observed.

B.1.2 Field of application

This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE as well as the development and maintenance of a MEDICAL DEVICE that includes SOUP.

The use of this standard requires the MANUFACTURER to perform MEDICAL DEVICE RISK MANAGEMENT that is compliant with ISO 14971. Therefore, when the MEDICAL DEVICE SYSTEM ARCHITECTURE includes an acquired component (this could be a purchased component or a component of unknown provenance), such as a printer/plotter that includes SOUP, the acquired component becomes the responsibility of the MANUFACTURER and must be included in the RISK MANAGEMENT of the MEDICAL DEVICE. It is assumed that through proper performance of MEDICAL DEVICE RISK MANAGEMENT, the MANUFACTURER would understand the component and recognize that it includes SOUP. The MANUFACTURER using this standard would invoke the software RISK MANAGEMENT PROCESS as part of the overall MEDICAL DEVICE RISK MANAGEMENT PROCESS.

The maintenance of released MEDICAL DEVICE SOFTWARE applies to the post-production experience with the MEDICAL DEVICE 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 released MEDICAL DEVICE

SOFTWARE ~~PRODUCT(S)~~. For example, this includes problem rectification, regulatory reporting, re-validation and preventive action. See ISO/IEC 14764 [10].

B.2 Normative references

ISO/IEC 90003 [15] provides guidance for applying a quality management system to software development. This guidance is not required by this standard but is highly recommended.

B.3 Terms and definitions

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

This standard chose to use three terms to describe the decomposition of a SOFTWARE SYSTEM (top level). The SOFTWARE SYSTEM can be a subsystem of the MEDICAL DEVICE (see IEC 60601-1-4 [2]) or a MEDICAL DEVICE in its own right, which then becomes a software 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 definition and granularity of the SOFTWARE ITEMS and SOFTWARE UNITS. Leaving these terms vague allows one to apply them to the many different development methods and types of software used in MEDICAL DEVICES.

B.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 MEDICAL DEVICE SOFTWARE:

– RISK MANAGEMENT;
– quality management;
– software engineering.

For the development and maintenance of safe MEDICAL DEVICE 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 MEDICAL DEVICE MANUFACTURER to follow a clearly structured and consistently repeatable decision-making PROCESS to promote SAFETY for MEDICAL DEVICE SOFTWARE.

B.4.1 Quality management system

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 standard on software engineering, these PROCESSES have been omitted from this standard. These PROCESSES are covered by a quality management system. ISO 13485 [8] 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 requirements does not automatically constitute conformity with national or regional regulatory requirements. It is the MANUFACTURER'S responsibility to identify and establish compliance with relevant regulatory requirements.

B.4.2 RISK MANAGEMENT

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

Rather than trying to define an appropriate RISK MANAGEMENT PROCESS in this software engineering standard, it is required that the MANUFACTURER apply a RISK MANAGEMENT PROCESS that is compliant with ISO 14971, which deals explicitly with RISK MANAGEMENT for MEDICAL DEVICES. Specific software RISK MANAGEMENT ACTIVITIES resulting from ~~HAZARDS~~ HAZARDOUS SITUATIONS that have software as a contributing cause are identified in a supporting PROCESS described in Clause 7.

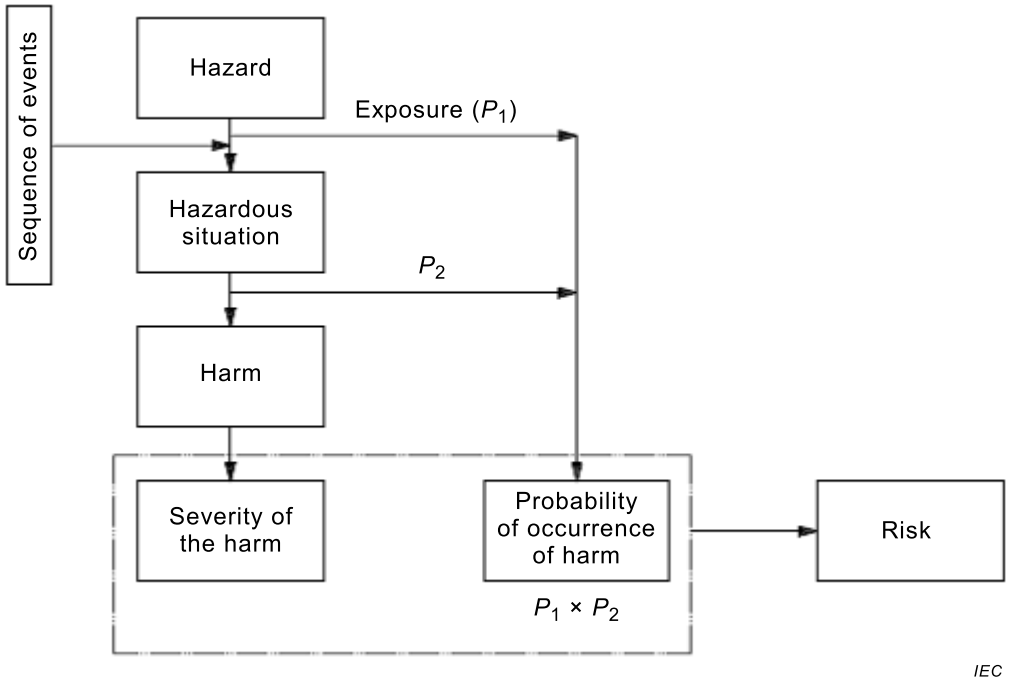
B.4.3 Software safety classification

The RISK associated with software as a part of a MEDICAL DEVICE, as an accessory to a MEDICAL DEVICE, or as a MEDICAL DEVICE in its own right, is used as the input to a software safety classification scheme, which then determines the PROCESSES to be used during the development and maintenance of software.

RISK is considered to be a combination of the severity of ~~injury~~ HARM and the probability of its occurrence. ~~However, there is no consensus on how to determine the probability of occurrence of software failures using traditional statistical methods. In this standard, therefore, SOFTWARE SYSTEM classification is based on the severity of the HAZARD resulting from failure of the software, assuming that the failure will occur. SOFTWARE SYSTEMS that contribute to the implementation of RISK CONTROL measures are classified based on the severity of the HAZARD they are controlling.~~ However, no consensus exists for a method of quantitatively estimating the probability of occurrence of a software failure. When software is present in a sequence or combination of events leading to a HAZARDOUS SITUATION, the probability of the software failure occurring cannot be considered in estimating the RISK for the HAZARDOUS SITUATION. In such cases, considering a worst case probability is appropriate, and the probability for the software failure occurring should be set to 1. When it is possible to estimate the probability for the remaining events in the sequence (as it may be if they are not software) that probability can be used for the probability of the HAZARDOUS SITUATION occurring (P_1 in Figure B.2).

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. Subjective rankings of probability can also be assigned based on clinical knowledge to distinguish failures that a clinician would be likely to detect from those that would not be detected and would be more likely to cause HARM.

Estimates of probability of a HAZARDOUS SITUATION leading to HARM (P_2 in Figure B.2) generally require clinical knowledge to distinguish between HAZARDOUS SITUATIONS where clinical practice would be likely to prevent HARM, and HAZARDOUS SITUATIONS that would be more likely to cause HARM.



NOTE P_1 is the probability of a hazardous situation occurring

P_2 is the probability of a hazardous situation leading to harm

Figure B.2 – Pictorial representation of the relationship of HAZARD, sequence of events, HAZARDOUS SITUATION, and HARM – from ISO 14971:2007 Annex E

If a SOFTWARE SYSTEM is decomposed into SOFTWARE ITEMS, then each SOFTWARE ITEM can have its own software safety classification.

It is only possible to determine the RISK associated with failure of a SOFTWARE ITEM:

– if a SYSTEM ARCHITECTURE and a software ARCHITECTURE define the role of the SOFTWARE ITEM in terms of its purpose and its interfaces with other software and hardware items;

– if changes to the SYSTEM are controlled;

– after RISK ANALYSIS has been done on the ARCHITECTURE and RISK CONTROL measures specified.

This standard requires the minimum number of ACTIVITIES that will achieve the above conditions for all classes of software.

The end of the software ARCHITECTURE ACTIVITY is the earliest point in the development when the full set of SOFTWARE ITEMS is defined and the RISK MANAGEMENT ACTIVITY has identified how the SOFTWARE ITEMS relate to SAFETY. This is therefore the earliest point at which SOFTWARE ITEMS can be classified definitively according to their SAFETY role.

This point corresponds to the point where RISK CONTROL is begun in ISO 14971.

Before this point, the RISK MANAGEMENT PROCESS identifies ARCHITECTURAL RISK CONTROL measures, for example adding protective subsystems, or reducing the opportunities for software failures to cause HARM. After this point, the RISK MANAGEMENT PROCESS uses PROCESSES aimed at reducing the probability of failure of SOFTWARE ITEMS. In other words, the classification of a SOFTWARE ITEM specifies PROCESS-based RISK CONTROL measures to be applied to that item.

It is expected that MANUFACTURERS will find it useful to classify software before this point, for example to focus attention on areas to be investigated, but such classification should be regarded as preliminary and should not be used to justify the omission of PROCESSES.

The software safety classification scheme is not intended to align with the RISK classifications of ISO 14971. Whereas the ISO 14971 scheme classifies RISK according to their severity and likelihood, the software safety classification scheme classifies SOFTWARE SYSTEMS and SOFTWARE ITEMS according to the PROCESSES to be applied in their development and maintenance.

As the design evolves, new RISKS might become evident. Therefore, RISK MANAGEMENT should be applied as an integral part of the development PROCESS. This permits the development of an ARCHITECTURAL design that identifies a complete set of SOFTWARE ITEMS, including those that are required to function correctly to assure safe operation and those that prevent faults from causing HARM.

The software ARCHITECTURE should promote segregation of software items that are required for safe operation and should describe the methods used to ensure effective segregation of those SOFTWARE ITEMS. Segregation is not restricted to physical (processor or memory partition) separation but includes any mechanism that prevents one SOFTWARE ITEM from negatively affecting another. The adequacy of a segregation is determined based on the RISKS involved and the rationale which is required to be documented.

As stated in B.3, this standard chooses to use three terms to describe the decomposition of a SOFTWARE SYSTEM (top level).

Figure B.1 illustrates the possible partitioning for SOFTWARE ITEMS within a SOFTWARE SYSTEM and how the software safety classes would be applied to the group of SOFTWARE ITEMS in the decomposition.

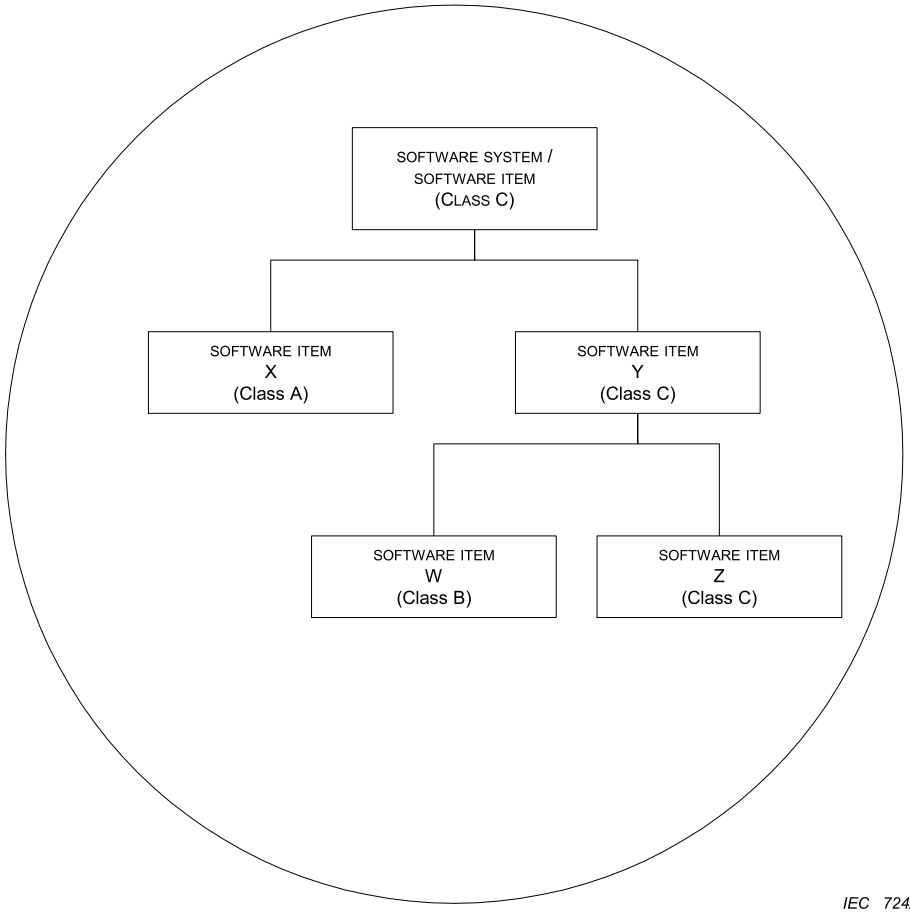


Figure B.1 – Example of partitioning of SOFTWARE ITEMS

For this example, the MANUFACTURER knows, due to the type of MEDICAL DEVICE SOFTWARE being developed, that the preliminary software safety classification for the SOFTWARE SYSTEM is software safety class C. During software ARCHITECTURE design the MANUFACTURER has decided to partition the SYSTEM, as shown, with 3 SOFTWARE ITEMS – X, W and Z. The MANUFACTURER is able to segregate all SOFTWARE SYSTEM contributions to ~~HAZARDS~~ HAZARDOUS SITUATIONS which could result in death or SERIOUS INJURY to SOFTWARE ITEM Z and all remaining SOFTWARE SYSTEM contributions to ~~HAZARDS~~ HAZARDOUS SITUATIONS which could result in a non-SERIOUS INJURY to SOFTWARE ITEM W. SOFTWARE ITEM W is classified as software safety class B and SOFTWARE ITEM Z is at software safety class C. SOFTWARE ITEM Y therefore must be classified as Class C, per 4.3 d). The SOFTWARE SYSTEM is also at a software safety class C per this requirement. SOFTWARE ITEM X has been classified at a software safety class 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 ~~partitioning~~ segregation is not possible between SOFTWARE ITEMS X and Y, then SOFTWARE ITEM X must be classified in software safety class C.

B.4.4 LEGACY SOFTWARE

Subclause 4.4 establishes a process for application of this standard to LEGACY SOFTWARE. Some geographies may require the MANUFACTURER to show conformity to the standard to obtain regulatory approval of the MEDICAL DEVICE SOFTWARE, even if that software was designed prior to the existence of the current version of the standard (LEGACY SOFTWARE). In this case, the requirements in 4.4 provide a method for the the MANUFACTURER to demonstrate compliance of LEGACY SOFTWARE to the standard.

A MANUFACTURER may determine that retrospective documentation of an already finished development-lifecycle 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 standard 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, and
– a MANUFACTURER should utilize resources as effectively as possible to effect a reduction of RISK.

In addition to a plan identifying the subset of ACTIVITIES to execute, the process also results in objective evidence supporting 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 MEDICAL DEVICE HAZARDS associated with the LEGACY SOFTWARE.

Subclause 4.4 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 may 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 may 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 should be assumed to be 1.

The MANUFACTURER determination of how the LEGACY SOFTWARE will be used in the overall MEDICAL DEVICE 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 use of 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 must take into account the modified set of HAZARDOUS SITUATIONS which can arise due to failures of the LEGACY SOFTWARE.
– A reused LEGACY SOFTWARE may be used for similar intended use but integrated into a new SOFTWARE SYSTEM. In this case the RISK assessment should take into account modification of architectural RISK CONTROL measures according to 5.3.

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 may be invalidated by the changes.

Changes to the LEGACY SOFTWARE should be performed according to Clauses 4 to 9 of this standard, including assessment of impact to RISK CONTROL measures according to 7.4. In the case of LEGACY SOFTWARE, existing RISK CONTROL measures may not be 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 4.4, 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 lifecycle model (as appropriate);

d) EVALUATION of the adequacy of existing RISK MANAGEMENT documentation, taking ISO 14971 into account.

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 may 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 SOFTARE reuse.

The rationale makes a positive case for the safe and reliable performace 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.

After LEGACY SOFTWARE has been re-used according to 4.4, those parts of the LEGACY SOFTWARE for which gaps in DELIVERABLES remain, continue to be LEGACY SOFTWARE and may be considered for further re-use again according to 4.4. When gaps in deliverables are closed by changing the LEGACY SOFTWARE, the changes should be performed according to Clauses 4 to 9 of this standard.

B.5 Software development PROCESS

B.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 the MEDICAL DEVICE 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 MEDICAL DEVICE 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 MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ 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 MEDICAL DEVICE SOFTWARE ~~PRODUCT~~. 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 rigor 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 safety classification 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 rigor needed for the SYSTEM or items in the SYSTEM to enable proper control over the development PROCESS.

B.5.2 Software requirements analysis

This ACTIVITY requires the MANUFACTURER to establish and verify the software requirements for the MEDICAL DEVICE SOFTWARE. Establishing verifiable requirements is essential for determining what is to be built, for determining that the MEDICAL DEVICE SOFTWARE exhibits acceptable behaviour, and for demonstrating that the completed MEDICAL DEVICE 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 MEDICAL DEVICE 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 with 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 standard. However, the decision to implement MEDICAL DEVICE 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 analyzing the requirements allocated to software by the SYSTEM requirements PROCESS and deriving a comprehensive set of software requirements that reflect the allocated requirements.

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 standard 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.

B.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 behavior should be described in the software ARCHITECTURE. This description is especially important for behaviour that can affect components of the MEDICAL DEVICE that are outside the software (see 5.3.5 and B.4.3). 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 safety classification 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 class. The software configuration management PROCESS (Clause 8) is used to ensure that all necessary rework is identified and completed.

B.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. ~~We have~~ This standard 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 must also be concerned with the packaging of the SOFTWARE PRODUCT. ~~It is necessary to document the design of each SOFTWARE UNIT and its interface so that the SOFTWARE UNIT can be implemented correctly. The~~

~~detailed design fills in the details necessary to construct the software.~~ 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 must also be concerned with the architecture of the MEDICAL DEVICE 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 safety class. However, the decision to do this is in itself part of the RISK MANAGEMENT PROCESS, and is documented in the RISK MANAGEMENT FILE.

Because implementation depends on detailed design, it is necessary to verify the detailed design before the ACTIVITY is complete. 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 are to 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.

B.5.5 SOFTWARE UNIT implementation and verification

This ACTIVITY requires the MANUFACTURER to write and verify the code for the SOFTWARE UNITS. The detailed design is to 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 MEDICAL DEVICE SOFTWARE will not perform as intended.

B.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 MEDICAL DEVICE hardware.

The rigor 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*, *structural*, *clear box* and *open box testing*, is a testing technique where explicit knowledge of the internal workings of the SOFTWARE ITEM being tested are 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, functional, opaque-box, 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 ~~a~~ MEDICAL DEVICE SOFTWARE ~~PRODUCT~~ 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 MEDICAL DEVICE.

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. These integrated SOFTWARE ITEMS must function properly for the entire MEDICAL DEVICE SOFTWARE to function correctly and safely.

B.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 previous section) 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 MEDICAL DEVICE.

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.6.3).

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 ~~HAZARD~~ RISK analysis to verify that they do not affect the SAFETY 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.

B.5.8 Software release

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

The MANUFACTURER should be able to show that the software that was developed using the development PROCESS is the software that is being released. 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.

B.6 Software maintenance PROCESS

B.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 the MEDICAL DEVICE SOFTWARE. This PROCESS is activated when the MEDICAL DEVICE 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 released MEDICAL DEVICE SOFTWARE while preserving its integrity. This PROCESS includes migration of the MEDICAL DEVICE SOFTWARE to environments or platforms for which it was not originally released. The ACTIVITIES provided in this clause are specific to the maintenance PROCESS; however, the maintenance PROCESS might use other PROCESSES in this standard.

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

B.6.2 Problem and modification analysis

This ACTIVITY requires the MANUFACTURER to analyze 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 ~~HAZARD~~ HAZARDOUS SITUATION or mitigate a RISK in software that previously did not cause a ~~HAZARD~~ HAZARDOUS SITUATION or mitigate RISKS. The software safety classification of a SOFTWARE ITEM might have changed if the software modification now can cause a HAZARD 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 the MEDICAL DEVICE SOFTWARE ~~PRODUCT~~. As part of a MEDICAL DEVICE, the software maintenance PROCESS needs to ensure that:

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

– MEDICAL DEVICE SOFTWARE ~~PRODUCTS are~~ is re-validated and re-released after modification with formal controls that ensure the rectification of the problem and the avoidance of further problems;

– the MANUFACTURER considers what other MEDICAL DEVICE SOFTWARE ~~PRODUCTS~~ 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;
- 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.

B.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 MEDICAL DEVICE SOFTWARE. Unless the MEDICAL DEVICE SOFTWARE is treated as a new development, analysis of the effect of a modification on the entire MEDICAL DEVICE SOFTWARE is necessary. Regression analysis and testing are employed to provide assurance that a change has not created problems elsewhere in the MEDICAL DEVICE 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 must be made that justifies the amount of REGRESSION TESTING that will be performed to ensure that the portions of the MEDICAL DEVICE SOFTWARE not being modified still perform as they did before the modification was made.

B.7 Software RISK MANAGEMENT PROCESS

Software RISK MANAGEMENT is a part of overall MEDICAL DEVICE RISK MANAGEMENT and cannot be adequately addressed in isolation. This standard requires the use of a RISK MANAGEMENT PROCESS that is compliant with ISO 14971. RISK MANAGEMENT as defined in ISO 14971 deals specifically with a framework for effective management of the RISKS associated with the use of MEDICAL DEVICES. One portion of ISO 14971 pertains to control of identified RISKS associated with each HAZARD identified during the RISK ANALYSIS. The software RISK MANAGEMENT PROCESS in this standard is intended to provide additional requirements for RISK CONTROL for software, including software that has been identified during the RISK ANALYSIS as potentially contributing to a hazardous situation, or software that is used to control MEDICAL DEVICE RISKS. The software RISK MANAGEMENT PROCESS is included in this standard for two reasons.

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

b) the general RISK MANAGEMENT standard, ISO 14971, provided as a normative reference in this standard, does not specifically address the RISK CONTROL of software and the placement of RISK CONTROL in the software development life cycle.

Software RISK MANAGEMENT is a part of overall MEDICAL DEVICE 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 MEDICAL DEVICE RISK MANAGEMENT ACTIVITIES and documentation as long as all requirements in this standard are met.

B.7.1 Analysis of software contributing to hazardous situations

It is expected that the device HAZARD 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 device 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 components, and the practicality of the RISK CONTROL measures assigned to software functions can be EVALUATED. At that time the device HAZARD analysis should be revised to include:

- revised hazardous situations;
- revised RISK CONTROL measures and software requirements;
- new hazardous situations arising from software, for example hazardous situations related to human factors.

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

B.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.

B.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 the MEDICAL DEVICE SOFTWARE.

B.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.

B.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.

