IEC/TR 80002-1

Edition 1.0 2009-09

TECHNICAL REPORT

colour inside

Medical device software – Part 1: Guidance on the application of ISO 14971 to medical device software

IEC/TR 80002-1:2009(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

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.

- Catalogue of IEC publications: www.iec.ch/searchpub

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 It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

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

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC/TR 80002-1

Edition 1.0 2009-09

TECHNICAL REPORT

colour inside

Medical device software –
Part 1: Guidance on the application of ISO 14971 to medical device software

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE XB

ICS 11.040.01

ISBN 2-8318-1061-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

MEDICAL DEVICE SOFTWARE –

Part 1: Guidance on the application of ISO 14971 to medical device software

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 provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
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.

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".

IEC 80002-1, which is a technical report, 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.

The text of this technical report is based on the following documents:

Enquiry draft	Report on voting
62A/639A/DTR	62A/664/RVC

Full information on the voting for the approval of this technical report can be found in the report on voting indicated in the above table. In ISO, the technical report has been approved by 16 P-members out of 17 having cast a vote.

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

In this technical report 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 TECHNICAL REPORT THAT HAVE BEEN DEFINED IN CLAUSE 2 AND ALSO GIVEN IN THE INDEX: IN SMALL CAPITALS.

A list of all parts of the IEC 80002 series, published under the general title *Medical device software,* can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result 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.

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 publication 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 implementation of the software fulfils those intentions without causing any unacceptable RISKS.

It is important to understand that software is not itself a HAZARD, but software may contribute to HAZARDOUS SITUATIONS. Software should always be considered in a SYSTEM perspective and software RISK MANAGEMENT cannot be performed in isolation from the SYSTEM.

Complex software designs can permit complex sequences of events which may contribute to HAZARDOUS SITUATIONS. Much of the TASK of software RISK MANAGEMENT consists of identifying those sequences of events that can lead to a HAZARDOUS SITUATION and identifying points in the sequences of events at which the sequence can be interrupted, preventing HARM or reducing its probability.

Software sequences of events which contribute to HAZARDOUS SITUATIONS may fall into two categories:

a) sequences of events representing unforeseen software responses to inputs (errors in specification of the software);

b) sequences of events arising from incorrect coding (errors in implementation of the software).

These categories are specific to software, arising from the difficulty of correctly specifying and implementing a complex SYSTEM and the difficulty of completely verifying a complex SYSTEM.

Since it is very difficult to estimate the probability of software ANOMALIES that could contribute to HAZARDOUS SITUATIONS, and since software does not fail randomly in use due to wear and tear, the focus of software aspects of RISK ANALYSIS should be on identification of potential software functionality and ANOMALIES that could result in HAZARDOUS SITUATIONS – not on estimating probability. RISKS arising from software ANOMALIES need most often to be evaluated on the SEVERITY of the HARM alone.

RISK MANAGEMENT is always a challenge and becomes even more challenging when software is involved. The following clauses contain additional details regarding the specifics of software and provide guidance for understanding ISO 14971:2007 in a software perspective.

- **Organization of the technical report**

This technical report is organized to follow the structure of ISO 14971:2007 and guidance is provided for each RISK MANAGEMENT activity in relation to software.

There is some intentional REDUNDANCY in the information provided due to the iterative nature of RISK MANAGEMENT activities in the software LIFE-CYCLE.

MEDICAL DEVICE SOFTWARE –

Part 1: Guidance on the application of ISO 14971 to medical device software

1 General

1.1 Scope

This technical report provides guidance for the application of the requirements contained in ISO 14971:2007, *Medical devices— Application of risk management to medical devices* to MEDICAL DEVICE SOFTWARE with reference to IEC 62304:2006, *Medical device software— Software life cycle processes*. It does not add to, or otherwise change, the requirements of ISO 14971:2007 or IEC 62304:2006.

This technical report is aimed at RISK MANAGEMENT practitioners who need to perform RISK MANAGEMENT when software is included in the MEDICAL DEVICE/SYSTEM, and at software engineers who need to understand how to fulfil the requirements for RISK MANAGEMENT addressed in ISO 14971.

ISO 14971, recognized worldwide by regulators, is widely acknowledged as the principal standard to use when performing MEDICAL DEVICE RISK MANAGEMENT. IEC 62304:2006, makes a normative reference to ISO 14971 requiring its use. The content of these two standards provides the foundation for this technical report.

It should be noted that even though ISO 14971 and this technical report focus on MEDICAL DEVICES, this technical report may be used to implement a SAFETY RISK MANAGEMENT PROCESS for all software in the healthcare environment independent of whether it is classified as a MEDICAL DEVICE.

This technical report does not address:

- areas already covered by existing or planned standards, e.g. alarms, usability engineering, networking, etc.;
- production or quality management system software; or
- software development tools.

This technical report is not intended to be used as the basis of regulatory inspection or certification assessment activities.

For the purposes of this technical report, “should” is used to indicate that amongst several possibilities to meet a requirement, one is recommended as being particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required. This term is not to be interpreted as indicating requirements.

1.2 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.

IEC 62304:2006, *Medical device software – Software life cycle processes*

ISO 14971:2007, *Medical devices – Application of risk management to medical devices*

2 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 14971:2007, IEC 62304:2006 and the following terms and definitions apply.

NOTE An index of defined terms is found beginning on page 63.

2.1
DIVERSITY
a form of REDUNDANCY in which the redundant elements use different (diverse) components, technologies or methods to reduce the probability that all of the elements fail simultaneously due to a common cause

2.2
REDUNDANCY
provision of multiple components or mechanisms to achieve the same function such that failure of one or more of the components or mechanisms does not prevent the performance of the function

2.3
SAFETY-RELATED SOFTWARE
software that can contribute to a HAZARDOUS SITUATION or software used in the implementation of RISK CONTROL measures

3 General requirements for RISK MANAGEMENT

3.1 RISK MANAGEMENT PROCESS

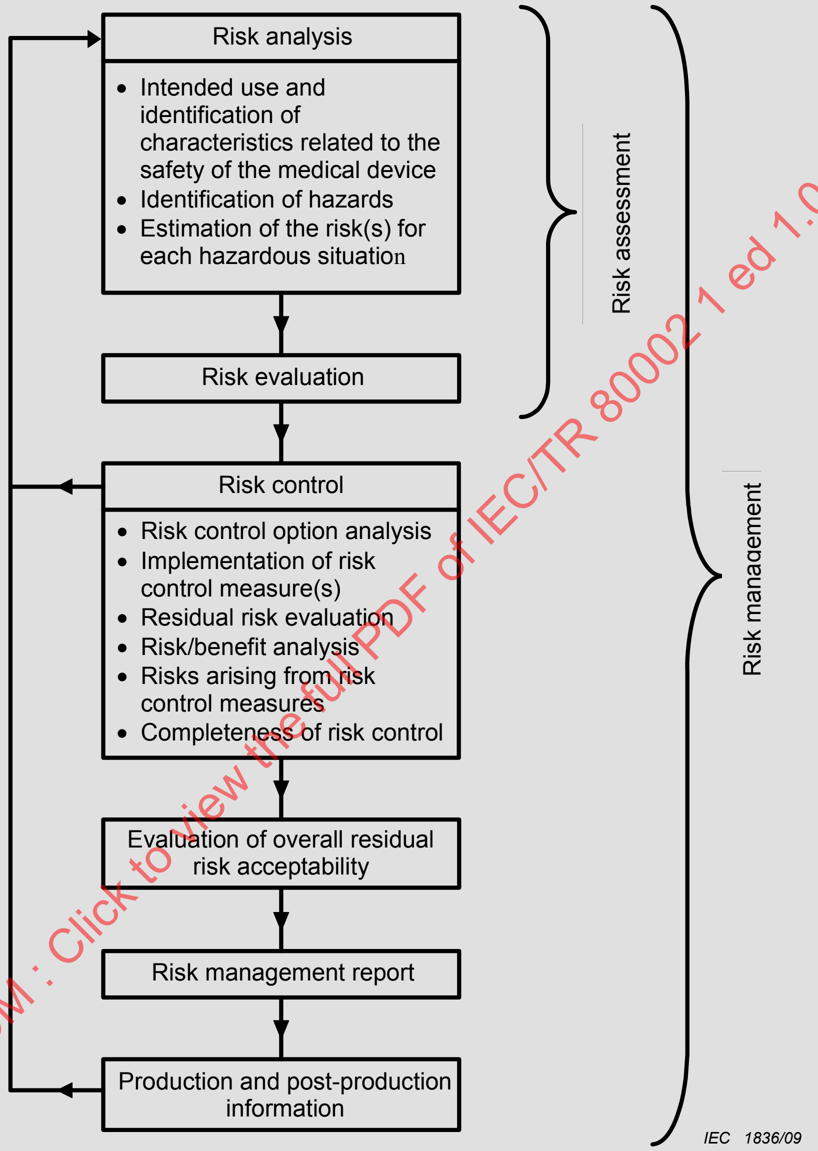
3.1.1 General

> **Text of ISO 14971:2007**
>
> **3 General requirements for RISK MANAGEMENT**
>
> **3.1 RISK MANAGEMENT PROCESS**
>
> The MANUFACTURER shall establish, document and maintain throughout the LIFE-CYCLE an ongoing PROCESS for identifying HAZARDS associated with a MEDICAL DEVICE, estimating and evaluating the associated RISKS, controlling these RISKS, and monitoring the effectiveness of the controls. This PROCESS shall include the following elements:
>
> - RISK ANALYSIS;
> - RISK EVALUATION;
> - RISK CONTROL;
> - production and POST-PRODUCTION information.
>
> Where a documented product realization PROCESS exists, such as that described in Clause 7 of ISO 13485:2003 [1][1], it shall incorporate the appropriate parts of the RISK MANAGEMENT PROCESS.
>
> NOTE 1 A documented quality management system PROCESS can be used to deal with SAFETY in a systematic manner, in particular to enable the early identification of HAZARDS and HAZARDOUS SITUATIONS in complex MEDICAL DEVICES and systems.

[1] Figures in square brackets refer to the Bibliography.

NOTE 2 A schematic representation of the RISK MANAGEMENT PROCESS is shown in Figure 1. Depending on the specific LIFE-CYCLE phase, individual elements of RISK MANAGEMENT can have varying emphasis. Also, RISK MANAGEMENT activities can be performed iteratively or in multiple steps as appropriate to the MEDICAL DEVICE. Annex B contains a more detailed overview of the steps in the RISK MANAGEMENT PROCESS.

Compliance is checked by inspection of appropriate documents.


Risk analysis
• Intended use and identification of characteristics related to the safety of the medical device
• Identification of hazards
• Estimation of the risk(s) for each hazardous situation
Risk evaluation
Risk control
• Risk control option analysis
• Implementation of risk control measure(s)
• Residual risk evaluation
• Risk/benefit analysis
• Risks arising from risk control measures
• Completeness of risk control
Evaluation of overall residual risk acceptability
Risk management report
Production and post-production information
Risk assessment
Risk management

IEC 1836/09

Figure 1 – A schematic representation of the RISK MANAGEMENT PROCESS

SAFETY is a property of the SYSTEM (in this case the whole MEDICAL DEVICE), which can include software. RISK MANAGEMENT should be performed on the SYSTEM comprising the software and its whole hardware environment. Software RISK MANAGEMENT activities should not be conducted in isolation from the SYSTEM.

While software aspects of RISK MANAGEMENT can not be effectively performed in isolation from overall MEDICAL DEVICE RISK MANAGEMENT, there are activities that may be best performed by software engineers as an integral part of the software LIFE-CYCLE. There are also elements of software that require more focus and different explanation than that provided in ISO 14971:2007 for overall MEDICAL DEVICE RISK MANAGEMENT. It is important to stress that even the software aspects of RISK MANAGEMENT need to focus on the MEDICAL DEVICE RISK in order to be effective.

NOTE 1 Software aspects of RISK MANAGEMENT can not be effectively performed in isolation from overall MEDICAL DEVICE RISK MANAGEMENT because of the interdependence of hardware failures, software failures, and hardware and software RISK CONTROL measures.

NOTE 2 For instance all software failures are systematic not random (as many types of hardware failures/breakdowns are) and their probability can not be accurately estimated. Therefore, the way in which the probability component of RISK is applied to software is quite different. See 4.4.3.

There are many opportunities for software engineers to contribute to the overall SAFETY of the MEDICAL DEVICE during the early stages of MEDICAL DEVICE design. The role of software in the SAFETY of the MEDICAL DEVICE should be considered before the SYSTEM design is finalised.

By participating in the MEDICAL DEVICE design PROCESS, the software engineer can contribute to SAFETY-related decisions concerning software-related RISKS as the design evolves. Such decisions should include but not be limited to:

- the provision of adequate hardware resources to support the software;
- the partitioning of functions between hardware and software;
- the intended use of the whole MEDICAL DEVICE and the intended use of the software user interfaces;
- the avoidance of unnecessarily complex software.

3.1.2 Iteration

Typical software development LIFE-CYCLES often use iteration. The use of iteration allows:

- investigation of the feasibility of different software designs;
- development of different SOFTWARE ITEMS at different times;
- staged delivery of different VERSIONS of the software;
- correction of errors made during the software development PROCESS.

IEC 62304:2006 requires iteration of RISK MANAGEMENT activities and coordination with SYSTEM design activities throughout the software LIFE-CYCLE. For example, during software development, Subclause 5.2.4 of IEC 62304:2006 requires the re-evaluation of the MEDICAL DEVICE RISK ASSESSMENT when software requirements are established. This re-evaluation may cause an update to SYSTEM requirement specifications and the MEDICAL DEVICE RISK ASSESSMENT. RISK EVALUATION should be repeated at all stages from requirements via ARCHITECTURE and design to the implementation of software.

ISO 14971 does not prescribe a design and development PROCESS, and it generally only requires RISK MANAGEMENT steps to be done before and after (not during) the implementation of the design (including RISK CONTROL measures). For example, when a RISK CONTROL measure has been implemented, ISO 14971 requires that it be reviewed to ensure that it has not caused any further HAZARDS and HAZARDOUS SITUATIONS. This should not be interpreted as an instruction to perform this review only after the implementation is complete—it is advantageous to address any further HAZARDS as soon as they become apparent. This implies iteration within the implementation of the RISK CONTROL measure.

It is important that all DELIVERABLES are kept consistent at all times. Iteration is a threat to the consistency of DELIVERABLES. It is therefore important that rigorous configuration management be used to ensure that all effects of a change are identified and all relevant DELIVERABLES are updated after a change. This is particularly important if software is involved, since software can be changed rapidly and an apparently small change can have unexpected side effects. All information related to the software needs to be up to date in order to avoid miscommunication between engineers. Proposals for software changes are examined for side-effects, especially side-effects which affect SAFETY. This may lead to repetition of parts of the RISK MANAGEMENT PROCESS.

3.1.3 Pro-active or reactive design approach to SAFETY

RISK MANAGEMENT should begin early with substantial input to the MEDICAL DEVICE specification, taking SAFETY into consideration in early design phases, i.e. a pro-active design approach is preferable to a reactive design approach. With a pro-active approach, SAFETY is considered along with other customer needs and captured as early SAFETY requirements. While a reactive approach is sometimes unavoidable (for example when a legacy product is updated), the proactive approach is usually the most effective, quickest and cheapest way to achieve a safe MEDICAL DEVICE.