B.9 Software problem resolution PROCESS

The software problem resolution PROCESS is a PROCESS for analyzing 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 analyzed 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 12207 [9] and IEC 60601-1-4 [2], Amendment 1. We have chosen to call it “software problem resolution” in this standard.

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 analyzed and EVALUATED for possible relevance to SAFETY (as specified in ISO 14971).

Software development plan(s) or procedures, as required in 5.1, are to 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.

Annex C
(informative)

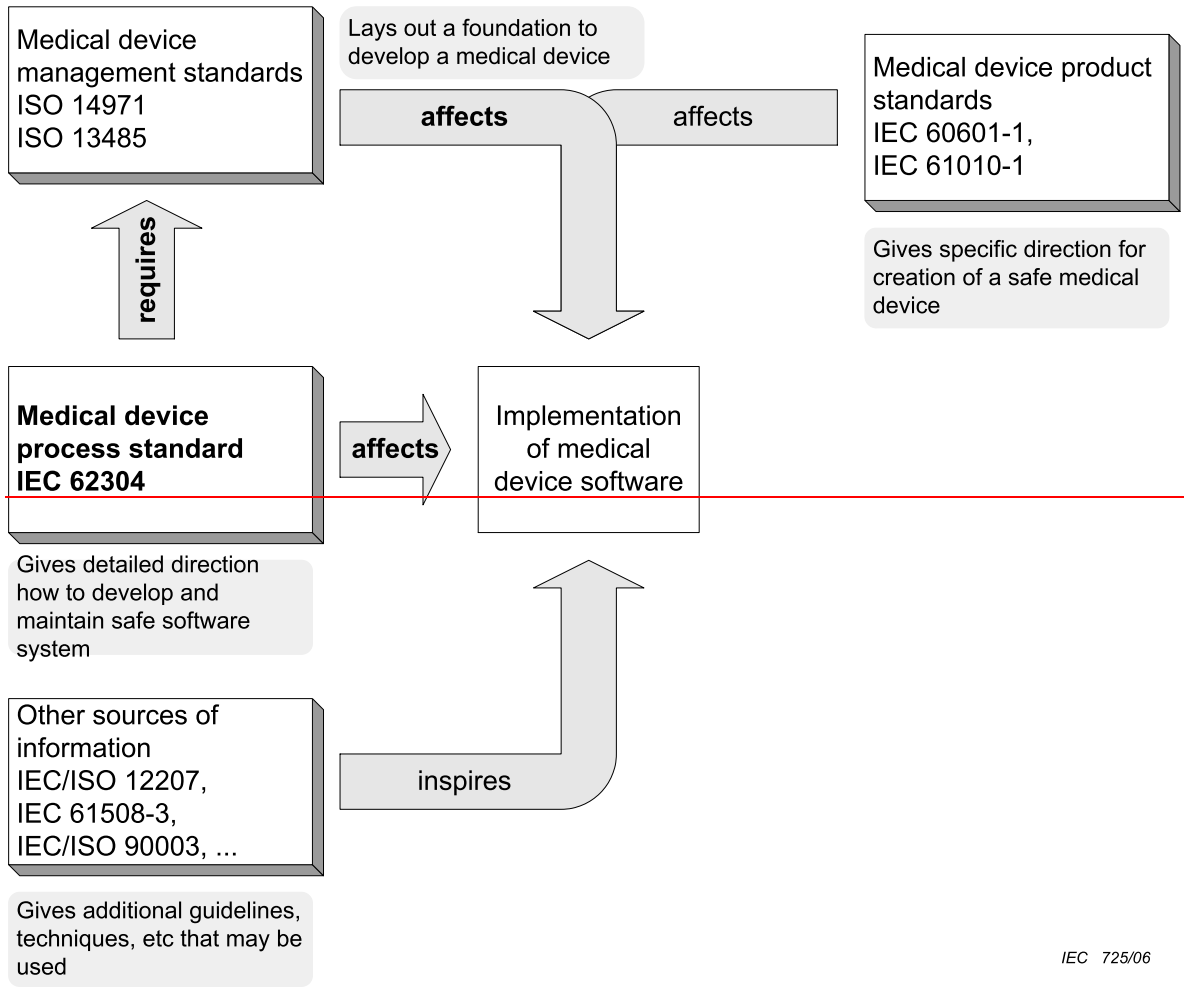
Relationship to other standards

C.1 General

This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE. The software is considered a subsystem of the MEDICAL DEVICE or is itself a MEDICAL DEVICE. This standard is to be used together with other appropriate standards when developing a MEDICAL DEVICE.

MEDICAL DEVICE management standards such as ISO 13485 [8] (see C.2 and Annex D) and ISO 14971 (see Annex C.3) provide a management environment that lays a foundation for an organization to develop products. Safety standards such as IEC 60601-1 [1] (see Annex C.4) and IEC 61010-1 [5] (see Annex C.5) give specific direction for creating safe MEDICAL DEVICES. When software is a part of these MEDICAL DEVICES, IEC 62304 provides more detailed direction on what is required to develop and maintain safe MEDICAL DEVICE SOFTWARE. Many other standards such as ISO/IEC 12207 [9] (see Annex C.6), IEC 61508-3 [4] (see Annex C.7) and ISO/IEC 90003 [15] can be looked to as a source of methods, tools and techniques that can be used to implement the requirements in IEC 62304. Figure C.1 shows the relationship of these standards.

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 standard.



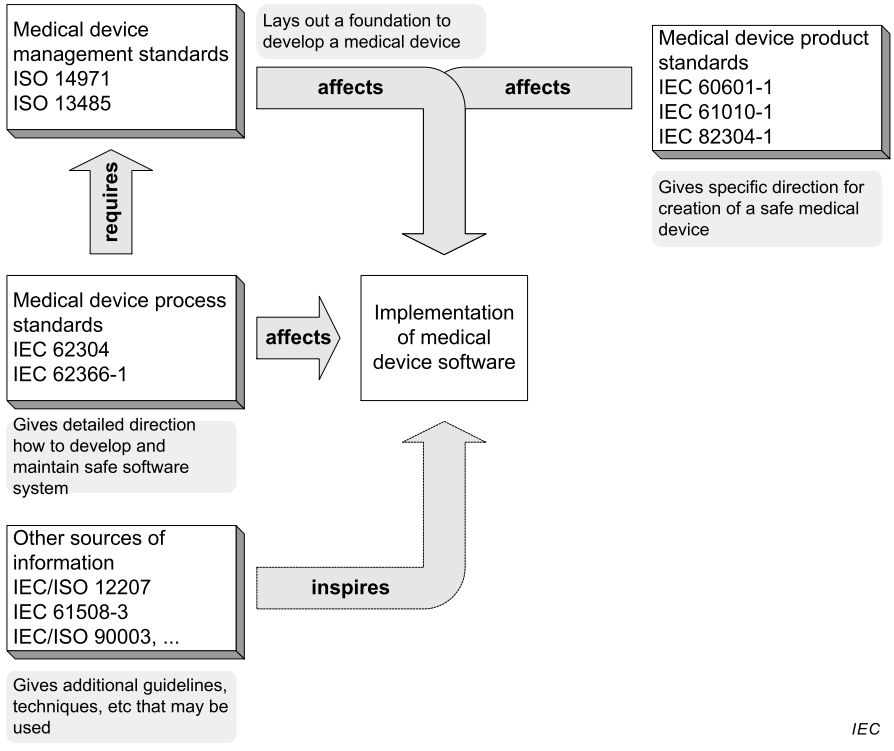


Figure C.1 – Relationship of key MEDICAL DEVICE standards to IEC 62304

C.2 Relationship to ISO 13485

This standard requires that the MANUFACTURER employs a quality management system. When a MANUFACTURER uses ISO 13485 [8], the requirements of ISO 62304 directly relate to some of the requirements of ISO 13485 as shown in Table C.1.

Table C.1 – Relationship to ISO 13485:2003

IEC 62304 clause	Related clause of ISO 13485:2003
5.1 Software development planning	7.3.1 Design and development planning
5.2 Software requirements analysis	7.3.2 Design and development inputs
5.3 Software ARCHITECTURAL design	
5.4 Software detailed design	
5.5 SOFTWARE UNIT implementation and verification	
5.6 Software integration and integration testing	
5.7 SOFTWARE SYSTEM testing	7.3.3 Design and development outputs 7.3.4 Design and development review
5.8 Software release	7.3.5 Design and development verification 7.3.6 Design and development validation
6.1 Establish software maintenance plan	7.3.7 Control of design and development changes
6.2 Problem and modification analysis	
6.3 Modification implementation	7.3.5 Design and development verification 7.3.6 Design and development validation
7.1 Analysis of software contributing to hazardous situations	
7.2 RISK CONTROL measures	
7.3 VERIFICATION of RISK CONTROL measures	
7.4 RISK MANAGEMENT of software changes	
8.1 Configuration identification	7.5.3 Identification and TRACEABILITY
8.2 Change control	7.5.3 Identification and TRACEABILITY
8.3 Configuration status accounting	
9 Software problem resolution PROCESS	

C.3 Relationship to ISO 14971

Table C.2 shows the areas where IEC 62304 amplifies requirements for the RISK MANAGEMENT PROCESS required by ISO 14971.

Table C.2 – Relationship to ISO 14971:~~2000~~ 2007

ISO 14971:~~2000~~ 2007 clause		Related clause of IEC 62304	
4.1	RISK ANALYSIS ~~procedure~~ process		
4.2	Intended use~~/intended purpose~~ and identification of characteristics related to the SAFETY of the MEDICAL DEVICE		
4.3	Identification of ~~known or foreseeable~~ HAZARDS	7.1	Analysis of software contributing to HAZARDOUS SITUATIONS
4.4	Estimation of the RISK(S) for each ~~HAZARD~~ HAZARDOUS SITUATION	4.3	Software safety classification
5	RISK evaluation		
6.1	RISK reduction		
6.2	RISK CONTROL option analysis	7.2.1	Define RISK CONTROL measures
6.3	Implementation of RISK CONTROL measures	7.2.2 7.3.1	RISK CONTROL measures implemented in software Verify RISK CONTROL measures
6.4	RESIDUAL RISK evaluation		
6.5	RISK/benefit analysis		
6.6	~~Other generated HAZARDS~~ RISKS arising from RISK CONTROL MEASURES	7.3.2	Document any new sequences of events
6.7	Completeness of RISK ~~evaluation~~ CONTROL		
7	Evaluation of overall RESIDUAL RISK ~~evaluation~~ acceptability		
8	RISK MANAGEMENT report	7.3.3	Document TRACEABILITY
9	Production and post-production information	7.4	RISK MANAGEMENT of software changes

C.4 Relationship to PEMS requirements of IEC 60601-1:2005 + IEC 606011:2005/AMD1:2012

C.4.1 General

Requirements for software are a subset of the requirements for a programmable electrical medical system (PEMS). This standard identifies requirements for software which are in addition to, but not incompatible with, the requirements of IEC 60601-1:2005 + IEC 60601-1:2005 /AMD1:2012 [1] for PEMS. Because PEMS include elements that are not software, not all of the requirements of IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 for PEMS are addressed in this standard. With the publication of IEC 60601-1:2005 + IEC 60601-1:2005 /AMD1:2012, IEC 62304 is now a normative reference of IEC 60601-1 and compliance with Clause 14 of IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 (and thus compliance with the standard) requires compliance with parts of IEC 62304 (not with the whole of IEC 62304 because IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 does not require compliance with post-production and maintenance requirements of IEC 62304). Finally, it is important to remember that IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 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 software standard 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.

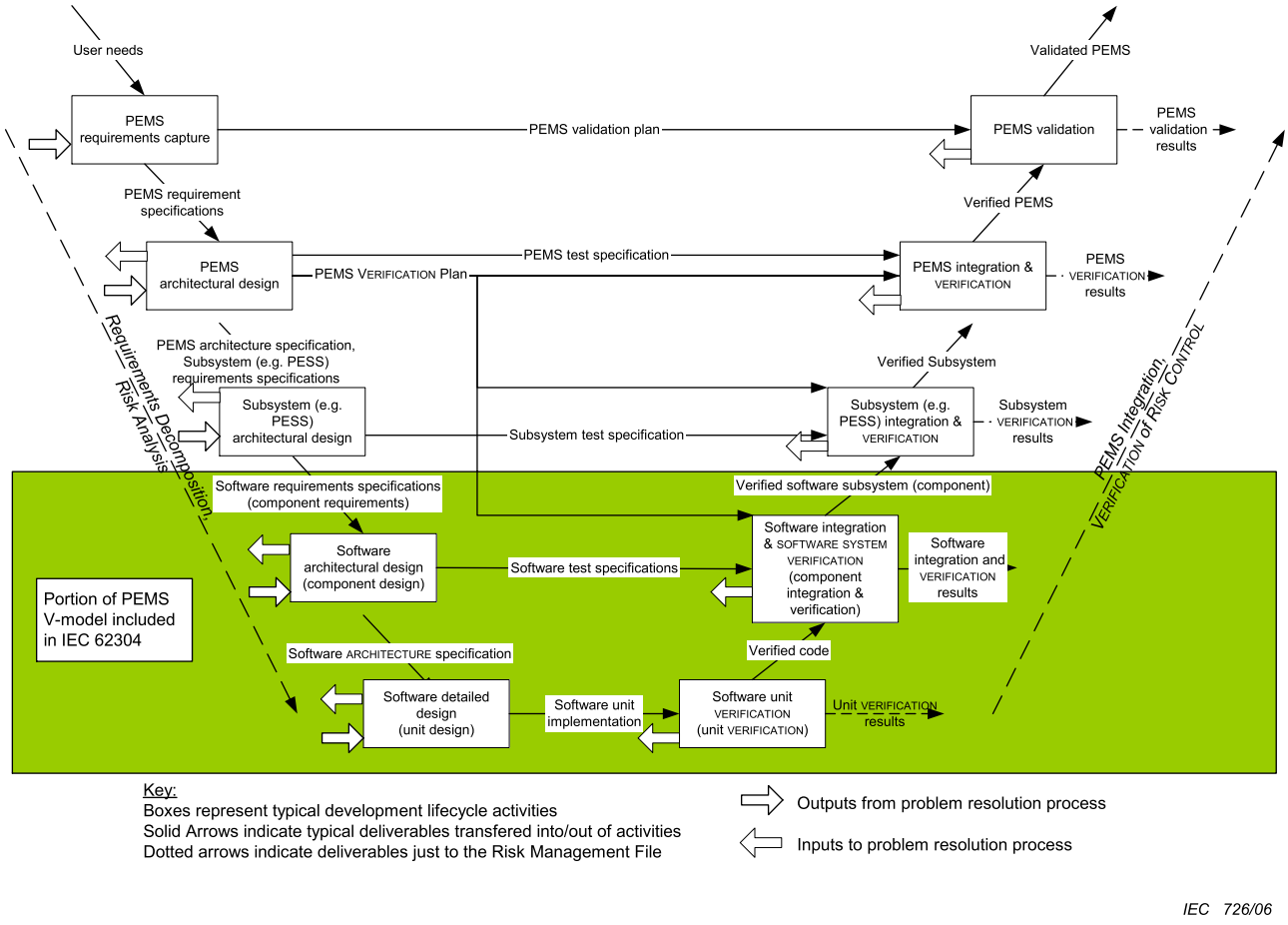


Figure C.2 – Software as part of the V-model

C.4.3 Development PROCESS

Compliance with the software development PROCESS of this standard (Clause 5) requires that a software development plan be specified and then followed; it does not require that any particular life cycle model is 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 for development life cycle, requirement specification, ARCHITECTURE, design and implementation, and VERIFICATION. The requirements in this standard provide greater detail about software development than those in IEC 60601-1.

C.4.4 Maintenance PROCESS

Compliance with the software maintenance PROCESS of this standard (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 standard for software maintenance provide greater detail about what must be done for software maintenance than the requirements for PEMS modification in IEC 60601-1.

C.4.5 Other PROCESSES

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

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

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

The software configuration management PROCESS in this standard 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 + IEC 606011:2005 /AMD1:2012

Table C.3 shows the PEMS requirements of IEC 60601-1 and the corresponding requirements in this standard.

Table C.3 – Relationship to IEC 60601-1

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
14.1 General The requirements ~~of this clause~~ in 14.2 to 14.12 (inclusive) shall apply to PEMS unless: – ~~the PESS~~ none of the PROGRAMMABLE ELECTRONIC SUBSYSTEMS (PESS) provides functionality necessary for ~~no~~ BASIC SAFETY or ESSENTIAL PERFORMANCE; or – ~~the application of ISO 14971~~ 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 subclause 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.	**4.3 Software safety classification** The PEMS requirements of IEC 60601-1 would only apply to software safety classes B and C. This standard includes some requirements for software safety class A. The software development PROCESS required for compliance with IEC 60601-1 does not include the post production monitoring and maintenance required by Clause 6 of IEC 62304:2006.
14.2 Documentation ~~In addition to the records and documents required by ISO 14971, the documents produced from application of Clause 14 shall be maintained and shall form part of the RISK MANAGEMENT FILE.~~	~~**4.2 RISK MANAGEMENT**~~
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 [8] requires control of the documents.
14.3 RISK MANAGEMENT PLAN The RISK MANAGEMENT plan required by ~~3.5 of ISO 14971~~ 4.2.2 shall also include a reference to the PEMS VALIDATION plan (see 14.11).	Not specifically required. There is no specific software validation plan. The PEMS validation plan is at the SYSTEM level and thus is outside the scope of this software standard. This standard 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 must be planned.
Each activity shall be defined including its inputs and outputs.	**5.1.1 Software development plan** ACTIVITIES are defined in this standard. Documentation to be produced is defined in each ACTIVITY.
Each milestone shall identify the RISK MANAGEMENT ACTIVITIES that must 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 standard allows the development life cycle to be documented in the development plan. This means the development plan contains a tailored development life cycle.

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
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 must 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 ~~NETWORK/DATA COUPLING~~ the incorporation of the PEMS into an IT-NETWORK, components of third-party origin and legacy subsystems.	**7.1 Analysis of software contributing to HAZARDOUS SITUATIONS** This standard does not mention network/data coupling specifically
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 standard 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 standard 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 standard 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	Requirements of IEC 62304 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: a) allocation of RISK CONTROL measures to subsystems and components of the PEMS; b) failure modes of components and their effects; c) common cause failures; d) systemic failures; e) test interval duration and diagnostic coverage; f) maintainability; g) protection from reasonably foreseeable misuse; h) the IT-NETWORK~~/DATA COUPLING~~ specification, if applicable.	This is not included in this standard.
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 standard does not require a test specification for detailed design.
Descriptive data regarding the design environment shall be included in the ~~RISK MANAGEMENT FILE~~ documentation.	**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

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
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 on 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 standard. 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~~, and shall require checks for unintended functioning of the PEMS~~.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
Methods used for PEMS VALIDATION shall be documented	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
The PEMS VALIDATION shall be performed according to the PEMS VALIDATION plan. The results of the PEMS VALIDATION activities shall be documented.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
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 standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
No member of a design team shall be responsible for the PEMS VALIDATION of their own design.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
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 standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
A reference to the methods and results of the PEMS VALIDATION shall be included in the RISK MANAGEMENT FILE.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
14.12 Modification If any or all of a design results from a modification of an earlier design then either all of this clause 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 standard 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 subclause 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	Requirements of IEC 62304 relating to the software subsystem of a PEMS
14.13 ~~Connection of PEMS by NETWORK/DATA COUPLING to other equipment~~ **PEMS intended to be incorporated into an IT-NETWORK** ~~If the PEMS is intended to be connected by NETWORK/DATA COUPLING to other equipment that is outside the control of the PEMS MANUFACTURER, the technical description shall:~~ ~~a) specify the characteristics of the NETWORK/DATA COUPLING necessary for the PEMS to achieve its INTENDED USE;~~ 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'S 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 and IEC 80001-1:2010). ~~b)~~f) list the HAZARDOUS SITUATIONS resulting from a failure of the IT-NETWORK~~/DATA COUPLING~~ to provide the ~~specified~~ characteristics required to meet the purpose of the PEMS connection to the IT-NETWORK. ~~c) Instruct the RESPONSIBLE ORGANIZATION that:~~ In the ACCOMPANYING DOCUMENTS, the MANUFACTURER shall instruct the RESPONSIBLE ORGANIZATION that: – connection of the PEMS to ~~a NETWORK/DATA COUPLING~~ 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;	Requirements for ~~network/data coupling~~ incorporation into an IT-network are not included in this standard.

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
NOTE 3 IEC 80001-1:2010 provides guidance for the RESPONSIBLE ORGANIZATION to address these risks. – subsequent changes to the IT-NETWORK~~/DATA COUPLING~~ could introduce new RISKS and require additional analysis; and – changes to the IT-NETWORK~~/DATA COUPLING~~ include: • changes in the IT-NETWORK~~/DATA COUPLING~~ configuration; • connection of additional items to the IT-NETWORK~~/DATA COUPLING~~; • disconnecting items from the IT-NETWORK~~/DATA COUPLING~~; • update of equipment connected to the IT-NETWORK~~/DATA COUPLING~~; • upgrade of equipment connected to the IT-NETWORK~~/DATA COUPLING~~.	

C.4.7 Relationship to requirements in IEC 60601-1-4

~~IEC 60601-1-4 will continue to be used until the transition period for IEC 60601-1:2005 is complete.~~

~~Table C.4 shows the requirements of IEC 60601-1-4 [2] and the related requirements in this standard. This does not indicate that the related requirements in this standard fully cover the requirements in IEC 60601-1-4. Many parts of the 60601-1-4 requirements are covered by compliance with ISO 14971. Some requirements in IEC 60601-1-4 are not addressed by IEC 62304.~~

~~Table C.4 – Relationship to IEC 60601-1-4~~

~~PEMS requirements from IEC 60601-1-4:1996 plus Amendment 1:1999~~	~~Related requirements of IEC 62304~~
~~6.8 Accompanying documents~~	
~~6.8.201~~	~~4.2 and 4.3 c)~~
~~52.201 Documentation~~	
~~52.201.1~~	~~4.1~~
~~52.201.2~~	~~4.1 and 4.2~~
~~52.201.3~~	~~4.2~~
~~52.202 RISK MANAGEMENT PLAN~~	
~~52.202.1~~	~~4.2~~
~~52.202.2~~	~~5.1.1, 5.1.5~~
~~52.202.3~~	~~4.1, 5.1.2~~
~~52.203 Development life-cycle~~	
~~52.203.1~~	~~5.1.1~~
~~52.203.2~~	~~5.1.1~~
~~52.203.3~~	
~~52.203.4~~	~~5.1.7~~
~~52.203.5~~	~~7~~
~~52.204 Risk management process~~	
~~52.204.1~~	~~4.2~~

PEMS requirements from IEC 60601-1-4:1996 plus Amendment 1:1999	Related requirements of IEC 62304
52.204.2	4.2, 7
52.204.3	
52.204.3.1	
52.204.3.1.1	4.2, 7.1
52.204.3.1.2	4.2, 7.1.2
52.204.3.1.3	4.2
52.204.3.1.4	4.2, 7.1.2 e)
52.204.3.1.5	4.2, 7.1.2
52.204.3.1.6	4.2, 7.1
52.204.3.1.7	4.2
52.204.3.1.8	4.2
52.204.3.1.9	4.2
52.204.3.1.10	4.2
52.204.3.2	
52.204.3.2.1	4.2
52.204.3.2.2	4.2, 4.3
52.204.3.2.3	
52.204.3.2.4	
52.204.3.2.5	4.2
52.204.4	
52.204.4.1	4.2
52.204.4.2	4.2
52.204.4.3	4.2
52.204.4.4	4.2
52.204.4.5	
52.204.4.6	4.2
52.205 Qualification of personnel	4.1
52.206 Requirement specification	
52.206.1	5.2
52.206.2	7.2.2
52.206.3	
52.207 Architecture	
52.207.1	5.3.1
52.207.2	5.3
52.207.3	
52.207.4	
52.207.5	
52.208 Design and implementation	
52.208.1	5
52.208.2	
52.209 Verification	
52.209.1	5.7.1
52.209.2	5.1.5, 5.1.6

~~PEMS requirements from IEC 60601-1-4:1996 plus Amendment 1:1999~~	~~Related requirements of IEC 62304~~
~~52.209.3~~	~~5.2.6, 5.3.6, 5.4.4, 5.5.5, 5.6, 5.7~~
~~52.209.4~~	
~~52.210 Validation~~	
~~52.210.1~~	~~4.1~~
~~52.210.2~~	~~4.1~~
~~52.210.3~~	~~4.1~~
~~52.210.4~~	
~~52.210.5~~	
~~52.210.6~~	
~~52.210.7~~	
~~52.211 Modification~~	
~~52.211.1~~	~~6~~
~~52.211.2~~	~~4.1, 6~~
~~52.212 Assessment~~	
~~52.212.1~~	~~4.1~~

IEC 60601-1-4 has been withdrawn.

C.5 Relationship to IEC 61010-1

The scope of IEC 61010-1 [~~4~~ 5] 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 must be 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 ~~HAZARD~~ HAZARDOUS SITUATION, for example failure caused by the software which results in an unwanted change of medical data (measuring results), IEC 62304 must be taken into account.

IEC 61010-1:2010 has a general requirement for risk assessment in Clause 17, which is more streamlined than the full risk management requirements of ISO 14971. Applying IEC 61010-1 Clause 17 alone does not meet the required criteria for risk management of IEC 62304, which is based on full ISO 14971 risk management requirements. With this in mind, it is expected by this standard 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. Compliance with Clause 17 of IEC 61010-1 will be achieved, as detailed in the Note to Clause 17 of IEC 61010-1:

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 ~~provides a useful aid to explain the principle way of the RISK MANAGEMENT PROCESS and~~ shows the application of IEC 62304 with IEC 61010-1, Clause 17:

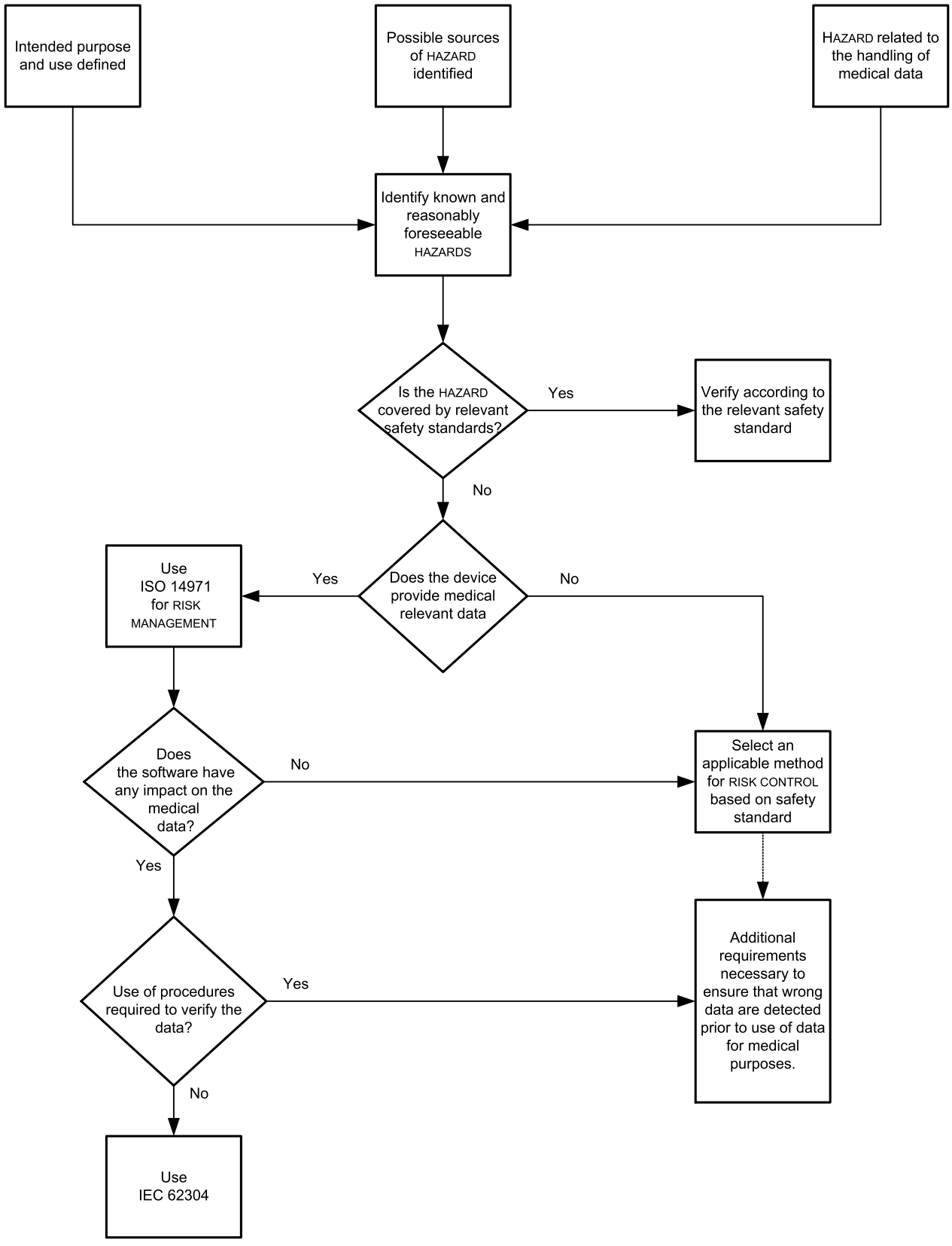


Figure C.3 – Application of IEC 62304 with IEC 61010-1

C.6 Relationship to ISO/IEC 12207

This standard has been derived from the approach and concepts of ISO/IEC 12207 [9], which defines requirements for software life cycle PROCESSES in general, i.e. not restricted to MEDICAL DEVICES.

This standard differs from ISO/IEC 12207 mainly with respect to the following. It:

- 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 12207 does.

Most of these changes were driven by the desire to tailor the standard 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 12207); and
- lowering the level of abstraction to make it easier to use.

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

Table C.5, which was prepared by ISO/IEC JTC1/SC7, shows the relationship between IEC 62304 and ISO/IEC 12207.

Table C.5 – Relationship to ISO/IEC 12207:2008

ISO/IEC 62304 processes		ISO/IEC 12207 processes	
Activity	Task	Activity	Task
5 Software development PROCESS		5.3 Development process 6.1 Documentation process 6.2 Configuration management process 6.4 Verification process 6.5 Validation process 6.8 Problem resolution process 7.1 Management process	
5.1 Software development planning		5.3.1 Process implementation 5.3.3 System architectural design 5.3.7 Software coding and testing 5.3.8 Software integration 5.3.9 Software qualification testing 5.3.10 System integration 6.1.1 Process implementation 6.2.1 Process implementation 6.2.2 Configuration identification 6.4.1 Process implementation 6.5.1 Process implementation 6.8.1 Process implementation 7.1.2 Planning 7.1.3 Execution and control 7.2.2 Establishment of the infrastructure 7.2.3 Maintenance of the infrastructure	
	5.1.1 Software development plan	5.3.1 Process implementation 7.1.2 Planning	5.3.1.1 5.3.1.3 5.3.1.4 7.1.2.1
	5.1.2 Keep software development plan updated	7.1.3 Execution and control	7.1.3.3
	5.1.3 Software development plan reference to SYSTEM design and development	5.3.3 System architectural design 5.3.10 System integration 6.5.1 Process implementation	5.3.3.1 5.3.10.1 6.5.1.4
	5.1.4 Software development standards, methods and tools planning	5.3.1 Process implementation	5.3.1.3 5.3.1.4
	5.1.5 Software integration and integration testing planning	5.3.8 Software integration.	5.3.8.1
	5.1.6 Software VERIFICATION planning	6.4.1 Process implementation 5.3.7 Software coding and testing 5.3.8 Software integration 5.3.9 Software qualification testing	6.4.1.4 6.4.1.5 5.3.7.5 5.3.8.5 5.3.9.3
	5.1.7 Software RISK MANAGEMENT planning	Amd.1:2002 – F.3.1.5 Risk management process	
	5.1.8 Documentation planning	6.1.1 Process implementation	6.1.1.1
	5.1.9 Software configuration management planning	6.2.1 Process implementation 6.8.1 Process implementation	6.2.1.1 6.8.1.1
	5.1.10 Supporting items to be controlled	7.2.2 Establishment of the infrastructure 7.2.3 Maintenance of the infrastructure	7.2.2.1 7.2.3.1
	5.1.11 Software CONFIGURATION ITEM control before VERIFICATION	6.2.2 Configuration identification	6.2.2.1
5.2 Software		5.3.3 System architectural	

ISO/IEC 62304 processes		ISO/IEC 12207 processes	
Activity	Task	Activity	Task
requirements analysis		design 5.3.4 Software requirements analysis 6.4.2 Verification	
	5.2.1 Define and document software requirements from SYSTEM requirements	5.3.3 System architectural design	5.3.3.1
	5.2.2 Software requirements content	5.3.4 Software requirements analysis	5.3.4.1
	5.2.3 Include RISK CONTROL measures in software requirements		
	5.2.4 Re-EVALUATE MEDICAL DEVICE RISK ANALYSIS		None
	5.2.5 Update SYSTEM requirements	5.3.4 Software requirements analysis	a) b)
	5.2.6 Verify software requirements	5.3.4 Software requirements analysis 6.4.2 Verification	5.3.4.2 6.4.2.3
5.3 Software ARCHITECTURAL design		5.3.5 Software architectural design	
	5.3.1 Transform software requirements into an ARCHITECTURE	5.3.5 Software architectural design	5.3.5.1
	5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS		5.3.5.2
	5.3.3 Specify functional and performance requirements of SOUP item		none
	5.3.4 Specify SYSTEM hardware and software required by SOUP item		none
	5.3.5 Identify segregation necessary for RISK CONTROL		none
	5.3.6 Verify software ARCHITECTURE	5.3.5 Software architectural design	5.3.5.6
5.4 Software detailed design		5.3.6 Software detailed design 6.4.2 Verification	
	5.4.1 Refine SOFTWARE ARCHITECTURE into SOFTWARE UNITS	5.3.6 Software detailed design	5.3.6.1
	5.4.2 Develop detailed design for each SOFTWARE UNIT		
	5.4.3 Develop detailed design for interfaces		5.3.6.2
	5.4.4 Verify detailed design	6.4.2 Verification	5.3.6.7
5.5 SOFTWARE UNIT implementation and verification		5.3.6 Software detailed design 5.3.7 Software coding and testing 6.4.2 Verification	
	5.5.1 Implement each SOFTWARE UNIT	5.3.7 Software coding and testing	5.3.7.1
	5.5.2 Establish SOFTWARE UNIT VERIFICATION PROCESS	5.3.6 Software detailed design 5.3.7 Software coding and testing	5.3.6.5 5.3.7.5
	5.5.3 SOFTWARE UNIT acceptance criteria	5.3.7 Software coding and testing	5.3.7.5
	5.5.4 Additional SOFTWARE UNIT acceptance criteria	5.3.7 Software coding and testing 6.4.2 Verification	5.3.7.5 6.4.2.5
	5.5.5 SOFTWARE UNIT VERIFICATION	5.3.7 Software coding and testing	5.3.7.2
5.6 Software integration and integration testing		5.3.8 Software integration 5.3.9 Software qualification	

ISO/IEC 62304 processes		ISO/IEC 12207 processes	
Activity	Task	Activity	Task
		testing 5.3.10 System integration 6.4.1 Process implementation 6.4.2 Verification	
	5.6.1 Integrate SOFTWARE UNITS	5.3.8 Software integration	5.3.8.2
	5.6.2 Verify software integration	5.3.8 Software integration 5.3.10 System integration	5.3.8.2 5.3.10.1
	5.6.3 Test integrated software	5.3.9 Software qualification testing.	5.3.9.1
	5.6.4 Integration testing content		5.3.9.3
	5.6.5 Verify integration tests procedures	6.4.2 Verification	6.4.2.2
	5.6.6 Conduct regression tests	5.3.8 Software integration	5.3.8.2
	5.6.7 Integration test record contents	5.3.8 Software integration	5.3.8.2
	5.6.8 Use software problem resolution PROCESS	6.4.1 Process implementation	6.4.1.6
5.7 SOFTWARE SYSTEM testing		5.3.8 Software integration 5.3.9 Software qualification testing 6.4.1 Process implementation 6.4.2 Verification 6.8.1 Process implementation	
	5.7.1 Establish tests for each software requirement	5.3.8 Software integration 5.3.9 Software qualification testing	5.3.8.4 5.3.9.1
	5.7.2 Use software problem resolution PROCESS	6.4.1 Process implementation	6.4.1.6
	5.7.3 Retest after changes	6.8.1 Process implementation	6.8.1.1
	5.7.4 Verify SOFTWARE SYSTEM testing	6.4.2 Verification 5.3.9 Software qualification testing	6.4.2.2 5.3.9.3
	5.7.5 Document data for each test SOFTWARE SYSTEM test record content	5.3.9 Software qualification testing	5.3.9.1
5.8 Software release		5.3.9 Software qualification testing 5.4.2 Operational testing 6.2.5 Configuration evaluation 6.2.6 Release management and delivery	
	5.8.1 Ensure software VERIFICATION is complete	5.4.2 Operational testing 6.2.6 Release management and delivery	5.4.2.1 5.4.2.2 6.2.6.1
	5.8.2 Document known residual ANOMALIES	6.2.5 Configuration evaluation 5.3.9 Software qualification testing	6.2.5.1 5.3.9.3
	5.8.3 Evaluate known residual ANOMALIES		
	5.8.4 Document released VERSIONS	6.2.6 Release management and delivery	6.2.6.1
	5.8.5 Document how released software was created		
	5.8.6 Ensure activities and tasks are complete		
	5.8.7 Archive software		
	5.8.8 Assure repeatability of software release		
6 Software maintenance PROCESS		5.5 Maintenance process 6.2 Configuration management process	
6.1 Establish software maintenance plan		5.5.1 Process implementation	5.5.1.1
6.2 Problem and modification analysis		5.5.1 Process Implementation 5.5.2 Problem and modification	

ISO/IEC 62304 processes		ISO/IEC 12207 processes	
Activity	Task	Activity	Task
		analysis 5.5.3 Modification implementation 5.5.5 Migration	
	6.2.1 Record and evaluate feedback		
	6.2.1.1 Monitor feedback	5.5.1 Process Implementation	5.5.1.1 5.5.1.2
	6.2.1.2 Document and EVALUATE feedback		
	6.2.1.3 Evaluate PROBLEM REPORT'S affects on SAFETY	5.5.2 Problem and modification analysis	5.5.2.1 5.5.2.2 5.5.2.3 5.5.2.4
	6.2.2 Use software problem resolution PROCESS	5.5.1 Process Implementation	5.5.1.2
	6.2.3 Analyse CHANGE REQUESTS	5.5.2 Problem and modification analysis	5.5.2.1
	6.2.4 CHANGE REQUEST approval	5.5.2 Problem and modification analysis	5.5.2.5
	6.2.5 Communicate to users and regulators	5.5.3 Modification implementation 5.5.5 Migration	5.5.3.1 5.5.5.3
6.3 Modification implementation		5.5.3 Modification implementation 6.2.6 Release management and delivery	
	6.3.1 Use established PROCESS to implement modification	5.5.3 Modification implementation	5.5.3.2
	6.3.2 Re-release modified SOFTWARE SYSTEM	6.2.6 Release management and delivery	6.2.6.1
7 Software RISK MANAGEMENT PROCESS		Amd.1:2002 – F.3.15 Risk management process Process in 62304 addresses risk / hazard issues that are not addressed in Amd 1. There is some commonality (risk measures, etc) but the focus of the analysis is quite different.	
8 Software configuration management PROCESS		5.5 Maintenance process 6.2 Configuration management process	
8.1 Configuration identification		6.2.2 Configuration identification	
	8.1.1 Establish means to identify CONFIGURATION ITEMS	6.2.2 Configuration identification	6.2.2.1
	8.1.2 Identify SOUP		none
	8.1.3 Identify SYSTEM configuration documentation	6.2.2 Configuration identification	6.2.2.1
8.2 Change control		5.5.3 Modification implementation 6.2.3 Configuration control	
	8.2.1 Approve CHANGE REQUESTS	6.2.3 Configuration control	6.2.3.1
	8.2.2 Implement changes	5.5.3 Modification implementation 6.2.3 Configuration control	5.5.3.2 6.2.3.1
	8.2.3 Verify changes	6.2.3 Configuration control	6.2.3.1
	8.2.4 Provide means for TRACEABILITY of change		
8.3 Configuration status accounting		6.2.4 Configuration status accounting	6.2.4.1
9 Software problem resolution PROCESS		5.5 Maintenance process 6.2 Configuration management 6.8 Problem resolution process	
9.1 Prepare PROBLEM REPORTS		6.8.1 Process implementation 6.8.2 Problem resolution	6.8.1.1 b) 6.8.2.1
9.2 Investigate the problem		6.8.2 Problem resolution 6.8.1 Process implementation	6.8.2.1 6.8.1.1 b)
9.3 Advise relevant parties		6.8.1 Process implementation	6.8.1.1 a)

~~ISO/IEC 62304 processes~~		~~ISO/IEC 12207 processes~~	
~~Activity~~	~~Task~~	~~Activity~~	~~Task~~
~~9.4 Use change control process~~		~~6.2.3 Configuration control.~~ ~~5.5.3 Modification implementation~~	
~~9.5 Maintain records~~		~~6.8.1 Process implementation~~	~~6.8.1.1 a)~~
~~9.6 Analyse problems for trends~~		~~6.8.1 Process implementation~~ ~~6.8.2 Problem resolution~~	~~6.8.1.1 b)~~ ~~6.8.2.1~~
~~9.7 Verify software problem resolution~~		~~6.8.1 Process implementation~~	~~6.8.1.1 d)~~
~~9.8 Test documentation contents~~			~~All testing tasks in 12207 require documentation~~

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
5 Software development PROCESS			
5.1 Software development planning			
	5.1.1 Software development plan	7.1.1 Software Implementation	7.1.1.3.1 Software implementation strategy 7.1.1.3.1.1 7.1.1.3.1.3 7.1.1.3.1.4 6.3.1.3.2 Project planning 6.3.1.3.2.1
	5.1.2 Keep software development plan updated	6.3.2 Project Assesment and Control	6.3.2.3.2 Project control 6.3.2.3.2.1
	5.1.3 Software development plan reference to SYSTEM design and development	6.4.3 System Architectural Design 6.4.5 System Integration 7.2.5 Software Validation Process	6.4.3.3.1 Establishing architecture 6.4.3.3.1.1 6.4.5.3.1 Integration 6.4.5.3.1.1 7.2.5.3.1 Process implementation 7.2.5.3.1.4
	5.1.4 Software development standards, methods and tools planning	7.1.1 Software Implementation	7.1.1.3.1 Software implementation strategy 7.1.1.3.1.3
	5.1.5 Software integration and integration testing planning	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.1
	5.1.6 Software VERIFICATION planning	7.2.4 Software Verification 7.1.5 Software Construction 7.1.6 Software Integration 7.1.7 Software Qualification Testing	7.2.4.3.1 Process implementation 7.2.4.3.1.4 7.2.4.3.1.5 7.1.5.3.1 Software construtution 7.1.5.3.1.5 7.1.6.3.1 Software integration 7.1.6.3.1.5 7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.1.7 Software RISK MANAGEMENT planning	6.3.4 Risk Management Process	
	5.1.8 Documentation planning	7.2.1 Software Documentation Management	7.2.1.3.1 Process implementation 7.2.1.3.1.1

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
	5.1.9 Software configuration management planning	7.2.2 Software Configuration Management 7.2.8 Software Problem Resolution	7.2.2.3.1 Process implementation 7.2.2.3.1.1 7.2.8.3.1 Process implementation 7.2.8.3.1.1
	5.1.10 Supporting items to be controlled	6.2.2 Infrastructure Management 6.2.2 Infrastructure Management	6.2.2.3.2 Establishment of the infrastructure 6.2.2.3.2.1 6.2.2.3.3 Maintenance of the infrastructure 6.2.2.3.3.1
	5.1.11 Software CONFIGURATION ITEM control before VERIFICATION	7.2.2 Software Configuration Management	7.2.2.3.2 Configuration identification 7.2.2.3.2.1
5.2 Software requirements analysis			
	5.2.1 Define and document software requirements from SYSTEM requirements	6.4.3 System Architectural Design	6.4.3.3.1 Establishing architecture 6.4.3.3.1.1
	5.2.2 Software requirements content	7.1.2 Software Requirements Analysis	7.1.2.3.1 Software requirements analysis 7.1.2.3.1.1
	5.2.3 Include RISK CONTROL measures in software requirements		
	5.2.4 Re-EVALUATE MEDICAL DEVICE RISK ANALYSIS	None	None
	5.2.5 Update SYSTEM requirements	7.1.2 Software Requirements Analysis	7.1.2.3.1 Software requirements analysis 7.1.2.3.1.1 a) & b)
	5.2.6 Verify software requirements	7.2.4 Software Verification	7.2.4.3.2 Verification 7.2.4.3.2.1
5.3 Software ARCHITECTURAL design	5.3.1 Transform software requirements into an ARCHITECTURE	7.1.3 Software Architectural Design	7.1.3.3.1 Software architectural design 7.1.3.3.1.1
	5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS		7.1.3.3.1 Software architectural design 7.1.3.3.1.2
	5.3.3 Specify functional and performance requirements of SOUP item	None	none
	5.3.4 Specify SYSTEM hardware and software required by SOUP item	None	none
	5.3.5 Identify segregation necessary for RISK CONTROL	None	none
	5.3.6 Verify software ARCHITECTURE	7.1.3 Software Architectural Design	7.1.3.3.1 Software architectural design 7.1.3.3.1.6
5.4 Software detailed design	5.4.1 Refine SOFTWARE ARCHITECTURE into SOFTWARE UNITS	7.1.4 Software Detailed Design	7.1.4.3.1 Software detailed design 7.1.4.3.1.1
	5.4.2 Develop detailed design for each SOFTWARE UNIT		
	5.4.3 Develop detailed design for interfaces		7.1.4.3.1 Software detailed design 7.1.4.3.1.2