The advantages of a pro-active SAFETY design are:

- from the outset the SYSTEM specification not only includes what the MEDICAL DEVICE should do but also identifies the SYSTEM behaviours that should be avoided in order to reduce the RISK;
- from the outset a SYSTEM ARCHITECTURE can be planned that can be demonstrated to provide the desired features while avoiding or preventing unsafe states;
- as the ARCHITECTURE is elaborated into a full design, RISK CONTROL measures can be developed while avoiding rework;
- the choice of SAFETY approaches and RISK CONTROL measures can be made early (for example, inherent SAFETY by design can be maximized and information for SAFETY minimized).

3.1.4 Characteristics of safe SYSTEMS incorporating software

Highly desirable characteristics of safe SYSTEMS include:

- the use of simple hardware SAFETY mechanisms to avoid excessive demands on SAFETY-RELATED SOFTWARE ITEMS;
- the use of only very simple SAFETY-RELATED SOFTWARE ITEMS;
- the distribution of SAFETY-RELATED SOFTWARE ITEMS between a number of independent processors;
- sufficient hardware to run all SAFETY-RELATED SOFTWARE when needed and without contention;
- the use of a deterministic design of software timing;
- the appropriate handling of failure conditions, for example
  - warning the user of failures and to allow opportunities for informed intervention;
  - providing reduced functionality in failure conditions;
  - shutting down safely when possible in failure conditions;
  - recovering quickly from failures;
- the means of preventing software code from being modified in its execution environment either through self-modification or as the result of data input;
- the means of detecting and/or preventing corruption of SAFETY-related data.

3.2 Management responsibilities

Text of ISO 14971:2007

3.2 Management responsibilities

TOP MANAGEMENT shall provide evidence of its commitment to the RISK MANAGEMENT PROCESS by:

- ensuring the provision of adequate resources

and

- ensuring the assignment of qualified personnel (see 3.3) for RISK MANAGEMENT.

> TOP MANAGEMENT shall:
>
> - define and document the policy for determining criteria for RISK acceptability; this policy shall ensure that criteria are based upon applicable national or regional regulations and relevant International Standards, and take into account available information such as the generally accepted state of the art and known stakeholder concerns;
> - review the suitability of the RISK MANAGEMENT PROCESS at planned intervals to ensure continuing effectiveness of the RISK MANAGEMENT PROCESS and document any decisions and actions taken; if the MANUFACTURER has a quality management system in place, this review may be part of the quality management system review.
>
> NOTE The documents can be incorporated within the documents produced by the MANUFACTURER'S quality management system and these documents can be referenced in the RISK MANAGEMENT FILE.
>
> Compliance is checked by inspection of the appropriate documents.

Both ISO 14971:2007 and IEC 62304:2006 assume that a quality management system is in place. The RISK MANAGEMENT requirements for TOP MANAGEMENT are listed in Subclause 3.2 of ISO 14971:2007.

NOTE Subclause 3.1 of ISO 14971:2007 states that RISK MANAGEMENT can be an integral part of a quality management system and Subclause 4.1 of IEC 62304:2006 states that the demonstration of the MANUFACTURER'S ability to consistently meet customer requirements and applicable regulatory requirements can be by the use of a quality management system that complies with ISO 13485 or a quality management system required by national regulation. IEC 62304:2006 also provides guidance on the provisions of Subclause 4.1 in Annex B.4, stating that 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.

TOP MANAGEMENT is responsible for putting in place the necessary organizational structure, adequate resources, accountability, and training (see 3.3) for an effective RISK MANAGEMENT PROCESS as well as for the safe design and maintenance of MEDICAL DEVICE SOFTWARE.

A MANUFACTURER may consider outsourcing software development or maintenance PROCESS activities (e.g. design, implementation, testing, or maintenance). In these situations, TOP MANAGEMENT is still fully responsible for ensuring that appropriate RISK MANAGEMENT activities occur for outsourced software development or maintenance PROCESSES activities and also ensuring that RISK CONTROL measures are appropriately applied.

When software development is outsourced, MANUFACTURERS should ensure by suitable contractual agreements that they will have sufficient control of the software and its design to ensure the performance of all RISK MANAGEMENT required by ISO 14971, during the whole LIFE-CYCLE of the MEDICAL DEVICE, including the correction of software ANOMALIES after the software has been released.

The MANUFACTURER should consider setting performance requirements on suppliers (see Subclause 7.4 of ISO 13485 [1] for supplier control), such as requiring the supplier to demonstrate:

- effective RISK MANAGEMENT by compliance to ISO 14971;
- effective software engineering practices by compliance to IEC 62304;
- ability to provide MEDICAL DEVICE SOFTWARE that consistently meets customer requirements and applicable regulatory requirements.

If there are RISK CONTROL measures applied to outsourced PROCESSES or products, the RISK CONTROL measures and their importance should be documented and clearly communicated to the supplier within a contractual agreement.

3.3 Qualification of personnel

3.3.1 General

Text of ISO 14971:2007

3.3 Qualification of personnel

Persons performing RISK MANAGEMENT tasks shall have the knowledge and experience appropriate to the tasks assigned to them. These shall include, where appropriate, knowledge and experience of the particular MEDICAL DEVICE (or similar MEDICAL DEVICES) and its use, the technologies involved or RISK MANAGEMENT techniques. Appropriate qualification RECORDS shall be maintained.

NOTE RISK MANAGEMENT tasks can be performed by representatives of several functions, each contributing their specialist knowledge.

Compliance is checked by inspection of the appropriate RECORDS.

Team members involved in the development and maintenance of the SOFTWARE SYSTEM should have the knowledge and experience appropriate to the TASKS assigned to them. It is fundamental that the person assigned to TASKS with RISK MANAGEMENT implications has the required knowledge of RISK MANAGEMENT. The involvement of a multidisciplinary team, including clinical experts (such as clinical support and technical service experts, and experts on other relevant subjects), software engineers, SYSTEM designers, experts on usability/human factors engineering, and domain experts, and the degree and type of their interaction with the software engineering and test staff should also be considered with respect to RISK MANAGEMENT.

This may require the development of a training program for the individuals to ensure full understanding of the required activities.

Also, the qualification of the member in the RISK MANAGEMENT team with respect to software should be considered and may require special training.

The following subclauses should provide an overview of the field of required knowledge which should be considered.

3.3.2 INTENDED USE/domain knowledge

At all stages of the design of a MEDICAL DEVICE, it is important to deploy knowledge of the INTENDED USE. This is particularly important both for designers of software and for staff carrying out RISK MANAGEMENT of software. The complex behaviour of software can easily contribute to misuse or to confusion of the user, leading to previously unforeseen HAZARDS and HAZARDOUS SITUATIONS. A thorough appreciation of clinical practice will allow RISK managers to identify HAZARDS and HAZARDOUS SITUATIONS, and allow software engineers to avoid HAZARDS and HAZARDOUS SITUATIONS or to design RISK CONTROL measures.

MANUFACTURERS should ensure that clinical experts (such as clinical support and technical service experts, and experts on other relevant subjects) are available to participate in, or at least advise on, both the design activities and the RISK MANAGEMENT activities.

In addition, MANUFACTURERS should consider training software engineers and RISK managers in the clinical use of the MEDICAL DEVICE.

3.3.3 Programming experience and attitude

Experienced software developers and testers learn to be realistic about the difficulty of discovering all software defects during testing, and hence the density of software defects remaining after testing. It is important to include experienced staff in the software development team and to give them appropriate authority to mentor, oversee and challenge less experienced staff.

It is particularly important to assign experienced staff to the following software TASKS:

– identification of the ways in which software can fail;
– analysis of RISKS associated with software failure;
– identification of RISK CONTROL measures;
– analysis of post-release PROBLEM REPORTS;
– design and implementation of changes, especially post-release.

In all these TASKS, experience will lead to an appreciation of the sort of things that can go wrong with software and with software development PROCESSES, and an awareness of the difficulties of making a change while maintaining the integrity of a software design.

3.4 RISK MANAGEMENT plan

3.4.1 General

Text of ISO 14971:2007

3.4 RISK MANAGEMENT plan

RISK MANAGEMENT activities shall be planned. Therefore, for the particular MEDICAL DEVICE being considered, the MANUFACTURER shall establish and document a RISK MANAGEMENT plan in accordance with the RISK MANAGEMENT PROCESS. The RISK MANAGEMENT plan shall be part of the RISK MANAGEMENT FILE.

This plan shall include at least the following:

a) the scope of the planned RISK MANAGEMENT activities, identifying and describing the MEDICAL DEVICE and the LIFE-CYCLE phases for which each element of the plan is applicable;

b) assignment of responsibilities and authorities;

c) requirements for review of RISK MANAGEMENT activities;

d) criteria for RISK acceptability, based on the MANUFACTURER'S policy for determining acceptable RISK, including criteria for accepting RISKS when the probability of occurrence of HARM cannot be estimated;

e) VERIFICATION activities;

f) activities related to collection and review of relevant production and POST-PRODUCTION information.

NOTE 1 Refer to Annex F for guidance on developing a RISK MANAGEMENT plan.

NOTE 2 Not all parts of the plan need to be created at the same time. The plan or parts of it can be developed over time.

NOTE 3 The criteria for RISK acceptability are essential for the ultimate effectiveness of the RISK MANAGEMENT PROCESS. For each RISK MANAGEMENT plan the MANUFACTURER should choose appropriate RISK acceptability criteria.

Options could include, among others:

- indicating in a matrix, such as Figures D.4 and D.5, which combinations of probability of HARM and SEVERITY of HARM are acceptable or unacceptable;
- further subdividing the matrix (e.g. negligible, acceptable with RISK minimization) and requiring that RISKS first be made as low as reasonably practicable before determining that they are acceptable (see D.8).

Whichever option is chosen, it should be determined according to the MANUFACTURER'S policy for determining criteria for RISK acceptability and thus be based upon applicable national or regional regulations and relevant International Standards, and take into account available information such as the generally accepted state of the art and known stakeholder concerns (see 3.2). Refer to D.4 for guidance on establishing such criteria.

If the plan changes during the LIFE-CYCLE of the MEDICAL DEVICE, a RECORD of the changes shall be maintained in the RISK MANAGEMENT FILE.

Compliance is checked by inspection of the RISK MANAGEMENT FILE.

The RISK MANAGEMENT plan should address the fact that software is part of the MEDICAL DEVICE by including:

– a description of the MEDICAL DEVICE including what functionality of the MEDICAL DEVICE will be implemented in software;

– a statement that software will be developed according to IEC 62304;

– a reference to software development aspects unique to software RISK MANAGEMENT (see Note);

– the RISK acceptance criteria for software-caused or software-controlled RISKS if they differ from acceptance criteria for other components of the MEDICAL DEVICE.

NOTE A reference to the software development plan may be the simplest way to address the inclusion of software development aspects unique to software RISK MANAGEMENT. See also 3.4.2 and 3.4.3 which discuss the relationship between the RISK MANAGEMENT plan and the software development plan and also specific RISK-related topics of the software development plan according to IEC 62304.

One reason that RISK acceptance criteria for software-caused or software-controlled RISKS might differ from acceptance criteria for other components is that the probability of HARM cannot be estimated. In this case the RISK acceptance criteria should be based on the SEVERITY of HARM. (See 4.4.3 for a discussion of probability for software caused HARM). If it can be concluded that the HAZARD is of little practical consequence, the RISK can be judged to be acceptable and no RISK CONTROL measures are necessary. However, for significant HAZARDS, that is, HAZARDS which could inflict HARM of high SEVERITY, no level of exposure can be identified that corresponds to a RISK so low that the RISK is acceptable. In this case RISK CONTROL measures need to be implemented.

RISK acceptance criteria for RESIDUAL RISK where probability cannot be estimated should take into account the RISK CONTROL measures that have been implemented and the effectiveness of those RISK CONTROL measures in reducing the probability of occurrence of HARM. RISK CONTROL measures should be a combination of all reasonable practicable measures, fulfil applicable standards and regulations, and be state of the art (see Annex D.4 of ISO 14971:2007).

When planning the activities related to collection and review of relevant production and POST-PRODUCTION information the following specific aspects for software should be considered:

– If SOFTWARE OF UNKNOWN PROVENANCE (SOUP) is used, then actively monitoring and evaluating publicly available ANOMALY lists and information about the SOUP field performance should be planned. Where possible, this should be supported by a contractual agreement with the SOUP supplier at the time of SOUP acquisition. If the users of the MEDICAL DEVICE may (intentionally or not) modify the SOUP of the MEDICAL DEVICE on their own (e.g. SOUP patches or updates), then special care should be taken in monitoring the provision of new SOUP VERSIONS for the market. See also Clause 9 regarding SOUP and POST-PRODUCTION monitoring.

– The MANUFACTURER should make it possible for the originator of a complaint to identify and report the software VERSION.

3.4.2 Relationship between RISK MANAGEMENT plan and software development plan

The requirements of ISO 14971 for a RISK MANAGEMENT plan and the requirements of IEC 62304 for a software development plan should not be taken as requiring specific documents with specific titles. Planning elements may be embodied in any documents to suit the MANUFACTURER'S quality management system, provided that:

- the combination of the planning documents satisfy the requirements of both standards in a verifiable manner;
- all plans are consistent with each other;
- all plans are available for use in a timely manner;
- all plans are kept up to date to reflect changing circumstances.

3.4.3 Specific RISK-related topics of the software development plan according to IEC 62304

The software development plan should ensure that the software development PROCESS, standards, methods, and tools associated with the development of software (described in the software development plan according to Clause 5 of IEC 62304:2006) are effective RISK CONTROL measures (see 6.2.2.6 for a discussion of PROCESS as a RISK CONTROL measure). This may be done by provision of evidence by other organisations, suppliers, and other projects within the organization. If not known, plan for and verify effectiveness within the project.

When establishing the MEDICAL DEVICE RISK MANAGEMENT PROCESS, aspects unique to software RISK MANAGEMENT should be considered, such as safe coding standards, VERIFICATION methods (e.g. formal proofs, peer reviews, walk-throughs, simulations, etc), and use of syntactic and logic checkers. If such aspects are considered RISK CONTROL measures, then they would also be subject to VERIFICATION (see Table B.2 for examples of verifying RISK CONTROL measures).

Software RISK MANAGEMENT activities should be addressed for each stage of MEDICAL DEVICE development in plans, PROCEDURES, and training, as appropriate.

3.5 RISK MANAGEMENT FILE

Text of ISO 14971:2007

3.5 RISK MANAGEMENT FILE

For the particular MEDICAL DEVICE being considered, the MANUFACTURER shall establish and maintain a RISK MANAGEMENT FILE. In addition to the requirements of other clauses of this International Standard, the RISK MANAGEMENT FILE shall provide traceability for each identified HAZARD to:

- the RISK ANALYSIS;
- the RISK EVALUATION;
- the implementation and VERIFICATION of the RISK CONTROL measures;
- the assessment of the acceptability of any RESIDUAL RISK(S).

NOTE 1 The RECORDS and other documents that make up the RISK MANAGEMENT FILE can form part of other documents and files required, for example, by a MANUFACTURER'S quality management system. The RISK MANAGEMENT FILE need not physically contain all the RECORDS and other documents; however, it should contain at least references or pointers to all required documentation. The MANUFACTURER should be able to assemble the information referenced in the RISK MANAGEMENT FILE in a timely fashion.