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
	5.4.4 Verify detailed design	7.1.4 Software Detailed Design	7.1.4.3.1 Software detailed design 7.1.4.3.1.7
5.5 SOFTWARE UNIT implementation and verification	5.5.1 Implement each SOFTWARE UNIT	7.1.5 Software Construction	7.1.5.3.1 Software construction 7.1.5.3.1.1
	5.5.2 Establish SOFTWARE UNIT VERIFICATION PROCESS	7.1.4 Software Detailed Design 7.1.5 Software Construction	7.1.4.3.1 Software detailed design 7.1.4.3.1.5 7.1.5.3.1 Software Construction 7.1.5.3.1.5
	5.5.3 SOFTWARE UNIT acceptance criteria	7.1.5 Software Construction	7.1.5.3.1 Software construction 7.1.5.3.1.5
	5.5.4 Additional SOFTWARE UNIT acceptance criteria	7.1.5 Software Construction 7.2.4 Software Verification	7.1.5.3.1 Software construction 7.1.5.3.1.2
	5.5.5 SOFTWARE UNIT VERIFICATION	7.1.5 Software Construction	7.1.5.3.1 Software construction 7.1.5.3.1.2
5.6 Software integration and integration testing	5.6.1 Integrate SOFTWARE UNITS	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2
	5.6.2 Verify software integration	7.1.6 Software Integration 6.4.5 System Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2 6.4.5.3.1 Integration 6.4.5.3.1.2
	5.6.3 Test integrated software	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.1
	5.6.4 Integration testing content	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.6.5 Verify integration tests procedures	None	None
	5.6.6 Conduct regression tests	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2
	5.6.7 Integration test record contents	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2
	5.6.8 Use software problem resolution PROCESS	7.2.4 Software Verification	7.2.4.3.1 Process implementation 7.2.4.3.1.6
5.7 SOFTWARE SYSTEM testing	5.7.1 Establish tests for each software requirement	7.1.6 Software Integration 7.1.7 Software Qualification Testing	7.1.6.3.1 Software integration 7.1.6.3.1.4 7.1.7.3.1 Software qualification testing 7.1.7.3.1.1
	5.7.2 Use software problem resolution PROCESS	7.2.4 Software Verification	7.2.4.3.1 Process implementation 7.2.4.3.1.6
	5.7.3 Retest after changes	7.2.8 Software Problem Resolution	7.2.8.3.1 Process implementation 7.2.8.3.1.1
	5.7.4 Verify SOFTWARE SYSTEM testing	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.3

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
	5.7.5 SOFTWARE SYSTEM test record contents	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.1
5.8 Software release	5.8.1 Ensure software VERIFICATION is complete	6.4.9 Software Operation 7.2.2 Software Configuration Management	6.4.9.3.2 Operation activation and check-out 6.4.9.3.2.1 6.4.9.3.2.2 7.2.2.3.6 Release management and delivery 7.2.2.3.6.1
	5.8.2 Document known residual ANOMALIES	7.2.2 Software Configuration Management 7.1.7 Software Qualification Testing	7.2.2.3.5 Configuration evaluation 7.2.2.3.5.1 7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.8.3 EVALUATE known residual ANOMALIES		
	5.8.4 Document released VERSIONS	7.2.2 Software Configuration Management Process	7.2.2.3.6 Release management and delivery 7.2.2.3.6.1
	5.8.5 Document how released software was created		
	5.8.6 Ensure ACTIVITIES and TASKS are complete		
	5.8.7 Archive software		
	5.8.8 Assure repeatability of software release		
6 Software maintenance PROCESS		6.4.10 Software Maintenance Process	
6.1 Establish software maintenance plan		6.4.10 Software Maintenance	None
6.2 Problem and modification analysis	6.2.1 Document and EVALUATE feedback	None	None
	6.2.1.1 Monitor feedback	6.4.10 Software Maintenance	None
	6.2.1.2 Document and EVALUATE feedback		
	6.2.1.3 EVALUATE PROBLEM REPORT'S effects on SAFETY	6.4.10 Software Maintenance	None
	6.2.2 Use software problem resolution PROCESS	6.4.10 Software Maintenance	None
	6.2.3 Analyse CHANGE REQUESTS	6.4.10 Software Maintenance	None
	6.2.4 CHANGE REQUEST approval	6.4.10 Software Maintenance	None
	6.2.5 Communicate to users and regulators	6.4.10 Software Maintenance	None
6.3 Modification implementation		None	None
	6.3.1 Use established PROCESS to implement modification	6.4.10 Software Maintenance	None
	6.3.2 Re-release modified SOFTWARE SYSTEM	7.2.2 Software Configuration Management	None
7 Software RISK MANAGEMENT PROCESS		6.3.4 Risk Management Process This is based on ISO/IEC 16085. 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			
8.1 Configuration identification	8.1.1 Establish means to identify CONFIGURATION ITEMS	7.2.2 Software Configuration Management	None

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
	8.1.2 Identify SOUP	None	None
	8.1.3 Identify SYSTEM configuration documentation	7.2.2 Software Configuration Management	None
8.2 Change control	8.2.1 Approve CHANGE REQUESTS	7.2.2 Software Configuration Management	None
	8.2.2 Implement changes	6.4.10 Software Maintenance	None
	8.2.3 Verify changes	7.2.2 Software Configuration Management	None
	8.2.4 Provide means for TRACEABILITY of change		
8.3 Configuration status accounting		7.2.2 Software Configuration Management	None
9 Software problem resolution PROCESS			
9.1 Prepare PROBLEM REPORTS		7.2.8 Software Problem Resolution	None
9.2 Investigate the problem		7.2.8 Software Problem Resolution	None
9.3 Advise relevant parties		7.2.8 Software Problem Resolution	None
9.4 Use change control process		7.2.2 Software Configuration Management 6.4.10 Software Maintenance	None
9.5 Maintain records		7.2.8 Software Problem Resolution	None
9.6 Analyse problems for trends		7.2.8 Software Problem Resolution	None
9.7 Verify software problem resolution		7.2.8 Software Problem Resolution	None
9.8 Test documentation contents		All testing TASKS in ISO 12207 require documentation	None

C.7 Relationship to IEC 61508

The question has been raised whether this standard, being concerned with the design of SAFETY-critical software, should follow the principles of IEC 61508. The approach to safety in IEC 62304 is fundamentally different than the one in IEC 61508. IEC 62304 takes into account that the effectiveness of medical devices justifies residual risks related to their use. The following explains the stance of this standard.

IEC 61508 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 standard 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 MEDICAL DEVICE SOFTWARE.

For issue 2), this standard takes a simpler approach than IEC 61508. The latter classifies software into 4 "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 standard simplifies issue 2) by ~~disallowing~~ defining the classification into 3 software safety classes based on the RISK caused by a failure ~~consideration of probability of software failure prior to classification~~. ~~Classification into 3 software safety classes is based only on the severity of that HARM caused by a failure.~~ After classification, different PROCESSES are required for different software safety classes: the intention is to further reduce the probability (and/or the severity) of failure of the software.

Issue 3) is not addressed by this standard. Readers of the standard are encouraged to use IEC 61508 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 standard 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 standard makes no requirement for an independent safety assessor, since this is a matter for ISO 14971.

Annex D
(informative)

Implementation

D.1 Introduction

This annex gives an overview of how this standard can be implemented into MANUFACTURERS' PROCESSES. It also considers that other standards like ISO 13485 [8] require adequate and comparable PROCESSES.

D.2 Quality management system

For MANUFACTURERS of MEDICAL DEVICES, including MEDICAL DEVICE SOFTWARE in the context of this standard, the establishment of a quality management system (QMS) is required in 4.1. This standard 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 QM 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 standard.

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

This standard can be implemented by adapting or extending the PROCESSES already installed in the QMS system, or integrating new PROCESSES. This standard does not specify how this is to 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 standard are suitably put into action when the MEDICAL DEVICE 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 safety classification (A, B or C) of the software. Table D.1 lists all ACTIVITIES described in this standard. The reference to ISO 13485 should help to define the place in the QMS. Based on the required software safety class, 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.

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

ACTIVITY	Related clause of ISO 13485:2003	Covered by existing procedure?	If yes: Reference	Actions to be taken
5.1 Software development planning	7.3.1 Design and development planning	Yes/No		
5.2 Software requirements analysis	7.3.2 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 and verification		Yes/No		
5.6 Software integration and integration testing		Yes/No		
5.7 SOFTWARE SYSTEM testing	7.3.3 Design and development outputs 7.3.4 Design and development review	Yes/No		
5.8 Software release	7.3.5 Design and development verification 7.3.6 Design and development validation	Yes/No		
6.1 Establish software maintenance plan	7.3.7 Control of design and development changes	Yes/No		
6.2 Problem and modification analysis		Yes/No		
6.3 Modification implementation	7.3.5 Design and development verification 7.3.6 Design and development validation	Yes/No		
7.1 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	7.5.3 Identification and traceability	Yes/No		
8.2 Change control	7.5.3 Identification and traceability	Yes/No		
8.3 Configuration status accounting		Yes/No		
9 Software problem resolution PROCESS		Yes/No		

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 60601-1-4:1996, *Medical electrical equipment – Part 1: General requirements for safety – 4.Collateral standard: Programmable electrical medical systems* (withdrawn)
IEC 60601-1-4:1996/AMD1:1999

[3] IEC 60601-1-6, *Medical electrical equipment – Part 1-6: General requirements for basic safety and essential performance – Collateral standard: Usability*

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

[5] IEC 61010-1:~~2001~~ 2010, *Safety requirements for electrical equipment for measurement, control, and laboratory use – Part 1: General requirements*

[6] ISO 9000:2005, *Quality management systems – Fundamentals and vocabulary*

[7] ISO 9001:~~2000~~ 2008, *Quality management systems – Requirements*

[8] ISO 13485:2003, *Medical devices – Quality management systems – Requirements for regulatory purposes*

~~[9] ISO/IEC 9126-1:2001, *Software engineering – Product quality – Part 1: Quality model*~~

[9] ISO/IEC 12207:~~1995~~ 2008, ~~*Information technology*~~ *Systems and software engineering – Software life cycle processes*
~~Amendment 1 (2002)~~
~~Amendment 2 (2004)~~

[10] ISO/IEC 14764:1999, ~~*Information technology – Software maintenance*~~ *Software Engineering – Software Life Cycle Processes – Maintenance*

[11] ISO/IEC 15504-5:2012, *Information technology – Process assessment – Part 5: An exemplar software life cycle process assessment model*

[12] ISO/IEC 25010:2011, *Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*

[13] ISO/IEC 33001:—[2)], *Information technology – Process assessment – Concepts and terminology*

[14] ISO/IEC 33004:—[2)], *Information technology – Process assessment – Requirements for process reference, process assessment and maturity models*

[15] ISO/IEC 90003:~~2004~~ 2014, *Software engineering – Guidelines for the application of ISO 9001:~~2000~~ 2008 to computer software*

[16] ISO/IEC Guide 51:~~1999~~ 2014, *Safety aspects – Guidelines for their inclusion in standards*

[17] IEEE 610.12:1990, *IEEE standard glossary of software engineering terminology*

[18] IEEE 1044:~~1993~~ 2009, *IEEE standard classification for software anomalies*

---

2 To be published.

[19] U.S. Department Of Health and Human Services, Food and Drug Administration, Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices, May 11, 2005, <://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089543.htm>

[20] U.S. Department Of Health and Human Services, Food and Drug Administration, *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*, January 11, 2002, <http://www.fda.gov/downloads/RegulatoryInformation/Guidances/ucm126955.pdf>

[21] IEC 62366-1:2015, *Medical devices – Part 1: Application of usability engineering to medical devices*

[22] IEC 82304-1:—[3)], *Healthcare Software Systems – Part 1: General requirements*

3 In preparation.

Index of defined terms

IEC 62304

Edition 1.1 2015-06

FINAL VERSION

colour inside

Medical device software – Software life cycle processes

IEC 62304:2006-05+AMD1:2015-06 CSV(en)

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

MEDICAL DEVICE 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.

DISCLAIMER
This Consolidated version is not an official IEC Standard and has been prepared for user convenience. Only the current versions of the standard and its amendment(s) are to be considered the official documents.

This Consolidated version of IEC 62304 bears the edition number 1.1. It consists of the first edition (2006-05) [documents 62A/523/FDIS and 62A/528/RVD] and its amendment 1 (2015-06) [documents 62A/1007/FDIS and 62A/1014/RVD]. The technical content is identical to the base edition and its amendment.

This Final version does not show where the technical content is modified by amendment 1. A separate Redline version with all changes highlighted is available in this publication.

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 and ISO Technical Committee 210, Quality management and corresponding general aspects for MEDICAL DEVICES. Table C.5 was prepared by ISO/IEC JTC 1/SC 7, Software and system engineering.

It is published as a dual logo standard.

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

In this standard the following print types are used:

- requirements and definitions: in roman type;
- informative material appearing outside of tables, such as notes, examples and references: in smaller type. Normative text of tables is also in a smaller type;
- terms used throughout this standard that have been defined in Clause 3 and also given in the index: in small capitals.

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 the base publication and its amendment will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

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

NOTE The attention of National Committees 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.

INTRODUCTION

Software is often an integral part of MEDICAL DEVICE technology. Establishing the SAFETY and effectiveness of a MEDICAL DEVICE containing software requires knowledge of what the software is intended to do and demonstration that the use of the software fulfils those intentions without causing any unacceptable RISKS.

This standard provides a framework of life cycle PROCESSES with ACTIVITIES and TASKS necessary for the safe design and maintenance of MEDICAL DEVICE SOFTWARE. This standard provides requirements for each life cycle PROCESS. Each life cycle PROCESS consists of a set of ACTIVITIES, with most ACTIVITIES consisting of a set of TASKS.

As a basic foundation it is assumed that MEDICAL DEVICE SOFTWARE is developed and maintained within a quality management system (see 4.1) and a RISK MANAGEMENT system (see 4.2). The RISK MANAGEMENT PROCESS is already very well addressed by the International Standard ISO 14971. Therefore IEC 62304 makes use of this advantage simply by a normative reference to ISO 14971. Some minor additional RISK MANAGEMENT requirements are needed for software, especially in the area of identification of contributing software factors related to HAZARDS. These requirements are summarized and captured in Clause 7 as the software RISK MANAGEMENT PROCESS.

Whether software is a contributing factor to a HAZARDOUS SITUATION is determined during the HAZARD identification ACTIVITY of the RISK MANAGEMENT PROCESS. HAZARDOUS SITUATIONS that could be indirectly caused by software (for example, by providing misleading information that could cause inappropriate treatment to be administered) need to be considered when determining whether software is a contributing factor. The decision to use software to control RISK is made during the RISK CONTROL ACTIVITY of the RISK MANAGEMENT PROCESS. The software RISK MANAGEMENT PROCESS required in this standard has to be embedded in the device RISK MANAGEMENT PROCESS according to ISO 14971.

The software development PROCESS consists of a number of ACTIVITIES. These ACTIVITIES are shown in Figure 1 and described in Clause 5. Because many incidents in the field are related to service or maintenance of MEDICAL DEVICE SYSTEMS including inappropriate software updates and upgrades, the software maintenance PROCESS is considered to be as important as the software development PROCESS. The software maintenance PROCESS is very similar to the software development PROCESS. It is shown in Figure 2 and described in Clause 6.

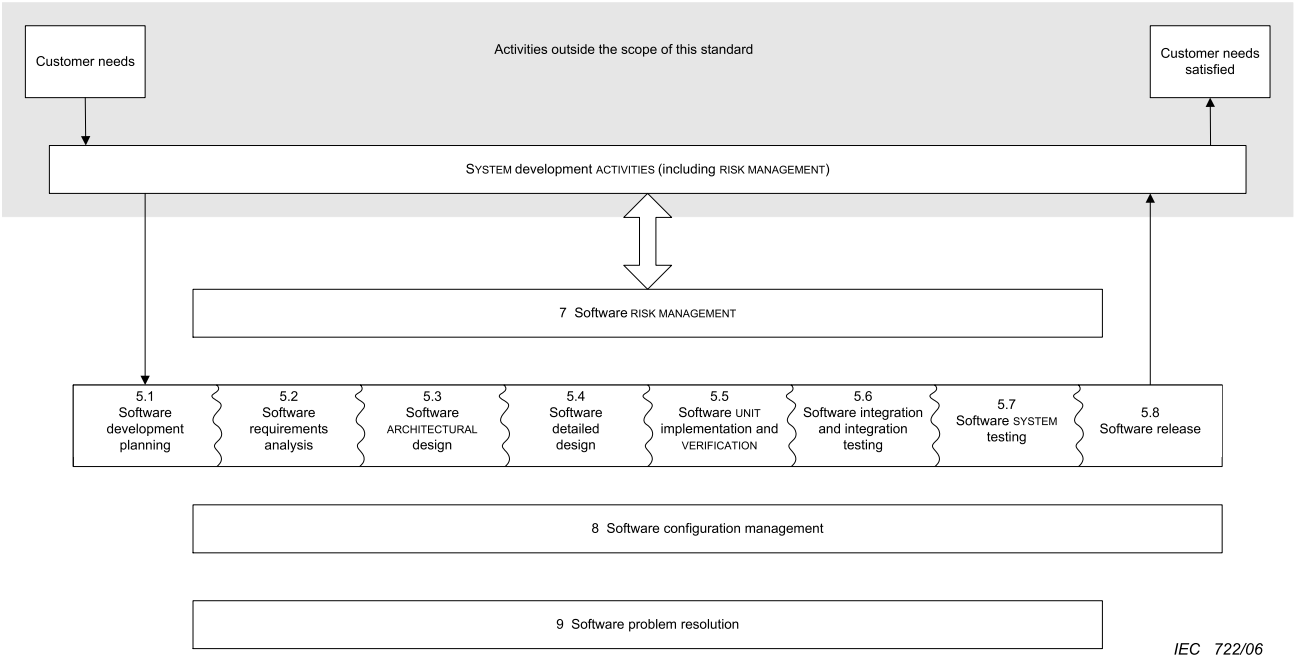


Figure 1 – Overview of software development PROCESSES and ACTIVITIES

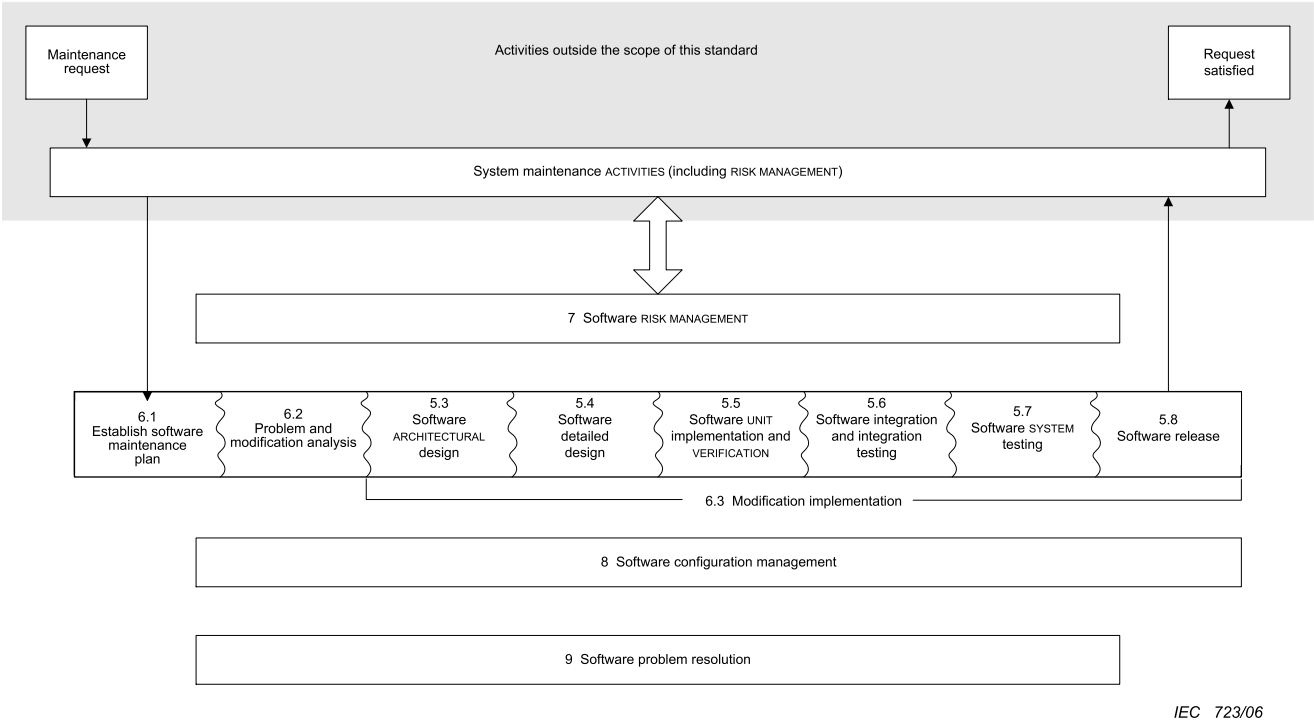


Figure 2 – Overview of software maintenance PROCESSES and ACTIVITIES

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

Amendment 1 updates the standard to add requirements to deal with LEGACY SOFTWARE, where the software design is prior to the existence of the current version, to assist manufacturers who must show compliance to the standard to meet European Directives. Software safety

classification changes include clarification of requirements and updating of the software safety classification to include a risk-based approach.

This standard does not specify an organizational structure for the MANUFACTURER or which part of the organization is to perform which PROCESS, ACTIVITY, or TASK. This standard requires only that the PROCESS, ACTIVITY, or TASK be completed to establish compliance with this standard.

This standard does not prescribe the name, format, or explicit content of the documentation to be produced. This standard requires documentation of TASKS, but the decision of how to package this documentation is left to the user of the standard.

This standard does not prescribe a specific life cycle model. The users of this standard are responsible for selecting a life cycle model for the software project and for mapping the PROCESSES, ACTIVITIES, and TASKS in this standard onto that model.

Annex A provides rationale for the clauses of this standard. Annex B provides guidance on the provisions of this standard.

For the purposes of this standard:

- “shall” means that compliance with a requirement is mandatory for compliance with this standard;
- “should” means that compliance with a requirement is recommended but is not mandatory for compliance with this standard;
- “may” is used to describe a permissible way to achieve compliance with a requirement;
- “establish” means to define, document, and implement; and
- where this standard 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.

INTRODUCTION to Amendment 1

The first edition of IEC 62304 was published in 2006. This amendment is intended to add requirements to deal with LEGACY SOFTWARE, where the software design is prior to the existence of the current version, to assist manufacturers who must show compliance to the standard to meet European Directives. Software safety classification changes needed for this amendment include clarification of requirements and updating of the software safety classification to include a risk-based approach. Work is continuing in parallel to develop the second edition of IEC 62304.

MEDICAL DEVICE SOFTWARE – SOFTWARE LIFE CYCLE PROCESSES

1 Scope

1.1 * Purpose

This standard defines the life cycle requirements for MEDICAL DEVICE SOFTWARE. The set of PROCESSES, ACTIVITIES, and TASKS described in this standard establishes a common framework for MEDICAL DEVICE SOFTWARE life cycle PROCESSES.

1.2 * Field of application

This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE when software is itself a MEDICAL DEVICE or when software is an embedded or integral part of the final MEDICAL DEVICE.

NOTE 1 This standard can be used in the development and maintenance of software that is itself a medical device. However, additional development activities are needed at the system level before this type of software can be placed into service. These system activities are not covered by this standard, but can be found in IEC 82304-1[1] [22].

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

This standard applies regardless of the method of delivery of the software (for example: transmission by network or email, optical disk, flash memory or EEPROM). The method of software delivery itself is not considered MEDICAL DEVICE SOFTWARE.

This standard does not cover validation and final release of the MEDICAL DEVICE, even when the MEDICAL DEVICE consists entirely of software.

NOTE 2 If a medical device incorporates embedded software intended to be executed on a processor, the requirements of this standard apply to the software, including the requirements concerning software of unknown provenance (see 8.1.2).

NOTE 3 Validation and other development activities are needed at the system level before the software and medical device can be placed into service. These system activities are not covered by this standard, but can be found in related product standards (e.g., IEC 60601-1, IEC 82304-1, etc.).

1.3 Relationship to other standards

This MEDICAL DEVICE SOFTWARE life cycle standard is to be used together with other appropriate standards when developing a MEDICAL DEVICE. Annex C shows the relationship between this standard and other relevant standards.

1.4 Compliance

Compliance with this standard is defined as implementing all of the PROCESSES, ACTIVITIES, and TASKS identified in this standard in accordance with the software safety class.

NOTE The software safety classes assigned to each requirement are identified in the normative text following the requirement.

---

1 In preparation.

Compliance is determined by inspection of all documentation required by this standard including the RISK MANAGEMENT FILE, and assessment of the PROCESSES, ACTIVITIES and TASKS required for the software safety class.

NOTE 1 This assessment could be carried out by internal or external audit.

NOTE 2 Although the specified PROCESSES, ACTIVITIES, and TASKS are performed, flexibility exists in the methods of implementing these PROCESSES and performing these ACTIVITIES and TASKS.

NOTE 3 Where any requirements contain "as appropriate" and were not performed, documentation for the justification is necessary for this assessment.

NOTE 4 The term "conformance" is used in ISO/IEC 12207 where the term "compliance" is used in this standard.

NOTE 5 For compliance of LEGACY SOFTWARE see 4.4.

2 * Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 14971, *Medical devices – Application of risk management to medical devices*.

3 * Terms and definitions

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

3.1
ACTIVITY
a set of one or more interrelated or interacting TASKS

3.2
ANOMALY
any condition that deviates from the expected based on requirements specifications, design documents, standards, etc. or from someone's perceptions or experiences. ANOMALIES may be found during, but not limited to, the review, test, analysis, compilation, or use of MEDICAL DEVICE SOFTWARE or applicable documentation

NOTE Based on IEEE 1044:1993, definition 3.1.

3.3
ARCHITECTURE
organizational structure of a SYSTEM or component

[IEEE 610.12:1990]

3.4
CHANGE REQUEST
a documented specification of a change to be made to a MEDICAL DEVICE SOFTWARE

3.5
CONFIGURATION ITEM
entity that can be uniquely identified at a given reference point

NOTE Based on ISO/IEC 12207:2008, 4,7.

3.6
DELIVERABLE
required result or output (includes documentation) of an ACTIVITY or TASK

3.7
EVALUATION
a systematic determination of the extent to which an entity meets its specified criteria

[ISO/IEC 12207:2008, 4.12]

3.8
HARM
physical injury, damage, or both to the health of people or damage to property or the environment

[ISO 14971:2007, 2.2]

3.9
HAZARD
potential source of HARM

[ISO 14971:2007, 2.3]

3.10
MANUFACTURER
natural or legal person with responsibility for designing, manufacturing, packaging, or labelling a MEDICAL DEVICE; assembling a SYSTEM; or adapting a MEDICAL DEVICE before it is placed on the market and/or put into service, regardless of whether these operations are carried out by that person or by a third party on that person's behalf

NOTE 1 Attention is drawn to the fact that the provisions of national or regional regulations can apply to the definition of manufacturer.

NOTE 2 For a definition of labelling, see ISO 13485:2003, definition 3.6.

[ISO 14971:2007, 2,8]

3.11
MEDICAL DEVICE
any instrument, apparatus, implement, machine, appliance, implant, in vitro reagent or calibrator, 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 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 for medical purposes by means of in vitro examination of specimens derived from the human body,

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

NOTE 1 This definition has been developed by the Global Harmonization Task Force (GHTF). See bibliographic reference [15] (in ISO 13485:2003).

[ISO 13485:2003, definition 3.7]

NOTE 2 Some differences can occur in the definitions used in regulations of each country.

NOTE 3 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).

3.12
MEDICAL DEVICE SOFTWARE
SOFTWARE SYSTEM that has been developed for the purpose of being incorporated into the MEDICAL DEVICE being developed or that is intended for use as a MEDICAL DEVICE

NOTE This includes a MEDICAL DEVICE software product, which then is a MEDICAL DEVICE in its own right.

3.13
PROBLEM REPORT
a record of actual or potential behaviour of a MEDICAL DEVICE SOFTWARE that a user or other interested person believes to be unsafe, inappropriate for the intended use or contrary to specification

NOTE 1 This standard does not require that every PROBLEM REPORT results in a change to the MEDICAL DEVICE SOFTWARE. A MANUFACTURER can reject a PROBLEM REPORT as a misunderstanding, error or insignificant event.

NOTE 2 A PROBLEM REPORT can relate to a released MEDICAL DEVICE SOFTWARE or to a MEDICAL DEVICE SOFTWARE that is still under development.

NOTE 3 This standard requires the MANUFACTURER to perform extra decision making steps (see Clause 6) for a PROBLEM REPORT relating to a released product to ensure that regulatory actions are identified and implemented.

3.14
PROCESS
a set of interrelated or interacting ACTIVITIES that transform inputs into outputs

[ISO 9000:2000, definition 3.4.1]

NOTE The term "ACTIVITIES" covers use of resources.

3.15
REGRESSION TESTING
the testing required to determine that a change to a SYSTEM component has not adversely affected functionality, reliability or performance and has not introduced additional defects

[ISO/IEC 90003:2004, definition 3.11]

3.16
RISK
combination of the probability of occurrence of HARM and the severity of that HARM

[ISO 14971:2007, 2.16]

3.17
RISK ANALYSIS
systematic use of available information to identify HAZARDS and to estimate the RISK

[ISO 14971:2007, 2.17]

3.18
RISK CONTROL
PROCESS in which decisions are made and RISKS are reduced to, or maintained within, specified levels

[ISO 14971:2007, 2.19]

3.19
RISK MANAGEMENT
systematic application of management policies, procedures, and practices to the TASKS of analyzing, evaluating, and controlling RISK

[ISO 14971:2007, 2.22, modified – The phrase "and monitoring" has been removed]

3.20
RISK MANAGEMENT FILE
set of records and other documents, not necessarily contiguous, that are produced by a RISK MANAGEMENT PROCESS

[ISO 14971:2007, 2.23]

3.21
SAFETY
freedom from unacceptable RISK

[ISO 14971:2007, 2.24]

3.22
SECURITY
protection of information and data so that unauthorized persons or systems cannot read or modify them an authorized persons or systems are not denied access to them

NOTE Based on ISO/IEC 12207: 2008, 4.39.

3.23
SERIOUS INJURY
injury or illness that:

a) is life threatening,

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

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

NOTE Permanent impairment means an irreversible impairment or damage to a body structure or function excluding trivial impairment or damage.

3.24
SOFTWARE DEVELOPMENT LIFE CYCLE MODEL
conceptual structure spanning the life of the software from definition of its requirements to its release, which:

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

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

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

NOTE Based on ISO/IEC 12207:1995, definition 3.11

3.25
SOFTWARE ITEM
any identifiable part of a computer program, i.e., source code, object code, control code, control data, or a collection of these items

NOTE 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 provide the granularity of the SOFTWARE ITEMS and SOFTWARE UNITS.

NOTE 2 Based on ISO/IEC 90003:2004, 3.14 and ISO/IEC 12207:2008, 4.41

3.26
Not used

3.27
SOFTWARE SYSTEM
integrated collection of SOFTWARE ITEMS organized to accomplish a specific function or set of functions

3.28
SOFTWARE UNIT
SOFTWARE ITEM that is not subdivided into other items

NOTE The granularity of SOFTWARE UNITS is defined by the MANUFACTURER (see B.3).

3.29
SOUP
software of unknown provenance (acronym)
SOFTWARE ITEM that is already developed and generally available and that has not been developed for the purpose of being incorporated into the MEDICAL DEVICE (also known as "off-the-shelf software") or SOFTWARE ITEM previously developed for which adequate records of the development PROCESSES are not available

NOTE A MEDICAL DEVICE SOFTWARE SYSTEM in itself cannot be claimed to be SOUP.

3.30
SYSTEM
integrated composite consisting of one or more of the PROCESSES, hardware, software, facilities, and people, that provides a capability to satisfy a stated need or objective

NOTE Based on ISO/IEC ISO/IEC 12207:2008, 4.48.

3.31
TASK
a single piece of work that needs to be done

3.32
TRACEABILITY
degree to which a relationship can be established between two or more products of the development PROCESS

[IEEE 610.12:1990]

NOTE Requirements, architecture, risk control measures, etc. are examples of deliverables of the development PROCESS.

3.33
VERIFICATION
confirmation through provision of objective evidence that specified requirements have been fulfilled

NOTE 1 "Verified" is used to designate the corresponding status.

[ISO 9000:2000, definition 3.8.4]

NOTE 2 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.

3.34
VERSION
identified instance of a CONFIGURATION ITEM

NOTE 1 Modification to a VERSION of MEDICAL DEVICE SOFTWARE, resulting in a new VERSION, requires software configuration management action.

NOTE 2 Based on ISO/IEC 12207:2008, 4.56.

3.35
HAZARDOUS SITUATION
circumstance in which people, property or the environment are exposed to one or more HAZARD(S)

[SOURCE: ISO 14971:2007, 2.4]

3.36
LEGACY SOFTWARE
MEDICAL DEVICE SOFTWARE which was legally placed on the market and is still marketed today but for which there is insufficient objective evidence that it was developed in compliance with the current version of this standard

3.37
RELEASE
particular VERSION of a CONFIGURATION ITEM that is made available for a specific purpose

NOTE Based on ISO/IEC 12207:2008, definition 4.35.

3.38
RESIDUAL RISK
RISK remaining after RISK CONTROL measures have been taken

NOTE 1 Adapted from ISO/IEC Guide 51:1999, definition 3.9.

NOTE 2 ISO/IEC Guide 51:1999, definition 3.9 uses the term “protective measures” rather than “RISK CONTROL measures.” However, in the context of this International Standard, “protective measures” are only one option for controlling RISK as described in 6.2 [of ISO 14971:2007].

[SOURCE: ISO 14971:2007, 2.15].

3.39
RISK ESTIMATION
PROCESS used to assign values to the probability of occurrence of HARM and the severity of that HARM

[SOURCE: ISO 14971:2007 2.20]

3.40
RISK EVALUATION
PROCESS of comparing the estimated RISK against given RISK criteria to determine the acceptability of the RISK

[SOURCE: ISO 14971:2007 2.21]

4 * General requirements

4.1 * Quality management system

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

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

- ISO 13485 [8]; 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 [15].

4.2 * RISK MANAGEMENT

The MANUFACTURER shall apply a RISK MANAGEMENT PROCESS complying with ISO 14971.

4.3 * Software safety classification

a) The MANUFACTURER shall assign to each SOFTWARE SYSTEM a software safety class (A, B, or C) according to the RISK of HARM to the patient, operator, or other people resulting from a HAZARDOUS SITUATION to which the SOFTWARE SYSTEM can contribute in a worst-case-scenario as indicated in Figure 3.

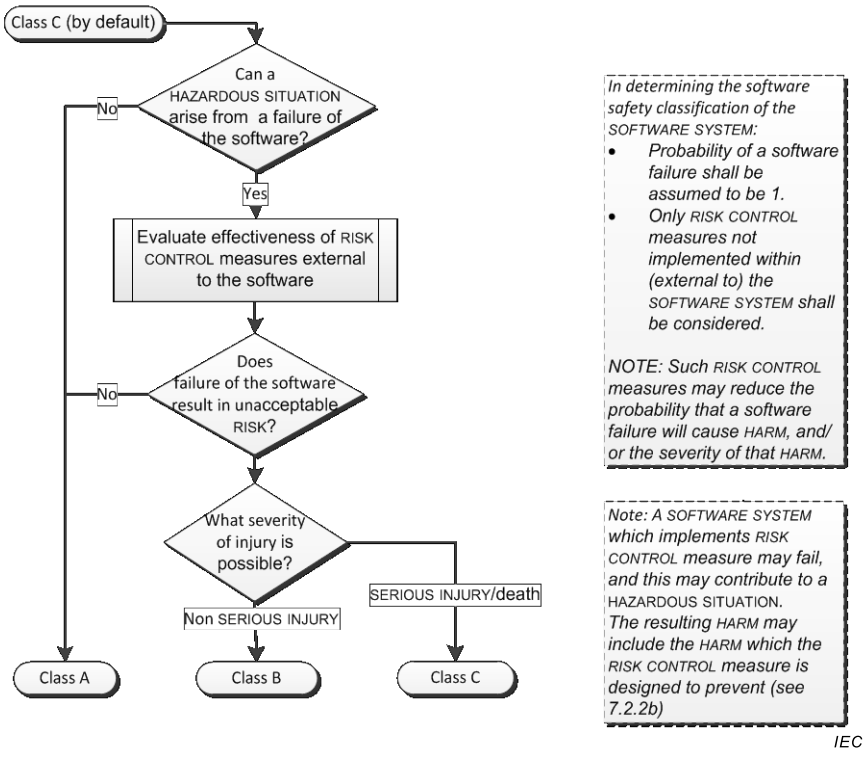


Figure 3 – Assigning software safety classification

The SOFTWARE SYSTEM is software safety class A if:

– the SOFTWARE SYSTEM cannot contribute to a HAZARDOUS SITUATION; or

– the SOFTWARE SYSTEM can contribute to a HAZARDOUS SITUATION which does not result in unacceptable RISK after consideration of RISK CONTROL measures external to the SOFTWARE SYSTEM.

The SOFTWARE SYSTEM is software safety class B if:

– the SOFTWARE SYSTEM can contribute to a HAZARDOUS SITUATION which results in unacceptable RISK after consideration of RISK CONTROL measures external to the SOFTWARE SYSTEM and the resulting possible HARM is non-SERIOUS INJURY.

The SOFTWARE SYSTEM is software safety class C if:

– the SOFTWARE SYSTEM can contribute to a HAZARDOUS SITUATION which results in unacceptable RISK after consideration of RISK CONTROL measures external to the SOFTWARE SYSTEM and the resulting possible HARM is death or SERIOUS INJURY.

For a SOFTWARE SYSTEM initially classified as software safety class 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 safety classification to the SOFTWARE SYSTEM.

NOTE 1 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.

NOTE 2 See ISO 14971:2007 subclause 3.2, *Management Responsibilities*, for the definition of risk acceptability.

b) Not used.

c) The MANUFACTURER shall document the software safety class 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 safety classification of the original SOFTWARE ITEM (or SOFTWARE SYSTEM) unless the MANUFACTURER documents a rationale for classification into a different software safety class (software safety classes assigned according to 4.3 a) replacing "SOFTWARE SYSTEM" with "SOFTWARE ITEM"). Such a rationale shall explain how the new SOFTWARE ITEMS are segregated so that they may be classified separately.

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

f) For compliance with this standard, when applying this standard to a group of SOFTWARE ITEMS, the MANUFACTURER shall use the PROCESSES and TASKS which are required by the classification 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.

g) For each SOFTWARE SYSTEM, until a software safety class is assigned, Class C requirements shall apply.

NOTE In the clauses and subclauses that follow, the software safety classesfor which a specific requirement applies are identified following the requirement in the form [Class . . .].

4.4 * LEGACY SOFTWARE

4.4.1 General

As an alternative to applying Clauses 5 through 9 of this standard, compliance of LEGACY SOFTWARE may be demonstrated as indicated in 4.4.2 to 4.4.5.

4.4.2 RISK MANAGEMENT ACTIVITIES

In accordance with 4.2 of this standard, the MANUFACTURER shall:

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

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 MEDICAL DEVICE 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;
- definition of RISK CONTROL measures for each potential cause of the LEGACY SOFTWARE contributing to a HAZARDOUS SITUATION.

4.4.3 Gap analysis

Based on the software safety class of the LEGACY SOFTWARE (see 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 SYSTEM test records (see 5.7.5).

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

4.4.4 Gap closure activities

a) The MANUFACTURER shall establish and execute a plan to generate the identified DELIVERABLES. Where available, objective evidence may 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 Clause 6.

4.4.5 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 4.4.

NOTE Fulfilling 4.4 enables further use of LEGACY SOFTWARE in accordance with IEC 62304.

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 safety classifications 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 (see Note 4);

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 the MEDICAL DEVICE SOFTWARE, DELIVERABLES and ACTIVITIES at each stage of the life cycle.

[Class A, B, C]

NOTE 1 The SOFTWARE DEVELOPMENT LIFE CYCLE MODEL can identify different elements (PROCESSES, ACTIVITIES, TASKS and DELIVERABLES) for different SOFTWARE ITEMS according to the software safety classification of each SOFTWARE ITEM of the SOFTWARE SYSTEM.

NOTE 2 These ACTIVITIES and TASKS can overlap or interact and can be performed iteratively or recursively. It is not the intent to imply that a specific life cycle model should be used.

NOTE 3 Other PROCESSES are described in this standard separately from the development PROCESS. This does not imply that they must be implemented as separate ACTIVITIES and TASKS. The ACTIVITIES and TASKS of the other PROCESSES can be integrated into the development PROCESS.

NOTE 4 The software development plan can reference existing PROCESSES or define new ones.

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

5.1.2 Keep software development plan updated

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

5.1.3 Software development plan reference to SYSTEM design and development

a) As inputs for software development, SYSTEM requirements shall be referenced in the software development plan by the MANUFACTURER.

b) In the software development plan, 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).

[Class 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 device).

5.1.4 Software development standards, methods and tools planning

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

a) standards,

b) methods, and

c) tools

associated with the development of SOFTWARE ITEMS of class C. [Class 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. [Class 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.

[Class 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. [Class 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;

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

[Class 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; and

f) when the problem resolution PROCESS is to be used.

[Class 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 the MEDICAL DEVICE SOFTWARE, which could impact the MEDICAL DEVICE SOFTWARE. [Class 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. [Class B, C]

5.1.12 Identification and avoidance of common software defects

The MANUFACTURER shall include or reference in the software development plan a procedure for:

a) identifying categories of defects that may be introduced based on the selected programming technology that are relevant to their SOFTWARE SYSTEM; and

b) documenting evidence that demonstrates that these defects do not contribute to unacceptable RISK.

NOTE See Annex B of IEC TR 80002-1:2009 for examples of categories of defects or causes contributing to HAZARDOUS SITUATIONS.

[Class B, C]

5.2 * Software requirements analysis

5.2.1 Define and document software requirements from SYSTEM requirements

For each SOFTWARE SYSTEM of the MEDICAL DEVICE, the MANUFACTURER shall define and document SOFTWARE SYSTEM requirements from the SYSTEM level requirements. [Class 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 device).

5.2.2 Software requirements content

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

a) functional and capability requirements;

NOTE 1 Examples include:

– performance (e.g., purpose of software, timing requirements),

– 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 requirements;

NOTE 3 Examples include:

– those related to the compromise of sensitive information,

– authentication,

– authorization,

– audit trail, and

– communication integrity,

– system security/malware protection.

f) user interface requirements implemented by software;

NOTE 4 Examples include those related to:

– support for manual operations,

– human-equipment interactions,

– constraints on personnel, and

– areas needing concentrated human attention.

NOTE 5 Information regarding usability engineering requirements can be found in IEC 62366-1 [21] among others (e.g., IEC 60601-1-6 [3]).

g) data definition and database requirements;

NOTE 6 Examples include:

– form;

– fit;

– function.

h) installation and acceptance requirements of the delivered MEDICAL DEVICE 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 9 Examples include those related to:

– networked alarms, warnings, and operator messages;

– network protocols;

– handling of unavailability of network services.

k) user maintenance requirements; and

l) regulatory requirements.

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

[Class A, B, C]

NOTE 7 All of these requirements might not be available at the beginning of the software development.

NOTE 8 Among others, ISO/IEC 25010 [12] provides information on quality characteristics that may be useful in defining software requirements.

5.2.3 Include RISK CONTROL measures in software requirements

The MANUFACTURER shall include RISK CONTROL measures implemented in software in the requirements as appropriate to the MEDICAL DEVICE SOFTWARE. [Class B, C]

NOTE 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.

5.2.4 Re-EVALUATE MEDICAL DEVICE RISK ANALYSIS

The MANUFACTURER shall re-EVALUATE the MEDICAL DEVICE RISK ANALYSIS when software requirements are established and update it as appropriate. [Class 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. [Class 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.

[Class A, B, C]

NOTE This standard does not require the use of a formal specification language.

5.3 * Software ARCHITECTURAL design

5.3.1 Transform software requirements into an ARCHITECTURE

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

5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS

The MANUFACTURER shall develop and document an ARCHITECTURE for the interfaces between the SOFTWARE ITEMS and the components external to the SOFTWARE ITEMS (both software and hardware), and between the SOFTWARE ITEMS. [Class 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 intended use. [Class 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. [Class 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. [Class 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 B.4.3).

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 MEDICAL DEVICE ARCHITECTURE supports proper operation of any SOUP items.

[Class 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 subdvide the software until it is represented by SOFTWARE UNITS. [Class 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. [Class 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. [Class 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.

[Class C]

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

5.5 * SOFTWARE UNIT implementation

5.5.1 Implement each SOFTWARE UNIT

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

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

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. [Class B, C]

NOTE Examples of acceptance criteria are:

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

– is the software code free from contradiction 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.

[Class C]

5.5.5 SOFTWARE UNIT VERIFICATION

The MANUFACTURER shall perform the SOFTWARE UNIT VERIFICATION and document the results. [Class 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). [Class 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.

[Class B, C]

NOTE This VERIFICATION is only 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. [Class B, C]

5.6.4 Software integration testing content

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

[Class 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. [Class 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. [Class 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) identify the tester.

[Class 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;
- 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. [Class B, C]

NOTE See Clause 9.

5.7 * SOFTWARE SYSTEM testing

5.7.1 Establish tests for software requirements

a) 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. [Class 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.

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

5.7.2 Use software problem resolution PROCESS

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

5.7.3 Retest after changes

When changes are made 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 testing appropriate to demonstrate that unintended side effects have not been introduced; and
c) perform relevant RISK MANAGEMENT ACTIVITIES as defined in 7.4.

[Class A, B, C]

5.7.4 Evaluate SOFTWARE SYSTEM testing

The MANUFACTURER shall EVALUATE the appropriateness 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.