NOTE 2 The RISK MANAGEMENT FILE can be in any form or type of medium.

The software PROCESS should set up a system that makes this TRACEABILITY possible, starting from the software-related HAZARDS and the software RISK CONTROL measures and tracing their implementation to the corresponding SAFETY-related software requirements and the SOFTWARE ITEMS that satisfy those requirements.

All of the above should be traceable to their VERIFICATION (see Subclause 7.3.3 of IEC 62304:2006).

Because software may change frequently during development, and because it may be released in different VERSIONS, that part of the RISK MANAGEMENT FILE relating to software may also be subject to change and multiple VERSIONS.

Table 1 lists IEC 62304:2006 requirements for documentation to be included in the RISK MANAGEMENT FILE in addition to ISO 14971:2007 requirements.

Table 1 – Requirements for documentation to be included in the RISK MANAGEMENT FILE in addition to ISO 14971:2007 requirements

IEC 62304:2006 Subclause	RISK MANAGEMENT FILE documentation
4.3c)	Software SAFETY class assigned to each SOFTWARE SYSTEM
4.3f)	Rationale for using a lower software SAFETY class (than the SOFTWARE SYSTEM) for a SOFTWARE ITEM in the SOFTWARE SYSTEM which does not implement SAFETY-related functions
7.1.4	Potential causes of the SOFTWARE ITEM contributing to a HAZARDOUS SITUATION
7.1.5	Sequences of events that could result in a HAZARDOUS SITUATION that are identified in Subclause 7.1.2 of IEC 62304:2006
7.2.1	RISK CONTROL measures defined for each potential cause of the SOFTWARE ITEM contributing to a HAZARDOUS SITUATION
7.3.2	If a RISK CONTROL measure is implemented as SOFTWARE ITEM, the MANUFACTURER is to evaluate the RISK CONTROL measure to identify and document any new sequences of events that could result in a HAZARDOUS SITUATION
9.5	The MANUFACTURER is to maintain RECORDS of PROBLEM REPORTS and their resolution including their VERIFICATION. The MANUFACTURER is to update the RISK MANAGEMENT FILE as appropriate

4 RISK ANALYSIS

4.1 RISK ANALYSIS PROCESS

Text of ISO 14971:2007

4 RISK ANALYSIS

4.1 RISK ANALYSIS PROCESS

RISK ANALYSIS shall be performed for the particular MEDICAL DEVICE as described in 4.2 to 4.4. The implementation of the planned RISK ANALYSIS activities and the results of the RISK ANALYSIS shall be recorded in the RISK MANAGEMENT FILE.

NOTE 1 If a RISK ANALYSIS, or other relevant information, is available for a similar MEDICAL DEVICE, that analysis or information can be used as a starting point for the new analysis. The degree of relevance depends on the differences between the devices and whether these introduce new HAZARDS or significant differences in outputs, characteristics, performance, or results. The extent of use of an existing analysis is also based on a systematic evaluation of the effects the changes have on the development of HAZARDOUS SITUATIONS.

NOTE 2 Some RISK ANALYSIS techniques are described in Annex G.

NOTE 3 Additional guidance on RISK ANALYSIS techniques for *IN VITRO* DIAGNOSTIC MEDICAL DEVICES is given in Annex H.

NOTE 4 Additional guidance on RISK ANALYSIS techniques for toxicological HAZARDS is given in Annex I.

In addition to the RECORDS required in 4.2 to 4.4, the documentation of the conduct and results of the RISK ANALYSIS shall include at least the following:

a) a description and identification of the MEDICAL DEVICE that was analyzed;

b) identification of the person(s) and organization who carried out the RISK ANALYSIS;

c) scope and date of the RISK ANALYSIS.

NOTE 5 The scope of the RISK ANALYSIS can be very broad (as for the development of a new device with which a MANUFACTURER has little or no experience) or the scope can be limited (as for analyzing the impact of a change to an existing device for which much information already exists in the MANUFACTURER'S files).

Compliance is checked by inspection of the RISK MANAGEMENT FILE.

As described in ISO 14971:2007, RISK ANALYSIS is a term used to encompass three distinct activities;

- identification of INTENDED USE,
- identification of known or foreseeable HAZARDS (and their causes), and
- estimating the RISK of each HAZARD and HAZARDOUS SITUATION.

It is essential to recognize that RISK ANALYSIS is performed as an integral part of the entire software development PROCESS—not as one or two discrete events—for it to be effective because HAZARD and failure mode information accrue over the software development LIFE-CYCLE PROCESS and need to be considered at each stage of design.

Since it is very difficult to estimate the probability of software ANOMALIES that could contribute to HAZARDOUS SITUATIONS, the focus of software aspects of RISK ANALYSIS is on identification of potential software functionality and ANOMALIES that could result in HAZARDOUS SITUATIONS—not on estimating probability. See 4.4.3 for more details on estimating probability.

The SEVERITY of the worst case HARM for which software is a contributing factor is a primary input in determining the level of rigour of software development PROCESSES (see Subclause 4.3 of IEC 62304:2006). The information provided in 4.2, 4.3, and 4.4 is intended to help identify software-specific aspects of an effective RISK MANAGEMENT PROCESS. In addition, software aspects of the RISK ANALYSIS should be identifiable in the resulting documentation and should include both software used to implement RISK CONTROL measures for hardware failures and software causes for HAZARDS and their associated RISK CONTROL measures.

4.2 INTENDED USE and identification of characteristics related to the SAFETY of the MEDICAL DEVICE

4.2.1 General

Text of ISO 14971:2007

4.2 INTENDED USE and identification of characteristics related to the SAFETY of the MEDICAL DEVICE

> For the particular MEDICAL DEVICE being considered, the MANUFACTURER shall document the INTENDED USE and reasonably foreseeable misuse. The MANUFACTURER shall identify and document those qualitative and quantitative characteristics that could affect the SAFETY of the MEDICAL DEVICE and, where appropriate, their defined limits. This documentation shall be maintained in the RISK MANAGEMENT FILE.
>
> NOTE 1 In this context, misuse is intended to mean incorrect or improper use of the MEDICAL DEVICE.
>
> NOTE 2 Annex C contains questions such as those relating to use that can serve as a useful guide in identifying MEDICAL DEVICE characteristics that could have an impact on SAFETY.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

While each MEDICAL DEVICE has an INTENDED USE, the potential for misuse—intentional or otherwise—should also be considered. While this is not a software-specific concern, the use of software may lead to an increased RISK of misuse because:

- the MEDICAL DEVICE's behaviour is more complex and therefore more difficult to master or understand;
- the user may become over-reliant on software, not understanding its limitations;
- the MEDICAL DEVICE may be configurable, and the user may be unaware of the current configuration.
- MEDICAL DEVICES may communicate with MEDICAL and non-MEDICAL DEVICES in a manner that cannot be anticipated in detail by the MEDICAL DEVICE MANUFACTURER.

The person responsible for producing SYSTEM requirements and the software engineer have a joint responsibility to RECORD in the RISK MANAGEMENT FILE the INTENDED USE of the SYSTEM including its software, together with all SYSTEM and software requirements that relate to SAFETY and safe use. The software engineer is specifically responsible for identifying aspects of INTENDED USE that are too subtle to be apparent at the SYSTEM level.

4.2.2 User interface

Software makes it possible to design flexible user interfaces, which may affect users' behaviour, leading to new forms of reasonably foreseeable misuse. Common misuses arise from misunderstanding of an over-complex user interface and over-reliance on software to avoid errors and unsafe states. It is important to anticipate such misuse and to change the design to avoid them as far as possible.

This includes implementation of multi-lingual labelling, especially when such labelling is a RISK CONTROL measure. Special care should be taken of:

a) different need for memory size for different languages;

b) use of different character sets;

c) use of characters instead of symbols;

d) use of different units which may require additional scaling of numerical results;

e) date formatting and numerical punctuation;

f) different layout requirements for different languages and/or character sets;

g) support of validation.

See IEC 62366 [5] for a usability PROCESS complementary to ISO 14971.

4.2.3 MEDICAL DEVICE interconnection

The use of software in MEDICAL DEVICES makes possible a range of interconnections and intercommunication between MEDICAL and non-MEDICAL DEVICES. Such connections and communications are likely to give rise to new uses (and misuses) of the SYSTEM comprising the MEDICAL DEVICE and the interconnected devices. While it is easy to foresee that such new uses and misuses may occur, it is not easy for the MEDICAL DEVICE MANUFACTURER to identify all such uses and misuses if the interconnections and intercommunication are unrestricted.

It is therefore important for MANUFACTURERS to specify a limited set of INTENDED USES for the MEDICAL DEVICE'S communication interfaces and to design the interfaces as far as possible to limit interconnections and communications to those which are safe.

For example, the software may check treatment data that is entered using the built-in interface of a MEDICAL DEVICE for consistency and reasonableness based on the identity of the user and patient and the context of the data preparation. If data is prepared elsewhere and imported into the MEDICAL DEVICE using a network connection, it may not be possible to apply the same checks. In such cases, the MANUFACTURER may consider making the software checks available to the network user as a network application, and/or restricting the data import to trusted sources, and writing a comprehensive manual for those responsible for network connections in a clinical environment.

IEC 80001-1 [6] covers the integration of MEDICAL DEVICES into IT networks in a clinical environment. In particular, it delineates the responsibilities of the MANUFACTURER and the person who integrates a MEDICAL DEVICE into the IT network.

4.3 Identification of HAZARDS

> **Text of ISO 14971:2007**
>
> **4.3 Identification of HAZARDS**
>
> The MANUFACTURER shall compile documentation on known and foreseeable HAZARDS associated with the MEDICAL DEVICE in both normal and fault conditions.
>
> This documentation shall be maintained in the RISK MANAGEMENT FILE.
>
> NOTE The examples of possible HAZARDS in E.2 and H.2.4 can be used as guidance by the MANUFACTURER to initiate HAZARD identification.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

The objective of HAZARD identification is to permit the analysis of all foreseeable HAZARDS, and the design and implementation of effective RISK CONTROL measures.

Unlike heat, electrical energy or suspended masses, software is not itself a HAZARD (a potential source of HARM); contact with software cannot cause injury. However, software may cause a person to be exposed to a HAZARD, in other words it may contribute to a HAZARDOUS SITUATION. Software failures (of any kind) often facilitate the transformation of a HAZARD into a HAZARDOUS SITUATION.

Thus, although software rarely introduces new HAZARDS, it often changes the HAZARDOUS SITUATION. More importantly for the MANUFACTURER, it may shift the responsibility for avoidance of HAZARDOUS SITUATIONS from user to the MANUFACTURER.

For example, a scalpel presents an obvious cutting HAZARD. However, the MANUFACTURER traditionally took no responsibility for this HAZARD beyond ergonomic design, as the HAZARD was assumed to be entirely in the control of the surgeon. If the scalpel is part of a remote surgery SYSTEM, the same HAZARD exists, but responsibility for avoiding the cutting HAZARD is now shared by the MANUFACTURER who supplies the software that controls the scalpel.

This means that RISK CONTROL of some HAZARDS which was, in the absence of software, solely dependent on the professional use of a MEDICAL DEVICE, is now shifted to software RISK MANAGEMENT by the MANUFACTURER.

An important case is the HAZARD of mistreatment due to the mishandling of data. This was always a HAZARD, but when the data was handled on the clinician's desk, it was not the MANUFACTURER'S responsibility. Many MEDICAL DEVICES now use software to generate, store, manipulate or use data; this makes the HAZARD partly the MANUFACTURER'S responsibility.

Software may contribute to a HAZARDOUS SITUATION in several ways, including some of the following (see also Annex B):

- the software may correctly implement an unsafe SYSTEM requirement, leading to behaviour whose hazardous nature is not appreciated until actual HARM occurs;
- the software specification may incorrectly implement a SYSTEM requirement, leading to undesired behaviour that is correct according to the software specification;
- the software design and implementation may be faulty, leading to behaviour that is contrary to the software specification. Obvious faults might arise from misunderstanding of the software specification, and errors converting the specification into code. Less obvious faults could arise from unforeseen interactions between SOFTWARE ITEMS and between the software and its infrastructure, including hardware and the operating SYSTEM.

In a MEDICAL DEVICE incorporating software, careful and comprehensive HAZARD identification may lead (in later stages of the RISK MANAGEMENT PROCESS) to the following important outcomes:

- hardware RISK CONTROL measures that will prevent software from causing HARM;
- the removal of a potentially harmful software function from the software specification;
- RISK CONTROL measures that use software to prevent HARM (see Subclause 5.2.3 of IEC 62304:2006);
- identification of the parts of the software that must be implemented with low defect density and parts of the software specification which must be targeted for special testing (Subclause 4.3 of IEC 62304:2006);
- identification of higher SAFETY class SOFTWARE ITEMS that must be segregated from other SOFTWARE ITEMS (in a lower software SAFETY class) to prevent HARM arising from unexpected side-effects (see Subclauses 4.3 and 5.3.5 of IEC 62304:2006). See further discussion of this in 6.2.2.2.4.

To adequately identify HAZARDS, clinical use of the MEDICAL DEVICE must be well understood. Also, software presents the particular challenge of complexity, including possible complex user interfaces. Therefore, software HAZARD identification cannot be done in isolation. It should be done at the SYSTEM level by a multidisciplinary team including clinical experts (such as clinical support and technical service experts), software engineers, SYSTEM designers, and experts on usability/human factors engineering (see also 3.3).

HAZARD identification should consider HARM that could result from the nature of the MEDICAL DEVICE (for example cutting, irradiating or electrocution of the patient) and also additional HAZARDS related to the use of software. The latter could include, for example:

- provision of misinformation to a clinician or patient;
- misidentification of the patient (in cases where the MEDICAL DEVICE stores patient details or prescriptions);
- delay or denial of treatment caused by software ANOMALY.

NOTE For many MEDICAL DEVICES delay or denial of treatment is not considered to HARM a patient.

Identified HAZARDS should include both HAZARDS related to software that is operating in accordance with its specification and also HAZARDS relating to software ANOMALIES (see also 6.1).

Often the user interface of the MEDICAL DEVICE is made more complex by software. In particular a MEDICAL DEVICE that incorporates software will often handle information. While this may be justified in terms of benefits to the patient, additional HAZARDS should be considered relating to incorrect, or incorrectly used, information, for example:

- incorrect data entry;
- user misreading displays;
- user misunderstanding or ignoring alarms;
- overloading of users with excessive data or excessive number of alarms (see IEC 62366 [5]).

4.4 Estimation of the RISK(S) for each HAZARDOUS SITUATION

4.4.1 General

Text of ISO 14971:2007

4.4 Estimation of the RISK(S) for each HAZARDOUS SITUATION

Reasonably foreseeable sequences or combinations of events that can result in a HAZARDOUS SITUATION shall be considered and the resulting HAZARDOUS SITUATION(S) shall be recorded.

NOTE 1 To identify HAZARDOUS SITUATIONS not previously recognized, systematic methods covering the specific situation can be used (see Annex G).

NOTE 2 Examples of HAZARDOUS SITUATIONS are provided in H.2.4.5 and E.4.

NOTE 3 HAZARDOUS SITUATIONS can arise from slips, lapses, and mistakes.