[Class 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) date tested; and

g) the identity of the person responsible for executing the test and recording the test results.

[Class A, B, C]

5.8 * Software RELEASE for utilization at a SYSTEM level

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. [Class A, B, C]

5.8.2 Document known residual ANOMALIES

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

5.8.4 Document released VERSIONS

The MANUFACTURER shall document the VERSION of the MEDICAL DEVICE SOFTWARE that is being released. [Class A, B, C]

5.8.5 Document how released software was created

The MANUFACTURER shall document the procedure and environment used to create the released software. [Class 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. [Class B, C]

NOTE See 5.1.3.b).

5.8.7 Archive software

The MANUFACTURER shall archive:

a) the MEDICAL DEVICE SOFTWARE and CONFIGURATION ITEMS; and

b) the documentation

for at least a period of time determined as the longer of: the life time of the MEDICAL DEVICE SOFTWARE as defined by the MANUFACTURER or a time specified by relevant regulatory requirements. [Class A, B, C]

5.8.8 Assure reliable delivery of released software

The MANUFACTURER shall establish procedures to ensure that the released MEDICAL DEVICE SOFTWARE can be reliably delivered to the point of use without corruption or unauthorised

change. These procedures shall address the production and handling of media containing the MEDICAL DEVICE SOFTWARE including as appropriate:

– replication,
– media labelling,
– packaging,
– protection,
– storage, and
– delivery.

[Class 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:
   – receiving,
   – documenting,
   – evaluating,
   – resolving and
   – tracking

   feedback arising after release of the MEDICAL DEVICE SOFTWARE;
b) criteria for determining whether feedback is considered to be a problem;
c) use of the software RISK MANAGEMENT PROCESS;
d) use of the software problem resolution PROCESS for analysing and resolving problems arising after release of the MEDICAL DEVICE SOFTWARE;
e) use of the software configuration management PROCESS (Clause 8) for managing modifications to the existing SOFTWARE SYSTEM; and
f) procedures to EVALUATE and implement:
   – upgrades,
   – bug fixes,
   – patches and
   – obsolescence

   of SOUP.

[Class 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 MEDICAL DEVICE SOFTWARE released for intended use.
[Class 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 released MEDICAL DEVICE SOFTWARE. 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. [Class A, B, C]

6.2.1.3 Evaluate PROBLEM REPORT'S affects on SAFETY

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

NOTE A problem could show that a SOFTWARE SYSTEM or SOFTWARE ITEM has not been placed in the correct software safety class. The problem resolution process can suggest changes of the software safety class. When the PROCESS has been completed, any change of safety class in the SOFTWARE SYSTEM or its SOFTWARE ITEMS should be 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 organization, MEDICAL DEVICE SOFTWARE released for intended use , and SYSTEMS with which it interfaces. [Class A, B, C]

6.2.4 CHANGE REQUEST approval

The MANUFACTURER shall EVALUATE and approve CHANGE REQUESTS which modify released MEDICAL DEVICE SOFTWARE. [Class A, B, C]

6.2.5 Communicate to users and regulators

The MANUFACTURER shall identify the approved CHANGE REQUESTS that affect released MEDICAL DEVICE SOFTWARE.

As required by local regulation, the MANUFACTURER shall inform users and regulators about:

a) any problem in released MEDICAL DEVICE SOFTWARE and the consequences of continued unchanged use; and

b) the nature of any available changes to released MEDICAL DEVICE SOFTWARE and how to obtain and install the changes.

[Class 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 repeated as a result of the modification. [Class 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. [Class 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 identified in the MEDICAL DEVICE RISK ANALYSIS ACTIVITY of ISO 14971 (see 4.2). [Class 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 identify potential causes of the SOFTWARE ITEM identified above contributing to a hazardous situation.

The MANUFACTURER shall consider potential causes including, 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) hardware failures or other software defects that could result in unpredictable software operation; and

e) reasonably foreseeable misuse.

[Class 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 MEDICAL DEVICE to determine if any of the known ANOMALIES result in a sequence of events that could result in a hazardous situation. [Class B, C]

7.1.4 Document potential causes

The MANUFACTURER shall document in the RISK MANAGEMENT FILE potential causes of the SOFTWARE ITEM contributing to a hazardous situation (see ISO 14971). [Class B, C]

7.2 RISK CONTROL measures

7.2.1 Define RISK CONTROL measures

For each case documented in the RISK MANAGEMENT FILE where a SOFTWARE ITEM could contribute to a HAZARDOUS SITUATION, the MANUFACTURER shall define and document RISK CONTROL measures in accordance with ISO 14971. [Class B, C]

NOTE The RISK CONTROL measures can be implemented in hardware, software, the working environment or user instruction.

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 requirements;

b) assign to each SOFTWARE ITEM that contributes to the implementation of a RISK CONTROL measure a software safety class based on the RISK that the RISK CONTROL measure is controlling (see 4.3 a)); and

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

[Class B, C]

NOTE This requirement provides additional detail for RISK CONTROL requirements of ISO 14971

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. [Class B, C]

7.3.2

Not used.

7.3.3 Document TRACEABILITY

The MANUFACTURER shall document TRACEABILITY of software HAZARDS as appropriate:

a) from 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.

[Class B, C]

NOTE See ISO 14971 – RISK MANAGEMENT report.

7.4 RISK MANAGEMENT of software changes

7.4.1 Analyse changes to MEDICAL DEVICE SOFTWARE with respect to SAFETY

The MANUFACTURER shall analyse changes to the MEDICAL DEVICE SOFTWARE (including SOUP) to determine whether:

a) additional potential causes are introduced contributing to a hazardous situation; and

b) additional software RISK CONTROL measures are required.

[Class 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 interfere with existing RISK CONTROL measures. [Class B, C]

7.4.3 Perform RISK MANAGEMENT ACTIVITIES based on analyses

The MANUFACTURER shall perform relevant RISK MANAGEMENT ACTIVITIES defined in 7.1, 7.2 and 7.3 based on these analyses. [Class 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. [Class A, B, C]

8.1.2 Identify SOUP

For each SOUP CONFIGURATION ITEM being used, including standard libraries, the MANUFACTURER shall document:

a) the title,

b) the MANUFACTURER, and

c) the unique SOUP designator

[Class 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. [Class A, B, C]

8.2 * Change control

8.2.1 Approve CHANGE REQUESTS

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

NOTE 1 The decision to approve a CHANGE REQUEST can be integral to the change control PROCESS or part of another PROCESS. This subclause only requires that approval of a change precede its implementation.

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).

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 safety classification of SOFTWARE SYSTEMS and SOFTWARE ITEMS. [Class A, B, C]

NOTE This subclause states how the change should be implemented to achieve adequate change control. It does not imply that the implementation is an integral part of the change control PROCESS. Implementation should use planned PROCESSES, see 5.1.1 e) 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. [Class A, B, C]

NOTE This subclause only requires that changes be VERIFIED. It does not imply that VERIFICATION is an integral part of the change control PROCESS. VERIFICATION should use planned PROCESSES, see 5.1.1 e) 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

[Class A, B, C]

8.3 * Configuration status accounting

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

9 * Software problem resolution PROCESS

9.1 Prepare PROBLEM REPORTS

The MANUFACTURER shall prepare a PROBLEM REPORT for each problem detected in the MEDICAL DEVICE SOFTWARE. PROBLEM REPORTS shall include a statement of criticality (for example, effect on performance, SAFETY, or SECURITY) as well as other information that may aid in the resolution of the problem (for example, devices affected, supported accessories affected).

[Class A, B, C]

NOTE Problems can be discovered before or after release, 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.

[Class A, B, C]

NOTE A problem does not have to be corrected for the MANUFACTURER to comply with the software problem resolution PROCESS, provided that the problem is not relevant to SAFETY.

9.3 Advise relevant parties

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

[Class 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.

9.4 Use change control process

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

9.6 Analyse problems for trends

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

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 in the appropriate MEDICAL DEVICE SOFTWARE and ACTIVITIES; and

d) additional problems have been introduced.

[Class 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:

a) test results;

b) ANOMALIES found;

c) the VERSION of software tested;

d) relevant hardware and software test configurations;

e) relevant test tools;

f) date tested; and

g) identification of the tester.

[Class A, B, C]

Annex A
(informative)

Rationale for the requirements of this standard

Rationale for the clauses of this standard is provided in this annex.

A.1 Rationale

The primary requirement of this standard is that a set of PROCESSES be followed in the development and maintenance of MEDICAL DEVICE 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 standard 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 standard requires the first category to be performed for all MEDICAL DEVICE SOFTWARE and the second category to be performed for selected SOFTWARE ITEMS.

A claim of compliance with this standard should therefore include a documented RISK ANALYSIS that identifies foreseeable sequences of events that include software and that can result in a HAZARDOUS SITUATION (see ISO 14971). HAZARDOUS SITUATIONS that can be indirectly caused by software (for example, by providing misleading information that could cause inappropriate treatment to be 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 "[Class A, B, C]", indicating that they are required irrespective of the classification of the software to which they apply.

ACTIVITIES are required for all classes A, B, and C for the following reasons:

– the ACTIVITY produces a plan relevant to RISK MANAGEMENT or software safety classification;

– the ACTIVITY produces an output that is an input to RISK MANAGEMENT or software safety classification;

– the ACTIVITY is a part of RISK MANAGEMENT or software safety classification;

– the ACTIVITY establishes an administration system, documentation or record-keeping system that supports RISK MANAGEMENT or software safety classification;

– the ACTIVITY normally takes place when the classification of the related software is unknown;

– the ACTIVITY can cause a change that could invalidate the current software safety classification of the associated software. This includes the discovery and analysis of safety related problems after release.

Other PROCESSES are required only for SOFTWARE SYSTEMS or SOFTWARE ITEMS classified in software safety classes B or C. ACTIVITIES required as parts of these PROCESSES are identified in the normative text as "[Class B, C]", or "[Class 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 classes 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 classes 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 class C for the following reasons:

– 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

Note that all PROCESSES and ACTIVITIES defined in this standard 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 class 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 HAZARD can be assured of SAFETY and effectiveness primarily through overall validation ACTIVITY during the design of a MEDICAL DEVICE (which is outside the scope of this standard) and through some simple software life cycle controls.

A.2 Summary of requirements by class

Table A.1 summarizes which software safety classes are assigned to each requirement. This table is informative and only provided for convenience. The normative section identifies the software safety classes for each requirement.

Table A.1 – Summary of requirements by software safety class

Clauses and subclauses		Class A	Class B	Class 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, 5.1.12		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

Annex B
(informative)

Guidance on the provisions of this standard

B.1 Scope

B.1.1 Purpose

The purpose of this standard is to provide a development PROCESS that will consistently produce high quality, safe MEDICAL DEVICE SOFTWARE. To accomplish this, the standard 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 MEDICAL DEVICE SOFTWARE.

This annex provides guidance for the application of the requirements of this standard. It does not add to, or otherwise change, the requirements of this standard. This annex can be used to better understand the requirements of this standard.

Note that in this standard, 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 and VERIFICATION, software integration and integration testing, SOFTWARE SYSTEM testing, and software release. The TASKS within these ACTIVITIES are the individual requirements.

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

Because of such logical dependencies between processes, it is easiest to describe the processes in this standard 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 12207 [9] include (see also Table B.1):

– Waterfall. The “once-through" strategy, also called “waterfall”, consists of performing the development PROCESS a single time. Simplistically: determine customer needs, define requirements, design the SYSTEM, implement the system, test, fix and deliver.
– Incremental: The “incremental” strategy determines customer needs and defines the SYSTEM requirements, then performs the rest of the development in a sequence of builds. The first build incorporates part of the planned capabilities, the next build adds more capabilities, and so on, until the SYSTEM is complete.
– Evolutionary: The “evolutionary” strategy also develops a SYSTEM in builds but differs from the incremental strategy in acknowledging that the user need is not fully understood and all requirements cannot be defined up front. In this strategy, customer needs and SYSTEM requirements are partially defined up front, then are refined in each succeeding build.

Table B.1 – Development (model) strategies as defined in ISO/IEC 12207

Development Strategy	Define all requirements first?	Multiple development cycles?	Distribute interim software?
Waterfall (Once-through)	yes	no	no
Incremental (Preplanned product improvement)	yes	yes	maybe
Evolutionary	no	yes	yes

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 standard;
– all PROCESS outputs should be available when needed as input to further work on the software.
– before any MEDICAL DEVICE SOFTWARE is released, all PROCESS outputs should be consistent with each other and all dependencies between PROCESS outputs explicitly or implicitly required by this standard should be observed.

B.1.2 Field of application

This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE as well as the development and maintenance of a MEDICAL DEVICE that includes SOUP.

The use of this standard requires the MANUFACTURER to perform MEDICAL DEVICE RISK MANAGEMENT that is compliant with ISO 14971. Therefore, when the MEDICAL DEVICE SYSTEM ARCHITECTURE includes an acquired component (this could be a purchased component or a component of unknown provenance), such as a printer/plotter that includes SOUP, the acquired component becomes the responsibility of the MANUFACTURER and must be included in the RISK MANAGEMENT of the MEDICAL DEVICE. It is assumed that through proper performance of MEDICAL DEVICE RISK MANAGEMENT, the MANUFACTURER would understand the component and recognize that it includes SOUP. The MANUFACTURER using this standard would invoke the software RISK MANAGEMENT PROCESS as part of the overall MEDICAL DEVICE RISK MANAGEMENT PROCESS.

The maintenance of released MEDICAL DEVICE SOFTWARE applies to the post-production experience with the MEDICAL DEVICE 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 released MEDICAL DEVICE SOFTWARE. For example, this includes problem rectification, regulatory reporting, re-validation and preventive action. See ISO/IEC 14764 [10].

B.2 Normative references

ISO/IEC 90003 [15] provides guidance for applying a quality management system to software development. This guidance is not required by this standard but is highly recommended.

B.3 Terms and definitions

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

This standard chose to use three terms to describe the decomposition of a SOFTWARE SYSTEM (top level). The SOFTWARE SYSTEM can be a subsystem of the MEDICAL DEVICE (see IEC 60601-1-4 [2]) or a MEDICAL DEVICE in its own right, which then becomes a software 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 definition and granularity of the SOFTWARE ITEMS and SOFTWARE UNITS. Leaving these terms vague allows one to apply them to the many different development methods and types of software used in MEDICAL DEVICES.

B.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 MEDICAL DEVICE SOFTWARE:

– RISK MANAGEMENT;
– quality management;
– software engineering.

For the development and maintenance of safe MEDICAL DEVICE 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 MEDICAL DEVICE MANUFACTURER to follow a clearly structured and consistently repeatable decision-making PROCESS to promote SAFETY for MEDICAL DEVICE SOFTWARE.

B.4.1 Quality management system

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 standard on software engineering, these PROCESSES have been omitted from this standard. These PROCESSES are covered by a quality management system. ISO 13485 [8] 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 requirements does not automatically constitute conformity with national or regional regulatory requirements. It is the MANUFACTURER'S responsibility to identify and establish compliance with relevant regulatory requirements.

B.4.2 RISK MANAGEMENT

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

Rather than trying to define an appropriate RISK MANAGEMENT PROCESS in this software engineering standard, it is required that the MANUFACTURER apply a RISK MANAGEMENT PROCESS

that is compliant with ISO 14971, which deals explicitly with RISK MANAGEMENT for MEDICAL DEVICES. 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.

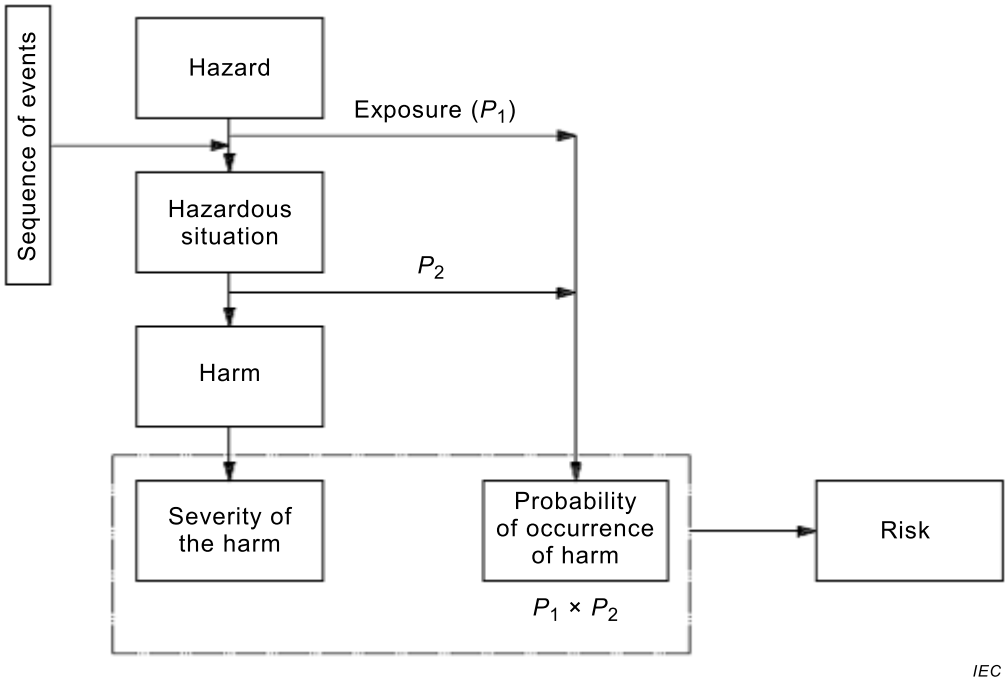
B.4.3 Software safety classification

The RISK associated with software as a part of a MEDICAL DEVICE, as an accessory to a MEDICAL DEVICE, or as a MEDICAL DEVICE in its own right, is used as the input to a software safety classification scheme, which then determines the PROCESSES to be used during the development and maintenance of software.

RISK is considered to be a combination of the severity of HARM and the probability of its occurrence. However, no consensus exists for a method of quantitatively estimating the probability of occurrence of a software failure. When software is present in a sequence or combination of events leading to a HAZARDOUS SITUATION, the probability of the software failure occurring cannot be considered in estimating the RISK for the HAZARDOUS SITUATION. In such cases, considering a worst case probability is appropriate, and the probability for the software failure occurring should be set to 1. When it is possible to estimate the probability for the remaining events in the sequence (as it may be if they are not software) that probability can be used for the probability of the HAZARDOUS SITUATION occurring (P_1 in Figure B.2).

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. Subjective rankings of probability can also be assigned based on clinical knowledge to distinguish failures that a clinician would be likely to detect from those that would not be detected and would be more likely to cause HARM.

Estimates of probability of a HAZARDOUS SITUATION leading to HARM (P_2 in Figure B.2) generally require clinical knowledge to distinguish between HAZARDOUS SITUATIONS where clinical practice would be likely to prevent HARM, and HAZARDOUS SITUATIONS that would be more likely to cause HARM.



NOTE P_1 is the probability of a hazardous situation occurring

P_2 is the probability of a hazardous situation leading to harm

Figure B.2 – Pictorial representation of the relationship of HAZARD, sequence of events, HAZARDOUS SITUATION, and HARM – from ISO 14971:2007 Annex E

If a SOFTWARE SYSTEM is decomposed into SOFTWARE ITEMS, then each SOFTWARE ITEM can have its own software safety classification.

It is only possible to determine the RISK associated with failure of a SOFTWARE ITEM:

– if a SYSTEM ARCHITECTURE and a software ARCHITECTURE define the role of the SOFTWARE ITEM in terms of its purpose and its interfaces with other software and hardware items;

– if changes to the SYSTEM are controlled;

– after RISK ANALYSIS has been done on the ARCHITECTURE and RISK CONTROL measures specified.

This standard requires the minimum number of ACTIVITIES that will achieve the above conditions for all classes of software.

The end of the software ARCHITECTURE ACTIVITY is the earliest point in the development when the full set of SOFTWARE ITEMS is defined and the RISK MANAGEMENT ACTIVITY has identified how the SOFTWARE ITEMS relate to SAFETY. This is therefore the earliest point at which SOFTWARE ITEMS can be classified definitively according to their SAFETY role.

This point corresponds to the point where RISK CONTROL is begun in ISO 14971.

Before this point, the RISK MANAGEMENT PROCESS identifies ARCHITECTURAL RISK CONTROL measures, for example adding protective subsystems, or reducing the opportunities for software failures to cause HARM. After this point, the RISK MANAGEMENT PROCESS uses PROCESSES aimed at reducing the probability of failure of SOFTWARE ITEMS. In other words, the classification of a SOFTWARE ITEM specifies PROCESS-based RISK CONTROL measures to be applied to that item.

It is expected that MANUFACTURERS will find it useful to classify software before this point, for example to focus attention on areas to be investigated, but such classification should be regarded as preliminary and should not be used to justify the omission of PROCESSES.

The software safety classification scheme is not intended to align with the RISK classifications of ISO 14971. Whereas the ISO 14971 scheme classifies RISK according to their severity and likelihood, the software safety classification scheme classifies SOFTWARE SYSTEMS and SOFTWARE ITEMS according to the PROCESSES to be applied in their development and maintenance.

As the design evolves, new RISKS might become evident. Therefore, RISK MANAGEMENT should be applied as an integral part of the development PROCESS. This permits the development of an ARCHITECTURAL design that identifies a complete set of SOFTWARE ITEMS, including those that are required to function correctly to assure safe operation and those that prevent faults from causing HARM.

The software ARCHITECTURE should promote segregation of software items that are required for safe operation and should describe the methods used to ensure effective segregation of those SOFTWARE ITEMS. Segregation is not restricted to physical (processor or memory partition) separation but includes any mechanism that prevents one SOFTWARE ITEM from negatively affecting another. The adequacy of a segregation is determined based on the RISKS involved and the rationale which is required to be documented.

As stated in B.3, this standard chooses to use three terms to describe the decomposition of a SOFTWARE SYSTEM (top level).

Figure B.1 illustrates the possible partitioning for SOFTWARE ITEMS within a SOFTWARE SYSTEM and how the software safety classes would be applied to the group of SOFTWARE ITEMS in the decomposition.

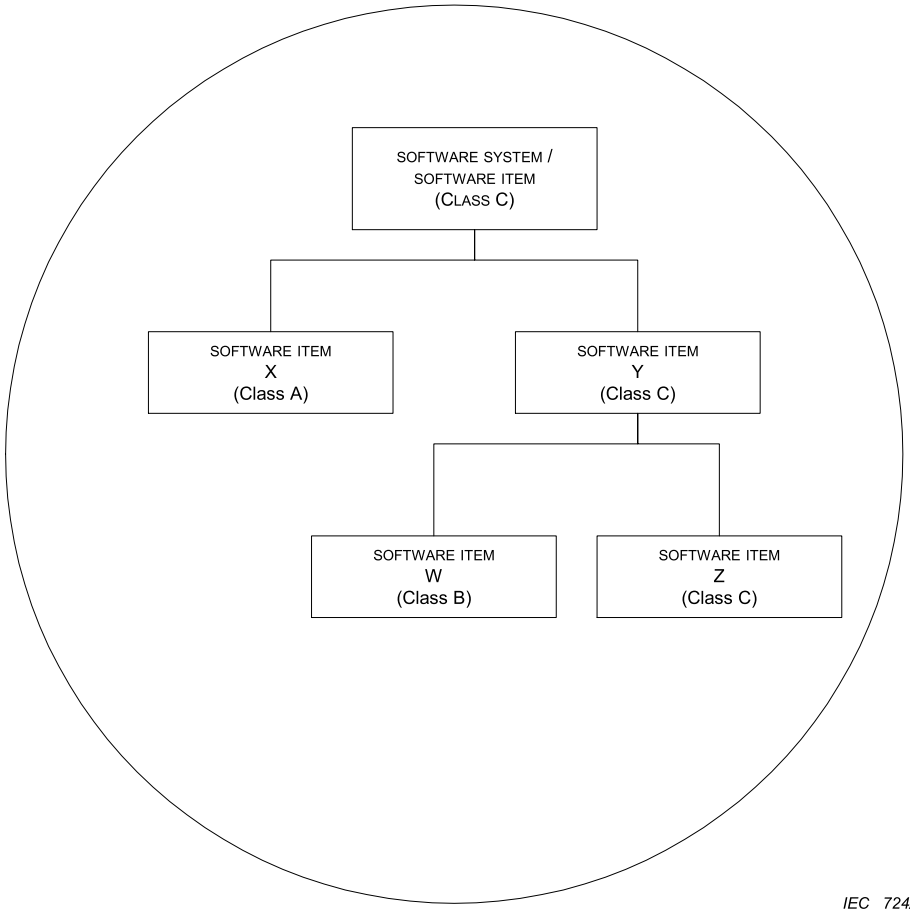


Figure B.1 – Example of partitioning of SOFTWARE ITEMS

For this example, the MANUFACTURER knows, due to the type of MEDICAL DEVICE SOFTWARE being developed, that the preliminary software safety classification for the SOFTWARE SYSTEM is software safety class C. During software ARCHITECTURE design the MANUFACTURER has decided to partition the SYSTEM, as shown, with 3 SOFTWARE ITEMS – X, W and Z. The MANUFACTURER is able to segregate all SOFTWARE SYSTEM contributions to HAZARDOUS SITUATIONS which could result in death or SERIOUS INJURY to SOFTWARE ITEM Z and all remaining SOFTWARE SYSTEM contributions to HAZARDOUS SITUATIONS which could result in a non-SERIOUS INJURY to SOFTWARE ITEM W. SOFTWARE ITEm W is classified as software safety class B and SOFTWARE ITEM Z is at software safety class C. SOFTWARE ITEM Y therefore must be classified as Class C, per 4.3 d). The SOFTWARE SYSTEM is also at a software safety class C per this requirement. SOFTWARE ITEM X has been classified at a software safety class 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 between SOFTWARE ITEMS X and Y, then SOFTWARE ITEM X must be classified in software safety class C.

B.4.4 LEGACY SOFTWARE

Subclause 4.4 establishes a process for application of this standard to LEGACY SOFTWARE. Some geographies may require the MANUFACTURER to show conformity to the standard to obtain regulatory approval of the MEDICAL DEVICE SOFTWARE, even if that software was designed prior to the existence of the current version of the standard (LEGACY SOFTWARE). In this case, the requirements in 4.4 provide a method for the the MANUFACTURER to demonstrate compliance of LEGACY SOFTWARE to the standard.

A MANUFACTURER may determine that retrospective documentation of an already finished development-lifecycle 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 standard 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, and
– a MANUFACTURER should utilize resources as effectively as possible to effect a reduction of RISK.

In addition to a plan identifying the subset of ACTIVITIES to execute, the process also results in objective evidence supporting 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 MEDICAL DEVICE HAZARDS associated with the LEGACY SOFTWARE.

Subclause 4.4 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 may 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 may 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 should be assumed to be 1.

The MANUFACTURER determination of how the LEGACY SOFTWARE will be used in the overall MEDICAL DEVICE 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 use of 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 must take into account the modified set of HAZARDOUS SITUATIONS which can arise due to failures of the LEGACY SOFTWARE.
– A reused LEGACY SOFTWARE may be used for similar intended use but integrated into a new SOFTWARE SYSTEM. In this case the RISK assessment should take into account modification of architectural RISK CONTROL measures according to 5.3.

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 may be invalidated by the changes.

Changes to the LEGACY SOFTWARE should be performed according to Clauses 4 to 9 of this standard, including assessment of impact to RISK CONTROL measures according to 7.4. In the case of LEGACY SOFTWARE, existing RISK CONTROL measures may not be 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 4.4, 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 lifecycle model (as appropriate);

d) EVALUATION of the adequacy of existing RISK MANAGEMENT documentation, taking ISO 14971 into account.

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 may 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 SOFTARE reuse.

The rationale makes a positive case for the safe and reliable performace 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.

After LEGACY SOFTWARE has been re-used according to 4.4, those parts of the LEGACY SOFTWARE for which gaps in DELIVERABLES remain, continue to be LEGACY SOFTWARE and may be considered for further re-use again according to 4.4. When gaps in deliverables are closed by changing the LEGACY SOFTWARE, the changes should be performed according to Clauses 4 to 9 of this standard.

B.5 Software development PROCESS

B.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 the MEDICAL DEVICE 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 MEDICAL DEVICE 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 MEDICAL DEVICE 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 MEDICAL DEVICE 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 rigor 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 safety classification 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 rigor needed for the SYSTEM or items in the SYSTEM to enable proper control over the development PROCESS.

B.5.2 Software requirements analysis

This ACTIVITY requires the MANUFACTURER to establish and verify the software requirements for the MEDICAL DEVICE SOFTWARE. Establishing verifiable requirements is essential for determining what is to be built, for determining that the MEDICAL DEVICE SOFTWARE exhibits acceptable behaviour, and for demonstrating that the completed MEDICAL DEVICE 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 MEDICAL DEVICE 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 with 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 standard. However, the decision to implement MEDICAL DEVICE 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 analyzing the requirements allocated to software by the SYSTEM requirements PROCESS and deriving a comprehensive set of software requirements that reflect the allocated requirements.

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 standard 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.

B.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 behavior should be described in the software ARCHITECTURE. This description is especially important for behaviour that can affect components of the MEDICAL DEVICE that are outside the software (see 5.3.5 and B.4.3). 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 safety classification 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 class. The software configuration management PROCESS (Clause 8) is used to ensure that all necessary rework is identified and completed.

B.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 standard 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 must also be 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 must also be concerned with the architecture of the MEDICAL DEVICE 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 safety class. However, the decision to do this is in itself part of the RISK MANAGEMENT PROCESS, and is documented in the RISK MANAGEMENT FILE.

Because implementation depends on detailed design, it is necessary to verify the detailed design before the ACTIVITY is complete. 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 are to 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.

B.5.5 SOFTWARE UNIT implementation and verification

This ACTIVITY requires the MANUFACTURER to write and verify the code for the SOFTWARE UNITS. The detailed design is to 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 MEDICAL DEVICE SOFTWARE will not perform as intended.

B.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 MEDICAL DEVICE hardware.

The rigor 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*, *structural*, *clear box* and *open box testing*, is a testing technique where explicit knowledge of the internal workings of the SOFTWARE ITEM being tested are 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, functional, opaque-box, 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 MEDICAL DEVICE 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 MEDICAL DEVICE.

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. These integrated SOFTWARE ITEMS must function properly for the entire MEDICAL DEVICE SOFTWARE to function correctly and safely.

B.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 previous section) 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 MEDICAL DEVICE.

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.6.3).

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 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.

B.5.8 Software release

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

The MANUFACTURER should be able to show that the software that was developed using the development PROCESS is the software that is being released. 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.

B.6 Software maintenance PROCESS

B.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

the MEDICAL DEVICE SOFTWARE. This PROCESS is activated when the MEDICAL DEVICE 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 released MEDICAL DEVICE SOFTWARE while preserving its integrity. This PROCESS includes migration of the MEDICAL DEVICE SOFTWARE to environments or platforms for which it was not originally released. The ACTIVITIES provided in this clause are specific to the maintenance PROCESS; however, the maintenance PROCESS might use other PROCESSES in this standard.

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

B.6.2 Problem and modification analysis

This ACTIVITY requires the MANUFACTURER to analyze 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 mitigate a RISK in software that previously did not cause a HAZARDOUS SITUATION or mitigate RISKS. The software safety classification of a SOFTWARE ITEM might have changed if the software modification now can cause a HAZARD 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 the MEDICAL DEVICE SOFTWARE. As part of a MEDICAL DEVICE, the software maintenance PROCESS needs to ensure that:

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

– MEDICAL DEVICE 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;

– the MANUFACTURER considers what other MEDICAL DEVICE 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;
- 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.

B.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 MEDICAL DEVICE SOFTWARE. Unless the MEDICAL DEVICE SOFTWARE is treated as a new development, analysis of the effect of a modification on the entire MEDICAL DEVICE SOFTWARE is necessary. Regression analysis and testing are employed to provide assurance that a change has not created problems elsewhere in the MEDICAL DEVICE 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 must be made that justifies the amount of REGRESSION TESTING that will be performed to ensure that the portions of the MEDICAL DEVICE SOFTWARE not being modified still perform as they did before the modification was made.

B.7 Software RISK MANAGEMENT PROCESS

Software RISK MANAGEMENT is a part of overall MEDICAL DEVICE RISK MANAGEMENT and cannot be adequately addressed in isolation. This standard requires the use of a RISK MANAGEMENT PROCESS that is compliant with ISO 14971. RISK MANAGEMENT as defined in ISO 14971 deals specifically with a framework for effective management of the RISKS associated with the use of MEDICAL DEVICES. One portion of ISO 14971 pertains to control of identified RISKS associated with each HAZARD identified during the RISK ANALYSIS. The software RISK MANAGEMENT PROCESS in this standard is intended to provide additional requirements for RISK CONTROL for software, including software that has been identified during the RISK ANALYSIS as potentially contributing to a hazardous situation, or software that is used to control MEDICAL DEVICE RISKS. The software RISK MANAGEMENT PROCESS is included in this standard for two reasons.

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

b) the general RISK MANAGEMENT standard, ISO 14971, provided as a normative reference in this standard, does not specifically address the RISK CONTROL of software and the placement of RISK CONTROL in the software development life cycle.

Software RISK MANAGEMENT is a part of overall MEDICAL DEVICE 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 MEDICAL DEVICE RISK MANAGEMENT ACTIVITIES and documentation as long as all requirements in this standard are met.

B.7.1 Analysis of software contributing to hazardous situations

It is expected that the device HAZARD 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 device 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 components, and the practicality of the RISK CONTROL measures assigned to software functions can be EVALUATED. At that time the device HAZARD analysis should be revised to include:

- revised hazardous situations;
- revised RISK CONTROL measures and software requirements;
- new hazardous situations arising from software, for example hazardous situations related to human factors.

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

B.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.

B.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 the MEDICAL DEVICE SOFTWARE.

B.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.

B.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.

B.9 Software problem resolution PROCESS

The software problem resolution PROCESS is a PROCESS for analyzing 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 analyzed 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 12207 [9] and IEC 60601-1-4 [2], Amendment 1. We have chosen to call it “software problem resolution” in this standard.

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 analyzed and EVALUATED for possible relevance to SAFETY (as specified in ISO 14971).

Software development plan(s) or procedures, as required in 5.1, are to 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.

Annex C
(informative)

Relationship to other standards

C.1 General

This standard applies to the development and maintenance of MEDICAL DEVICE SOFTWARE. The software is considered a subsystem of the MEDICAL DEVICE or is itself a MEDICAL DEVICE. This standard is to be used together with other appropriate standards when developing a MEDICAL DEVICE.

MEDICAL DEVICE management standards such as ISO 13485 [8] (see C.2 and Annex D) and ISO 14971 (see Annex C.3) provide a management environment that lays a foundation for an organization to develop products. Safety standards such as IEC 60601-1 [1] (see Annex C.4) and IEC 61010-1 [5] (see Annex C.5) give specific direction for creating safe MEDICAL DEVICES. When software is a part of these MEDICAL DEVICES, IEC 62304 provides more detailed direction on what is required to develop and maintain safe MEDICAL DEVICE SOFTWARE. Many other standards such as ISO/IEC 12207 [9] (see Annex C.6), IEC 61508-3 [4] (see Annex C.7) and ISO/IEC 90003 [15] can be looked to as a source of methods, tools and techniques that can be used to implement the requirements in IEC 62304. Figure C.1 shows the relationship of these standards.

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 standard.

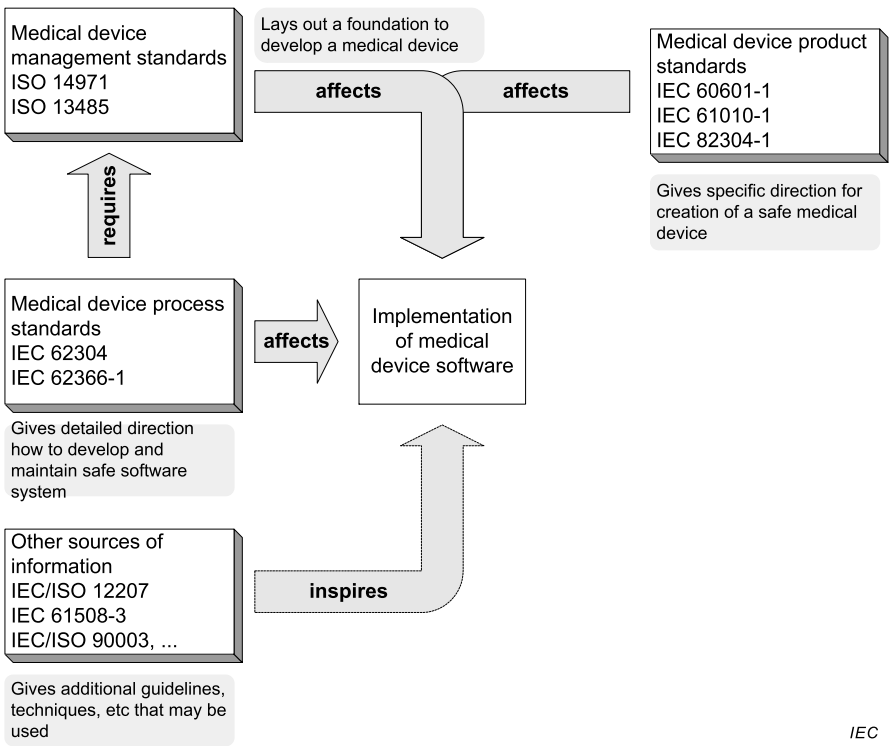


Figure C.1 – Relationship of key MEDICAL DEVICE standards to IEC 62304

C.2 Relationship to ISO 13485

This standard requires that the MANUFACTURER employs a quality management system. When a MANUFACTURER uses ISO 13485 [8], the requirements of ISO 62304 directly relate to some of the requirements of ISO 13485 as shown in Table C.1.

Table C.1 – Relationship to ISO 13485:2003

IEC 62304 clause	Related clause of ISO 13485:2003
5.1 Software development planning	7.3.1 Design and development planning
5.2 Software requirements analysis	7.3.2 Design and development inputs
5.3 Software ARCHITECTURAL design	
5.4 Software detailed design	
5.5 SOFTWARE UNIT implementation and verification	
5.6 Software integration and integration testing	
5.7 SOFTWARE SYSTEM testing	7.3.3 Design and development outputs 7.3.4 Design and development review
5.8 Software release	7.3.5 Design and development verification 7.3.6 Design and development validation
6.1 Establish software maintenance plan	7.3.7 Control of design and development changes
6.2 Problem and modification analysis	
6.3 Modification implementation	7.3.5 Design and development verification 7.3.6 Design and development validation
7.1 Analysis of software contributing to hazardous situations	
7.2 RISK CONTROL measures	
7.3 VERIFICATION of RISK CONTROL measures	
7.4 RISK MANAGEMENT of software changes	
8.1 Configuration identification	7.5.3 Identification and TRACEABILITY
8.2 Change control	7.5.3 Identification and TRACEABILITY
8.3 Configuration status accounting	
9 Software problem resolution PROCESS	

C.3 Relationship to ISO 14971

Table C.2 shows the areas where IEC 62304 amplifies requirements for the RISK MANAGEMENT PROCESS required by ISO 14971.

Table C.2 – Relationship to ISO 14971:2007

ISO 14971:2007 clause	Related clause of IEC 62304
4.1 RISK ANALYSIS process	
4.2 Intended use and identification of characteristics related to the SAFETY of the MEDICAL DEVICE	
4.3 Identification of HAZARDS	7.1 Analysis of software contributing to HAZARDOUS SITUATIONS
4.4 Estimation of the RISK(S) for each HAZARDOUS SITUATION	4.3 Software safety classification
5 RISK evaluation	
6.1 RISK reduction	
6.2 RISK CONTROL option analysis	7.2.1 Define RISK CONTROL measures
6.3 Implementation of RISK CONTROL measures	7.2.2 RISK CONTROL measures implemented in software 7.3.1 Verify RISK CONTROL measures
6.4 RESIDUAL RISK evaluation	
6.5 RISK/benefit analysis	
6.6 RISKS arising from RISK CONTROL MEASURES	7.3.2 Document any new sequences of events
6.7 Completeness of RISK CONTROL	
7 Evaluation of overall RESIDUAL RISK acceptability	
8 RISK MANAGEMENT report	7.3.3 Document TRACEABILITY
9 Production and post-production information	7.4 RISK MANAGEMENT of software changes

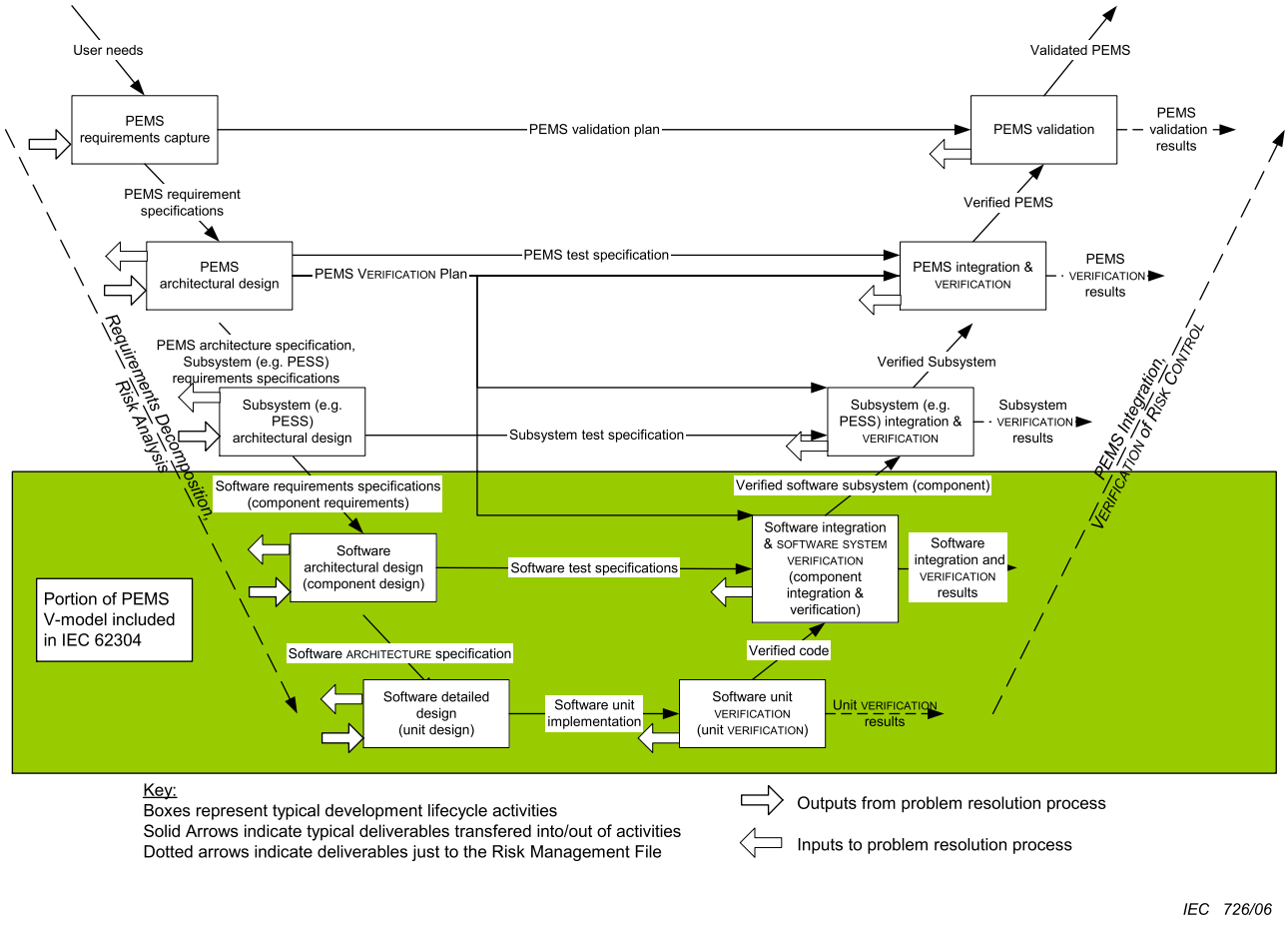
C.4 Relationship to PEMS requirements of IEC 60601-1:2005 + IEC 606011:2005/AMD1:2012

C.4.1 General

Requirements for software are a subset of the requirements for a programmable electrical medical system (PEMS). This standard identifies requirements for software which are in addition to, but not incompatible with, the requirements of IEC 60601-1:2005 + IEC 60601-1:2005 /AMD1:2012 [1] for PEMS. Because PEMS include elements that are not software, not all of the requirements of IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 for PEMS are addressed in this standard. With the publication of IEC 60601-1:2005 + IEC 60601-1:2005 /AMD1:2012, IEC 62304 is now a normative reference of IEC 60601-1 and compliance with Clause 14 of IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 (and thus compliance with the standard) requires compliance with parts of IEC 62304 (not with the whole of IEC 62304 because IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 does not require compliance with post-production and maintenance requirements of IEC 62304). Finally, it is important to remember that IEC 60601-1:2005 + IEC 60601-1:2005/AMD1:2012 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 software standard 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.



IEC 726/06

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

C.4.3 Development PROCESS

Compliance with the software development PROCESS of this standard (Clause 5) requires that a software development plan be specified and then followed; it does not require that any particular life cycle model is 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 for development life cycle, requirement specification, ARCHITECTURE, design and implementation, and VERIFICATION. The requirements in this standard provide greater detail about software development than those in IEC 60601-1.

C.4.4 Maintenance PROCESS

Compliance with the software maintenance PROCESS of this standard (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 standard for software maintenance provide greater detail about what must be done for software maintenance than the requirements for PEMS modification in IEC 60601-1.

C.4.5 Other PROCESSES

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

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

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

The software configuration management PROCESS in this standard 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 + IEC 606011:2005 /AMD1:2012

Table C.3 shows the PEMS requirements of IEC 60601-1 and the corresponding requirements in this standard.