For each identified HAZARDOUS SITUATION, the associated RISK(S) shall be estimated using available information or data. For HAZARDOUS SITUATIONS for which the probability of the occurrence of HARM cannot be estimated, the possible consequences shall be listed for use in RISK EVALUATION and RISK CONTROL. The results of these activities shall be recorded in the RISK MANAGEMENT FILE.

Any system used for qualitative or quantitative categorization of probability of occurrence of HARM or SEVERITY of HARM shall be recorded in the RISK MANAGEMENT FILE.

NOTE 4 RISK ESTIMATION incorporates an analysis of the probability of occurrence and the consequences. Depending on the application, only certain elements of the RISK ESTIMATION PROCESS might need to be considered. For example, in some instances it will not be necessary to go beyond an initial HAZARD and consequence analysis. See also D.3.

NOTE 5 RISK ESTIMATION can be quantitative or qualitative. Methods of RISK ESTIMATION, including those resulting from systematic faults, are described in Annex D. Annex H gives information useful for estimating RISKS for *in vitro* diagnostic MEDICAL DEVICES.

NOTE 6 Information or data for estimating RISKS can be obtained, for example, from:

a) published standards;

b) scientific technical data;

c) field data from similar MEDICAL DEVICES already in use, including published reported incidents;

d) usability tests employing typical users;

e) clinical evidence;

f) results of appropriate investigations;

g) expert opinion;

h) external quality assessment schemes.

Compliance is checked by inspection of the RISK MANAGEMENT FILE.

To estimate software related RISK it is first necessary to identify the HAZARDOUS SITUATIONS that include software. The software may be either the initiating cause of the sequence of events leading to a HAZARDOUS SITUATION, or it may be elsewhere in the sequence, as in the case of software intended to detect a hardware malfunction. The software could include a SOUP component or the reuse of a previously developed component.

RISK ESTIMATION is based on the probability of HARM and the SEVERITY of HARM from each identified HAZARDOUS SITUATION. Because it is very difficult to estimate the probability of HARM arising from a software ANOMALY (see 4.4.3), the probability of a software ANOMALY occurring must be used with care in estimating the RISK of a HAZARDOUS SITUATION including software ANOMALY in the sequence of events leading to HARM.

4.4.2 Methods of identification

A variety of methods can be used to identify potential software roles in HAZARDOUS SITUATIONS. These techniques take different approaches and may be useful at different times in the software development. No single one of them is the only correct method. See also Annex G of ISO 14971:2007 for information on some available techniques for RISK ANALYSIS.

Fault tree analysis (FTA) is a traditional top down method (see IEC 61025 [3]) often used beginning with the MEDICAL DEVICE as a whole. FTA is primarily used to analyze causes of HARM. It postulates that a HARM occurs and uses Boolean logic to identify the events or conditions that must be present for the HARM to occur. The events or conditions are analysed in increasing detail until a point is reached where one or more RISK CONTROL measures can be identified which would prevent the HARM. FTA can be used to identify the SOFTWARE ITEMS involved in a sequence of events that results in a HAZARDOUS SITUATION.

Failure modes and effect analysis (FMEA) is a bottom-up approach (see IEC 60812 [2]) that begins with a component or subsystem (for software this is a SOFTWARE ITEM in IEC 62304), and poses the question: If this element failed, what would be the consequences?

In view of the difficulty of anticipating which software defects will be present in each SOFTWARE ITEM, the starting point for FMEA would be to list the safety-related requirements of each SOFTWARE ITEM and consider the question: If this requirement is not met, what would be the consequences?

This leads to the identification of SOFTWARE ITEMS whose failure could cause HARM, and identification of what types of failure need to be prevented.

When identifying sequences or combinations of events that can result in a HAZARDOUS SITUATION, it is easiest to focus on software directly related to the essential performance of the MEDICAL DEVICE (e.g. an algorithm that calculates glucose levels in blood) and the specific causes for related HAZARDS. It is also important to consider software causes that could result in subtle failure modes and therefore cause one or more MEDICAL DEVICE HAZARDS. Refer to Annex B for examples of software causes.

NOTE Specific causes are defects in software whose functionality is clearly related to the clinical functionality of the device and lead to one of the device HAZARDS. An example is a defect in an algorithm for calculating a test result.

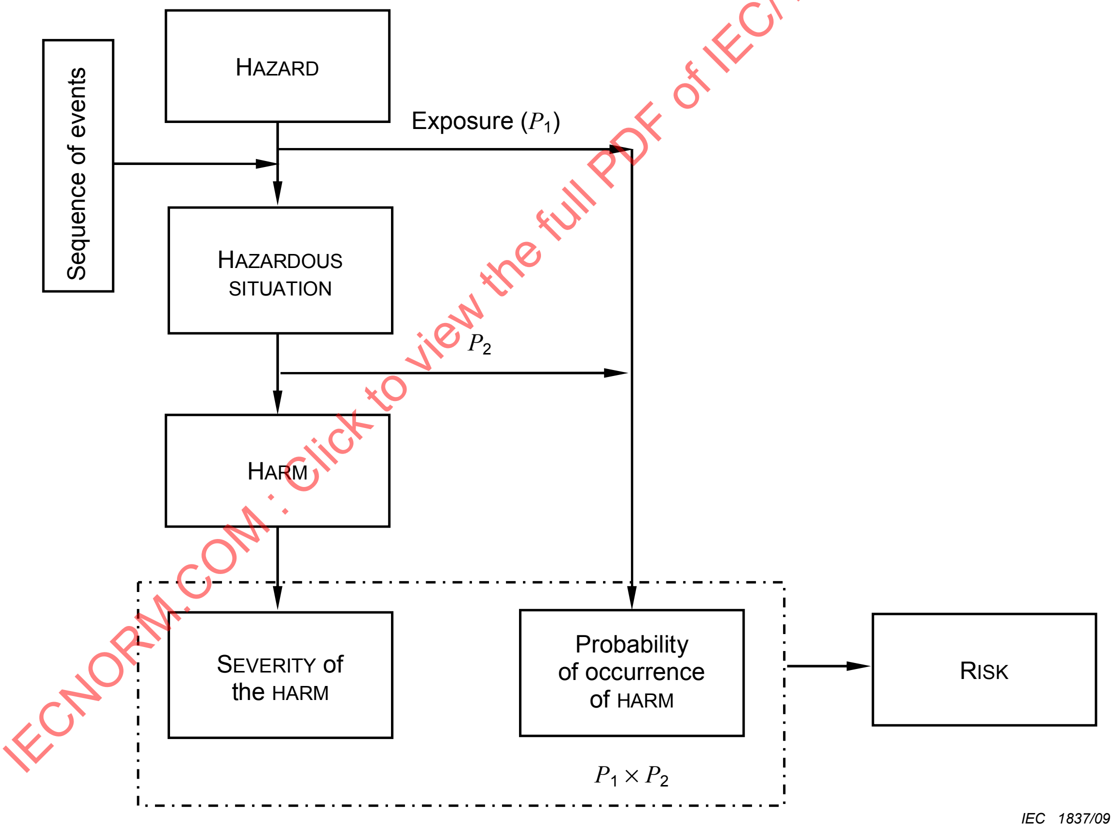
While it is difficult to anticipate exactly what may fail in a SOFTWARE ITEM, it is possible to identify categories of defects, each of which has well-known RISK CONTROL measures. For example, corruption of data is a class of fault which could be detected and prevented by using a checksum procedure. See Annex B for examples of software causes with suggested ways of handling them. MANUFACTURERS should maintain their own lists of categories of software defects relevant to their own products.

4.4.3 Probability

Software ANOMALIES in a particular VERSION of software will be present in all copies of that software. However, the probability of a software ANOMALY leading to a software failure is very difficult to estimate, because of the random nature of the inputs to each separate copy of the software.

No consensus exists for a method of estimating the probability of occurrence of a software failure. When software is present in a sequence 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 worse 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 may be used for the probability of the HAZARDOUS SITUATION occurring (P_1 in Figure 1). If this is not possible, the probability of the HAZARDOUS SITUATION occurring should be set to 1.

Estimates of probability of a HAZARDOUS SITUATION leading to HARM (P_2 in Figure 1) 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 a HARM.

Figure 1 – Pictorial representation of the relationship of HAZARD, sequence of events, HAZARDOUS SITUATION and HARM – from ISO 14971:2007 Annex E

In many cases, estimating the probability of occurrence of HARM may not be possible, and the RISK should be evaluated on the basis of the SEVERITY of the HARM alone. RISK ESTIMATION in these cases should be focused on the SEVERITY of the HARM resulting from the HAZARDOUS SITUATION.

Although it may not be possible to estimate the probability of the occurrence of a software failure, it is obvious that many RISK CONTROL measures reduce the probability that such a failure would lead to a HAZARDOUS SITUATION. Consider, for example, memory corruption resulting from a software ANOMALY. A checksum of the memory could detect the failure and reduce the probability of a HAZARDOUS SITUATION. The checksum does not guarantee any possible corruption would be detected. Rather, it would detect the vast majority of such corruption and thus lower the RISK to an acceptable level. Although the probability of a HAZARDOUS SITUATION cannot be estimated either before or after the checksum is implemented, it can be asserted that the probability of a HAZARDOUS SITUATION after the checksum is in place is lower than it was before implementing the checksum. It is the responsibility of the MANUFACTURER to demonstrate that the RISK CONTROL measure is effective in meeting the acceptability criteria for RESIDUAL RISK that was identified in the RISK MANAGEMENT plan.

In summary, software RISK ESTIMATION should focus primarily on SEVERITY and the relative probability of HARM if a failure should occur rather than on attempts to estimate the probability of each possible software failure.

NOTE This helps provide distinctions between HAZARDS of the same SEVERITY to allow greater focus on those with higher probability of actual HARM.

4.4.4 SEVERITY

The SEVERITY estimate for a software-caused RISK has an impact on the software development PROCESS to be used. According to IEC 62304 the rigour of the PROCESS depends on the SEVERITY of the HARM the software might cause.

Whereas it is open to the MANUFACTURER how SEVERITY levels are defined for the purpose of RISK EVALUATION according to ISO 14971, it is helpful to define these SEVERITY levels such that there is a relationship to the software SAFETY classification of IEC 62304. Otherwise it might be necessary to classify the SEVERITY twice, once for RISK EVALUATION needed for the overall RISK MANAGEMENT PROCESS and in addition for determining the SAFETY class of the software according to IEC 62304.

5 RISK EVALUATION

> **Text of ISO 14971:2007**
>
> **5 RISK EVALUATION**
>
> For each identified HAZARDOUS SITUATION, the MANUFACTURER shall decide, using the criteria defined in the RISK MANAGEMENT plan, if RISK reduction is required. If RISK reduction is not required, the requirements given in 6.2 to 6.6 do not apply for this HAZARDOUS SITUATION (i.e., proceed to 6.7). The results of this RISK EVALUATION shall be recorded in the RISK MANAGEMENT FILE.
>
> NOTE 1 Guidance for deciding on RISK acceptability is given in D.4.
>
> NOTE 2 Application of relevant standards, as part of the MEDICAL DEVICE design criteria, might constitute RISK CONTROL activities, thus meeting the requirements given in 6.3 to 6.6.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

As described in 4.4.3, it is difficult to estimate the probability of software failures. When this results in the inability to estimate the probability of HARM then RISK should be evaluated on the basis of the SEVERITY of the HARM alone.

6 RISK CONTROL

6.1 RISK reduction

> **Text of ISO 14971:2007**
>
> **6 RISK CONTROL**
>
> **6.1 RISK reduction**
>
> When RISK reduction is required, RISK CONTROL activities, as described in 6.2 to 6.7, shall be performed.

Software aspects of RISK reduction are discussed in 6.2 to 6.7.

6.2 RISK CONTROL option analysis

> **Text of ISO 14971:2007**
>
> **6.2 RISK CONTROL option analysis**
>
> The MANUFACTURER shall identify RISK CONTROL measure(s) that are appropriate for reducing the RISK(S) to an acceptable level.
>
> The MANUFACTURER shall use one or more of the following RISK CONTROL options in the priority order listed:
>
> a) inherent SAFETY by design;
>
> b) protective measures in the MEDICAL DEVICE itself or in the manufacturing PROCESS;
>
> c) information for SAFETY.
>
> NOTE 1 If implementing option b) or c), MANUFACTURERS can follow a PROCESS where reasonably practicable RISK CONTROL measures are considered and the option providing the appropriate reduction in RISK is chosen before determining whether the RISK is acceptable.
>
> NOTE 2 RISK CONTROL measures can reduce the SEVERITY of the HARM or reduce the probability of occurrence of the HARM, or both.
>
> NOTE 3 Many standards address inherent SAFETY, protective measures, and information for SAFETY for MEDICAL DEVICES. In addition, many other MEDICAL DEVICE standards have integrated elements of the RISK MANAGEMENT PROCESS (e.g. electromagnetic compatibility, usability, biocompatibility). Relevant standards should be applied as part of the RISK CONTROL option analysis.
>
> NOTE 4 For RISKS for which the probability of occurrence of HARM cannot be estimated, see D.3.2.3.
>
> NOTE 5 Guidance on information for SAFETY is provided in Annex J.
>
> The RISK CONTROL measures selected shall be recorded in the RISK MANAGEMENT FILE.
>
> If, during RISK CONTROL option analysis, the MANUFACTURER determines that required RISK reduction is not practicable, the MANUFACTURER shall conduct a RISK/benefit analysis of the RESIDUAL RISK (proceed to 6.5).
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

6.2.1 Choosing RISK CONTROL options for complex SYSTEMS

6.2.1.1 General

In a complex SYSTEM there may be many sequences of events which can lead to HAZARDOUS SITUATIONS. It may not be possible or necessary to apply a RISK CONTROL measure to each event in such sequences. It is sufficient to apply RISK CONTROL measures to selected events to reduce the overall probability of HARM to an acceptable level.

In the following three subclauses, an overview is given how three types of RISK CONTROL measures can be implemented in software. In addition, it is discussed which events need RISK CONTROL measures at all (see 6.2.1.5).

6.2.1.2 Inherent SAFETY by design

Inherent SAFETY by design is generally achieved by removing unsafe features of a MEDICAL DEVICE, or by changing the design to implement a feature in a safer way, i.e. a way that avoids or minimises HAZARDOUS SITUATIONS. This often has the effect of simplifying the design, making it easier to implement and easier for the user to operate.

This is particularly true of features implemented in software. There is a temptation in software SYSTEMS to include all possible customer desires without discrimination. This may result in an excessive number of ways in which software components can interact, introducing unexpected HAZARDOUS SITUATIONS. By applying RISK MANAGEMENT early in the development of the MEDICAL DEVICE and its software, this can be avoided while still satisfying the majority of customers.

In most cases, inherent SAFETY by design, applied to software, will involve:

- eliminating unnecessary features;
- changing the software ARCHITECTURE to avoid sequences of events that lead to HAZARDOUS SITUATIONS;
- simplifying the user interface to reduce the probability of human errors in use;
- specifying software design rules to avoid software ANOMALIES.

An example of the latter would include:

- using only static memory allocation to avoid software ANOMALIES related to dynamic memory allocation;
- using a restricted VERSION of a programming language to avoid structures which are likely to lead to programming errors.

6.2.1.3 Protective measures

Protective measures for a MEDICAL DEVICE that uses software can be implemented in either hardware or software. The design of a protective measure should demonstrate that the protective measure is independent of the function to which it is applied. This is relatively easy to achieve if a software protective measure is applied to hardware or vice-versa.

In choosing protective measures that are implemented in software and applied to software, it is important to avoid the possibility of multiple failures arising from one cause. If a protective measure detects and/or prevents a HAZARDOUS SITUATION, the MANUFACTURER should demonstrate an adequate segregation between the protective measure and software features that provide essential performance.

For example, software which provides treatment for a patient may run on one processor, while the software which implements software protective measures runs on a separate processor.

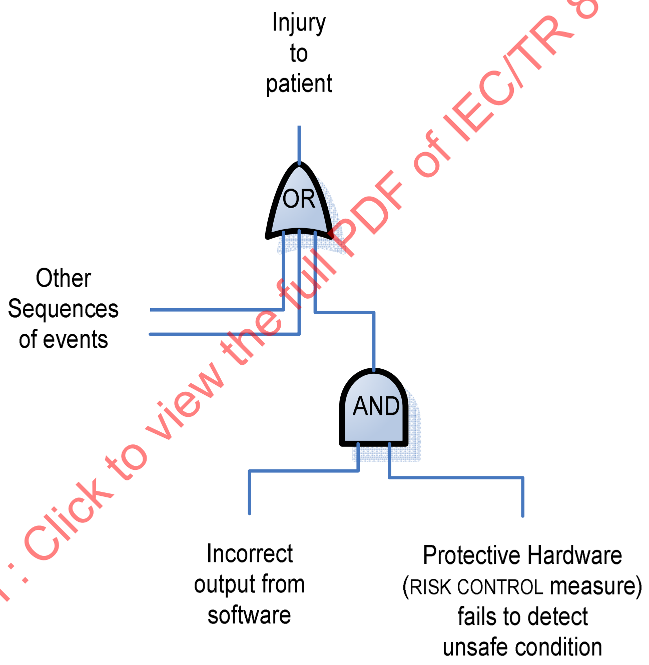
6.2.1.4 Information for SAFETY

The use of software in a MEDICAL DEVICE is likely to lead to more complex behaviour as seen by the user. This is likely to lead to an increased reliance on information for SAFETY, ranging from simple on-screen warnings to complex user manuals and defined training courses. The size and complexity of such written material can be reduced by attention to good user-interface design (see IEC 62366 [5]).

6.2.1.5 Which events need RISK CONTROL measures?

Many sequences of events can lead to HAZARDOUS SITUATIONS. It may not be possible or necessary to apply a RISK CONTROL measure to each event in such sequences. It is sufficient to apply RISK CONTROL measures to carefully selected events to reduce the overall probability of HARM to an acceptable level.

In deciding which events should be detected, prevented, or made less likely to occur, it is obviously helpful to map out the sequences of events that could lead to HAZARDOUS SITUATIONS. While a Fault Tree Analysis (see 4.4.2) does not show sequences of events, it can be used to identify such sequences. Correct operation of a RISK CONTROL measure would appear as a FALSE input to an AND gate, leading to the prevention of HARM irrespective of the other inputs to the AND gate. Figure 2 shows part of an FTA diagram, in which an incorrect software output (itself the output of a sequence of events), is prevented from causing injury to a patient by a RISK CONTROL measure which is designed to detect unsafe conditions and prevent the harmful effects of the output (for example by halting an action). The incorrect software output can only cause injury to the patient if the RISK CONTROL measure fails. Note that the sequence of events leading to the incorrect software output does not need to be explored in detail to ensure that it cannot lead to an injury to the patient.


Injury to patient
OR
Other Sequences of events
AND
Incorrect output from software
Protective Hardware (RISK CONTROL measure) fails to detect unsafe condition

IEC 1838/09

Figure 2 – FTA showing RISK CONTROL measure which prevents incorrect software outputs from causing HARM

Obvious points at which RISK CONTROL measures may be applied include:

- inputs to the SOFTWARE SYSTEM as a whole;
- outputs from the SOFTWARE SYSTEM as a whole;
- internal interfaces between software modules.

A RISK CONTROL measure which limits the range of an input to the software can prevent unsafe outputs. Less obviously, it can also reduce the probability of the input leading to HARM due to an ANOMALY in the software, because it reduces the probability that the software will operate in unexpected ways that may not have been tested (see 4.4.3).

Limiting the range of an input to the software can be done using a software or hardware RISK CONTROL measure. For example:

- a software RISK CONTROL measure can check input and reject unsafe or inconsistent values;
- a hardware RISK CONTROL measure could consist of a locked room to prevent unauthorised people from inputting data.

A RISK CONTROL measure placed at the output of a MEDICAL DEVICE or its software could check that the software's output values are within a safe range and internally consistent and could prevent HARM if it is not. This could be achieved, for example, by:

- a software RISK CONTROL measure that checks the output values and prevents them from departing from a safe range;
- a hardware RISK CONTROL measure that limits the energy applied to a patient;
- a RISK CONTROL measure consisting of a combination of a warning label and a hard-wired STOP switch. This RISK CONTROL measure assumes that a competent operator is able to detect the HAZARDOUS SITUATION.

In addition to RISK CONTROL measures applied at inputs and outputs of the MEDICAL DEVICE or its software, RISK CONTROL measures may also be applied to inputs and outputs of software components. This allows inputs and outputs of smaller parts of the software to be checked, and HARM prevented.

It may not be possible to specify a single range for a parameter within which the MEDICAL DEVICE operates safely. However, it may be possible to specify a "safe operating envelope", in other words a combination of parameters that form a boundary within which the MEDICAL DEVICE operates safely. Software can be used to assess whether the MEDICAL DEVICE is operating inside the safe operating envelope. For example, software could combine the measured temperature of an applied part and the time of exposure to detect the possibility of burning the patient.

There are cases when the safe range for the software's inputs and outputs is known by a clinician but cannot be anticipated in the design of a MEDICAL DEVICE. In such cases, RISK CONTROL measures can be used to ensure that the MEDICAL DEVICE does exactly what is specified by the clinician. A hardware or software RISK CONTROL measure can be used to detect inconsistencies between the software's outputs and inputs.

For example, the clinician may prescribe treatments, using a MEDICAL DEVICE, which vary widely from one patient to the next. A HAZARDOUS SITUATION cannot be detected only by analysing input or output values. A software RISK CONTROL measure can nonetheless be applied that ensures that the outputs of the MEDICAL DEVICE accurately match the inputs (the intended prescription).

6.2.2 RISK CONTROL methods

6.2.2.1 Overview

In order to efficiently implement appropriate RISK CONTROL measures for software, an understanding of the product development and software LIFE-CYCLES should be carefully considered. Some types of RISK CONTROL measures are very easy to implement early in design and impossible or very costly to implement later in development. If software is not carefully considered from a RISK MANAGEMENT perspective early in the product development PROCESS, hardware decisions may be made that inadvertently place excessive reliance on proper software operation for the SAFETY of the MEDICAL DEVICE.

It can be useful to segregate SOFTWARE ITEMS and assign software SAFETY classes to the SOFTWARE ITEMS to distinguish highly critical SOFTWARE ITEMS (e.g. those that could result in death if they are defective) from SOFTWARE ITEMS that could not affect SAFETY. See Subclause 4.3 of IEC 62304:2006 for software SAFETY classification.

Assigning software SAFETY classes can serve as a basis for greater rigour and focus in VERIFICATION and configuration management activities for more critical SOFTWARE ITEMS. If this is done, side-effects should be carefully considered and the less critical SOFTWARE ITEMS should be rated the same as any more critical SOFTWARE ITEMS they could affect. It should also be noted that IEC 62304 allows for different methods to be used within an ACTIVITY or TASK (See Subclause 5.1.4 of IEC 62304:2006 which requires that methods be defined). The MANUFACTURER may decide to create a scheme for differentiating SOFTWARE ITEMS which have the software SAFETY classification of Class C. For example, the MANUFACTURER may use a more formal method of VERIFICATION (i.e., code inspection versus code review) for SOFTWARE ITEMS with high complexity.

Note that SOFTWARE ITEMS could be classified initially as SAFETY-related and then through certain RISK CONTROL measures or design choices be treated as less critical. Properly performed RISK MANAGEMENT can result in reducing SAFETY-related software to the smallest possible subset through isolation and inherent safe design.

Ensuring software SAFETY requires a variety of activities throughout the product development LIFE-CYCLE. Reliability techniques such as formal methods for failure analysis do not comprise complete RISK MANAGEMENT methods. It is also important to recognize that reliability and SAFETY, although often related, are not identical attributes. A LIFE-CYCLE PROCESS that focuses on reliability may not achieve adequate SAFETY.

Some specific RISK CONTROL measures are described in more detail and guidance is given on how to address specific causes for RISKS in 6.2.2.2, 6.2.2.3, 6.2.2.4, 6.2.2.5, and 6.2.2.6.

6.2.2.2 RISK CONTROL measures and software architectural design

6.2.2.2.1 Overview

Software ARCHITECTURE should describe features of the software used to control RISK by inherently safe design, as well as software mechanisms for protective measures to reduce RISK.

6.2.2.2.2 Inherently safe design by ARCHITECTURE features

A HAZARD associated with a software control function might be avoided, for example, by using hardware to implement the function. (Similarly, a HAZARD associated with a hardware function (wear, fatigue) might be avoided by using software.)

Sometimes a HAZARD may be completely avoided by a high-level design decision. For example, from a hardware perspective, use of batteries as a power source in place of AC power could eliminate the RISK of electrocution. Similarly a whole class of programming errors that could lead to a HAZARD could be eliminated based on high level design decisions. For example, memory leaks could be avoided by using only static data structures.

A particular problem in SYSTEMS using software is the perception that there is no limit to the extent to which software can share a physical infrastructure. This perception is false.

It is a normal rule of SYSTEM design that the SYSTEM should include sufficient resources to perform all necessary TASKS when required. This rule should be applied to software as well as to hardware. If a SOFTWARE ITEM has a role in SAFETY, then RISK ASSESSMENT should address the following questions:

- Can the SAFETY-RELATED SOFTWARE ITEM gain access to its processor when needed?
- Can the SAFETY-RELATED SOFTWARE ITEM be guaranteed enough processor time to complete its TASK before an unsafe state develops into an accident?
- Can it be demonstrated that no other SOFTWARE ITEM can corrupt or otherwise interfere with the SAFETY-RELATED SOFTWARE ITEM?

If SAFETY-RELATED SOFTWARE has to share a processor with non-SAFETY-RELATED SOFTWARE, then the above questions are particularly important, as SAFETY functions will compete for resources with non-SAFETY functions (see 6.2.2.2.4 on segregation).

Development methods should be chosen to make all of the above issues visible to the designer. For example, it is not enough to design a SAFETY-RELATED SOFTWARE ITEM as a PROCESS which, all being well, will run when the operating system gets around to it. The development method should support deliberate design of scheduling, priority and timing.

6.2.2.2.3 Fault tolerant ARCHITECTURES

Many functions of a MEDICAL DEVICE may be required to be available to assure the SAFETY of the patient or user. Such functions may include clinical functions that cannot be interrupted or delayed, and functions that implement protective RISK CONTROL measures.

Fault-tolerant design is a very common approach to improving MEDICAL DEVICE dependability (references for software engineering practitioners include Pullum [7] and Banatre [8]). The objective of fault-tolerant design is to ensure that the SAFETY-related functions will continue to operate in the presence of component faults, including software ANOMALIES.

Fault tolerant design will usually make use of REDUNDANCY. This may be a simple duplication of a vital component to provide continued operation when one component fails, or it may consist of additional components which detect a failure and switch operation to an alternative mode of operation, possibly with limited functionality.

Fault tolerance may be used to continue vital functions in the event of a software failure. In this case, simple REDUNDANCY using multiple copies of the same software will probably be insufficient, as the same defect will be present in each copy of the software.

In such cases, DIVERSITY is required. For example, additional software may be used to detect a software error and to perform a recovery program. The additional software should avoid sharing any features with the software that it monitors, thereby eliminating the possibility that one software defect will cause the failure of both.

In more critical cases, two or more SOFTWARE ITEMS may perform the same function but they may be independently designed and implemented, starting from a common specification. This is "DIVERSITY programming". Note, however, that there is a tendency for the same mistakes to be made by different development engineers, which would invalidate the DIVERSITY. Note also that the common specification may contain an incorrect requirement. Finally, some method, such as voting, must be used to ensure that the malfunctioning software is prevented from having any effect. At least 3 diverse SOFTWARE ITEMS would be needed to implement a voting scheme.

When using REDUNDANCY, with or without DIVERSITY, to provide fault tolerance, it is important to signal to the user that there is a failure. Otherwise, a fault tolerant MEDICAL DEVICE may appear to operate safely when in fact it is operating with reduced SAFETY.

6.2.2.2.4 Segregation to reduce RISK from software causes

It is possible for software defects to lead to errors in unrelated software running on the same hardware. The manufacturer should select methods of segregating SAFETY-RELATED SOFTWARE ITEMS from the non-SAFETY-RELATED SOFTWARE ITEMS such that non-SAFETY-RELATED SOFTWARE ITEMS cannot interfere with the operation of the SAFETY-RELATED SOFTWARE ITEMS (see Subclause 5.3.5 of IEC 62304:2006), and should demonstrate that the segregation is effective. This includes demonstrating the appropriate use of resources (physical or time) by the SOFTWARE ITEMS to avoid unintended contention between the items.

Effective segregation between SOFTWARE ITEMS must address the following possible ways in which SOFTWARE ITEMS may be subject to unexpected interaction.

SOFTWARE ITEMS may interact in unintended ways when they contend for time on shared hardware (for example processors, storage devices and other input/output devices). This can prevent a SOFTWARE ITEM from running at the intended time. The provision of sufficient hardware is an architectural feature (see 6.2.2.2.2) which should be subject to adequate specification and planning to ensure that sufficient time is available to run all SOFTWARE ITEMS when required.

SOFTWARE ITEMS may co-exist in the same memory; this can cause one SOFTWARE ITEM unexpectedly to change data belonging to another SOFTWARE ITEM. In extreme cases, one SOFTWARE ITEM may accidentally change the coding of another SOFTWARE ITEM. Many processors and operating systems offer hardware-assisted methods of segregating memory usage. Where such methods exist, they should always be used. Most such methods will guard against unintended interaction even when there is a defect in one of the SOFTWARE ITEMS.

SOFTWARE ITEMS may also interact in unintended ways when they share variables, including global variables, environment variables, and operating system parameters; this can result in unintended communication between SOFTWARE ITEMS if there is a defect in one of the SOFTWARE ITEMS. The sharing of variables between SOFTWARE ITEMS should be minimised. If it is necessary, rules should be published to all engineers to ensure that the shared variables are only changed by a small number of specified SOFTWARE ITEMS and that all other SOFTWARE ITEMS only read the shared variables without changing them.

The strongest form of segregation consists of running SOFTWARE ITEMS that should not interact on separate processors. However, careful architectural design as recommended above may provide an appropriate degree of segregation on a single processor.

Testing the SYSTEMS in a lab environment may indicate sufficient physical and time resources for the test cases given, while the application load or the execution environment (other processes running on the same box) in the field makes the software fail in a way that causes HARM.

On the other hand, when test cases in the lab do show that there is low performance and invalid measures are taken to hastily speed-up the software, these measures possibly break the design and add other RISKS through unforeseen side-effects.

Effective segregation should demonstrate that under normal operation:

a) data flow corruption is prevented: non-SAFETY-related SOFTWARE ITEMS cannot modify SAFETY-related data;

b) control flow corruption is prevented:

   – SAFETY-related functions can always execute at the correct time, without being effected by the actions of the non SAFETY-RELATED SOFTWARE ITEMS;

   – non-SAFETY-RELATED SOFTWARE ITEMS cannot modify the SAFETY-RELATED SOFTWARE items;

c) corruption of the execution environment is prevented: corruption of parts of the SOFTWARE SYSTEM used by both SAFETY-related and non SAFETY-RELATED SOFTWARE ITEMS (e.g. processor registers, device registers and memory access privileges) cannot occur.

Events that cause any of the above to be violated, e.g. hardware failure, should be detected and cause the SYSTEM to take necessary actions to ensure continued SAFETY.

6.2.2.3 Details on protective measures

In many cases, it is not practical to avoid all HAZARDS by inherent safe design or to implement fault-tolerance for all potential failures. In those cases, protective measures are the next best approach to managing a potential HAZARD. These measures typically operate by detecting a potentially HAZARDOUS SITUATION, and either intervening automatically to mitigate consequences, or generating an alarm so that the user may intervene.

For example, a therapeutic x-ray SYSTEM may have an interlock SYSTEM using software logic or hardware that shuts down the x-ray generator if any door to the facility is opened. The interlock function has no role in delivering the therapy. Its sole purpose is to mitigate the HARM of unintentional radiation exposure.

In some cases (i.e., where loss of MEDICAL DEVICE functionality does not pose a HAZARD), SAFETY may be achieved at the expense of mission. For example, a failure of a laboratory blood analyzer to provide a result may not in some cases be hazardous, but providing an incorrect result could be. In this example, shutting down the analyzer when defensive programming checks indicate unexpected faults, rather than continuing to operate, reduces RISKS. In a *fail-safe* ARCHITECTURE, a SYSTEM or component fault, or other hazardous condition, may lead to a loss of function, but in a manner that preserves the SAFETY of operators and patients. In a *fail-operational* SYSTEM, the SYSTEM may continue to operate safely, but with degraded performance (e.g. reduced capacity or slower response time).

6.2.2.4 Preventing and announcing HAZARDOUS SITUATIONS promptly

An important class of RISK CONTROL measure is one which improves the probability of preventing a HAZARDOUS SITUATION.

In addition to preventing HARM, consideration should be given to announcing the detected condition to the user. Without this there is a possibility that a subsequent failure of the RISK CONTROL measure would allow HARM to occur.

Consideration should be given to the frequency with which software RISK CONTROL measures should run. The software RISK CONTROL measure should run sufficiently frequently to detect the condition before it causes HARM.

6.2.2.5 Risk control measures for software anomalies

Software presents a particular difficulty: some events in a sequence leading to a HAZARDOUS SITUATION may result from undiscovered software ANOMALIES and it is difficult to anticipate where such ANOMALIES may occur or what their effects might be.

RISK CONTROL measures can reduce the probability of HARM originating from software ANOMALIES. There will usually be places in the software ARCHITECTURE where a RISK CONTROL measure can reduce the probability of HARM irrespective of the nature of preceding events. If this is done carefully, it is not necessary to anticipate the exact nature of software ANOMALIES in order to prevent them from causing HARM.

In cases where this approach is not practicable, for example if a preventive measure is implemented in software, methods should be used to assure the integrity of the software (see 6.2.2.6).

6.2.2.6 Process as a risk control measure

If software ANOMALIES could contribute to a sequence of events leading to a HAZARDOUS SITUATION, it may not be possible to devise RISK CONTROL measures to prevent HARM from occurring. The best solution in this case is inherent SAFETY by design, so that software ANOMALIES cannot result in a HAZARDOUS SITUATION.

When this is not possible, an effective software development PROCESS may be used to reduce the probability of occurrence of the software ANOMALIES. There is strong consensus that PROCESS RISK CONTROL measures are beneficial when considered in combination with other types of RISK CONTROL measures and if defined in detail.

If it can be established that the confidence in the software to perform its intended function reliably is very high, and the confidence that the software is fault-free is very high, the software may be treated as a high-integrity component. To achieve this high level of confidence, the MANUFACTURER must demonstrate that the software development PROCESS can predictably produce highly reliable, fault-free software. The use of such a PROCESS can then be claimed to reduce the probability of occurrence of software ANOMALIES.

It is accepted that increasing the rigour of the software development PROCESS can reduce the number of software ANOMALIES. It should be noted that while testing on MEDICAL DEVICE SOFTWARE can reduce the number of software ANOMALIES, it cannot be assumed that when the software passes all planned tests, no software ANOMALIES remain. This is because in clinical use the inputs to the software will include sequences that were not part of the planned tests. Since MEDICAL DEVICE SOFTWARE is too complex to test exhaustively, rigorous testing can only be viewed as a method to lower the probability of HAZARDOUS SITUATIONS. Testing by itself, however, is not sufficient to establish confidence that the software can be treated as a high-integrity component.

A starting point for defining a rigorous software development PROCESS would be to include the activities and TASKS specified in IEC 62304 in the software development PROCESS. Additional items to consider when developing a rigorous software development PROCESS might include:

- staff competency – skills, qualifications, experience and training (who develops the software?);
- methods –the suitability of the specification, design, coding and testing methods (what is the PROCESS of development?);
- rigour, formality, and scope of reviews and inspections (how much static analysis is performed?);
- tools –quality of tools such as compilers, requirements TRACEABILITY, and configuration management tools (what tools are used during development of software?).

The predictability of developing high-integrity software depends on having a repeatable PROCESS that is consistently followed.

When implementation of a rigorous development PROCESS is used to reduce RISK of HAZARDOUS SITUATIONS resulting from software ANOMALIES, the effectiveness of the RISK CONTROL measures should be demonstrated by collecting and analyzing data showing the frequency of failures resulting from software ANOMALIES. To support the claim that a PROCESS produces high-integrity software, there should be evidence that there are no or very infrequent software failures.

6.2.3 SOFTWARE OF UNKNOWN PROVENANCE (SOUP) considerations

The use of SOUP is often determined during the SYSTEM design. The higher the potential RISKS of the MEDICAL DEVICE, the more closely potential failure modes of SOUP should be analyzed and RISK CONTROL measures identified. SOUP generally can not be modified to incorporate new RISK CONTROL measures needed to monitor or isolate SOUP to prevent it from contributing to HAZARDS or HAZARDOUS SITUATIONS if it fails. Nor is there adequate internal design information available to identify all the potential HAZARDS caused by SOUP. Therefore the SYSTEM and software ARCHITECTURE should be designed to provide RISK CONTROL measures needed to monitor or isolate SOUP to prevent it from causing HAZARDS if it fails.

When SOUP is included in the MEDICAL DEVICE, care must be taken to prevent compromising MEDICAL DEVICE SAFETY. Such a situation may call for the introduction of “wrappers” or a middleware ARCHITECTURE. The middleware may:

a) prevent the use of features of the SOUP that one does not want to use,

b) perform logical checks to ensure correct information is transferred between the SOUP and the MEDICAL DEVICE SOFTWARE, or

c) provide additional information needed by the MEDICAL DEVICE.

Another critical issue related to SOUP is the use of commercial operating and communication systems. An ARCHITECTURE should be established that permits changes (e.g. stability, SECURITY) to the software platform, based on a thorough RISK ASSESSMENT without compromising SAFETY. Such a RISK ASSESSMENT should include an analysis of the frequency of changes necessary to ensure MEDICAL DEVICE SAFETY integrity; such as installing network SECURITY patches.

6.3 Implementation of RISK CONTROL measure(s)

> **Text of ISO 14971:2007**
>
> **6.3 Implementation of RISK CONTROL measure(s)**
>
> The MANUFACTURER shall implement the RISK CONTROL measure(s) selected in 6.2.
>
> Implementation of each RISK CONTROL measure shall be verified. This VERIFICATION shall be recorded in the RISK MANAGEMENT FILE.
>
> The effectiveness of the RISK CONTROL measure(s) shall be verified and the results shall be recorded in the RISK MANAGEMENT FILE.
>
> NOTE The VERIFICATION of effectiveness can include validation activities.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

Once RISK CONTROL measures are identified they need to be implemented and their effectiveness verified.

VERIFICATION that the RISK CONTROL measures have been properly implemented and are effective at controlling the RISK is essential for software. Both analysis and testing are likely to be necessary. Key aspects to consider include:

a) TRACEABILITY to assure that all SAFETY-RELATED SOFTWARE ITEMS are identified and all SAFETY-related functionality is specified, implemented, and tested in all relevant VERSIONS and variants (e.g. for different platforms, languages, or MEDICAL DEVICE models) of the software;

b) greater rigour and coverage when testing RISK CONTROL measures including testing under a wide range of abnormal and stress conditions;

c) focus on regression testing RISK CONTROL measures and SAFETY-related functionality when changes are made, even if the changes are not intended to affect SAFETY.

If a process is used that is thought to be rigorous enough to create high-integrity software, the results must still be verified.

ANOMALY information collected during VERIFICATION and validation activities is often useful if root cause analysis was performed and anomaly criticality ratings (see Subclause 9.1 of IEC 62304:2006) associated with SAFETY are tracked and evaluated. ANOMALIES with SAFETY consequences can be evaluated to determine if they were identified in the RISK ASSESSMENT and if identified RISK CONTROL measures would suffice for them once implemented. Data on software ANOMALIES may be used to demonstrate software development PROCESS effectiveness, or to identify aspects of the software development PROCESS that need improvement in order to claim it reduces RISK.

The adequacy of software RISK CONTROL measures may not be as obvious as the adequacy of hardware RISK CONTROL measures. For this reason when dealing with software, one should consider if the documentation of the RISK ASSESSMENT requires other formats than traditionally would be required. One way that can be useful is to write "SAFETY cases" (see Annex E).

6.4 RESIDUAL RISK EVALUATION

> **Text of ISO 14971:2007**
>
> **6.4 RESIDUAL RISK EVALUATION**
>
> After the RISK CONTROL measures are applied, any RESIDUAL RISK shall be evaluated using the criteria defined in the RISK MANAGEMENT plan. The results of this evaluation shall be recorded in the RISK MANAGEMENT FILE.
>
> If the RESIDUAL RISK is not judged acceptable using these criteria, further RISK CONTROL measures shall be applied (see 6.2).
>
> For RESIDUAL RISKS that are judged acceptable, the MANUFACTURER shall decide which RESIDUAL RISKS to disclose and what information is necessary to include in the ACCOMPANYING DOCUMENTS in order to disclose those RESIDUAL RISKS.
>
> NOTE Guidance on how RESIDUAL RISK(S) can be disclosed is provided in Annex J.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE and the ACCOMPANYING DOCUMENTS.

RESIDUAL RISK due to software needs to be included in the RESIDUAL RISK associated with the MEDICAL DEVICE at the SYSTEM level. Given the difficulty in estimating the probability of software ANOMALIES, RESIDUAL RISK EVALUATION usually involves determining whether all sequences of events which lead to unacceptable RISK have RISK CONTROL measures to reduce the probability of their occurrence or to limit the SEVERITY of HARM to an acceptable level as defined in the RISK MANAGEMENT plan (see 3.4.1).

Any software ANOMALIES identified (during VERIFICATION and validation activities) that are not corrected should be analyzed to determine if they affect SAFETY-RELATED SOFTWARE (see Subclause 5.8.3 of IEC 62304:2006). If so, RISK from these ANOMALIES needs to be evaluated and used in evaluating the RESIDUAL RISK of any HAZARDOUS SITUATION they can impact.

6.5 RISK/benefit analysis

> **Text of ISO 14971:2007**
>
> **6.5 RISK/benefit analysis**
>
> If the RESIDUAL RISK is not judged acceptable using the criteria established in the RISK MANAGEMENT plan and further RISK CONTROL is not practicable, the MANUFACTURER may gather and review data and literature to determine if the medical benefits of the INTENDED USE outweigh the RESIDUAL RISK. If this evidence does not support the conclusion that the medical benefits outweigh the RESIDUAL RISK, then the RISK remains unacceptable. If the medical benefits outweigh the RESIDUAL RISK, then proceed to 6.6.
>
> For RISKS that are demonstrated to be outweighed by the benefits, the MANUFACTURER shall decide which information for SAFETY is necessary to disclose the RESIDUAL RISK.
>
> The results of this evaluation shall be recorded in the RISK MANAGEMENT FILE.
>
> NOTE See also D.6.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

No additional guidance for software.

6.6 RISKS arising from RISK CONTROL measures

> **Text of ISO 14971:2007**
>
> **6.6 RISK arising from RISK CONTROL measures**
>
> The effects of the RISK CONTROL measures shall be reviewed with regard to:
>
> a) the introduction of new HAZARDS or HAZARDOUS SITUATIONS;
>
> b) whether the estimated RISKS for previously identified HAZARDOUS SITUATIONS are affected by the introduction of the RISK CONTROL measures
>
> Any new or increased RISKS shall be managed in accordance with 4.4 to 6.5.
>
> The results of this review shall be recorded in the RISK MANAGEMENT FILE.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

A rigorous software configuration management PROCESS, including rigorous change control (see Subclause 8.2 of IEC 62304:2006), is essential so that the introduction of a software RISK CONTROL measure is examined carefully for its impact on other parts of the MEDICAL DEVICE (see also Subclause 7.4 of IEC 62304:2006).

ISO 14971 does not prescribe a design and development PROCESS. One effect of this is that when a RISK CONTROL measure has been implemented, ISO 14971 simply requires that it be reviewed to ensure that it has not caused any further HAZARDS. This should not be interpreted as an instruction to examine this question only after the implementation is complete.

For software RISK CONTROL measures, it is especially important that this review should not be left until the software has been implemented. As soon as the software RISK CONTROL measure has been specified, it should be placed under configuration management and reviewed to discover adverse side effects including the inadvertent creation of new_HAZARDS or_HAZARDOUS SITUATIONS.

Implementation of a RISK CONTROL measure that makes the software design significantly more complex may increase the potential for additional software ANOMALIES or cause new HAZARDOUS SITUATIONS. RISK CONTROL measures should be as simple as practicable and should always be subjected to a new RISK ASSESSMENT.

This review should be repeated (as a minimum) after software design and after SOFTWARE SYSTEM test.

6.7 Completeness of RISK CONTROL

> **Text of ISO 14971:2007**
>
> **6.7 Completeness of RISK CONTROL**
>
> The MANUFACTURER shall ensure that the RISK(S) from all identified HAZARDOUS SITUATIONS have been considered. The results of this activity shall be recorded in the RISK MANAGEMENT FILE.
>
> Compliance is checked by inspection of the RISK MANAGEMENT FILE.

Subclause 7.3.3 of IEC 62304:2006 should be considered for inclusion in the completeness of RISK CONTROL when software is a contributing factor in HAZARDOUS SITUATIONS.

7 Evaluation of overall residual risk acceptability

Text of ISO 14971:2007

7 Evaluation of overall RESIDUAL RISK acceptability

After all RISK CONTROL measures have been implemented and verified, the MANUFACTURER shall decide if the overall RESIDUAL RISK posed by the MEDICAL DEVICE is acceptable using the criteria defined in the RISK MANAGEMENT plan.

NOTE 1 For guidance on overall RESIDUAL RISK EVALUATION, see D.7.

If the overall RESIDUAL RISK is not judged acceptable using the criteria established in the RISK MANAGEMENT plan, the MANUFACTURER may gather and review data and literature to determine if the medical benefits of the INTENDED USE outweigh the overall RESIDUAL RISK. If this evidence supports the conclusion that the medical benefits outweigh the overall RESIDUAL RISK, then the overall RESIDUAL RISK can be judged acceptable.

Otherwise, the overall RESIDUAL RISK remains unacceptable.

For an overall RESIDUAL RISK that is judged acceptable, the MANUFACTURER shall decide which information is necessary to include in the ACCOMPANYING DOCUMENTS in order to disclose the overall RESIDUAL RISK.

NOTE 2 Guidance on how RESIDUAL RISK(S) can be disclosed is provided in Annex J.

The results of the overall RESIDUAL RISK EVALUATION shall be recorded in the RISK MANAGEMENT FILE.

Compliance is checked by inspection of the RISK MANAGEMENT FILE and the ACCOMPANYING DOCUMENTS.

The evaluation of overall RESIDUAL RISK requires implementation of all RISK CONTROL measures. This includes the software being evaluated in the context of each different SYSTEM configuration that the software is used in.

The results of SYSTEM test activities (with respect to all software functionalities and hardware RISK CONTROL) should be evaluated in conjunction with the acceptance criteria. All remaining residual software ANOMALIES are to be documented in the RISK MANAGEMENT FILE and should be evaluated to ensure that they do not contribute to an unacceptable RISK (see Subclauses 5.8.2 and 5.8.3 of IEC 62304:2006). Where necessary, the evaluation should be accepted by independent interdisciplinary reviews with clinical/application experts. It may also be necessary to include information in the ACCOMPANYING DOCUMENTS.

8 Risk management report

Text of ISO 14971:2007

8 RISK MANAGEMENT report

Prior to release for commercial distribution of the MEDICAL DEVICE, the MANUFACTURER shall carry out a review of the RISK MANAGEMENT PROCESS. This review shall at least ensure that:

- the RISK MANAGEMENT plan has been appropriately implemented;
- the overall RESIDUAL RISK is acceptable;
- appropriate methods are in place to obtain relevant production and POST-PRODUCTION information.

The results of this review shall be recorded as the RISK MANAGEMENT report and included in the RISK MANAGEMENT FILE.

The responsibility for review should be assigned in the RISK MANAGEMENT plan to persons having the appropriate authority [see 3.4 b)].

Compliance is checked by inspection of the RISK MANAGEMENT FILE.

Clause 6 and Subclause 7.3.3 of IEC 62304:2006 should be considered for inclusion as part of the review of the RISK MANAGEMENT PROCESS.

9 Production and POST-PRODUCTION information

Text of ISO 14971:2007

9 Production and POST-PRODUCTION information

The MANUFACTURER shall establish, document and maintain a system to collect and review information about the MEDICAL DEVICE or similar devices in the production and the POST-PRODUCTION phases.

When establishing a system to collect and review information about the MEDICAL DEVICE, the MANUFACTURER should consider among other things:

a) the mechanisms by which information generated by the operator, the user, or those accountable for the installation, use and maintenance of the MEDICAL DEVICE is collected and processed;

or

b) new or revised standards.

The system should also collect and review publicly available information about similar MEDICAL DEVICES on the market.

This information shall be evaluated for possible relevance to SAFETY, especially the following:

- if previously unrecognized HAZARDS or HAZARDOUS SITUATIONS are present or
- if the estimated RISK(S) arising from a HAZARDOUS SITUATION is/are no longer acceptable.

If any of the above conditions occur:

1) the impact on previously implemented RISK MANAGEMENT activities shall be evaluated and shall be fed back as an input to the RISK MANAGEMENT PROCESS and

2) a review of the RISK MANAGEMENT FILE for the MEDICAL DEVICE shall be conducted; if there is a potential that the RESIDUAL RISK(S) or its acceptability has changed, the impact on previously implemented RISK CONTROL measures shall be evaluated.

The results of this evaluation shall be recorded in the RISK MANAGEMENT FILE.

NOTE 1 Some aspects of POST-PRODUCTION monitoring are the subject of some national regulations. In such cases, additional measures might be required (e.g. prospective POST-PRODUCTION evaluations).

NOTE 2 See also 8.2 of ISO 13485:2003 [1].

Compliance is checked by inspection of the RISK MANAGEMENT FILE and other appropriate documents.

Software RISK MANAGEMENT continues throughout the software LIFE-CYCLE, including the software maintenance PROCESS (see Clause 6 of IEC 62304:2006) and software problem resolution PROCESS (see Clause 9 of IEC 62304:2006).

Clause 6 of IEC 62304:2006 requires the MANUFACTURER to establish software maintenance plans which address the use of PROCEDURES to receive, document, evaluate, resolve and track feedback after the release of the MEDICAL DEVICE SOFTWARE. The maintenance plan(s) are also to address the use of the software RISK MANAGEMENT PROCESS and the use of the software problem resolution PROCESS for analyzing and resolving problems arising after the release of the MEDICAL DEVICE SOFTWARE.

The use of the software problem resolution PROCESS (see Clause 9 of IEC 62304:2006) integrates the RISK MANAGEMENT activities in the investigation of the software problem and the evaluation of the problem's relevance to SAFETY. It is important to involve a multidisciplinary team including clinical experts, software engineers, SYSTEM designers, and experts on usability/human factors engineering in this investigation and evaluation of the problem (see 3.3).

SOUP is also an important aspect of the software maintenance plan and POST-PRODUCTION RISK MANAGEMENT activities. Some SOUP, by its characteristics (e.g. virus protect software), may have frequent updates which the MANUFACTURER should take into consideration for the software maintenance plan(s).

Failures or unexpected results of SOUP and the obsolescence of SOUP (discontinuation of support) may impact the overall RESIDUAL RISK acceptability of the MEDICAL DEVICE. Therefore, it is necessary to implement SOUP monitoring and evaluation activities in the development and maintenance of the SOFTWARE SYSTEM. These activities should address SOUP updates, upgrades, bug fixes, patches, and obsolescence. Actively monitoring and evaluating publicly available ANOMALY lists and information about the SOUP field performance allows the MANUFACTURER to proactively determine if any of the known ANOMALIES result in a sequence of events that could result in a HAZARDOUS SITUATION (see Subclauses 6.1 f), 7.1.2 c), 7.1.3, and 7.4.2 of IEC 62304:2006).

Released SOUP patches or updates from MANUFACTURERS may include additional functionality that is not essential for the SAFETY and effectiveness of the MEDICAL DEVICE. These SOUP updates should be analyzed for excessive components which can be eliminated from the medical software release to avoid unexpected changes which may lead to a HAZARDOUS SITUATION.

As with any SOFTWARE ITEM change, the MANUFACTURER should know which SOFTWARE ITEMS are affected by the SOUP update and perform regression testing (see Subclauses 7.4, 8.2, and 9.7 of IEC 62304:2006).

Annex A
(informative)

Discussion of definitions

A couple of key terms from ISO 14971 are HAZARD and HARM. Understanding these is important for understanding the standard. HARM is physical injury or damage to the health of people, or damage to property or the environment. HAZARD is defined as a "potential source of HARM", and there are (many) causes for the HAZARDS.

According to the definitions in ISO 14971:2007, a HAZARD cannot result in HARM until such time as a sequence of events or other circumstances (including normal use) lead to a HAZARDOUS SITUATION (see Figure A.1). This sequence of events includes both single events and combination of events. Annex D.2 of ISO 14971:2007 provides guidance on HAZARDS and HAZARDOUS SITUATIONS. Annex E of ISO 14971:2007 provides guidance on examples of HAZARDS, foreseeable sequences of events, and HAZARDOUS SITUATIONS.

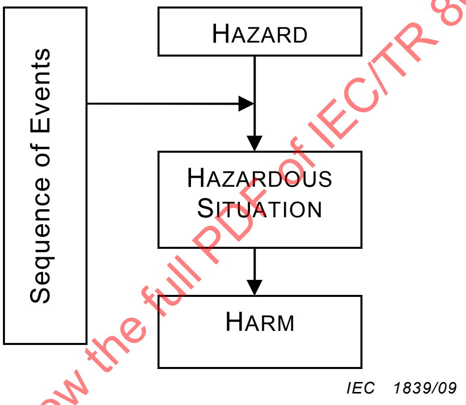


Figure A.1 – Relationship between sequence of events, HARM and HAZARD

A **cause** of a HAZARD or HAZARDOUS SITUATION is any sequence of events, the combination of which might reasonably be expected to result in a HAZARDOUS SITUATION. A given HAZARD might have one, several, or many possible causes (sequence of events).

Unlike heat, electrical energy or suspended masses, software is not itself a HAZARD (a potential source of HARM); contact with software cannot cause injury. However, software may cause a person to be exposed to a HAZARD, in other words it may contribute to a HAZARDOUS SITUATION. Software failures (of any kind) often facilitate the transformation of a HAZARD into a HAZARDOUS SITUATION.

Software contributes to HAZARDOUS SITUATIONS primarily by contributing to the sequence of events which exposes the HAZARD and creates a HAZARDOUS SITUATION.

Table A.1 – Relationship between HAZARDS, foreseeable sequences of events, HAZARDOUS SITUATIONS and the HARM that can occur

HAZARD	Foreseeable sequence of events involving software	Initiating cause	HAZARDOUS SITUATION	HARM
Electrical energy	Software output controlling current applied by an eye implant is too high	(1) Software algorithm has limitations. (2) Software is correctly specified but has an ANOMALY	Excessive current applied to patient's eye by implant	Serious burns Loss of vision
Loss of clinical function	Software fails to provide life-supporting function for which it is designed	(1) Software is unable to handle unusual input data. (2) Software fails to detect incorrect setup of equipment (3) Hardware does not provide sufficient resources to support timely operation of software	(1) Device cannot provide treatment when needed (2) Device operates when not correctly setup (3) Device fails to warn of life-threatening condition	Deterioration of patient's condition Death
Neglect of patient by clinical staff	Software inputs and outputs confuse or mislead the user	(1) Software-human interface confuses the user (2) Software outputs exceeds the user's capacity to respond (3) User does not understand the software's limitations	(1) Mistreatment of patient (2) Lack of timely response to emergencies (3) Over-reliance on software replaces personal initiative	Deterioration of patient's condition Death

Annex B
(informative)

Examples of software causes

Table B.1 lists functional areas of software often related to HAZARDS and provides examples of conditions that are potential causes for HAZARDS. It also provides example questions to ask during software development that can help lead to improved RISK CONTROL. Some of the information in this table may not apply to all MEDICAL DEVICE SOFTWARE. The relevancy for specific MEDICAL DEVICES depends on the INTENDED USE of the MEDICAL DEVICE, the SYSTEM-level design of the MEDICAL DEVICE, the role of the software in the MEDICAL DEVICE, and other factors. This table is intended only as a starting point.

This table is not exhaustive but is an aid to the thought PROCESS used for developing safe and effective software.

Table B.1 – Examples of causes by software function area

Software function area	Example causes for HAZARDS	Questions to ask
Alarms and alerts:		
Priority	Lower priority alarms mask the display or audible output of higher priority alarms Critical alarm does not persist	Do specifications identify how the SYSTEM reacts to multiple alarm conditions? Are there multiple levels of alarms? Do higher level alarms override the audio for lower level alarms? Should any of the alarms persist until the user can acknowledge the alarm?
Protective actions	Protective actions are not clear for each alarm condition or alarm category Removal of protective actions once alarm clears are not specified	Does the protective action create a usability problem, i.e. can the user safely navigate from the protective action?
Shutdown/safe mode/ recovery	Safe mode action is not adequate Safe mode action creates new HAZARDS	Is safe mode action appropriate for INTENDED USE? Has the clinical staff reviewed safe mode scenarios? Is safe mode state apparent to the user?
User interface	Crowded or poor user interface masks the “real” alarm condition Actions to take in response to the alarm are not clear	Has the clinical staff reviewed protective measures for usability?
Log	Persistent error is signalling a pending failure Logged alarms are associated with wrong patient	Are detected errors logged? Is log large enough? Is log storage reliable? How is log cleared? Is the user aware when log is cleared?
Audibility	Background noise suppresses alarm sound Alarm volume so loud that operators find alternative means to disable alarm Audio SYSTEM has failed and user is unaware of failure	Has INTENDED USE environment been considered for audible alarm design? Have users been involved in developing requirements for user interface design? How is audio SYSTEM verified per power up or per patient?

Table B.1 – Examples of causes by software function area *(continued)*

Software function area	Example causes for HAZARDS	Questions to ask
Critical power cycle states		
Data integrity	Non-volatile write in progress at shutdown Critical parameters are not preserved to resume treatment on power cycle Critical parameters are inadvertently overwritten by latent software ANOMALY during operation	What happens to a memory write that was in progress when power was lost? Is the software made aware of impending power loss? Is non-volatile storage verified on power up? Are critical parameters checked before use?
Reset	Failure to synchronize with component after an unexpected reset Persistent resets go undetected but could be signalling a pending failure	Is reset being used as a RISK CONTROL measure? Will input/output control be compromised during reset cycle? Is user made aware of resets?
Recovery	The MEDICAL DEVICE is unavailable for INTENDED USE while power-on initializations are occurring Non-volatile failures at power-on – what do you do? User is not aware that critical setting is restored to factory defaults	Is recovery time a SAFETY issue? Is availability of the MEDICAL DEVICE a SAFETY issue? How is non-volatile storage impacted by failsafe protective measure?
Power modes	Audible functions or other critical user interface functions are not available during low power states Transition to low power state inadvertently disables critical interrupt	Are any RISK CONTROL measures compromised during low power modes? Has software recovery from low power modes been considered as a possible start-up state for VERIFICATION and validation activities?
Critical user controls/Usability		
Adjustment/navigation/ selection	User interface software PROCESS handles value change, but control PROCESS never gets the new value	Is user notified if adjustment is made to a new value but never selected or confirmed? Should the parameter being adjusted require a two-step operation for change?
Data entry	User input is an out of range value User inputs in-range but unintended value	Should user be prompted for confirmation? Does software check data entry for validity? Does it require supervisory user login to confirm highly critical inputs or error overrides?
Accessibility	Concealed shutoff control Touch screen controls do not function with surgical gloves	How many "layers" must user navigate to access SAFETY- related function?
Screen changes	Alarm "automatically" causes display to switch screens	Have all automatic screen switches been evaluated?
User interface design	Use of colours without regard for common forms of colour blindness User can't determine which condition causes alarm	How will colour blind operator interpret error message? Have users been involved in developing requirements for user interface design?

Table B.1 – Examples of causes by software function area *(continued)*

Software function area	**Example causes for HAZARDS**	**Questions to ask**
Display		
Diagnostic images	Orientation reversal Patient incorrectly associated with patient	Is there a technique being used to ensure correct orientation of image? How are images associated with patient?
Diagnostic waveforms	Improper display filter Aliasing, distortion, scaling errors, timebase errors, lossy compression	What frequency content is required for display? Has the clinical staff reviewed that requirement? Has display filter been fully characterized, i.e. what is rejected and what is passed, over full range of inputs?
Hardware controls		
Algorithms Motor control	Integral windup, aliasing, timing, overflow, port error (serial port/parallel port)	What is the sampling rate? If PID control, is integral gain limited? Has algorithm been characterized over the full variation of manufactured hardware? If feedback control, what checks are made to the validity of the feedback signal? Have all data types been evaluated for the microprocessor and compiler in use?
Applying energy	Not checking all energy "gates" initially and continuously during therapy SAFETY SYSTEM has failed but user is not aware	Are all assertions continuously verified on a scheduled basis? Could a "common mode" error exist in therapy control software and SAFETY monitor software? Are SAFETY monitors verified per power up or per patient?
Discrete	Stuck bit Changes in bit go undetected due to polling interval	Does software detect that bit is stuck (never changes)? Has polling rate been discussed with SYSTEM or hardware engineer?
Calibration/Self-test Range checks Assay calibration (Software specific calibration)	Poor instructions for use cause the user to fail to correctly calibrate the MEDICAL DEVICE which results in erroneous calibration constant Auto-zeroing operation done with non-zero signal, i.e. unexpected pressure in cuff or in line, or force on transducer	Does software perform a reasonableness or validity check of calibration values, i.e. slope or offset? Is user aware of auto-cal or auto-zero?
Hardware fault detection	Hardware fault is detectable but never reported to user The MEDICAL DEVICE continues to be used in this condition Hardware fault occurs after power-up software only checks for hardware fault at power-up	Are all hardware faults reported to user? Should hardware fault be checked at power-up, before each treatment or session or on a continuous basis such as once per second?
Self-cleaning	User aborts cleaning or disinfection PROCESS mid-cycle	Does software enforce completion of cycle? Can software detection of incomplete cleaning or disinfection cycle be defeated?

Table B.1 – Examples of causes by software function area *(continued)*

Software function area	Example causes for HAZARDS	Questions to ask
Fluid delivery	Detecting improper calibration Not checking all fluid "gates" initially and continuously during therapy	Are all assertions continuously verified on a scheduled basis? Can SAFETY SYSTEMS be defeated, i.e. pump operated without tube in SAFETY clamp?
Life support	Safe state never defined Out of many shutdown paths, one does not disable interrupts No backups for life support functions	Have SAFE STATES been defined and analyzed thoroughly including impact of delay of treatment and safe shutdown sequences for the range of target populations (e.g. adults, neonates)? Can software support a "limited functionality" mode and inform user of situation?
Monitoring		
Decision	Common Mode error in monitoring software Race condition causes incorrect decision result	Has therapy control and therapy monitor software been developed independently? Has software design eliminated or minimized possibility for race condition for this decision point?
Deactivation	Control SYSTEM unaware that monitoring SYSTEM has shut down subsystem Networked SYSTEM logging parameter that is deactivated at MEDICAL DEVICE	Is control subsystem aware of monitor subsystem actions? How are deactivated parameters communicated to user or networked SYSTEMS?
Display	Displayed value is not being updated but user is unaware Display writes are performed at two or more priority levels.	How is the user made aware of "frozen" display? Is video "context" saved before pre-emption?
Measurement	Wrong data acquisition timing or sampling rate	Is sampling rate appropriate for frequency content of signal? Is the measurement value stored in consistent units throughout software layers?
Interfaces		
Bad argument passing	Function passes value in microlitres but driver expects value in millilitres Bad pointer passed Pointer to volatile memory passed and values are lost before processing	Does each software function verify passed parameters? Does software language support more robust type checking? Is software designed with consistent units for values throughout the software package? Are arguments modified at higher priority processing layer?
Network	Software enters endless loop waiting on response from host computer MEDICAL DEVICES on network are given identical "names" resulting in data loss Networking data processing dominates CPU cycles starving SAFETY or INTENDED USE functions	Has software been designed to tolerate any physical network connection condition? Can remote connection degrade SYSTEM performance by repeatedly sending commands or bogus data? Does the MEDICAL DEVICE check that the network name is not already in use?

Table B.1 – Examples of causes by software function area *(continued)*

Software function area	Example causes for HAZARDS	Questions to ask
Data		
Clinical information	SYSTEM accesses wrong patient RECORD and user interface display does not make it apparent SYSTEM stores data from patient in wrong archive	Can there be display of multiple independent identifiers to put the user in the loop of detecting mix-ups? Can critical identifiers be embedded with actual data as a cross-check?
Reports	Report provides incorrect data or identifies it in the wrong sequence or without units	What reports will be used for clinical purposes? What is the SEVERITY of HARM if the data is incorrect? How likely is it that a clinician would notice the problem?
Databases	Data corruption due to side-effects from SYSTEM level failures or SOUP	How can data corruption be detected prior to use of the data? Can this be done with each use instead of only at start-up?
Diagnostic		
Decision-making	Artefact detection indication suppresses asystole indication on the display	Has the alarm indication hierarchy been thoroughly reviewed and also reviewed with clinical staff?
Data conversion	Arithmetic precision errors result in invalid result Algorithm uses or displays incorrect units	What arithmetic precision is required? How should mathematical formulas be coded to ensure adequate precision?
Automated preventive maintenance	Background diagnostic modifies data temporarily but while application code is retrieving the data for actual use Background diagnostic interferes with proper timing	Are application PROCESSES locked out during diagnostics at appropriate times? Are diagnostics locked out during critical timed cycles?
SECURITY		
Configuration options	No or inadequate protection of access to critical configuration parameters or data	What data is critical and should not be modifiable by the user or should require supervisory authorization to do so? Is an audit trail needed?
Functional access	No or inadequate protection of access to controls of therapy or instrument operation	Should operators be required to login before operation? Can patients inadvertently operate the MEDICAL DEVICE?
Interface access	No or inadequate protection from data and commands submitted through communications interfaces or networks	What should be allowed remotely? Should assumed controls at the remote SYSTEMS be relied on and if so why?

Table B.1 – Examples of causes by software function area *(continued)*

Software function area	Example causes for HAZARDS	Questions to ask
Performance		
Capacity/load/response time	Critical timing is affected during peak load Sequence of transactions/inputs/outputs is affected or data is lost under peak loads Motor control is affected under peak SYSTEM loads	During peak load or when capacity limits are reached will data or timing be lost or affected in undetectable ways? Will inputs and outputs be queued up in a correct deterministic sequence under peak loads? Have critical functionality and RISK CONTROL measures been tested under these stress conditions? Have RISK CONTROL measures been implemented to detect limits? Can interrupts be set to segregate critical time constrained functionality from other functionality?

In addition to the potential software causes of HAZARDS in SAFETY-RELATED SOFTWARE functionality shown in Table B.1, there are some types of software causes that may result in side-effects in software unrelated to the software where the failure occurred. If a certain type of software defect can have unpredictable effects on SAFETY-related parts of the software, then this potential should be identified and a RISK CONTROL strategy and specific RISK CONTROL measures developed.

Table B.2 identifies examples of these potential software causes for HAZARDS and some possible RISK CONTROL measures to consider for each example. Requirements-based SYSTEM testing is often ineffective at identifying these types of software causes or verifying the RISK CONTROL measures associated with them and the effectiveness of the RISK CONTROL measures. The columns on the right of the table provide guidance on the type of static or dynamic VERIFICATION method(s) that might be appropriate for each example. Table B.3 provides examples of static or dynamic VERIFICATION method(s).

Table B.2 – Examples of software causes that can introduce side-effects

	VERIFICATION types:	Inspection	Test (unit, integration)	Analysis: static / dynamic / timing
Software causes	**RISK CONTROL measures**			
Arithmetic				
Divide by zero	Run-time error traps, defensive coding	✦	✦	D
Numeric overflow/underflow	Range checks, Floating-point data representations	✦	✦	D
Floating point rounding	Robust algorithms	✦		
Improper range/bounds checking	Defensive coding	✦	✦	S
Off-by-one (OBO)	Defensive coding	✦	✦	

Table B.2 – Examples of software causes that can introduce side-effects *(continued)*

	VERIFICATION types:	Inspection	Test (unit, integration)	Analysis: static / dynamic / timing
Software causes	**RISK CONTROL measures**			
Hardware-related				
EEPROM usage: long access times, wear-out	Use special access modes (Page/Burst mode), write only when data changes, cache writes and update EEPROM only at power-loss	✦		T
CPU/Hardware failures	Power-on CPU Check, program image CRC check, RAM test, Clock check, watchdog check, non-volatile storage check, timeouts and reasonability checks on hardware responses, short-to power/ground checks on sensors, test sensor response with known signal		✦	
Noise	Debounce digital inputs, filter analogue inputs, all interrupts – used and unused – have interrupt service routines (ISRs)			
Peripheral interface ANOMALIES	Start-up delays for ADC/DAC, verify timing and other interface requirements are always met, reasonability checks	✦		T
Timing				
Race conditions	Identify and protect (lock) shared resources during updates, Implement a single, non-re-entrant PROCESS to handle all accesses to the shared resource, shared resource analyses			S
Missed time deadlines	Specified timing requirements, appropriate real-time design algorithms, eliminate priority-inversion and deadlock issues by design, avoid non-deterministic timing constructs, verify all timing assumptions in completed code NOTE Non-deterministic timing constructs include: recursive routines, waiting for a response from hardware, dynamic memory allocation, and virtual memory (swapping memory pages to/from disk)			T
Missed interrupts	Simple interrupt ARCHITECTURE (fewest priorities), ISRs are short and fast, disabling of interrupt(s) is infrequent and of short duration, appropriate real-time design			T
Excessive jitter in outputs	ISRs are short and fast, avoid non-deterministic timing constructs, appropriate real-time design, update all outputs at beginning of periodic TASK or in high-priority ISR			T
Watchdog time-outs	Determine longest time between watchdog timer resets and set time-out appropriately, set time-out to longest period possible while still avoiding HAZARDOUS SITUATION			T
Moding				
Abnormal termination	Exit traps, run-time error traps, appropriate watchdog timer design, validity checks on all program inputs, power-up self test, remove all “debugging” code and other non-production features from program before release, insure “special” modes (service, manufacturing) can not be entered inadvertently			D
Power loss/recovery/sequencing problems	Power-up checks: CPU, program image CRC, RAM, clock, watchdog, non-volatile storage, peripherals, etc., appropriate state design, initialization of time-averaged values, re-initialization of peripherals, store/recover SYSTEM state from non-volatile storage, external voltage monitor / reset circuitry			
Start-up/shut-down ANOMALIES	Power-up checks (see above), proper initialization of peripherals and data, appropriate non-volatile memory usage, appropriate state design			
Enter/exit of low power modes	Appropriate interrupt design	✦		T

Table B.2 – Examples of software causes that can introduce side-effects *(continued)*

	VERIFICATION types:	Inspection	Test (unit, integration)	Analysis: static / dynamic / timing
Software causes	**RISK CONTROL measures**			
Data issues				
Data corruption	Shadow RAM, block CRCs or checksums, encapsulate data access through functions, minimize global data, keep data structures simple, be aware of how compiler aligns data in structures, avoid casts, (also see “errant pointers” and “intermediate data” below)			S
Resource contention issues	Shared resource analysis, (see also “race conditions” above)			
Errant pointers	Defensive coding: test for validity before de-referencing, use a strongly-typed language, minimize the use of pointers, avoid pointer casts	✦	✦	S
Data conversion errors: type-casting, scaling	Avoid type-casts, use floating-point representations	✦	✦	S
Incorrect initialization	Pre-initialize time-averaged variables, clear all data memory to “0” at power-up	✦	✦	S
Averaged data out of range	Insure enough samples are taken before calculating average (especially at power-up) or pre-initialize average to a known (or last) good value	✦		
Rollovers	Reasonability checks	✦	✦	
Volatile data	Verify volatile storage class is used for all data changed by hardware, ISRs, or different priority TASKS	✦		S
Unintended aliasing	Sample data at least 2x faster than the largest frequency component of the signal (Nyquist criteria), limit the bandwidth of the signal			
Use of intermediate data NOTE A sequence of calculations should never be performed on a global (or shared) variable when the calculation can be interrupted or pre-empted. Instead, perform all the calculations on a temporary variable and update the global variable with a single, un-interruptible instruction.	Insure that any data which is assumed to be synchronous in time is updated all at once, shared resource analysis	✦		
Interface issues				
Failing to update display	Continuous update instead of event-driven			
Human factors: MISUSE	Logs for re-constructing scenarios, context-sensitive help, simple user interface design			
Network issues, e.g. multi-user	Load testing			T
Hardware/software configuration errors / wrong drivers	Software development PROCESS, configuration management tools			
Bad patches/updates	Program image CRC and VERSION check at power-up, check protocol revisions, expiration dates			
SOUP failure modes : Hangs/doesn’t return, Locks interrupts too long, etc.	Examine SOUP Errata, Robust design (e.g. timeouts on all blocking calls), Lock memory pages which are used frequently or used by ISRs, Use only the SOUP features required: remove all others	✦		T
Virus	Virus Checkers			
Browser/web incompatibility	Integrate VERSION checking at start-up, compatibility testing			

Table B.2 – Examples of software causes that can introduce side-effects *(continued)*

Software causes	RISK CONTROL measures	VERIFICATION types: Inspection	Test (unit, integration)	Analysis: static / dynamic / timing
Miscellaneous				
Memory leaks	Avoid dynamic allocation of memory	✦	✦	D
SYSTEM deadlocks	Simple locking strategies (PROCESS can only have one resource locked at a given time), deadlock analysis			S
Re-entrancy	Ensure that all functions, including those in third-party libraries, which are called from interrupts (or multiple TASKS of different priorities) are designed to be re-entrant			D
Stack overflow	Run-time stack guards, high-water marks, stack analysis			S
Logic errors/syntax	Use source code analysis tool (such as Lint) and/or max. compiler warning level Dual DIVERSITY and cross-checks at critical control points	✦	✦	S
Infinite loops	Loop counters, loop timeouts, watchdog timer	✦	✦	
Code corruption	Power-up and run-time program image CRC check			
Dead code	If not removed insert an error check that will alarm or perform a safe shutdown if dead code (in custom or for off-the-shelf software components) begins to execute.			D
Incorrect conditional code	Ensure conditional compilation is used appropriately and only when necessary	✦		
Unintended macro side-effects	Use parenthesis around all macro parameters			S
Resource depletion	Stack, heap, and timing analyses			T
Incorrect alarm/alert prioritization	Stress testing			
Unauthorized features ("gold plating", "back doors", etc.)	Requirements and design reviews, trace matrices			
Incorrect order of operations/precedence	"Bread-crumbing", Called from tracking			
SAFE state	Independent monitor			

There are many methods available to facilitate assurance that RISK CONTROL methods are likely to perform as intended, some more resource-intensive than others. No single method is sufficient. Some of these methods are identified below in Table B.3.

Table B.3 – Methods to facilitate assurance that RISK CONTROL methods are likely to perform as intended

Static analysis	Dynamic testing	Modelling
Walkthroughs	Functional testing	Environmental modelling
Design reviews	Timing and memory tests	Timing simulation
Sneak circuit analysis	Boundary value analysis	Use case/user workflows
	Performance testing	
	Stress testing	
	Statistical testing	
	Error guessing	
	Thread-based testing	
	Use-based testing	
	Cluster testing	

Testing should account for a variety of types of tests (e.g. stress, boundary, timing, power failure, fault, SOUP failure, etc.) to assure SAFETY-RELATED SOFTWARE is tested under an adequate range of conditions rather than focusing exclusively on requirement-based testing.