Table C.3 – Relationship to IEC 60601-1

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 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 subclause 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.	**4.3 Software safety classification** The PEMS requirements of IEC 60601-1 would only apply to software safety classes B and C. This standard includes some requirements for software safety class A. The software development PROCESS required for compliance with IEC 60601-1 does not include the post production monitoring and maintenance required by Clause 6 of IEC 62304:2006.
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 [8] requires control of the documents.
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 software validation plan. The PEMS validation plan is at the SYSTEM level and thus is outside the scope of this software standard. This standard 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 must be planned.
Each activity shall be defined including its inputs and outputs.	**5.1.1 Software development plan** ACTIVITIES are defined in this standard. Documentation to be produced is defined in each ACTIVITY.
Each milestone shall identify the RISK MANAGEMENT ACTIVITIES that must 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 standard 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**

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
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 must 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.	**7.1 Analysis of software contributing to HAZARDOUS SITUATIONS** This standard does not mention network/data coupling specifically
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 standard 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 standard 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 standard 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	Requirements of IEC 62304 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: a) allocation of RISK CONTROL measures to subsystems and components of the PEMS; b) failure modes of components and their effects; c) common cause failures; d) systemic failures; e) test interval duration and diagnostic coverage; f) maintainability; g) protection from reasonably foreseeable misuse; h) the IT-NETWORK specification, if applicable.	This is not included in this standard.
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 standard does not require a test specification for detailed design.
Descriptive data regarding the design environment shall be included in the documentation.	**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

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
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 on 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 standard. 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 standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
Methods used for PEMS VALIDATION shall be documented	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
The PEMS VALIDATION shall be performed according to the PEMS VALIDATION plan. The results of the PEMS VALIDATION activities shall be documented.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
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 standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
No member of a design team shall be responsible for the PEMS VALIDATION of their own design.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
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 standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
A reference to the methods and results of the PEMS VALIDATION shall be included in the RISK MANAGEMENT FILE.	This standard does not cover software validation. PEMS validation is a SYSTEM level ACTIVITY and is outside the scope of this standard.
14.12 Modification If any or all of a design results from a modification of an earlier design then either all of this clause 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 standard 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 subclause 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	Requirements of IEC 62304 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'S 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 and IEC 80001-1:2010). f) list 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. 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;	Requirements for incorporation into an IT-network are not included in this standard.

PEMS requirements from IEC 60601-1:2005	Requirements of IEC 62304 relating to the software subsystem of a PEMS
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.	

C.4.7 Relationship to requirements in IEC 60601-1-4

IEC 60601-1-4 has been withdrawn.

C.5 Relationship to IEC 61010-1

The scope of IEC 61010-1 [5] 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 must be 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 must be taken into account.

IEC 61010-1:2010 has a general requirement for risk assessment in Clause 17, which is more streamlined than the full risk management requirements of ISO 14971. Applying IEC 61010-1 Clause 17 alone does not meet the required criteria for risk management of IEC 62304, which is based on full ISO 14971 risk management requirements. With this in mind, it is expected by this standard 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. Compliance with Clause 17 of IEC 61010-1 will be achieved, as detailed in the Note to Clause 17 of IEC 61010-1:

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 IEC 62304 with IEC 61010-1, Clause 17:

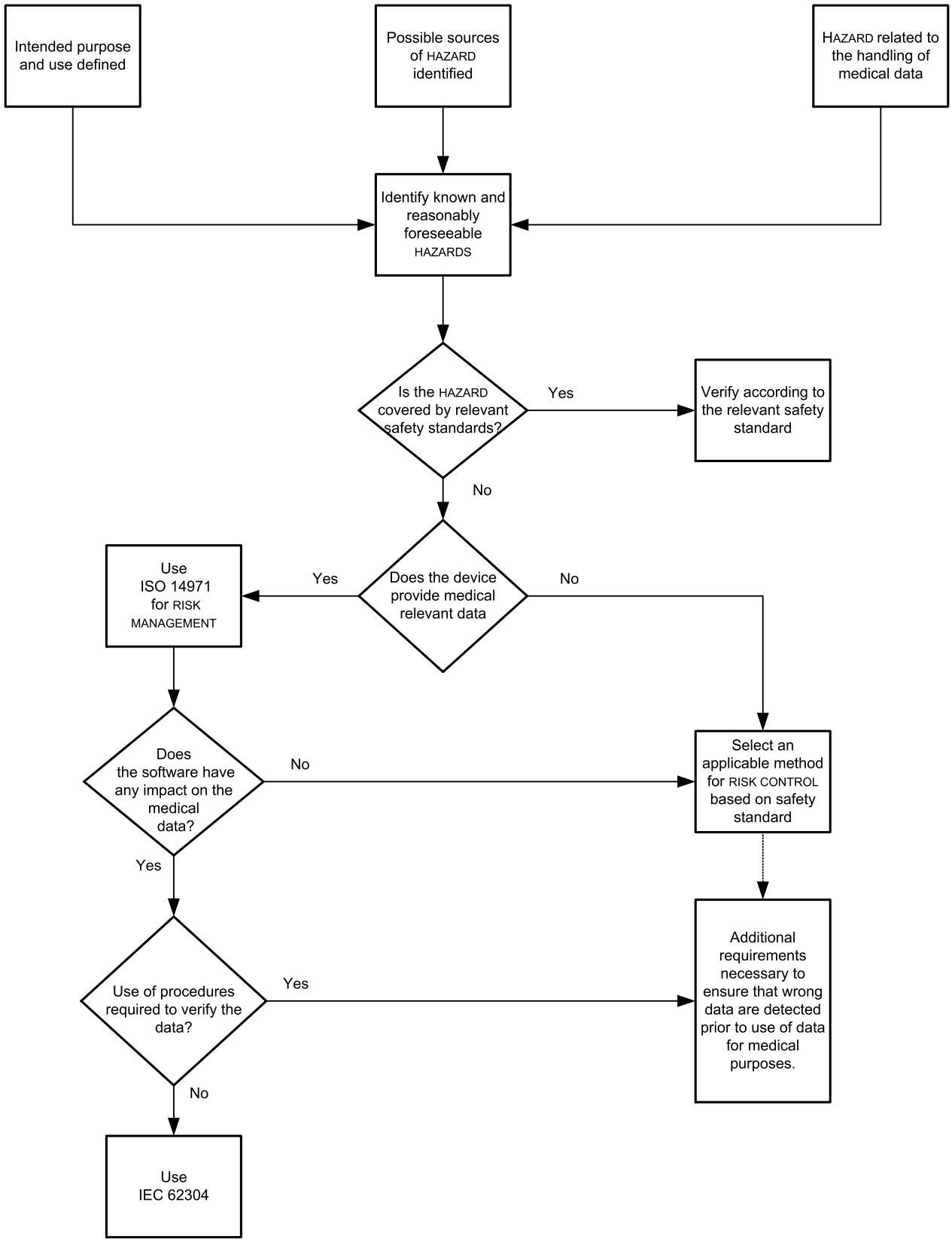


Figure C.3 – Application of IEC 62304 with IEC 61010-1

C.6 Relationship to ISO/IEC 12207

This standard has been derived from the approach and concepts of ISO/IEC 12207 [9], which defines requirements for software life cycle PROCESSES in general, i.e. not restricted to MEDICAL DEVICES.

This standard differs from ISO/IEC 12207 mainly with respect to the following. It:

- 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 12207 does.

Most of these changes were driven by the desire to tailor the standard 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 12207); and
- lowering the level of abstraction to make it easier to use.

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

Table C.5, which was prepared by ISO/IEC JTC1/SC7, shows the relationship between IEC 62304 and ISO/IEC 12207.

Table C.5 – Relationship to ISO/IEC 12207:2008

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
5 Software development PROCESS			
5.1 Software development planning			
	5.1.1 Software development plan	7.1.1 Software Implementation	7.1.1.3.1 Software implementation strategy 7.1.1.3.1.1 7.1.1.3.1.3 7.1.1.3.1.4 6.3.1.3.2 Project planning 6.3.1.3.2.1
	5.1.2 Keep software development plan updated	6.3.2 Project Assesment and Control	6.3.2.3.2 Project control 6.3.2.3.2.1
	5.1.3 Software development plan reference to SYSTEM design and development	6.4.3 System Architectural Design 6.4.5 System Integration 7.2.5 Software Validation Process	6.4.3.3.1 Establishing architecture 6.4.3.3.1.1 6.4.5.3.1 Integration 6.4.5.3.1.1 7.2.5.3.1 Process implementation 7.2.5.3.1.4
	5.1.4 Software development standards, methods and tools planning	7.1.1 Software Implementation	7.1.1.3.1 Software implementation strategy 7.1.1.3.1.3
	5.1.5 Software integration and integration testing planning	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.1
	5.1.6 Software VERIFICATION planning	7.2.4 Software Verification 7.1.5 Software Construction 7.1.6 Software Integration 7.1.7 Software Qualification Testing	7.2.4.3.1 Process implementation 7.2.4.3.1.4 7.2.4.3.1.5 7.1.5.3.1 Software constrution 7.1.5.3.1.5 7.1.6.3.1 Software integration 7.1.6.3.1.5 7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.1.7 Software RISK MANAGEMENT planning	6.3.4 Risk Management Process	
	5.1.8 Documentation planning	7.2.1 Software Documentation Management	7.2.1.3.1 Process implementation 7.2.1.3.1.1
	5.1.9 Software configuration management planning	7.2.2 Software Configuration Management 7.2.8 Software Problem Resolution	7.2.2.3.1 Process implementation 7.2.2.3.1.1 7.2.8.3.1 Process implementation 7.2.8.3.1.1
	5.1.10 Supporting items to be controlled	6.2.2 Infrastructure Management 6.2.2 Infrastructure Management	6.2.2.3.2 Establishment of the infrastructure 6.2.2.3.2.1 6.2.2.3.3 Maintenance of the infrastructure 6.2.2.3.3.1
	5.1.11 Software CONFIGURATION ITEM control before VERIFICATION	7.2.2 Software Configuration Management	7.2.2.3.2 Configuration identification 7.2.2.3.2.1

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
5.2 Software requirements analysis			
	5.2.1 Define and document software requirements from SYSTEM requirements	6.4.3 System Architectural Design	6.4.3.3.1 Establishing architecture 6.4.3.3.1.1
	5.2.2 Software requirements content	7.1.2 Software Requirements Analysis	7.1.2.3.1 Software requirements analysis 7.1.2.3.1.1
	5.2.3 Include RISK CONTROL measures in software requirements		
	5.2.4 Re-EVALUATE MEDICAL DEVICE RISK ANALYSIS	None	None
	5.2.5 Update SYSTEM requirements	7.1.2 Software Requirements Analysis	7.1.2.3.1 Software requirements analysis 7.1.2.3.1.1 a) & b)
	5.2.6 Verify software requirements	7.2.4 Software Verification	7.2.4.3.2 Verification 7.2.4.3.2.1
5.3 Software ARCHITECTURAL design	5.3.1 Transform software requirements into an ARCHITECTURE	7.1.3 Software Architectural Design	7.1.3.3.1 Software architectural design 7.1.3.3.1.1
	5.3.2 Develop an ARCHITECTURE for the interfaces of SOFTWARE ITEMS		7.1.3.3.1 Software architectural design 7.1.3.3.1.2
	5.3.3 Specify functional and performance requirements of SOUP item	None	none
	5.3.4 Specify SYSTEM hardware and software required by SOUP item	None	none
	5.3.5 Identify segregation necessary for RISK CONTROL	None	none
	5.3.6 Verify software ARCHITECTURE	7.1.3 Software Architectural Design	7.1.3.3.1 Software architectural design 7.1.3.3.1.6
5.4 Software detailed design	5.4.1 Refine SOFTWARE ARCHITECTURE into SOFTWARE UNITS	7.1.4 Software Detailed Design	7.1.4.3.1 Software detailed design 7.1.4.3.1.1
	5.4.2 Develop detailed design for each SOFTWARE UNIT		
	5.4.3 Develop detailed design for interfaces		7.1.4.3.1 Software detailed design 7.1.4.3.1.2
	5.4.4 Verify detailed design	7.1.4 Software Detailed Design	7.1.4.3.1 Software detailed design 7.1.4.3.1.7
5.5 SOFTWARE UNIT implementation and verification	5.5.1 Implement each SOFTWARE UNIT	7.1.5 Software Construction	7.1.5.3.1 Software construction 7.1.5.3.1.1
	5.5.2 Establish SOFTWARE UNIT VERIFICATION PROCESS	7.1.4 Software Detailed Design 7.1.5 Software Construction	7.1.4.3.1 Software detailed design 7.1.4.3.1.5 7.1.5.3.1 Software Construction 7.1.5.3.1.5
	5.5.3 SOFTWARE UNIT acceptance criteria	7.1.5 Software Construction	7.1.5.3.1 Software construction 7.1.5.3.1.5

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
	5.5.4 Additional SOFTWARE UNIT acceptance criteria	7.1.5 Software Construction 7.2.4 Software Verification	7.1.5.3.1 Software construction 7.1.5.3.1.2
	5.5.5 SOFTWARE UNIT VERIFICATION	7.1.5 Software Construction	7.1.5.3.1 Software construction 7.1.5.3.1.2
5.6 Software integration and integration testing	5.6.1 Integrate SOFTWARE UNITS	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2
	5.6.2 Verify software integration	7.1.6 Software Integration 6.4.5 System Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2 6.4.5.3.1 Integration 6.4.5.3.1.2
	5.6.3 Test integrated software	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.1
	5.6.4 Integration testing content	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.6.5 Verify integration tests procedures	None	None
	5.6.6 Conduct regression tests	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2
	5.6.7 Integration test record contents	7.1.6 Software Integration	7.1.6.3.1 Software integration 7.1.6.3.1.2
	5.6.8 Use software problem resolution PROCESS	7.2.4 Software Verification	7.2.4.3.1 Process implementation 7.2.4.3.1.6
5.7 SOFTWARE SYSTEM testing	5.7.1 Establish tests for each software requirement	7.1.6 Software Integration 7.1.7 Software Qualification Testing	7.1.6.3.1 Software integration 7.1.6.3.1.4 7.1.7.3.1 Software qualification testing 7.1.7.3.1.1
	5.7.2 Use software problem resolution PROCESS	7.2.4 Software Verification	7.2.4.3.1 Process implementation 7.2.4.3.1.6
	5.7.3 Retest after changes	7.2.8 Software Problem Resolution	7.2.8.3.1 Process implementation 7.2.8.3.1.1
	5.7.4 Verify SOFTWARE SYSTEM testing	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.7.5 SOFTWARE SYSTEM test record contents	7.1.7 Software Qualification Testing	7.1.7.3.1 Software qualification testing 7.1.7.3.1.1
5.8 Software release	5.8.1 Ensure software VERIFICATION is complete	6.4.9 Software Operation 7.2.2 Software Configuration Management	6.4.9.3.2 Operation activation and check-out 6.4.9.3.2.1 6.4.9.3.2.2 7.2.2.3.6 Release management and delivery 7.2.2.3.6.1
	5.8.2 Document known residual ANOMALIES	7.2.2 Software Configuration Management 7.1.7 Software Qualification Testing	7.2.2.3.5 Configuration evaluation 7.2.2.3.5.1 7.1.7.3.1 Software qualification testing 7.1.7.3.1.3
	5.8.3 EVALUATE known residual ANOMALIES		

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
	5.8.4 Document released VERSIONS	7.2.2 Software Configuration Management Process	7.2.2.3.6 Release management and delivery 7.2.2.3.6.1
	5.8.5 Document how released software was created		
	5.8.6 Ensure ACTIVITIES and TASKS are complete		
	5.8.7 Archive software		
	5.8.8 Assure repeatability of software release		
6 Software maintenance PROCESS		6.4.10 Software Maintenance Process	
6.1 Establish software maintenance plan		6.4.10 Software Maintenance	None
6.2 Problem and modification analysis	6.2.1 Document and EVALUATE feedback	None	None
	6.2.1.1 Monitor feedback	6.4.10 Software Maintenance	None
	6.2.1.2 Document and EVALUATE feedback		
	6.2.1.3 EVALUATE PROBLEM REPORT'S effects on SAFETY	6.4.10 Software Maintenance	None
	6.2.2 Use software problem resolution PROCESS	6.4.10 Software Maintenance	None
	6.2.3 Analyse CHANGE REQUESTS	6.4.10 Software Maintenance	None
	6.2.4 CHANGE REQUEST approval	6.4.10 Software Maintenance	None
	6.2.5 Communicate to users and regulators	6.4.10 Software Maintenance	None
6.3 Modification implementation		None	None
	6.3.1 Use established PROCESS to implement modification	6.4.10 Software Maintenance	None
	6.3.2 Re-release modified SOFTWARE SYSTEM	7.2.2 Software Configuration Management	None
7 Software RISK MANAGEMENT PROCESS		6.3.4 Risk Management Process This is based on ISO/IEC 16085. 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			
8.1 Configuration identification	8.1.1 Establish means to identify CONFIGURATION ITEMS	7.2.2 Software Configuration Management	None
	8.1.2 Identify SOUP	None	None
	8.1.3 Identify SYSTEM configuration documentation	7.2.2 Software Configuration Management	None
8.2 Change control	8.2.1 Approve CHANGE REQUESTS	7.2.2 Software Configuration Management	None
	8.2.2 Implement changes	6.4.10 Software Maintenance	None
	8.2.3 Verify changes	7.2.2 Software Configuration Management	None
	8.2.4 Provide means for TRACEABILITY of change		

ISO/IEC 62304 PROCESSES		ISO/IEC 12207:2008	
ACTIVITY	TASK	PROCESSES	ACTIVITY/TASK
8.3 Configuration status accounting		7.2.2 Software Configuration Management	Non e
9 Software problem resolution PROCESS			
9.1 Prepare PROBLEM REPORTS		7.2.8 Software Problem Resolution	None
9.2 Investigate the problem		7.2.8 Software Problem Resolution	None
9.3 Advise relevant parties		7.2.8 Software Problem Resolution	None
9.4 Use change control process		7.2.2 Software Configuration Management 6.4.10 Software Maintenance	None
9.5 Maintain records		7.2.8 Software Problem Resolution	None
9.6 Analyse problems for trends		7.2.8 Software Problem Resolution	None
9.7 Verify software problem resolution		7.2.8 Software Problem Resolution	None
9.8 Test documentation contents		All testing TASKS in ISO 12207 require documentation	None

C.7 Relationship to IEC 61508

The question has been raised whether this standard, being concerned with the design of SAFETY-critical software, should follow the principles of IEC 61508. The approach to safety in IEC 62304 is fundamentally different than the one in IEC 61508. IEC 62304 takes into account that the effectiveness of medical devices justifies residual risks related to their use. The following explains the stance of this standard.

IEC 61508 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 standard 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 MEDICAL DEVICE SOFTWARE.

For issue 2), this standard takes a simpler approach than IEC 61508. The latter classifies software into 4 "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 standard simplifies issue 2) by defining the classification into 3 software safety classes based on the RISK caused by a failure. After classification, different PROCESSES are required for different software safety classes: the intention is to further reduce the probability (and/or the severity) of failure of the software.

Issue 3) is not addressed by this standard. Readers of the standard are encouraged to use IEC 61508 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 standard 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 standard makes no requirement for an independent safety assessor, since this is a matter for ISO 14971.

Annex D
(informative)

Implementation

D.1 Introduction

This annex gives an overview of how this standard can be implemented into MANUFACTURERS' PROCESSES. It also considers that other standards like ISO 13485 [8] require adequate and comparable PROCESSES.

D.2 Quality management system

For MANUFACTURERS of MEDICAL DEVICES, including MEDICAL DEVICE SOFTWARE in the context of this standard, the establishment of a quality management system (QMS) is required in 4.1. This standard 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 QM 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 standard.

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

This standard can be implemented by adapting or extending the PROCESSES already installed in the QMS system, or integrating new PROCESSES. This standard does not specify how this is to 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 standard are suitably put into action when the MEDICAL DEVICE 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 safety classification (A, B or C) of the software. Table D.1 lists all ACTIVITIES described in this standard. The reference to ISO 13485 should help to define the place in the QMS. Based on the required software safety class, 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.

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

ACTIVITY	Related clause of ISO 13485:2003	Covered by existing procedure?	If yes: Reference	Actions to be taken
5.1 Software development planning	7.3.1 Design and development planning	Yes/No		
5.2 Software requirements analysis	7.3.2 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 and verification		Yes/No		
5.6 Software integration and integration testing		Yes/No		
5.7 SOFTWARE SYSTEM testing	7.3.3 Design and development outputs 7.3.4 Design and development review	Yes/No		
5.8 Software release	7.3.5 Design and development verification 7.3.6 Design and development validation	Yes/No		
6.1 Establish software maintenance plan	7.3.7 Control of design and development changes	Yes/No		
6.2 Problem and modification analysis		Yes/No		
6.3 Modification implementation	7.3.5 Design and development verification 7.3.6 Design and development validation	Yes/No		
7.1 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	7.5.3 Identification and traceability	Yes/No		
8.2 Change control	7.5.3 Identification and traceability	Yes/No		
8.3 Configuration status accounting		Yes/No		
9 Software problem resolution PROCESS		Yes/No		

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 60601-1-4:1996, *Medical electrical equipment – Part 1: General requirements for safety – 4.Collateral standard: Programmable electrical medical systems* (withdrawn)
IEC 60601-1-4:1996/AMD1:1999

[3] IEC 60601-1-6, *Medical electrical equipment – Part 1-6: General requirements for basic safety and essential performance – Collateral standard: Usability*

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

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

[6] ISO 9000:2005, *Quality management systems – Fundamentals and vocabulary*

[7] ISO 9001:2008, *Quality management systems – Requirements*

[8] ISO 13485:2003, *Medical devices – Quality management systems – Requirements for regulatory purposes*

[9] ISO/IEC 12207:2008, *Systems and software engineering – Software life cycle processes*

[10] ISO/IEC 14764:1999, *Software Engineering – Software Life Cycle Processes – Maintenance*

[11] ISO/IEC 15504-5:2012, *Information technology – Process assessment – Part 5: An exemplar software life cycle process assessment model*

[12] ISO/IEC 25010:2011, *Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*

[13] ISO/IEC 33001:—[2)], *Information technology – Process assessment – Concepts and terminology*

[14] ISO/IEC 33004:—[2)], *Information technology – Process assessment – Requirements for process reference, process assessment and maturity models*

[15] ISO/IEC 90003:2014, *Software engineering – Guidelines for the application of ISO 9001:2008 to computer software*

[16] ISO/IEC Guide 51:2014, *Safety aspects – Guidelines for their inclusion in standards*

[17] IEEE 610.12:1990, *IEEE standard glossary of software engineering terminology*

[18] IEEE 1044:2009, *IEEE standard classification for software anomalies*

2 To be published.

[19] U.S. Department Of Health and Human Services, Food and Drug Administration, Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices, May 11, 2005, <://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089543.htm>

[20] U.S. Department Of Health and Human Services, Food and Drug Administration, *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*, January 11, 2002, <http://www.fda.gov/downloads/RegulatoryInformation/Guidances/ucm126955.pdf>

[21] IEC 62366-1:2015, *Medical devices – Part 1: Application of usability engineering to medical devices*

[22] IEC 82304-1:—[3)], *Healthcare Software Systems – Part 1: General requirements*

3 In preparation.

Index of defined terms

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch