Implementing a Medical Device Software Risk Management Process by ISO 14971 in compliance with Agile Principles

Tuomas Granlund

University of Tampere
School of Information Sciences
Degree Programme in Computer Sciences
M. Sc. thesis
Supervisor: Timo Poranen
December 2016

University of Tampere
School of Information Sciences
Degree Programme in Computer Sciences
Tuomas Granlund: Implementing a Medical Device Software Risk Management Process by ISO 14971 in compliance with Agile Principles
M.Sc. thesis, 66 pages and 4 index pages
December 2016

The development of medical device software is strictly regulated by competent authorities. In addition to producing significant medical benefits, the medical device software can be a potential source of serious safety hazard to patients or healthcare professionals. The International Standard ISO 14971 was created to minimize the risks related to treatment of patients with the medical devices.

Although agile software development has become a widely used method for developing software products, medical device manufacturers and regulators have been uncertain whether these practices are appropriate for the regulated environment. The purpose of this thesis is to research similarities and differences between ISO 14971 risk management process and agile principles. Furthermore, the aim is also to provide guidance and produce practical ideas for the implementation of the risk management process that meets the regulatory requirements and follows agile values and principles.

The risk management standard ISO 14971 was thoroughly analyzed in order to find all process requirements. Similarly, the agile practices were studied through the Agile Manifesto and other essential resources of the field. The synthesis of the two concepts was produced based on the information gathered.

The ideas produced in the research are presented as an example development process model which can be used as a reference implementation. The relatively high abstraction level of the model secures the generalizability of the research.

When designing the risk management process implementation, it is essential to thoroughly understand the goals and principles of the regulatory framework. By following the guidance and instructions provided in this thesis, medical device software manufacturers should be able to create the applicable risk management process and to claim conformity to ISO 19471.

Key words and terms: Medical device software development, Software development, Quality management systems, Risk management, Agile methods.

Contents

1. Introduction

Manufacturing of medical devices is strictly controlled by national authorities. The safety of people and healthcare are among the main concerns of governments. In the EU region, there is a regulatory framework run by the European Commission and it consists of three core directives. The European Commission has ongoing international cooperation between the most important interest parties and is also participating to a multilateral framework, the International Medical Device Regulators' Forum (IMDRF) [European Commission, 2016]. In Finland, medical device manufacturing is controlled by the National Supervisory Authority for Welfare and Health Valvira [Valvira, 2009].

One of the core directives is the *Council Directive 93/42/EEC on Medical Devices* (MDD) [1993]. MDD is the most crucial directive that regulates the medical device manufacture process in the EU area (excluding only active implantable and In Vitro devices that are being addressed in specific directives) [European Commission, 1990, 1998]. A company marketing their medical devices in the EU must prove compliance with the MDD [European Commission, 1993] and in the US market regulations of The Food and Drugs Administration (FDA) must be met. One of the most essential ways to meet these regulations is to implement the requirements of the Quality Management System (QMS) as defined in ISO 13485 standard. It is a QMS standard for medical devices and it is harmonized with MDD in the EU and it is also accepted by the FDA [2012]. It also defines the risk management process that is required within the medical device manufacturing process by making normative reference to ISO 14971.

1.1. Risk management regulation

The International Standard ISO 14971 was prepared with a mandate by the European Commission. It provides a means of conforming the risk management requirements of MDD [ISO, 2012]. It is a de facto standard and commonly recognized as one of the best processes to implement the risk management process for medical devices, taking care of the whole lifecycle of the medical device in question [Catelani et al., 2011]. Faris [2006] estimated that over half of the medical devices on sale in the US market contained some form of software and it is safe to assume that the ratio has not decreased since, on the contrary. However, MDD does not differentiate between the physical medical device (with or without software) or medical device software that is used as a medical device and exists without any specific physical device - the software is considered to be a medical device as such [European Commission, 1993]. Therefore, the same medical device standards and regulations must be used when manufacturing software for use in the medical domain.

1.2. Agile development success

At the same time, agile software development has grown increasingly popular over the last decade [VersionOne, 2016]. Agile practices have pervaded from small pioneering companies to the whole field of software development organizations. The growth has definitely been rapid, considering that the agile movement is practically only 15 years old. However, the roots of the movement are considerably older [Poppendieck & Poppendieck, 2003].

The importance and impact of agile philosophy cannot be denied. The number of software development organizations using agile practices continues to increase every year and the phenomenon is growing globally. According to VersionOne survey [2016], besides North America and Europe, agile is getting more practitioners also in Asia, South America, Oceania and Africa.

The popularity of agile is based on the belief that adopting agile practices helps organizations to succeed. The promise of agile is accelerated development, increased product quality, customer satisfaction and ability to respond to change. Furthermore, the increased development process visibility can be seen as an advantage. Besides gaining more popularity inside the software development domain, agile has become a largely used methodology also in other complex development environments. This evolvement is natural from the historical viewpoint since agile has taken strong influence from lean manufacturing industry.

1.3. Barriers to agile adoption in regulated environment

Although adopting agile practices is generally seen as a desirable goal, there can be certain obstacles to overcome. It is common that organizational culture prevents development teams from being truly agile when top management is not giving consistent support for new ways to work. Besides the company culture, existing development framework and lack of experience in agile practices can become a complication [VersionOne, 2016].

Highsmith [2004] states that every development organization, team and project has some compliances to meet. These compliances can be seen as obstacles to agile adoption, although the restrictive impact of certain compliance depends highly on the type and the complexity of the obstacle in question. It is evident that the medical device development regulatory framework produces a great deal of complicated compliances.

McHugh et al. [2014] studied the perceived barriers to following agile practices when developing medical device software. The questionnaire-based survey part of the study revealed that exactly regulatory control and traceability issues were seen as barriers for agile adoption by participant organizations. However, McHugh et al. [2014] argues that, based on the study, these barriers are only superficial and no actual external barriers

to adopting agile practices exists. Nevertheless, the survey acts as an evidence that guidance for agile adoption in medical device software domain is needed.

AAMI TIR45 Technical Information Report [2012] provides practical guidance for agile adoption for medical device software development. The guidance given in the document is based on the information from the actual agile implementations in the medical device software domain. While AAMI TIR45 is an important and highly recommended reference for agile adoption, it is mainly focused on software lifecycle processes requirements of IEC 62304, therefore the risk management requirements of ISO 14971 are not addressed with great level of detail. Conversely, previous studies of ISO 14971 risk management, for example Schmuland [2005] and Flood et al. [2015], do not address aspects of agile development.

The risk management is a key activity for organizations developing medical device software. However, it is a difficult and time-consuming task to define a suitable process to meet the regulatory requirements [Flood et al, 2015]. In conclusion, practical guidance for risk management process implementation and agile adoption is needed.

1.4. Research objectives, methods and outcomes

The aim of this research is to produce practical ideas how to implement the requirements of ISO 14971 while consistently following agile principles. The agile principles and the risk management process are examined. The research aims to discover the principles of integration between these two concepts and should also gain knowledge about the suitable integration.

The research questions are:

1) How to implement the requirements of the ISO 14971 in compliance with the agile principles?
2) Are there any non-compliances between the ISO 14971 process model and agile principles?
3) If there are non-compliances, how to solve them? Should the agile software development framework be tailored in some way?

The research is a case study of the risk management process standard ISO 14971. The standard is analyzed thoroughly and all the requirements are gathered. The risk management process is sectioned to appropriate phases and inside the phases to more detailed steps. All the requirements of the standard should be satisfied by following these defined steps.

Simultaneously, the agile development principles are examined from the general point of view. Agile software development is a general topic and considerable amount of quality literature can be found. Therefore, this research concentrates on seminal or otherwise essential works of the field. It is important to limit the irrelevant details of some specific agile software development method. Instead, the focus is on the general

principles and guidelines of the agile philosophy. This approach secures the generalizability of the research. On the other hand, it is essential to notice that practical quality management system and risk management process implementations can only be made on a case by case basis [Schmuland, 2005].

The rest of this thesis is arranged as follows. Chapter 2 presents the background of regulatory framework behind the risk management process of ISO 14971. Chapter 3 analyzes the process requirements of ISO 14971 and addresses also the risk management requirements of IEC 62304. Chapter 4 defines the agile grounding and perspective of this study. Chapter 5 studies the similarities and differences between agile and regulatory perspectives and justifies the rationale behind agile methodology adaptation. Chapter 6 presents the practical guidance and a reference development model for risk management process and agile development synthesis. Chapter 7 concludes the thesis.

2. Regulatory perspective

The manufacturing of medical devices and medical device software inside EU region is strictly controlled by EU directives. The directive covering the medical device software is *Council Directive 93/42/EEC on Medical Devices* [MDD, 1993], although other directives may also apply, depending on the type of the software product. For example, *The European Parliament and Council Directive 2011/24/EU* controls the use of patient's personal data [European Commission, 2011]. As the directives require EU countries to achieve a certain result, they must be properly implemented in national legislation [European Union, 2016]. The main goal of the regulatory framework is to protect the health and safety of patients and medical device users by guarding public against unsafe medical products.

It is obvious that it would be practical if the similar medical device regulations applied in all countries. This kind of harmonization work has been made first by the Global Harmonization Task Force (GHTF) and then by the International Medical Device Regulators Forum (IMDRF). The organizations like the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) are producing international standards from which notable part is harmonized inside EU. These standards are called harmonized standards in EU region and recognized standards in USA and Canada.

MDD provides only general requirements for the medical device manufacturing process while the details of the implementation are left to manufacturers to define [AAMI TIR45, 2012]. MDD does not demand manufacturers to use the harmonized standards, however strong justification must be provided if standards are not followed. In practice, the standards form a very useful tool to prove that the regulations of MDD have been implemented. The standards can also provide some ready-made solutions for implementation of MDD while allowing manufacturers to create their own practices to align with regulations.

The harmonized standards that must be currently taken into consideration in the medical device software manufacture process are presented in Figure 1 and addressed with more detail later in this chapter. One of the most central theme in these standards is that the medical device software must be safe and effective. The standards also cover the post-production phase of the software with the concepts of post-market surveillance and incident reports. Overall, the regulatory requirements and standards provide a framework for medical device software development practices while leave a responsibility to tailor the details of the development process for the manufacturer.

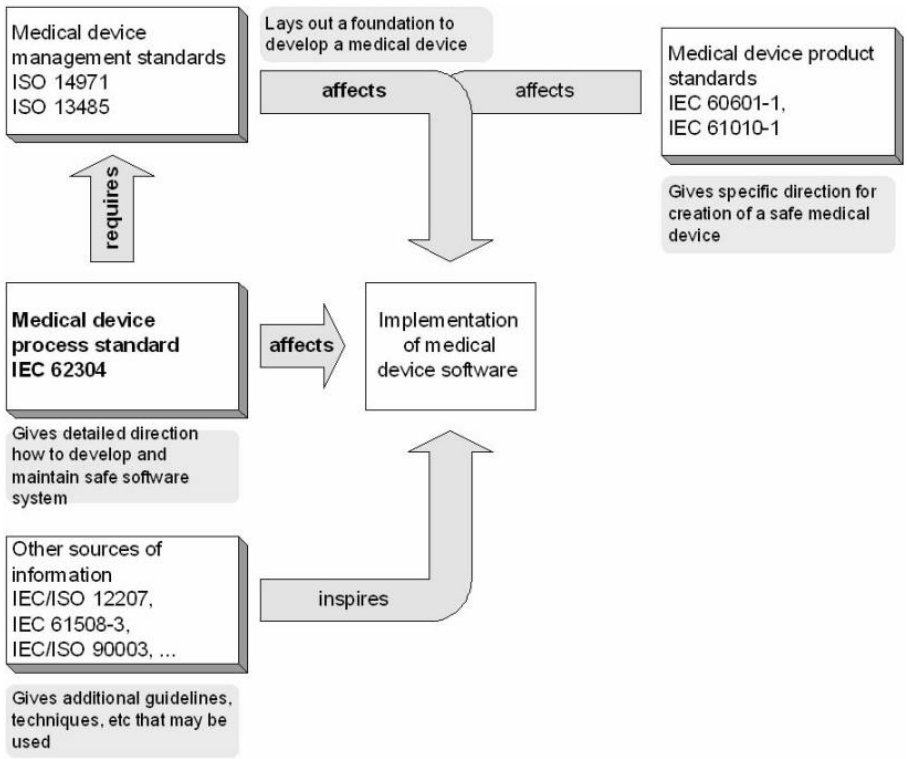


Figure 1. Relationship of key medical device standards to IEC 62304 [IEC 62304, 2006].

2.1. ISO 13485

ISO 13485:2003, Medical devices – Quality management systems – Requirements for regulatory purposes is a widely acknowledged standard that defines requirements for the quality management system for the design and manufacture process of medical devices. ISO 13485 is heavily based on ISO 9001, which is maybe the most commonly known quality management system standard, but it is a stand-alone standard tailored for application to the medical device sector and its requirements are specific to medical device domain. ISO 13485 states that the adoption of the quality management system should be a strategic decision of an organization and the implementation is influenced by varying aspects [ISO, 2003]. It also adds an obligation for organization to establish a risk management system.

It is worth to notice that some of the definitions provided in ISO 13485 can be superseded by local regulations. Furthermore, it excludes some of ISO 9001 requirements that are not suitable as regulatory requirements in the medical domain. For this reason, organizations implementing ISO 13485 must also conform to all the requirements of ISO 9001 if they want to claim conformity to ISO 9001 [ISO, 2003]. ISO 9001 is not sufficient for manufacture process of medical devices, but it can be otherwise beneficial for example in competitive tendering. Broadly, the requirement level of ISO 13485 is more demanding than ISO 9001.

ISO 13485 defines general requirements for the quality management system. The principle of the standard is that everything must be documented and what is not documented does not exist [Ståhlberg, 2015]. Terms "if appropriate" and "where appropriate" are used in standard several times. When these terms are used in conjunction with a regulation, the regulation in question is deemed to be appropriate unless the organization can document a justification otherwise. If the regulation is specified to be "documented", it must also be implemented and maintained.

Furthermore, quality manual and control of document and record requirements are interpreted. The purpose of these documents and records is to be able to demonstrate that predefined processes of the standard and the quality management system itself have been followed on all occasions. Traceability is one of the most central subjects in the quality management system. All the occurred nonconformities must be handled with corrective and preventive action process.

ISO 13485 emphasizes management responsibility of the organization. Management commitment, responsibilities, authorities, quality policy and internal communication are regulated topics. The quality management system must have a top management representative who is responsible for the system and the system must be reviewed by the management at planned intervals. As a part of the quality management system, organization must be aware about the laws, regulations and requirements that apply to operation of the organization. The personnel whose work affects the quality of the end product must be competent and the competence must be evaluated.

Other topics of the standard are external supplier purchasing process, production and service provision, identification of end product and control of possible monitoring and measuring devices. The production process of the medical device must be carefully planned and plan must be followed accordingly during the production phase. Also monitoring the quality of the quality management system itself must be done by gathering feedback from the appropriate feedback system and by performing internal audits. Design and development controls of the standard can be excluded from the quality management system if other effective regulatory requirements permit.

One of the interesting differences between ISO 9001 and ISO 13485 is that the later has eliminated the commitment to continually improve the quality management system in favor for continually maintain the effectiveness of the system. ISO 13485 [2003, Annex B, 8.1] states that "the objective of medical device regulations is the maintenance of the effectiveness of the quality management system to consistently produce medical devices that are safe and effective, not the continual improvement of the quality management system". However, as noted previously, if the organization wishes to meet also the requirements of ISO 9001, it has to make a commitment to continuous improvement. The process model of ISO 9001 is presented in Figure 2.

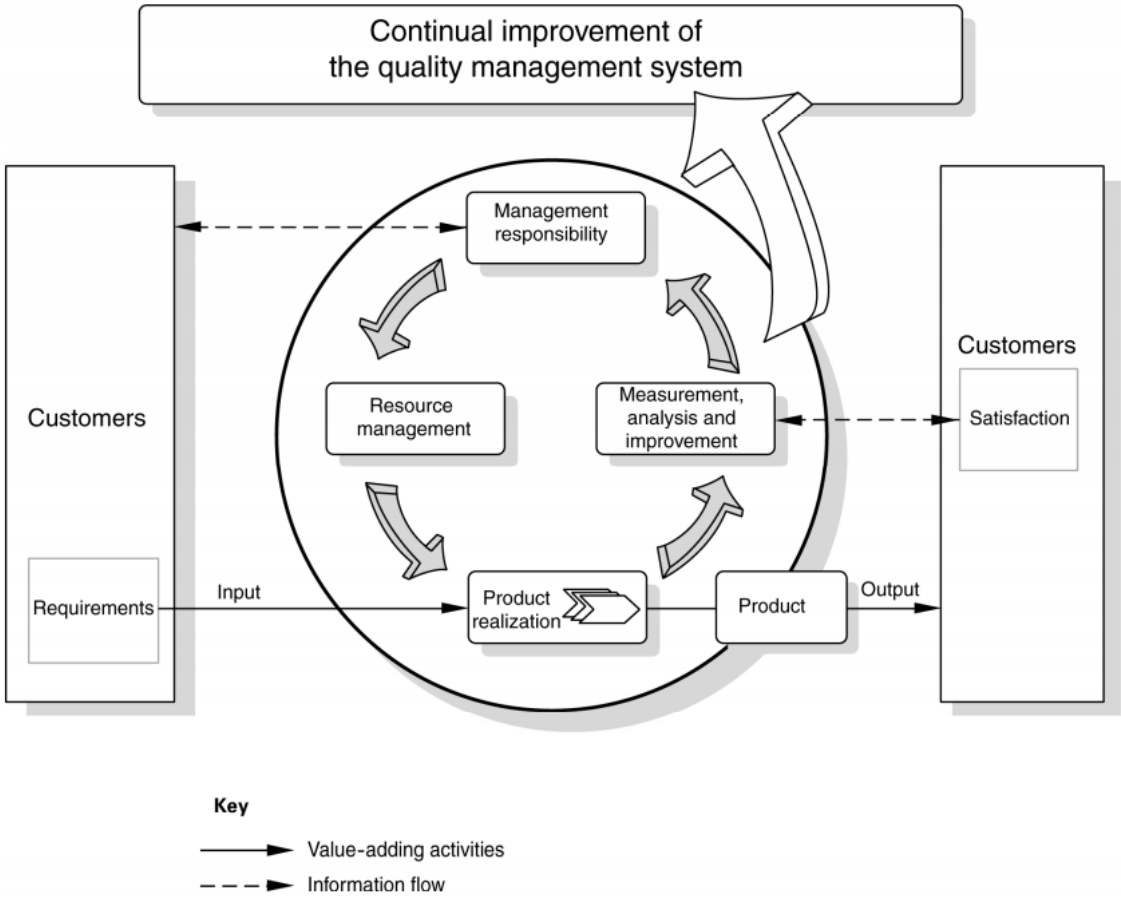


Figure 2. Model of a process-based quality management system [ISO 9001:2008].

The practical guidelines for ISO 13485 implementation can be found from ISO/TR 14969 technical report.

2.2. IEC 62304

IEC 62304:2006, Medical device software – Software lifecycle processes -standard defines the lifecycle requirements for medical device software. It consists of processes, activities and tasks that combine a common framework for software lifecycle processes. The general goal of the standard is to improve safety and effectiveness of medical device software by establishing a process that reveals what the software is intended to do and to demonstrate that the software fulfills those objectives without unacceptable risks [IEC, 2006]. The most important matters of IEC 62304 are software development and software maintenance.

According to IEC 62304 [2006], the basic assumption is that the medical device software is being developed and maintained within the quality and risk management systems. As the risk management system is well covered by the ISO 14971 standard, IEC 62304 addresses the risk management process by only making normative reference to ISO 14971 and adding some minor software related additions. These additions are covered later in Chapter 3 where ISO 14971 is discussed with more detail.

IEC 62304 emphasizes the importance of the software maintenance process as many incidents in the field are related to maintenance or service of medical device systems. For this reason, the maintenance process is considered to be as important as the actual development process. According the Clause 5, the software development process includes following activities:

1) software development planning,
2) software requirements analysis,
3) software architectural design,
4) software detailed design,
5) software unit implementation and verification,
6) software integration and integration testing,
7) software system testing,
8) software release.

The activities are further divided to more specific tasks. Similarly, Clause 6 contains the maintenance process with following activities:

1) establish maintenance plan,
2) problem and modification analysis,
3) modification implementation.

The maintenance activities are also divided to specific tasks. Besides these concepts, the software configuration management and the software problem resolution processes are identified in IEC 62304.

IEC 62304 clearly states that its scope includes both embedded software in physical medical devices and standalone software that itself is a medical device. However, the standard does not cover validation or final release of the device. One crucial concept in IEC 62304 is software safety classification since different standard clauses are applied to different device safety classes, thus allowing manufacturers to tailor their processes to match with the classification.

The safety classes are assigned based on severity [IEC 62304]:

1) Class A: No injury or damage to health is possible.
2) Class B: Non-serious injury possible.
3) Class C: Death or serious injury possible.

It is worth to notice that the assigned software safety classification can be reduced by one class (that is, from C to B or from B to A) if appropriate hardware risk control measure is applied. The risk control process is covered in more detail in Chapter 3.

2.3. IEC 62366-1

The usability and safe use of medical devices has become increasingly important concern since the medical devices have become more complex and they are being used even by the patients themselves. *IEC 62366-1:2015 Medical Devices – Part 1: Application of usability engineering to medical devices* is a usability engineering process standard that

defines processes to analyze, specify, develop and to evaluate medical device usability. The general goals of the standard are to achieve reasonable usability to minimize use errors and use-associated risks and to protect the safety of the patient and the medical device operator [IEC, 2015].

The original purpose of IEC 62366 was to present a usability engineering process for physical medical devices. The standard, especially the revised version, can be used also with medical device software, although special consideration for software related matters is not offered besides few incoherent examples.

The main intention of the standard is to optimize the usability of medical device as it relates to safety. Part 1 of the standard describes the usability engineering process, while part 2 provides guidance how to comply the first part of the standard. The scope of the standard includes normal use of the medical device, that is correct use and typical use errors. It can also be used to identify abnormal use related risks, but it cannot be used to assess or mitigate them.

The scope and implementation details of the usability engineering process can be selected case by case. This allows the organization to tailor the process based on the type of the medical device in question. If the user interface is simple, or there otherwise exists little safety issues, the complexities of the usability process can be reduced. The standard presents some implementation methods for different tasks, but using the offered methods is not compulsory. The implementation suggestions are not overly detailed so there might be need to seek guidance from external literature sources. From this point of view, it is not enough just to follow the text from the standard - some additional usability engineering understanding is needed in order be able to select the applicable methods.

It is appropriate to notice that IEC 62366-1 has strong relation to ISO 14971 risk management standard. For example, when manufacturer uses information for safety as a risk control measure, this information must be conducted following the usability engineering process (ISO 14971 and the concept of risk control measure is presented in the chapter 3). In practice, this information for safety, for example warning tag or operation manual, must be designed with the same iterative process that the user interface.

The closer analysis of IEC 62366-1 is left outside the scope of this thesis.

3. Analysis of ISO 14971

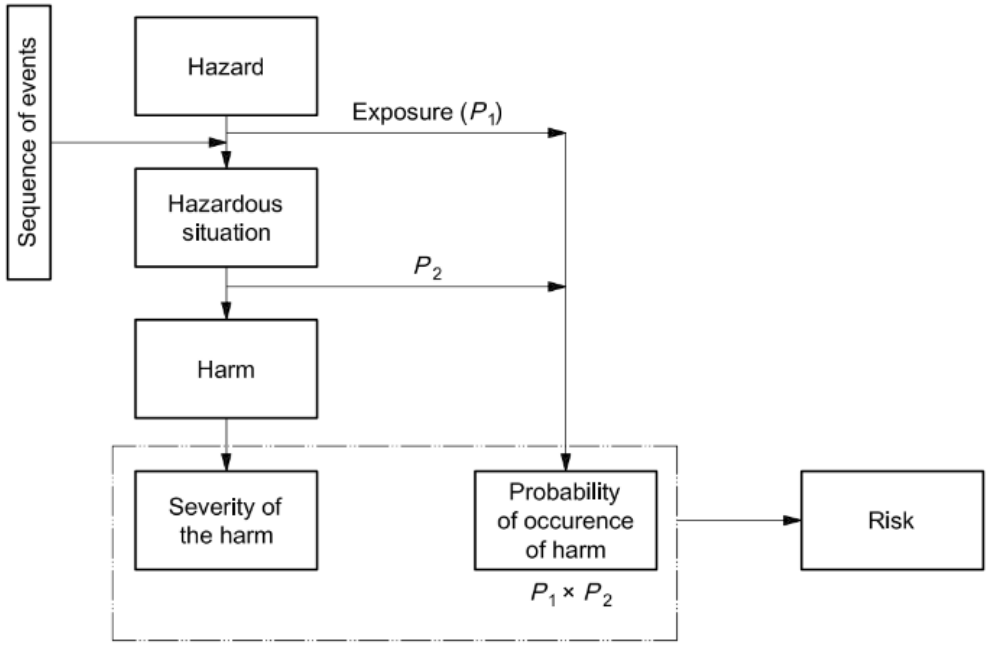
ISO 14971:2012, Medical devices – Application of risk management to medical devices is a worldwide recognized standard used for medical device risk management. It defines a framework and minimum boundaries for risk management activities. The most important goal of performing formal risk management process activities is the assessment of overall residual risk [Schmuland, 2005].

ISO 14971 is needed when the manufacturer wants to claim conformity to IEC 62304 since IEC 62304 makes a normative reference to ISO 14971 requiring its use [IEC, 2006]. The compliance to ISO 14971 is needed to sell medical devices in EU region and Canada and it is recognized by the FDA as a way to meet the intention of the regulatory requirements.

3.1. Key terms of ISO 14971

ISO 14971 uses certain risk management related key terms that are important in order to understand the meaning of the standard. First, harm means physical injury or damage to the health of people, or damage to property or the environment. It is worth to notice that this definition has a major effect regarding the scope of the risk management process of ISO 14971 as it limits out all the development project related issues. Second, hazard is a potential source of harm and there can be several different causes for the hazards [IEC, 2009]. Third, a hazardous situation is a circumstance where hazard occurs. Therefore, as defined in ISO 14971, a hazard cannot directly result in harm – a hazardous situation is needed before harm can occur. IEC/TR 80002-1 [2009] provides a clarifying example:

> "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."



NOTE P_1 is the probability of a hazardous situation occurring.
P_2 is the probability of a hazardous situation leading to harm.

Figure 3. Pictorial representation of the relationship of hazard, sequence of events, hazardous situation and harm [ISO 14971, 2012].

Furthermore, risk is defined as combination of probability of occurrence of harm and the severity of that harm. Consequently, the concept of specific risk does not exist before risk estimation is given - the hazard transforms to risk during the risk estimation phase. The difference is little more than a semantic in spoken language, but it is useful to be able to differentiate the two concepts in the risk management context. The pictorial presentation of the key terms is presented in Figure 3.

3.2. Risk management personnel and responsibilities

ISO 14971 states that the personnel performing risk management tasks must have required knowledge of risk management and qualification records to provide objective evidence must be maintained. The personnel must be trained to ensure full understanding of risk management process requirements. Furthermore, domain knowledge, both clinical and software development, is needed to be able to identify and estimate potential hazard scenarios. According to ISO 14971, the understanding is needed from:

1) How the device is constructed.
2) How the device works.
3) How the device is produced.
4) How the device is actually used.
5) How to apply the risk management process.

ISO 14971 recommends the use of multidisciplinary team in practical risk management related work. For confidentiality reasons, the qualification records can be maintained outside of the quality management system.

The top management commitment is essential for the effectiveness of the risk management process because:

1) Risk management activities are less effective if resources are inadequate.
2) Risk management is a specialized discipline and requires training.
3) Top management is required to establish a risk management policy that defines how risk acceptance is determined.
4) Risk management is an evolving process and should be periodically evaluated.

3.3. Risk management plan and documentation

ISO 14971 states that the risk management activities for a certain product development must be planned and these plans are presented in the risk management plan document. The risk management plan must include:

1) The scope of the risk management process.
2) Assignment of responsibilities and authorities.
3) Requirements of reviews.
4) Risk acceptability criteria (based on the company risk management policy), including criteria for risks when the probability cannot be estimated.
5) Verification activities.
6) Production and post-production activities.

The risk management plan should prove that the organization uses an organized approach to risk management and can act as a checklist to ensure that all essential issues have been addressed. The plan does not need to be created at once, it can evolve over time. In addition, the risk management plan should also 1) describe the functionality of the medical device implemented in software, 2) state that the software will be developed according to standards and 3) reference to development aspects unique to software risk management [IEC, 2009]. The risk management plan does not need to be a specific file with a specific title, the actual document can be embedded into the quality management system.

Every risk management process requirement that needs to be documented will be documented in the risk management file. The risk management file can be in any form or type of medium. It does not need to physically contain all the documents; references and pointers are sufficient. The term *risk management file* is used in standard to signify where all risk management related records and documents can be found. In practice, the risk management file is commonly embedded into the quality management system. Besides collecting evidence and records from other phases of the risk management process, the risk management file must provide traceability from each identified hazard to:

1) The risk analysis.
2) The risk evaluation.
3) The implementation and verification of the risk control measures.
4) The assessment of the acceptability of any residual risk(s) [ISO, 2012].

Certain parts of the risk management file should be able to address software related changes and different release versions without compromising the traceability. Traceability is needed to demonstrate that the risk management process has been applied to all identified hazards and thus to prove the completeness of the risk management process.

3.4. Risk management process lifecycles

ISO 14971 defines the basic elements to be included to the risk management process. As can be seen in Figure 4, the process baseline can be divided to six specific lifecycle phases:

1) Risk analysis.
2) Risk evaluation.
3) Risk control.
4) Overall residual risk evaluation.
5) Risk management report.
6) Production and post-production information.

3.5. Risk analysis

Risk analysis is an activity to systematically use available information to identify hazards and estimate risks. In risk analysis phase, the analyzed medical device is described and identified, with the identification of persons and organization who performed risk analysis. Furthermore, the date and scope of the analysis must be recorded. Standard informs that if any previous risk analysis data or other relevant information is available for similar medical device, this information should be used as a starting point to save time and effort. However, some caution is obviously needed when evaluating the compatibility of the previous analysis and the current one.

The intended use of the medical device is documented, as well as reasonably foreseeable incorrect use. When considering these issues, the focus should be on all those characteristics of the device that could affect to the safety. For example, it's not unusual that medical device is being used outside of normal intended circumstances - the manufacturer should be prepared for the potential hazards arising from these situations.

Based on the previously collected list of safety related characteristics, all the predictable potential hazards should be identified. It is noticeable that the hazards should be considered in both routine and inadequate conditions.

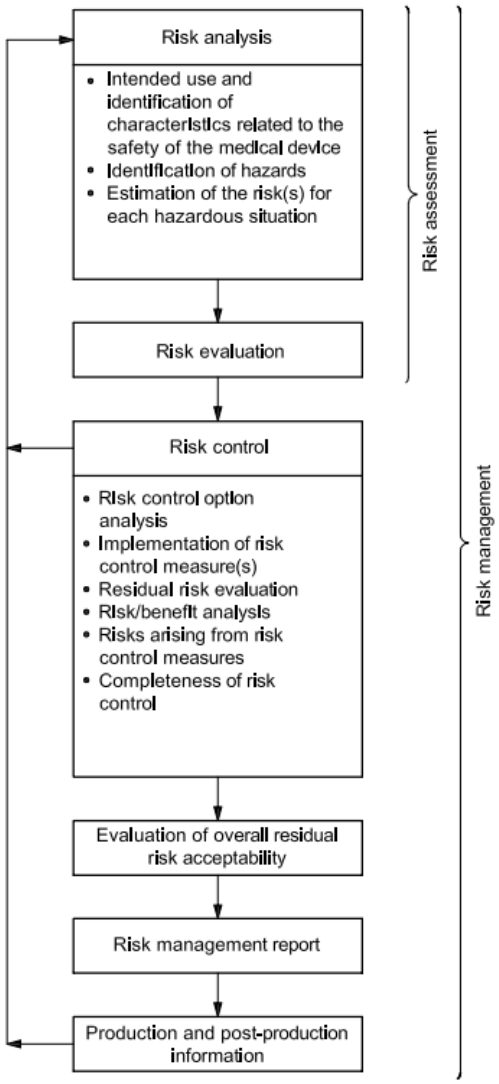


Figure 4. A schematic representation of the Risk Management Process [ISO 14971, 2012].

The final step of risk analysis is to identify hazardous situations and to estimate related risks. All the sequences of events that can lead to an identified hazard are hazardous situations which shall be documented and further analyzed. Every identified hazardous situation must be estimated with the help of available data. This information can be, for example, expert opinion, scientific technical data, field data from similar medical devices already in production use or usability tests. In estimation, the risk related to a hazardous situation is assessed with probability of occurrence and severity of harm (of realized hazard). The risk estimation is difficult task since the key factors of estimation vary between every hazardous situation.

3.6. Risk evaluation

Risk evaluation is a straightforward task: every identified and estimated risk is compared to acceptance criteria defined in the risk management plan. The decision whether risk

reduction is needed or not is solely based on the acceptance criteria. It is worth to notice that the standard does not specify any kind of concrete criteria, so the decision is left to the manufacturer. The justification to risk acceptance criteria must be given to a regulatory-approval body.

3.7. Risk control

When risk is evaluated to be unacceptable, risk reduction is needed and risk control activities must be performed. The first step is to do risk control option analysis. Risk control options are (in prioritized order):

1) inherent safety by design,
2) protective measures (in the device or in the manufacturing process),
3) information for safety.

The prioritized order is an important concept and can be found also from MDD. If possible, the manufacturer should always try to alter the design of the product in a way that the identified risk would reduce. When this is not convenient, some protective measures could be used, for example automatic cut-offs or visual and acoustic alarms. The least preferred option is information for safety. In practice, information for safety provides instructions how to safely perform a specific task or how to avoid a hazard. Information for safety can be, for example, a warning tag or safety information in the operation manual.

Selected risk control action can reduce probability of occurrence or severity of harm or, in an ideal case, both. Also, often more than one action can be used to control certain risk. On the other hand, one possible result of the option analysis is that no practicable reduction option was found. In this case, risk/benefit analysis is carried out to decide if the medical benefits outweigh the residual risk.

The next phase of risk control is to implement the previously selected risk control measures. Every implemented measure and the effectiveness of the measure should be verified and documented. This leads to two different verifications: the first verification ensures that the risk control measure is part of the final design and the second one ensures that the implemented risk control measure is actually effective.

After the selected risk control measures are implemented, the residual risk shall be evaluated. The evaluation is based again to the acceptance criteria defined in the risk management plan. If the risk in question is not acceptable, further risk control measures must be carried out. However, if the residual risk is acceptable, the risk can be disregarded.

As previously discussed, risk/benefit analysis is needed if there is no practicable option to reduce a certain not acceptable risk. This process step is used to analyze if the device provides more medical benefits than the potential harm. In most instances, the design should be dropped or at least altered if the related risks cannot be judged to be acceptable. The risk/benefit analysis may be the only opportunity for the manufacturer to

avoid this problem in an exceptional occasion where risk reduction is not possible. Standard does not provide any defined method for medical benefit estimation, so it is essentially a matter of experienced individual judgement.

The implemented risk control measures should be reviewed to ensure that they have not introduced any new hazards or caused any unwanted side effects. This review must also be documented.

The last step of risk control phase is to review the completeness of risk control. The object of this exercise is to ensure that all risks from all identified hazardous situations have been considered and covered appropriately.

3.8. Overall residual risk evaluation

In overall residual risk evaluation phase the overall residual risk of the medical device is considered from a wide perspective. Residual risk denotes the risk remaining after risk control measures have been taken. ISO 14971 does not define any standardized method for overall residual risk evaluation. The only requirement is that the evaluation is based on the acceptance criteria defined in the risk management plan. In this way, the decision how to implement the evaluation is left to the manufacturer. Nevertheless, the evaluation should be carried out by qualified personnel with a knowledge, experience and authority and the evaluation must be documented.

3.9. Risk management report

The last phase before commercial release of the medical device is to conduct a risk management report. It ensures that the risk management plan has been implemented, the overall residual risk is acceptable and the appropriate methods to collect production and post-production information are in place. The responsibility of this review is high, so it should be carried out by qualified person with the authority.

3.10. Production and post-production information

According to ISO 14971, the manufacturer must establish a system to collect and review production and post-production information. This information includes data generated by operators, users, administrators and installation accountables as well as new or revised laws and standards. The gathered information should be considered for relevance to safety. There is a possibility that some previously unrecognized hazards have emerged or some certain controlled risk becomes unacceptable. If this situation occurs, new information must be fed as an input to the risk management process and the overall residual risk must be re-evaluated.

It is noticeable that national regulations might create additional requirements for post-production monitoring.

3.11. Risk management process requirement extensions from IEC 62304

As previously mentioned, the risk management process of ISO 14971 is extended by IEC 62304 Clause 7. These extensions must be taken into account when the medical device software product is manufactured. The medical device software safety classification, declared in IEC 62304, has again a significant role in Clause 7 since only one additional risk management requirement is assigned to class A software. As some other important IEC 62304 related terms are also present, it is essential to be familiar with both of the standards, even when focusing only on risk management process.

The extensions from IEC 62304 should be embedded to corresponding ISO 14971 risk management lifecycle phase.

3.11.1. Analysis of software contributing to hazardous situations

The software items that can contribute to a hazardous situation should be identified. Also the potential causes for these contributions should be considered, for example incorrect or incomplete specification of functionality, software defects in functionality, unexpected results from unknown provenance software components, hardware failures or other software defects that could result in unpredictable operation and reasonably foreseeable misuses. If any potential causes from unknown provenance software components is found, those components' anomaly lists must be evaluated.

All the potential causes and sequences of events that could result in a hazardous situation must be documented.

3.11.2. Risk control measures

Risk control measures must be identified and documented for each previously identified potential cause. Risk control measures can be implemented in hardware, software, the working environment or in user instructions.

If the risk control measure is implemented as a part of the functions of a software item, the measure must be included in the software requirements, safety class must be assigned to the affected software item and the item must be developed in accordance with software development model of IEC 62304 (Clause 5).

3.11.3. Verification of risk control measures

Every implemented risk control measure should be verified and verification should be documented. If risk control measure was implemented as a software item, the measure is evaluated to identify any new sequences of events that could result in hazardous situation. The traceability of software hazards should be documented as appropriate 1) from the hazardous situation to the software item, 2) from the software item to the software cause, 3) from the software cause to the risk control measure and finally 4) from the risk control measure to the verification of the risk control measure.

3.11.4. Risk management of software changes

After software change, the change must be analyzed to determine if additional potential causes for hazardous situation are introduced or additional risk control measures are required. The change affects for existing risk control measures must also be analyzed. Based on these analyses, relevant previously described risk management activities are performed.

4. Agile perspective

Agile is a complex concept with many different definitions that are related to the context of use. Nowadays agile is being applied in many different fields of industry, but the origin of the term can be found from the domain of software development. Ángel Medinilla, an agile evangelist and a frequent agile conference speaker, suggests that agile is, at least to some degree, evolution of lean that is practiced in software development [Medinilla, 2012]. This idea seems to be coherent with the fact that most agile approaches share common elements with lean philosophy, either implicitly or explicitly [AAMI TIR45, 2012].

The term agile is generally used when the development process in question follows the spirit of the Agile Manifesto, is more empirical than deterministic and is iterative and evolutionary.

4.1. The essence of agile

Manifesto for Agile Software Development was written in 2001 by seventeen software practitioners. In the Manifesto, Beck et al. [2001] defines four high-level value statements of agile software development:

> "Individuals and interactions over processes and tools
>
> Working software over comprehensive documentation
>
> Customer collaboration over contract negotiation
>
> Responding to change over following a plan".

The left side of the value statements shows what authors consider to be more important than the item on the right side. It does not mean that the items on the right are considered to be completely unimportant [Highsmith, 2002].

When further crafting the definition of agility, Ron Jeffries [2016], who is one of the original signatories of the Agile Manifesto, states:

> "Like many of the ideas in software development, modern 'Agile' software development offers to make software development more productive and better controlled by making it simpler. Agile is simple. Four values, a dozen of principles. How complex could it be? Well, it still seems to get pretty darn complex"

On the other hand, agile is an umbrella term for lightweight software development methodologies that emerged during 90s as a counter reaction against plan-driven, heavy and document-focused processes that were widely used at that time [Poppendieck & Poppendieck, 2003]. These lightweight methodologies were, among others, Scrum, Extreme Programming (XP), Feature Driven Development and Crystal Clear Methods and they had lot of common goals and principles [Highsmith, 2002]. Afterwards, a few new important methods have stand out, namely Lean software development, Kanban and

Scrumban, a hybrid model combining Scrum and Kanban. There has also been evolution amongst the original lightweight methods.

Against this background, the term agile seems to have at least two distinct meanings:

1) An ideal approach and mindset for collaborative and adaptable software development driven by the values and principles defined in the Agile Manifesto.
2) A collective noun representing different lightweight software development ecosystems, that is, agile methods.

For the purposes of this thesis, the first definition is applied. Rather than focusing on some specific agile implementation, the broader view of agile values and principles is examined.

4.2. Agile principles

The 12 principles inspired by the goals and outlined in the Agile Manifesto by Beck et al. [2001] are:

1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4) Business people and developers must work together daily throughout the project.
5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7) Working software is the primary measure of progress.
8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9) Continuous attention to technical excellence and good design enhances agility.
10) Simplicity - the art of maximizing the amount of work not done - is essential.
11) The best architectures, requirements, and designs emerge from self-organizing teams.
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

These principles are common to all agile development methods, but the actual implementation and influence of certain principle varies with the method in question. Each method has its own specific purpose and the fundamental nature of agile is to be adaptable to the context of use. Agile aims to provide benefits in quality and correctness of the product, productivity and efficiency of the development process, improved

estimation and manageable response to change and finally product effectiveness by meeting the needs of customer.

4.3. Traditional software development process

In contrast to agile iterative process model, the traditional software development process can be seen as a single-pass model where all process phases should be completed as a whole before entering to next phase. This kind of process is commonly called a waterfall process model and it was first formally outlined by Dr. Winston Royce in his white paper titled "Managing the Development of Large Software Systems" [Royce, 1970]. However, it is worth to notice that in his paper Royce actually suggested iterative path to process through the "essential steps of computer program development" and warned that the single pass is "risky and invites failure" [Royce, 1970]. For some reason this idea was later ignored by the followers. Despite this possible common misconception, the term waterfall is used here to stand for traditional plan-driven software development model.

The waterfall model, or at least its established perception, is largely criticized by the agile community. The infamous Chaos Report case study by the Boston-based project management and consulting firm Standish Group revealed that the average success rate of software projects was alarmingly low: only 16,2% of the projects were completed on-time and on-budget and this rate fell to 9% in large projects [Standish Group, 1995]. The agile methodology aims to solve the identified problems of single-delivery waterfall model.

4.4. Agile software development in medical device domain

At first glance, the medical device standards seem to be closely following the traditional plan- and design-driven process model with a sequential order of phases. This is especially the case when considering IEC 62304 standard that describes the medical device software lifecycle process regulations. The plan-driven nature of the standard is explained by its age: the revisited 2006 version is based on the original ANSI/AAMI SW68 document that was published in 2001. It is worth to notice that the ISO 14971 risk management process has a built-in iterative loop and therefore the nature of the standard aligns with the values of agile more intuitively. However, the guidance for agile implementation is needed.

Technical Information Report AAMI TIR45 provides recommendations and guidance for complying with international standards when using agile practices to develop medical device software [AAMI TIR45, 2012]. As it is Technical Information Report (TIR), it has not been a subject to the same formal approval process as a standard but it has been approved by a technical committee and the Association for the Advancement of Medical Instrumentation (AAMI) Standards Board. It is not a regulatory document so its recommended practices are voluntary to the user of the document. AAMI TIR45 provides a comprehensive list of suggestions how to align the goals, values,

principles and practices between agile and regulatory perspective and therefore it is central document when designing agile software development model in medical device context.

The evolutionary software development lifecycle model of AAMI TIR45 is presented in Figure 5. As can be seen, the model is well aligned with the agile approach.

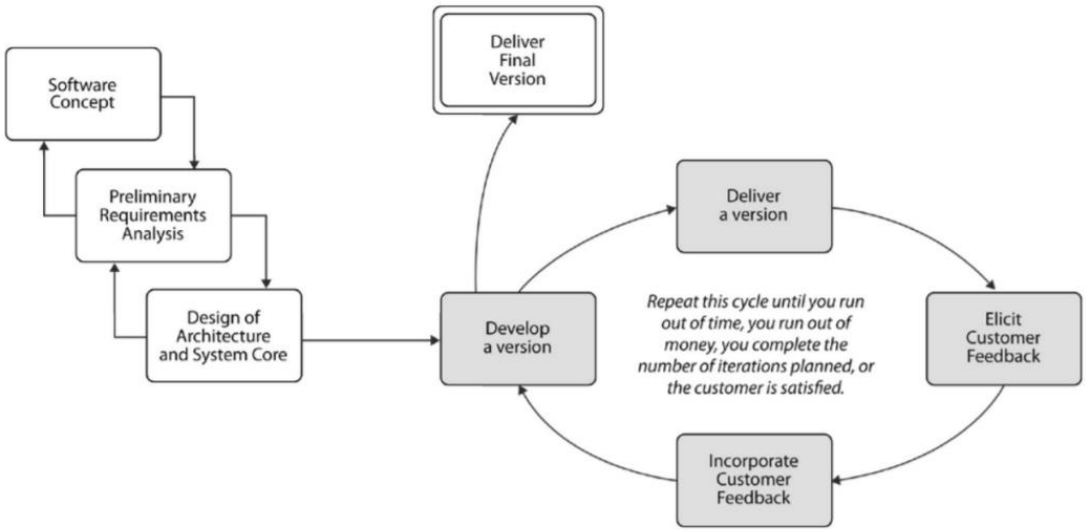


Figure 5. Evolutionary lifecycle [AAMI TIR45, 2012].

5. Synthesizing the regulatory and agile perspectives

Regulatory framework defines a set of activities that must be performed during the software development process. A considerable amount of these activities must be documented in order to satisfy the regulatory requirements. This documentation forms an audit trail that is used as evidence to prove that the activity was performed in accordance with the quality management system.

Following the principles of agile and lean development, the proposed software development model is never ready - it is continuously improving and evolving. The effectiveness of the model must be periodically evaluated during the retrospective meeting. The same applies to the quality management system if conformity to ISO 9001 is claimed. In the medical device domain, it is usual that the quality management system is certified; thus, it must be periodically evaluated by a regulatory-approval body. This audition can be seen as a good opportunity for the organization to ensure that the quality management system implementation is in compliance with the regulatory framework. Furthermore, new ideas for further improvement targets can be gathered.

This chapter discusses the alignment between agile and regulatory perspectives. As agile and regulatory perspectives are two very different models, the contextual mapping between the concepts is needed. A crucial aspect of regulatory development process construction is to thoroughly understand the goals, principles and practices of the regulatory framework, including all relevant regulations, standards and guidance documents. It is obvious that this deep understanding and knowledge is needed to ensure that all regulative requirements are fulfilled.

While the main subject of this thesis is ISO 14971 standard and the risk management process that it defines, the general regulatory background cannot be totally excluded from the discussion. There are several cross-references between the relevant standards, and the implementation of risk management process must be consistent within the whole regulatory perspective.

5.1. Mismatch between regulatory and agile perspectives

Jim Highsmith, acknowledged thought leader in agile community and one of the original signatories of the Agile Manifesto, defines *Chaordic Perspective* [Highsmith, 2002]. It is a term that illustrates the nature of any organization - that there is always both, chaos and order, defying the use of linear planning and execution practices. This makes using traditional management and development processes dysfunctional since the operation environment is always too unpredictable. According to Highsmith [2002, p. xi], there are two consequences:

1) Product goals are achievable, but they are not predictable.
2) Processes can aid consistency, but they are not repeatable.

Further on, Highsmith even implies that when process repeatability increases, the project predictability decreases.

Another popular model to visualize the level of complexity of management decision making is Stacey Matrix [2007], presented in Figure 6. In this context, the horizontal x-axis which represents the level of certainty in decision making, is especially interesting as it aligns well with Highsmith's idea. When the decision can be made based on pre-existing information, the rational decision tactics can be used and the outcome is fairly predictable [Stacey, 2007]. Normally, this is not the case when developing software – the product is unique and no relevant pre-existing information is available and, as a result, the level of uncertainty increases.

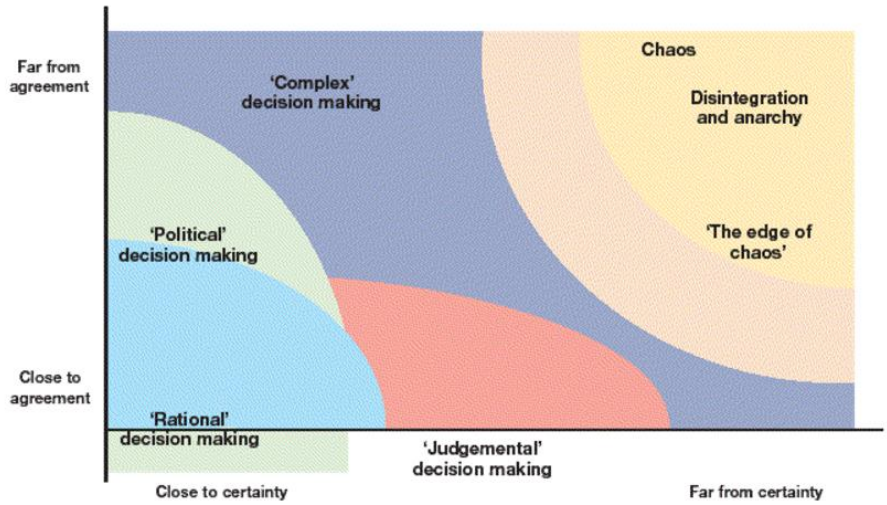


Figure 6. The Stacey Matrix [Stacey, 2007].

To address the uncertainty in decision making, the agile approach emphasizes the ability to respond to change and even the ability to create change in order to create value and competitive advantage [Jeffries, 2016]. An agile process is in a constant state of change. On the contrary, the regulative perspective is not tolerant towards change. For example, the FDA considers the changing requirement (design input) to be a sign of danger that the development process is not properly in control [Ståhlberg, 2015]. The change is also considered to be an indication of poor design.

Agile approaches emphasise empirical and incremental development cycles. The best practice is considered to be the ability to frequently release small and well-focused features that will maximize the current customer value of the system [Jeffries, 2016]. Furthermore, customer collaboration and feedback has an important role in the agile perspective. This practice highlights the incremental and evolutionary nature of agile development - the requirements and even the definition of the product can evolve during the development process. Although the regulatory framework does not explicitly require a certain development lifecycle model, the mindset behind it seems to be strongly based

on plan-driven, linear and design upfront model. This brings some challenges for agile implementation.

It is not difficult to see that the linear, single pass model is a relatively simple process to understand. The activities can be cleanly defined and they are completed without parallel tasks in a simple sequence. However, this is not the case with agile processes, where the same activities are completed in more complex and dynamic sequences. Robust change management control is essential in such a dynamic agile environment.

One of the widely used tools in common agile development methods is self-organizing and cross-functional teams that have total autonomy over their own processes and practices [Medinilla, 2012]. However, the regulatory framework requires an ongoing, functional and robust quality management system and disciplined processes that dramatically limit the freedom of choice of the development team. Concerning the regulatory framework, the controlled processes of the quality management system are essential in order to produce high quality and safe software.

In order to demonstrate compliance to regulations, comprehensive documentation and collection of evidence of followed practices and process steps is needed. IEC 62304 puts considerable interest on documentation of planning and process execution activities - especially the risk management process is covered with great detail, as the risk management process should be embedded throughout the development activities. IEC 62304 and other regulations require a substantial amount of documentation to be used as audit trail, and the use of agile practices cannot be used as an excuse to discard that obligation. Even if documentation is not considered to be totally unimportant in agile practices [Highsmith, 2002], the working software is indisputably cherished over comprehensive documentation - the working software is the best indicator of the project progress and the realized value.

In conclusion, it is clear that not all agile values and principles are suitable when developing medical device software. However, it is equally evident that using traditional software development methods is not the most efficient way to organize the development process [Standish Group, 1995]. For this reason, an agile approach can bring value also to medical device software development [AAMI TIR45, 2012]. As the regulatory framework does not prohibit the use of any specific software development methodology, it is possible to use selected set of agile practices and tailor them to be applicable in the medical device context.

5.2. Similarities between regulatory and agile perspectives

In addition to mismatches, there are also some similarities between the agile and regulatory goals. One of the most apparent concept that aligns well is quality - both agile and the regulatory framework value high-quality software. Despite using different terms and emphasizing different aspects of the quality, they both clearly place high value to the concept. The agile perspective focuses on a broad view of meeting customer requirements

and fulfilling expectations with technical excellence, whereas regulatory perspective concentrates more on performance and safety of the product. These different aspects of the quality should be realised in a way that they are in good balance between each other, without sacrificing the safety.

The regulatory framework does not elaborate on the efficiency of the development process as agile does, but this is not an incompliancy - every organization, including medical device manufacturers, benefit from effective development processes. The customer satisfaction is central value in agile perspective – the agile project has an implicit desire to produce a product that maximizes the satisfaction of the customer at every phase of the project. The customer focus is also one of the quality management principles as the strive to exceed customer expectations is hoped to achieve [ISO 9000, 2005].

It is a common mistake to underestimate the significance of planning in agile methods. This phenomenon might be the result of the Agile Manifesto value statement which values responding to change more than following a plan. This form of statement makes a comparison between these two things and shows only which one is more important in the context of agile practices - it does not indicate that there is absolutely no value in the latter [Highsmith, 2002]. In fact, planning is something that happens all the time during agile project, on every phase of development cycle. Plans are frequently created and adjusted based on the feedback - it is impossible to respond to change without actually planning the response action before implementation. As planning is also important part of the regulatory framework, the value of planning is well aligned between the agile and regulatory perspectives [AAMI TIR45, 2012].

5.3. Software development lifecycle model

Neither ISO 14971 nor the other standards of the regulatory framework mandate organization to use any particular development lifecycle model. However, IEC 62304 requires that some certain lifecycle model is chosen and the implementation details are documented within the quality management system [IEC 62304, 2006]. This leaves the manufacturer considerable amount of freedom for lifecycle model design and implementation.

The standards describe various activities that must be performed within the software development lifecycle. The activities can have distinctive requirements, completion criteria and cross references between each other. In IEC 62304 these activities are organized in a manner that to a large extent resemble linear waterfall development cycle. However, the agile approach is indeed feasible if the organization carefully documents the lifecycle design and implementation details [AAMI TIR45, 2012]. Especially close attention should be paid to documenting the timing and sequence of the activities and the audit trail of the accomplished tasks. The lifecycle model and risk management process integration should also be in the centre of attention.

5.4. Dividing the regulatory process activities to different layers of abstraction

One practical solution suggested by AAMI TIR45 [2012] for agile and regulatory perspective alignment is to divide the regulatory process to different layers of abstraction. Certain regulatory activities are completed only once during the lifetime of the development process while the others are carried out repeatedly. These activity types are divided into different abstraction layers. This practice is very useful in the context of risk management process.

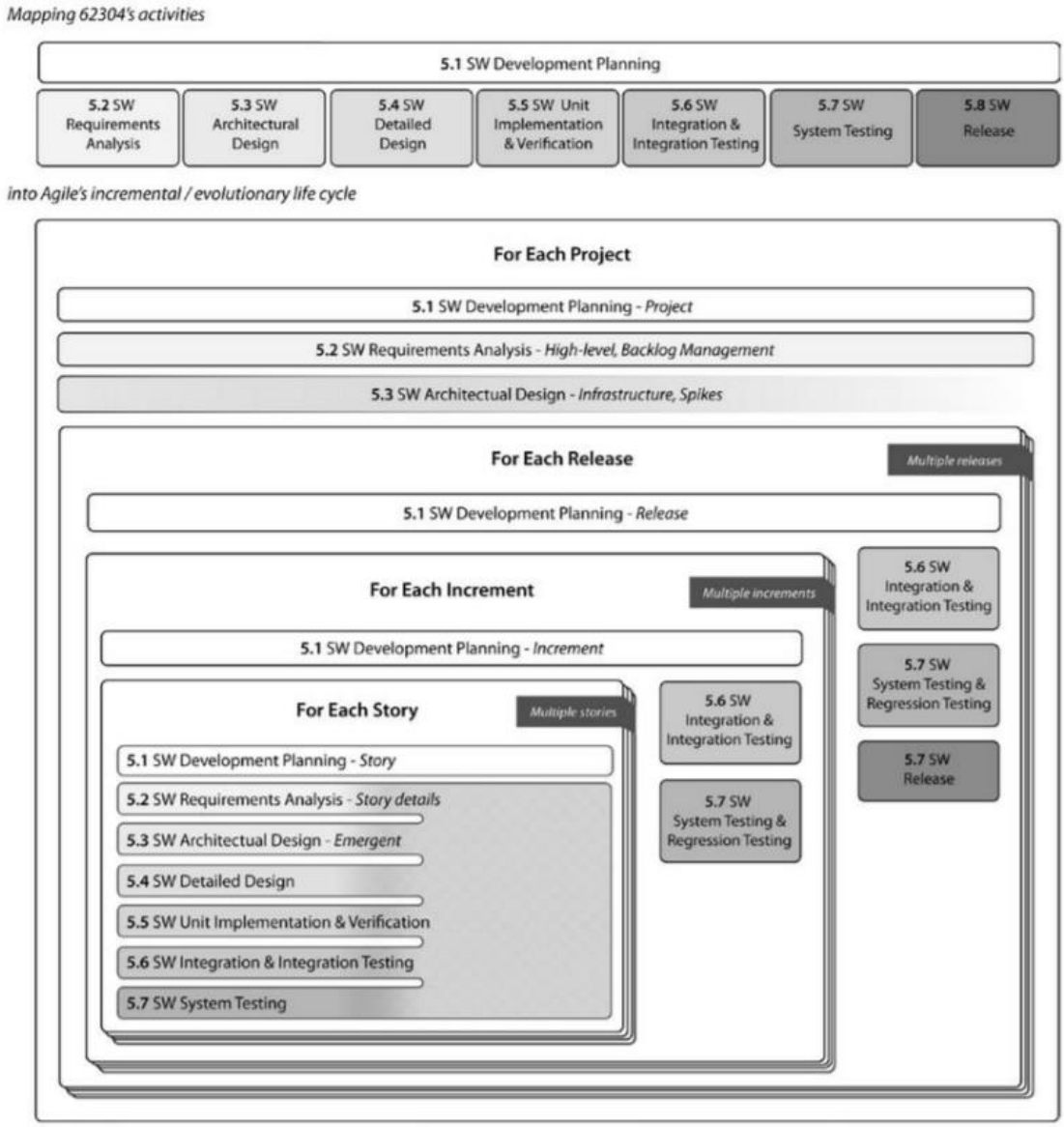


Figure 7. Mapping IEC 62304's activities into agile's incremental/evolutionary lifecycle [AAMI TIR45, 2012].

As seen in Figure 7, the layers of abstraction are derived from agile practices. In the figure, the sequential process model phases of IEC 62304 are mapped with agile incremental lifecycle. The most abstract layer or context where regulatory development

activities are completed is project or product layer, depending on the type of the organization and the development environment. These activities, typically planning and design tasks, are related to the whole project or product. For example, risk management plan must concern the whole project and it must be produced and validated before any actual development has begun.

The second layer is related to each release or version of the product. The regulatory activities belonging to this layer are typically verification measures where the quality of the release is verified. Following the mindset of agile practices, a release is a pre-planned and coherent set of value-focused features. The release can be made based on one or more increments.

According to AAMI TIR45 [2012], the third layer is increment layer. The term is relative to applied agile convention, thus the iteration layer is equally suitable term. The result of each increment should be potentially shippable product. The requirement to produce potentially shippable product is useful mindset since it creates high level of confidence that all work done in the increment is actually finished. The decision whether to create a release to be deployed to the production environment or not can be made based on the evaluation of that time. It is fairly usual that the result of one increment does not bring enough value compared to costs of the deployment, especially in the cases where the development cycle is relatively short. However, if this scenario is frequent, it is a strong evidence that the deployment process is too ineffective and should be investigated and presumably improved. The project delivers value only when deployed and the best value comes from small and focused features [Jeffries, 2016].

From the viewpoint of regulatory requirements, the only actual difference between the release layer and the increment layer is the required ceremonies of the release activity. In the context of risk management process, this includes the activities of the overall residual risk evaluation and producing the risk management report. It is debatable if the division between release and incremental layer should be highlighted or even initially made. One central concern related to this practice is that the release ceremonies, which are not frequently performed, will increase the length of the feedback cycle of those tasks. For example, if the residual risk evaluation is done not until the end of planned release cycle, the unexpected findings can postpone the release schedule. Similar problems might occur during the risk management report processing. Besides the feedback cycle of the release ceremonies, the feedback cycle of the end user feedback also increases. This effect has several drawbacks that agile practises are trying to address - the short feedback cycle is essential agile principle. The short feedback cycle improves the quality of the software [Highsmith, 2002] and the frequent deliveries make the project more predictable [Jeffries, 2016] - the level of unexpected and considerable problems will decrease.

In the context of risk management process, the residual risk evaluation can be performed simultaneously with the development during the iteration and the composing

of the risk management report can be automated using the data from the risk management tool. These approaches are highly recommended.

The last abstraction layer is the story layer. The agile term story represents a small piece of functionality that can be an incomplete set. Depending on applied agile convention, also term use case could be used. From the regulatory perspective, story is the smallest reasonable process management item that is addressed in the requirements [AAMI TIR45, 2012]. The layers and the corresponding process activities are presented in Table 1.

Table 1. Process activities in different layers.

Layer	Process activities
Project layer	• project development planning • high-level requirements analysis • infrastructure and architectural design
Release layer	• release planning • integration • integration testing • system testing • regression testing • release
Increment layer	• increment planning • integration • integration testing • system testing • regression testing
Story layer	• story planning • story requirements analysis • design (as needed) • implementation and verification • integration • integration testing • system testing

5.5. Definition of value in medical device development context

Agile does not especially emphasize the importance of software safety or seek benefits by improving it. However, agile emphasizes the concept of customer value and concentrates on maximizing the value at all times during the software development

lifecycle. From the viewpoint of medical device software customers and stakeholders, safety is arguably the highest valuable quality of any medical device product [AAMI TIR45, 2012]. Thus, the agile concept of value can be redefined in the context of medical device followingly:

1) Safety of the use of the medical device.
2) Medical benefits of the medical device.
3) Other customer value that provides competitive advance over the similar products in the market.

This definition applies to the whole end product as well as every separate feature of it. This definition of value must be taken into account in every step of the development process design and also in product design itself. It is a very useful definition especially during the development phase when prioritizing the product features in product backlog. With the respect to agile perspective to value, it must be noted that value will be realized only when product is shipped and put to use.

5.6. Definition of done as a tool to fulfil the regulatory requirements

Definition of done is a very important concept in agile practices. It is a collaborative tool to ensure that all team members agree what "being done" means in different development activities. Definition of done is trying to address the problem of partially done work. Partially done work causes several problems: it has tendency to become obsolete, get in the way of other development and it causes uncertainty if the feature is working at all, or if it is a working solution to the business problem it tries to solve. Partially done work ties resources and thus generates financial risk. The elimination of partially done work is a project management risk-reduction strategy [Poppendieck & Poppendieck, 2003].

For example, definition of done can intend that the feature is designed, implemented, integrated, documented, tested and validated. Kniberg [2015] has a very practical approach when he states that the best possible way to define done is to intend that the feature is "ready to be deployed to production". On the other hand, he also admits that not all types of stories are able to follow this convention. He suggests that if this is a problem, definition of done could be defined separately for every story. Development team can use checklists as a backup for common sense. The specific items selected to be in the checklist will depend on the type of the feature, technologies used and other current impediments of the project [Rubin, 2012].

Jeffries [2016] emphasizes the evolving nature of definition of done. To summarise his theory, evolving nature of the term means that at the beginning of the project the definition can be loose but it gets more stringent over time. For example, there might be some actual project related impediments that prevent the features be production ready at the end of the increment and when these impediments are later removed, the definition of done can evolve. While this approach seems practical and natural in agile perspective, it does not utilize the full potential of the concept regarding the regulatory framework.

Definition of done is a very useful concept when aligning the regulatory and agile perspective. It can be used to illustrate the regulative process requirements that must be performed before a certain issue or story is completed. When used in this manner, it is functioning as a risk management process related acceptance criteria to ensure that all risk controls are implemented and verified - if definition of done is used consistently with every feature, story, use case, backlog item, increment and release. It can ensure the completeness of risk control (however, this approach should not be confused with the item-specific acceptance criteria, which is a set of conditions of business satisfaction for that specific feature). Furthermore, appropriate audit trail must be generated during the process in order to satisfy the requirements of the regulatory framework. In addition, automate acceptance tests should be used to protect the features not become undone at some later point of time during the development project.

5.7. Process inputs and outputs

The regulative framework emphasizes the importance of process inputs and outputs. When evaluating the efficiency of the process, it is possible to determine how well the planned process output meets the process input. In linear model these concepts are easy to understand and thus fairly convenient abstractions. The problem with the linear model is that it resembles the traditional waterfall development model and thus is not very well suited for software development in a complex domain – the agile approach is more practical solution. However, the dynamic nature of the agile process adds complexity to the way in which inputs are used and outputs are generated [AAMI TIR45, 2012].

In the linear model, it is easy to follow the process flow: the process is started with the inputs and the process produces the outputs. In the incremental development this flow is less apparent. In agile development, it is extremely important for every process to define the possible inputs, the entry criteria of these inputs and how ready the inputs should be before the process starts. Similarly, it is essential to define the types of the possible outputs, the exit criteria of these outputs and how ready the outputs should be before process can be regarded as done.

The agile approach gets complicated especially with design inputs and outputs. Design inputs can be defined as the physical and performance requirements that are used as a basis for device design and design outputs as the result of design effort (at each design phase and at the end of the total design effort). With respect to this definition, a design input can be for example a story, a use case, a user need, a functional and non-function requirement and intended use of the device. Equally, the concrete design output can be the design artifact that demonstrates that design process has been completed, that is, the design itself.

In an agile perspective, the relationship between the inputs and outputs is not linear, it is bidirectional - in the iterative process new information is gathered and part of the output affects to the input through the feedback cycle. This denotes that the inputs can

change during the activity process, at the same time while the outputs are being produced. Therefore, the relationship between the inputs and outputs should be considered in a context of small pieces of information.

It is important to notice that large amount of the design input can be verbal communication between the development team and the customer and this communication can last over the whole development phase. When considering a certain activity, the scope should be defined and the minimum amount of relevant input selected. To be able to start the activity, the inputs should ready enough to allow proceeding [AAMI TIR45, 2012]. If there exists a regulatory requirement to get an approval or validation for the input before proceeding, the validation task should be documented and included in the process. Naturally, it is important that the process implementation demonstrates that the regulatory requirements have been satisfied.

IEC 62304 standard elaborates the concept of inputs and outputs with great detail. Similarly, AAMI TIR45 technical report provides high quality guidance for this fairly complex issue. The closer analysis of the subject is out of the scope of this thesis.

5.8. Rationale behind agile methodology adaptation

While agile itself does not provide full functioning processes for software development, certain agile development methods can give detailed guidance for development activities. The problem with predefined and exact processes is that every organization, environment, project and team is different - the processes need to be modified to fit the domain. The agile perspective emphasizes the need for tailoring the development methods to fit for the special context where used and the same applies to regulatory perspective [AAMI TIR45, 2012]. In order to design an effective agile development method implementation, it is essential to understand the underlying principles and values of the practices. As Kent Beck says in his foreword of Lean From The Trenches: "Applying those practices required wisdom, patience and persistence, which is why you can't just copy the story to fix your project" [Kniberg, 2011]. The organizations should understand the values and practices of the agile development first, and only after that find the necessary processes and tools to support the vision. It is important to adapt to reality, not try to make the reality adapt to vision [Medinilla, 2012].

The speciality in medical device domain is that it requires considerable amount of methodology ceremony to meet the regulatory requirements. There is no commonly known existing agile method that would offer sufficient discipline in its processes so adaptation is inevitably needed. This should not be seen as a problem, since the basic nature of agile is to be adaptable. However, the well-documented set of best practices of a certain agile methodology can help organization to learn and understand the principles and values of agile software development [Medinilla, 2012].

When designing the agile method implementation, the project objectives and characteristics should be carefully analyzed. Besides business-value-oriented objectives,

Highsmith [2002] has found four related project characteristics: 1) team size, 2) project criticality, 3) risk and uncertainty and 4) activity scope. The team size effects on suitable communication practices, also the location of team members must be taken into consideration. Highsmith suggests that the criticality of the project could be simplified to the question: can software failure threaten life? However, this seems to be oversimplification from the viewpoint of medical device regulations, where medical device safety classification controls the applicable quality management requirements. Naturally, risk and uncertainty characteristics are related to project criticality, but Highsmith uses these terms to describe the volatility of the project. Finally, the activity scope describes the limits of the development model lifecycle - which activities are defined to be in the lifecycle of the model and which are defined to be out of the scope. For example, the activity scope in medical device domain is fairly wide on account of regulatory requirements that must be fulfilled within the development model.

6. Towards a reference implementation model

The proposed reference implementation model is presented in this chapter. The idea of the reference model is not to cover every aspect of an agile methodology. Instead, the intention is to present concrete ideas how the risk management process regulatory requirements can be fulfilled by using commonly known best practices of agile perspective. The model is a small set of easily understood conventions.

The implicit nature of regulatory and agile perspective is to be adaptable to the specific context of use, thus the development process model and the risk management process should always be tailored for organizations individual needs. With respect to this, the reference implementation model is not excessively specific about the implementation details and thus can be applied to all organizations that wish to claim conformity to ISO 14971.

The general agile terminology used in this chapter is derived from AAMI TIR45 [2012] where applicable. As AAMI TIR45 does not cover the whole scope of agile perspective, certain terms are also derived from two of the most commonly used agile frameworks, Scrum and XP [VersionOne, 2015].

6.1. Basis of the model design

Arguably the most important goal when designing the risk management implementation model is to ensure that the model is in compliance with ISO 14971 and with the other relevant standards. The scope of this thesis is limited to ISO 14971, but the implementation is not practicable or useful if it is conflicting with the other parts of the regulatory framework. For this reason, special focus is given to ensure that the recommendations and ideas provided in this thesis are not conflicting with the other requirements of the regulatory framework.

On the other hand, from the agile perspective, the product that is compliant but is never delivered to the customer produces no value. Thus the primary goal of the development is to produce working yet safe product. In conclusion, this primary goal of delivery is actually supported by the secondary goal of compliance - both are needed to satisfy the business requirements of medical device domain.

The most efficient way to implement risk management process is to integrate it into the overall product development process [Schmuland, 2005].

6.1.1. Adding value with lightweight risk management process

When considering the implementation of ISO 14971 in compliance with agile principles without prior experience, it is easy to make a hasty conclusion and to view the requirements of the standard as a burdensome and extra work. For this reason, it is extremely important to design the development process in a way that only the absolutely required regulatory tasks are added. Furthermore, the implementation of these tasks

should be as lightweight as possible and, at the same time, they should be implemented in which way they produce maximum value to the development team.

Schmuland [2005] states that risk management, if done properly, can powerfully complement the overall development process by improving its ability to identify the factors that can fail to meet the customer requirements. However, it is unfortunately common to overreact and create particularly detailed process that cannot be practically followed. According to Schmuland, the value-added risk management process consists of three fundamental factors:

1) The intent of process is not lost in the details and process is not impractical.
2) Process deliverables actually add value.
3) The people performing the risk management activities must have a sense of unity and common interest towards the process goals.

The nature of the software development is not similar that manufacturing production where large quantities of same product is reproduced and where detailed process is the most efficient option. The nature of the software development resembles craftsmanship more than a static manufacturing assembly line [Hunt & Thomas, 2000]. Consequently, too detailed process reduces the focus on essential activities, thus every process step should produce value.

The value-adding process implementation is one of the most important design goals and principles when aligning the agile and regulatory perspectives, thus it is a basis of the implementation model design. It is essential to find a right balance point where business and regulatory goals are met.

6.1.2. Necessary compliance

Highsmith [2004] introduces the concept of necessary compliance. According to Highsmith, there are good reasons for some compliance activities - the medical device regulatory framework certainly being one of them. Highsmith even suggests that every organization has legitimate compliance reasons. When these reasons are present, they must be kept from impeding the development process since usually these compliance activities are not producing customer value. On the contrary, they are the cost of doing business in the selected domain [Highsmith, 2004]. The concept of necessary compliance is a strategy to do compliance activities just enough, but no more than is absolutely required - they are considered to be overhead and thus subject to minimization.

When following the ISO 14971 standard, certain fundamental regulative requirements must be satisfied. The risk management system must be present during the whole lifecycle of the medical device [AAMI TIR45, 2012]. As the risk management process of ISO 14971 is part of the quality management system, the quality management system must also be present. In practice, this means that the quality management system must be implemented in compliance with ISO 13485 and it must be properly understood

by the organization. The maturity of the quality management system implementation must be at adequate level to be effective and controlled.

The agile values and principles of the development process can be fulfilled only without compromising the requirements of the regulatory framework. According to AAMI TIR45 [2012], agile must meet the requirements of the quality management system. Agile can adapt, whereas the minimum requirements of the regulative framework cannot.

6.1.3. Supportive project management tool and documentation

In practice, some supportive project management tool should be used to track both the development issues and the risk management tasks. According to VersionOne State of Agile Survey Report [2015], the most popular tool used in agile project management was, quite surprisingly, Microsoft Excel. While Excel can undoubtedly be useful tool when calculating estimations and designing releases, it has certain limitations regarding to ability to produce project documentation.

When following regulations, the agile project management tool is needed to produce evidence that risk management process has been followed as defined in the risk management plan. For the solution to be as lightweight as possible, the risk management process flow should be embedded into the development process flow and the selected tool should support this practice. For example, the second popular agile project management tool by the VersionOne survey [2015], Atlassian JIRA, offers these features as built-in. In addition, several other suitable tools exist in the market.

Besides producing evidence about followed practices, the project management tool is useful also in other ways. It can be used to bring discipline to a development work by guiding the team to follow the preferred process and workflow. In this way, the process is visible to the whole team and other interest parties. If the information security policy allows, some up to date views can be revealed to the customer, which further improves the visibility of the project. Furthermore, the project management and issue tracker tool can be used as a primary documentation tool of the project.

Agile values “working software over comprehensive documentation” [Beck et al., 2001]. With respect to this value statement, the documentation produced during the agile project should be valuable to the development team. When properly using correct project management tool, the documentation is semi-automatically generated by the tool during the process – it is a byproduct of the development process. Similarly, certain software design related documentation can be automatically generated from the source code and executable specifications itself can document the business requirements of the product at the same time while being automated tests. Regardless of the origin, the documentation must meet the requirements of the regulatory framework and it must be understandable for regulators [AAMI TIR45, 2012].

6.1.4. Focus on safety by team collaboration

ISO 14971 defines that risk management activities may be carried out only by the personnel with appropriate training. In addition, these persons must be named in the risk management plan. This can become a bottleneck for the development process if risk management activities block the development, or management problem if the risk management activities are carried out in a late phase of the development. Therefore, it is important that risk management activities are done simultaneously and parallel with the software development.

A practical approach is that every member of the development team has an authorization to identify potential hazards at any phase of the development by entering the findings to the management tool, but only the trained members are allowed to do further analysis. It is important to notice that the requirement for training in this context is related particularly to risk management, in contrast with the technical aspects of the development. As a result, the person carrying out the risk management activities might need consultation about the technical details from other members of the development team.

6.1.5. Commitment through understanding

Regardless of the details of the development model implementation, the team must be trained to understand the basic fundamentals behind the model design. The model and the processes cannot function optimally if the development team is not fully committed to follow and endorse them. As noted in the previous chapter, from agile perspective the team owns its processes and workflows. This idea should be cherished even when the risk management requirements must be met. The team needs to understand also the goals and values of the risk management system in order to be able to continuously improve the development process while, at the same time, taking responsibility for the regulatory scope. This aim is supported by the common agile practice of collective ownership of design, code and tests. With collective ownership, the expertise is distributed throughout the whole team [AAMI TIR45, 2012].

It is important to notice that ISO 13485 requires development team members to be trained in their tasks and this training activity must be documented. The practices of collective ownership, pair working and peer review can be effective when fulfilling the regulatory requirements.

6.1.6. Executable requirements

The agile practice of executable requirements is a procedure where product requirements are written in formal, domain specific language and can be executed as automatic tests to ensure the correct functionality of the product. This technique has a straight relationship to acceptance test-driven development methodology but it is not a direct synonym.

The concept *Specification By Example* (SBE) was introduced by Gojko Adzic [2011] and it is recommended by Kniberg [2011]. It is a set of practices and process patterns to implement agile acceptance testing and behavior driven development and to bridge the communication gap between business stakeholders and development team. The main aim of SBE is to ensure that right software product is delivered effectively and SBE promises to provide just enough documentation at the right time and to improve the quality of the software product. Furthermore, a living documentation system will be generated by following the SBE practices. Overall, SBE is compelling approach for agile executable requirements implementation.

Executable requirements will make the process of validating the software system cheap and efficient. The validation can be made repetitively during the development which decreases the feedback delay. The main difference between executable requirements and traditional unit and functional testing approaches is the business focus – the requirements are written as business requirements and, as a result, are accessible also to business users [Adzic, 2011]. In practice, system specifications are described with concrete examples that can be used to validate the functionality of the system. These specifications are automated and used as an executable acceptance test. The executable requirements are written collaboratively and can be maintained by every member of the development team.

Executable requirements must be frequently validated and updated to obtain the full benefits of SBE [Adzic, 2011]. During the development project, the domain knowledge of the development team deepens and the requirements must be updated to reflect these changes. This procedure ensures that the software is always behaving as expected and specified. Furthermore, executable requirements form an authoritative and reliable source of information that is easy to read and understand [Adzic, 2011]. The artifact of executable requirements is genuinely living documentation and important part of the delivery process.

In order to be used as a part of final requirement documentation, the executable requirements must be clear enough to be understood by the regulators. To facilitate readability, the requirements should focus the scope on to business requirements and use domain specific and ubiquitous language [Vernon, 2016]. The regulatory requirements do not prevent the use of executable requirements as a part of requirement documentation [AAMI TIR45, 2012].

6.1.7. Requirements as use cases

User story or plainly *story* are terms commonly used in agile practices to describe a lightweight requirement of certain functionality or feature that creates business value. Accordingly, term *story* is used also in AAMI TIR45 [2012]. A common format for user story is to define the user role, to describe the user goal and to explain the benefits achieved [Cohn, 2004]. The fact that the concept of story is used in AAMI TIR45

indicates that story should be suitable tool to be used in regulated environment. However, more formal approach of *use case* can produce some additional benefits.

Alistair Cockburn [2001] has created a template that addresses the different aspects of use case requirements in a comprehensive manner. According to Cockburn, the template is adaptive and can be tailored by the unique needs. Even though the template is fairly detailed and not particularly lightweight format, it is an excellent introduction to the approach. However, Jacobson et al. [2016] has further developed the concept to include newer related ideas making use cases to be well-suited for regulatory environment.

According to Jacobson et al. [2016], use cases includes all the techniques provided by the stories. Besides telling user stories, they offer more. For example, they offer support for architecture, design, test and user experience. Jacobson et al. defines six basic principles for use case adoption [2016, p. 63-64]:

1) Keep it simple by telling stories
2) Understand the big picture
3) Focus on value
4) Build the system in slices
5) Deliver the system in increments
6) Adapt to meet the team's needs

The analysis of the *Use Case 2.0* of Jacobson et al. reveals that the stories used in agile practices are actually included in the use cases as narratives with additional relevant information. This information includes, for example, relation to the big picture of the system, better organization of test assets, test-case generation and analysis, active scope management and easier identification of missing functionality. Furthermore, use cases undergo several state changes with respect to concurrent development cycle and the different use case slices progress in parallel and actually drive the development system. The set of all use cases forms the functional requirements of the system and can also be used in applications that are not user-intensive [Jacobson et al., 2016].

The characteristics of Use Case 2.0 as defined by Jacobson et al. are ideal from the viewpoint of the regulatory framework. Therefore, the concept is adopted to the reference implementation model.

6.1.8. Testing and verification

While test planning is not central issue in agile development, it is very important activity in regulatory perspective. Agile testing is done in different phases of the development cycle; therefore, it is crucial to plan testing carefully and to document these plans accordingly.

In agile practices, testing is used to ensure the correctness of the system and as a method to provide instant feedback during development. In practice, testing should be as automated as possible in order to enable efficient regression testing. In incremental

development model, efficient regression testing is needed to demonstrate the correctness of the software version that is going to be released [AAMI TIR45, 2012]. Besides the new features added, the previous features must also be verified. From the viewpoint of the regulatory framework, tests are as important as the actual code since every use case that gets implemented must also be verified. The practical method to make certain that use cases will be entirely tested is to use test-driven development and write automated tests in different abstraction levels of the system. The test results must be documented as evidence to prove that testing and verification activities are performed as planned.

To ensure that every requirement is verified, the requirement must be linked to verification record. When tests are used as a tool to provide verification, the test results must be traceable back to the requirement. The process to write and run tests is integrated to use case implementation process and can be controlled by definition of done - convention. As a result, traceability will be built-in to development process.

6.2. The essence of risk management process

As previously discussed, in order to be able to perform risk management activities efficiently it is important to understand the primary rationale behind the process. The main goal of the regulatory framework is to protect society against unsafe medical devices. With respect to this, the most important goal of the risk management process is to gather knowledge about the risks related to the product and to control the overall residual risk before releasing the device to production use.

In practice, the process model of ISO 14971 can be roughly divided to two parts: to activities that are performed during product development and to activities that are performed after development [Schmuland, 2005]. Naturally, this division is not that evident when following agile practices since the development phase and the phase after release are actually overlapping with each other. Table 2 presents all requirements for documentation related to medical device software risk management process.

With respect to agile philosophy, the risk management process should be transparent during the whole product development. This practice ensures that the possible safety issues are visible from the beginning and no risk related surprises emerge at the end of the development cycle. In addition, top management representative should be actively present in the risk management decision making during product development. This is important since the risk acceptability level is defined by the top management in the risk policy and should be endorsed accordingly.

Table 2. Requirements for documentation from ISO 14971 and IEC 62304.

Standard	Clause / Subclause	Requirements for documentation
ISO 14971	3.1	• risk management process
	3.2	• policy for determining criteria for risk acceptability
	3.3	• qualification records of persons performing risk management tasks
	3.4	• risk management plan (including changes)
	3.5	• traceability for each identified hazard to o risk analysis o risk evaluation o implementation and verification of the risk control measures o the assessment of the acceptability of residual risk(s)
	4.1	• implementation of the planned risk analysis activities • result of risk analysis
	4.2	• intended use • reasonable foreseeable misuse • other characteristics related to safety
	4.3	• known and foreseeable hazards (in normal and fault conditions)
	4.4	• hazardous situation(s) • risk estimations • system used for o qualitative or quantitative categorization of probability of occurrence o categorization of severity of harm
	5	• results of risk evaluation
	6.2	• selected risk control measures
	6.3	• verification of each implemented risk control measure • verification of effectiveness of each implemented risk control measure
	6.4	• residual risk evaluation

	6.5	• results of risk/benefit analysis (if performed, should generally be avoided)
	6.6	• review of the (unwanted) effects of the risk control measures
	6.7	• completeness of risk control
	7	• evaluation of overall residual risk acceptability
	8	• risk management report
	9	• evaluation of risks arising from production or post-production
IEC 62304	4.3	• software safety class assigned to software system • rationale for using lower software safety class for a certain software item (lower than the overall system)
	7.1	• potential causes of software item contributing to a hazardous situation • sequences of events that could result in a hazardous situation (for each software item)
	7.2	• risk control measures defined (for each software item)
	7.3	• evaluation of risk control measures implemented as software item (to identify new hazards)
	9.5	• problem reports and their resolution and verification

Schmuland [2005] has found four principles encountered when performing risk management activities:

1) All hazardous situations cannot be avoided. However, exposure to certain hazardous situation does not necessarily result in harm.
2) The severity and type of harm can vary within a certain harm occurrence.
3) Risk controls can limit the probability and/or severity of the possible harm. Nevertheless, all risk controls are imperfect and thus can fail.
4) Regarding the safety of the product, the overall residual risk is the most important factor over each individual harm scenario.

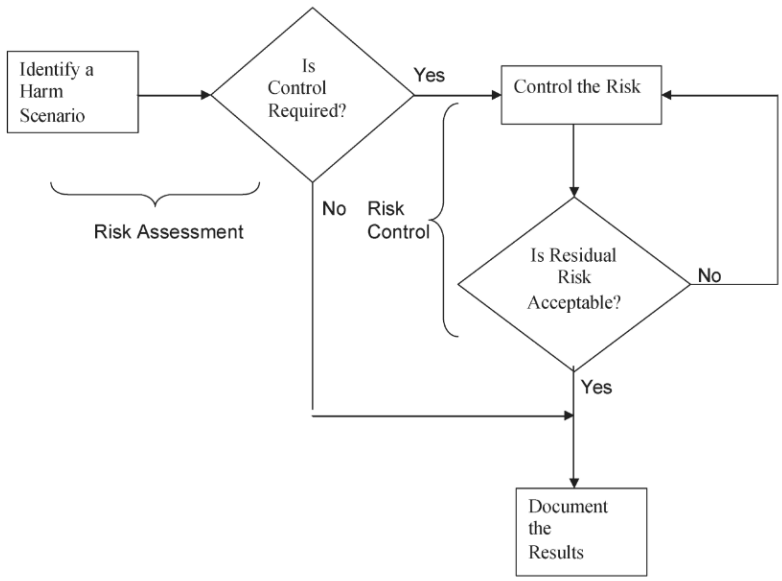


Figure 8. Essential product-development risk-management process [Schmuland, 2005].

As can be seen in Figure 8, not all harm scenarios need to be controlled. If the risk is evaluated to be acceptable, risk reduction is not needed and that specific harm scenario can be disregarded. However, even if the risk reduction is not needed from the regulatory viewpoint, there are no barriers to implement risk control measures, if judged to be suitable, outside of the risk management process. Naturally, it is only reasonable to try to reduce the risks to the lowest level practicable. It is possible to also include this approach to risk management process, ISO 14971 Annex D.8 provides guidance for appropriate implementation [ISO 2012].

Schmuland [2005] emphasizes the importance of certain characteristics of persons performing a successful risk management activities. First, product understanding is needed in order to understand the potential harm scenarios and suitable solutions. Second, advocacy is needed to be able to use good judgment when considering the right balance between business interest and patient interest. Finally, passion should be present to actively and honestly seek safety improvements, instead of just focusing to satisfying the regulatory agencies. According to Schmuland, this mindset will eventually lead to quality improvements and consequently to competitive advance in the market.

6.3. Risk management activities preceding the development phase

Certain amount of risk management activities must be performed before the active development phase. In regulated perspective, the software development planning is formalized activity and it must be documented accordingly. During software development planning, the risk management planning should also be conducted. The risk management planning should consider all the items that are required in the risk management plan.

Special attention should be given to risk acceptability criteria definition as the risk acceptability definition has a substantial impact to rest of the risk management process – the risk acceptability eventually decides the overall effectiveness of the risk management process [ISO, 2012]. As the ISO 14971 does not define the acceptability level, the decision is left entirely to the manufacturer. The possible methods determining acceptability level are, amongst others, to seek further guidance from applicable standards for that specific type of medical device, to use information from similar devices as a reference and to evaluate data from clinical studies. In addition, certain type of risks can be a matter of general interest thus additional weighting should be given to meet the expectations of public opinion. The acceptability level can be presented in a risk matrix.

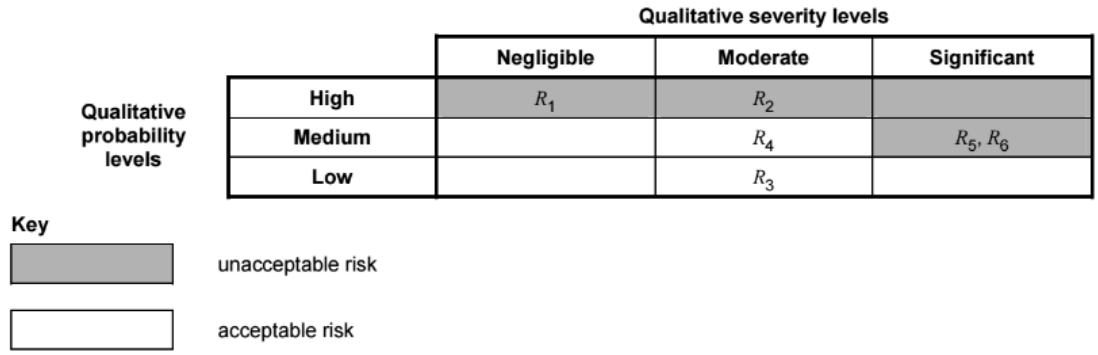


Figure 9. Example of qualitative 3 x 3 risk evaluation matrix [ISO, 2012].

Risk acceptability criteria for software-caused or software-controlled risks can differ from the matrix presented in Figure 9 as the probability of the harm cannot be estimated. In this case, the estimation is based solely on the severity of the harm [IEC, 2009].

Finally, system to collect evidence from performed risk management activities and to enable traceability from identified hazards to controlling risk control measures is required. The ISO 14971 term for this system is a risk management file and also this document can be embedded into the quality management system. As the software is a subject of constant change during the development, also the risk management file can change. In this case the risk management file must be versioned accordingly.

6.4. Medical device software risk estimation considerations

ISO 14971 requires that the risk management activity of risk estimation must be carried out, but it does not define the method how it should be performed. Risk estimation can be qualitative or quantitative and it is analysis of two distinct components: the probability of the occurrence and the severity of possible consequences. The estimation is based on available information or data. According to ISO 14971, the information and data be obtained, for example, from:

1) published standards,
2) scientific technical data,

3) field data from similar medical devices already in use, including published reported incidents,
4) usability tests employing typical users,
5) clinical evidence,
6) results of appropriate investigations,
7) expert opinion,
8) external quality assessment schemes.

Besides being a product of multiplication of probability of the occurrence and severity of the harm, the risk estimate can also be described differently, for example with two-dimensional risk chart [ISO, 2012].

The difficult in the estimation is that every hazardous situation is different. The systematic way to do estimation is needed.

6.4.1. Probability estimation

Quantitative categorization of probability level is recommended when suitable data is available. If data is not present, qualitative description should be given. Manufacturer can decide how many probability levels is needed, but at least three levels should commonly be used. The categories should be clearly defined in order to avoid misunderstandings and confusion. When evaluating the probability, circumstances and the complete sequence of events leading to harm should be considered - the concept of exposure is in central position. Seven approaches of ISO 14971 to probability estimation are:

1) use of relevant historical data,
2) prediction of probabilities using analytical or simulation techniques,
3) use of experimental data,
4) reliability estimates,
5) production data,
6) post-production information,
7) use of expert judgment.

From these approaches, the first three are complementary and have different strengths and weaknesses. It is recommended to use as many approaches as convenient since multiple iterations will increase the reliability of the results. The expert judgment should be used only as a last resort if none of the others are sufficient.

As can be seen in Figure 3 from Section 3.1, the probability of occurrence of harm consists of two components: probability of a hazardous situation occurring and the probability of a hazardous situation leading to harm. The original figure (*Figure E.1* in ISO 14971) is not included in the main part of ISO 14971, instead it is located in Annex E. Therefore, it is possible that certain risk management process implementations are realized without using this division.

The reliable probability estimation can be made when suitable data is available or when reasonable qualitative estimate is possible. However, this is not always the case,

there are certain situations where reliable estimations are particularly difficult to make. IEC [2009] states that there is no consensus for method of estimating the probability of occurrence of software failure. When the probability estimation cannot be reached, the risk should be evaluated on the basis of the harm alone. If the hazard is minor without significant consequences the risk can be judged to be acceptable. On the other hand, if the hazard has notable outcomes, the probability estimation should be based on a reasonable worst-case estimate. This means, in practice, that the risk control measure must prevent the harm scenario entirely or reduce the severity of the harm to acceptable level.

IEC [2009] states:

> "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."

Although the probability estimation cannot be given before or after the risk control measure implementation, it is evident that the probability of certain hazardous situation can reduce. The manufacturer shall demonstrate the effectiveness of the implemented risk reduction measure in order to prove that the risk acceptability level is achieved.

6.4.2. Severity estimation

The severity level categorization should be descriptive and based on a product characteristics. Manufacturer can decide how many categories is used and how they are defined. It is important that they do not include any element of probability and they are valid in clearly defined use conditions [ISO, 2012].

6.4.3. Overall residual risk estimation

Overall residual risk is combined impact of all the residual risks that still exists after risk control measures have been implemented. The overall residual risk can exceed the limit of acceptable risk even if individual residual risks do not. The high number of identified risks and the complexity of the system can increase the overall residual risk [IEC, 2009]. All risk control measures must be implemented before residual risk estimation.

Schmuland [2005] suggest that the overall residual risk estimation should be quantitative. In practice, quantitative estimation is the expected number of injuries for each of the specified harm over the whole lifecycle of the product - if the product is released with associated harm scenarios, it is only realistic to expect harms to occur with definite quantity. The quantitative estimation has certain benefits over qualitative estimation. First, quantitative estimation can be used as a benchmark to validate the effectiveness of the quality management system. Second, quantitative estimation can also be used to validate field performance of the device after the release, whereas qualitative estimates cannot be added afterwards to be compared to actual results. Finally, the

quantitative estimation is less abstract than qualitative, therefore it is effective method for clarifying and visualizing the risks related to the product.

6.5. Medical device software risk control considerations

If identified risk is evaluated to be unacceptable based on the criteria defined in risk management plan, risk control measures are needed. The evaluation is based on the acceptance criteria defined in risk management plan.

The suitable risk control measures must be identified for software requirement and software failure related risks. The suitable measures can increase detectability, reduce severity and/or reduce the probability of a hazardous situation [IEC, 2009]. It should be noted that the most effective risk control measure is the one that eliminates the harm scenario altogether. As defined in ISO 14971, the preferred method for controlling action is always a change in the design. When the design change is not viable solution, protective measures, for example automatic cut-offs or alarms, can be added. The last risk control measure, information for safety, is limited in effectiveness and should therefore be generally avoided [Schmuland, 2005]. The additional requirement from IEC 62304 states that if the risk control measure is implemented as a part of the function of the software, it must be included in the software requirements [IEC, 2006].

Information for safety promotes risk awareness. It explains the overall residual risk in a manner which allows users to minimize exposure to the residual risks. The information for safety might be communicated in different ways, for example with warning label, user interface or operation manual [ISO, 2012]. The applicable communication media should be analyzed and decided as necessary. Furthermore, the level of detail, the wording and understandability and the immediate recipients should be considered.

ISO 14971 offers a shortcut for risk control to the situations where risk reduction is not practicable. However, the risk/benefit argument should be cautiously used only in exceptional circumstances as the remaining risk seriously compromises the safety and thus suitability of the product. ISO 14971 defines that risk/benefit argument can be used only in situations where the product brings substantial medical benefit [ISO, 2012]. In practice, medical benefit denotes the likelihood and extent of improvement of health expected from use of the device. Usually this kind of medical device is a breakthrough product for which there is no alternative treatment available [Schmuland, 2005]. The decision to use risk/benefit argument is a matter of judgment by experienced person and requires knowledge about technical, clinical, regulatory, economic, sociological and political context [ISO, 2012]. Therefore, it is recommended that the decision is essentially accepted by the top management representative as it has such considerable effect to the overall quality of the product.

There exist several appropriate ways to control the risks related to architectural design. For example, critical components can be isolated and thus specific hazard causes

can be prevented. Effective method to control inputs from external interfaces is to use whitelisting that prevents all unknown commands and data formats [Pinto & Stuttard, 2011]. Suitable redundancy strategies should be used in the complex systems including critical interface dependencies.

The risks of detailed software design can be controlled with defensive design and coding practices. It is important to ensure that no unspecified functionality will be implemented [IEC, 2009]. In certain safety-critical situations, it might be necessary to clearly separate the safety-related and non-safety-related code. However, this practice has a high maintenance cost and thus cannot be commonly recommended.

6.6. Proposed development model

IEC/TR 80002-1 [2009] is a technical report that provides practical guidance for the application of the ISO 14971 to medical device software. The document presents a mapping grid between software development lifecycle and risk management in Annex D. While the grid does not include agile concepts, it is not intended to represent a strict sequential waterfall lifecycle and thus it is extremely useful as a reference when planning ISO 14971 implementation. The suitable agile concepts can be added as practicable. The proposed development model discussed in this section is influenced by the guidance provided in IEC/TR 80002-1.

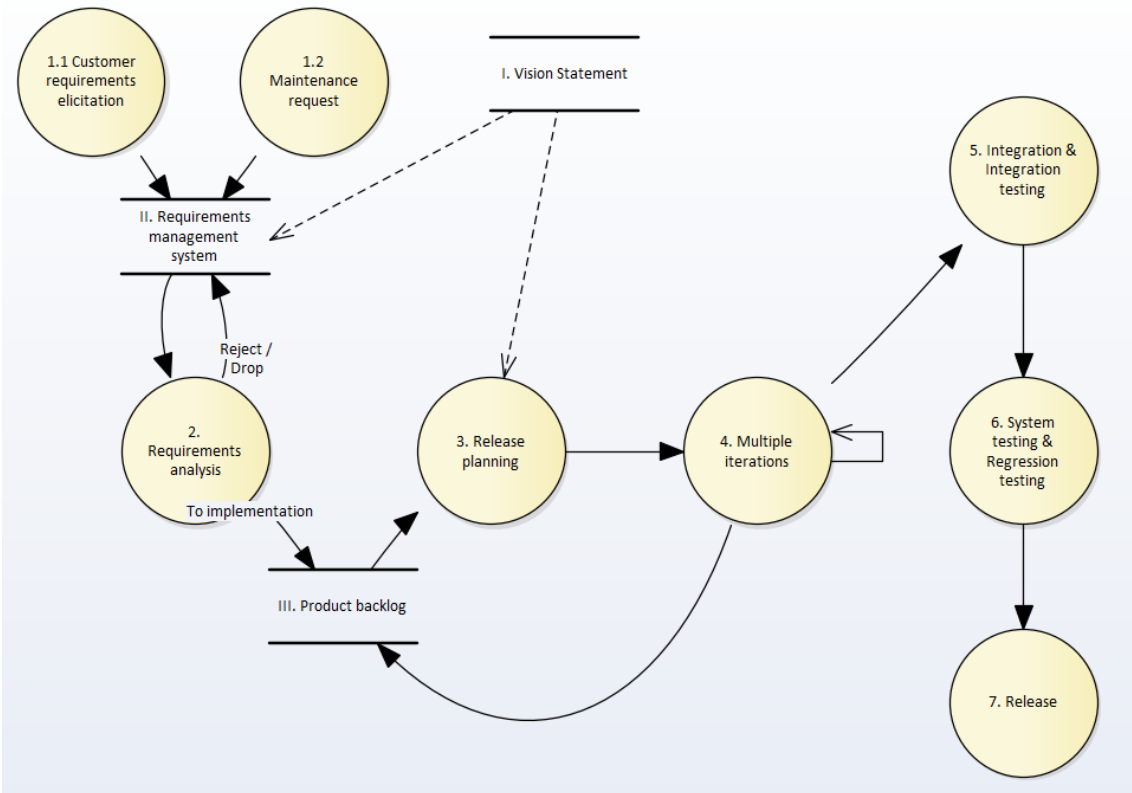


Figure 10. Reference development model: high level of abstraction.

Figure 10 presents the proposed development model at a high abstraction level. The regulatory requirements are addressed within different process phases as applicable. The requirement management activities are not addressed with more details in the scope of this thesis as they are not part of ISO 14971. Nevertheless, they are present in Figure 10 to secure the compliance with other related standards.

6.6.1. The Product vision statement as a documentation of intended use

The product vision statement is a relatively short, single page document that describes the essence of the product to be developed. The vision describes why the product should be manufactured and the problems or the opportunities it addresses [Shore & Warden, 2007]. The emphasis should be on value – how the software will benefit the customer and why it is valuable. Success criteria of the project should also be defined. Schwaber [2004] suggests that the major features and functions of the software could be listed in the vision statement. However, the implementation details should be scoped out as they should be defined by the development team later in the project.

ISO 14970 states that intended use of the medical device with characteristics related to safety must be documented. Reasonable agile process artifact for this documentation is the vision statement. In addition to intended use, the reasonably foreseeable misuse should be considered. In the context of software product, the misuse can happen, for example, as a result of configuration error.

As a recommended addition to previously mentioned vison statement sections, the business strategy benefits of the manufacturing organization should be addressed to ensure the profitability of the development project.

6.6.2. Product backlog

Backlog mechanism is very useful method to conduct agile requirements management activity. From regulatory perspective, requirements documentation must cover all areas that are required by regulations [AAMI TIR45, 2012]. Applicable documentation techniques are, for example, stories and use cases, textual descriptions, user interface mockups, UML-diagrams and control flows. With respect to agile perspective, the requirements documentation should be useful for the development team by providing actual value.

Product backlog is a prioritized list of all the features that might be needed in the product. It is an evolving artifact that obtains new items as the collective knowledge increases. A typical form of the valuable backlog item is epic, use case or story, other forms are used for non-functional requirements and infrastructure items. It is arguable if these technical issues should be placed to the backlog as independent items or if they should be presented only through a certain customer-meaningful requirement that actually has customer value. Non-functional requirements can be particularly expensive since they typically affect the design and testing of large number of other requirement

implementations. Therefore, the need and scope of certain non-functional requirement should be considered carefully before being accepted. One reasonable place for non-functional requirement is acceptance criteria (and definition of done) of the related use case. Regardless of the selected practice, the detected software defects yet not fixed and exploration activities should be placed to backlog as stand-alone items.

The activity of prioritization is important since the items on the top of the list are the most valuable features and thus should be implemented next. Furthermore, the backlog should be proactively administrated, managed and organized in order to stay in consistent state [Rubin, 2013]. The product backlog management is ongoing process lasting the whole lifecycle of the project.

The importance of ability to respond to change is emphasized in agile perspective, thus the change management process must be solid. The change request items can be placed to product backlog and prioritized accordingly. As with new features in backlog, the same development process can be used to implement the change requests. This practice ensures that change management process is controlled and change requests are well tested and verified.

6.6.3. Release planning and verification

Release is a coherent set of features which add the largest possible amount of value to the product. As the release activities do carry a fixed cost, it is important to understand all related cost components. When this information is visible, it is possible to seek ways to reduce the total cost. All the release activities that can be carried out at the end of each iteration should be placed accordingly. The routine to release often shortens the feedback cycle and reduces distress, risks and the amount of incorrect assumptions [Kniberg, 2011].

While the agile perspective has a vision that the product could be released after every increment, this approach might not be practical with every project in medical device domain. Even when the release activities are cut to the minimum, the inevitable cost can be too high to be paid after every iteration. The practical solution to this is to strive for working software after every iteration but to plan a broad level releases and to allocate needed time for release activities and verification [AAMI TIR45, 2012]. Thus, the end result of the iteration is near-shippable product that is missing only regulatory approval.

The final requirements of the release might not be ready at release planning phase. Instead, requirements evolve during every iteration between the release cycle. Nevertheless, the release planning is natural development phase to perform risk analysis concerning the software requirements. The challenge of evolving requirements can be addressed by keeping the risk analysis of software requirements open until the requirements are finalized at the end of the release cycle

In software requirement risk analysis, the known and foreseeable hazards are identified, including sequences or combination of events that could result in hazardous

situation. Moreover, the activities of installation, training, usage, upgrade and maintenance should be taken into account [IEC, 2009]. It is important to identify as many harm scenarios as possible. Finally, the identified risks should be estimated in order to enable the risk control measures at the later phases of the development, at the latest before the release cycle end.

Even though it is impossible to identify every possible harm scenario, the most severe and probable should be feasible to discover with expert knowledge. One common method for risk identification is the brainstorming session by multidisciplinary group. According to Schmuland [2005], the design inputs should be analyzed regarding to safety – if the product is consistently built based on the design input specification, is it safe? Secondly, the design outputs should be validated to find possible safety compromises, for example operations that need certain sequence of events or decisions to be safe.

Jeffries [2016] states that developers are often trained to design the architecture of the system at early state of the development. Upfront design is not ideal in an agile project since the requirement details are not known in advance. Therefore, the evolving agile architecture design should be done in several different development phases. Release planning phase should include a big-picture design that acts as a framework for more detailed design done in later phases of the development. The broad architectural vision is needed in order to be able to plan and estimate the release in a credible manner. The documentation of architecture must be complete at the time of release verification.

Architectural design should be a subject of risk analysis. Critical data, components and classes of defects are identified with associated hazards. External interfaces can act as a source of unpredictable input and timing and therefore they must be carefully analyzed. Complex systems can include critical dependencies to external interfaces and thus provide potential hazards. Furthermore, the performance criteria and limitations must be considered. The software safety classification (as required in IEC 62304) has a notable effect on how rigorous the risk analysis process is.

The agile practices that can be used to verify the architecture amongst others are continuous integration, automated testing and working software at the end of the iteration [AAMI TIR45, 2012]. These tools provide rapid feedback at regular intervals and therefore reduce the possibility of major problems.

The final software version to be released must be verified with regulatory approval before releasing. In practice, this means that all risk management requirements must be fulfilled and the completeness of risk management is verified. All identified hazards are covered and all risks have been closed - no unhandled risks can remain when decision to release is made. Furthermore, regression testing of the implemented risk control measures is performed with complete traceability and coverage analysis to ensure that all risk control measures are implemented and tested [IEC, 2009]. The final requirements of the release must be gathered and documented.

Before releasing the product, overall residual risk must be evaluated to be acceptable. In overall residual risk evaluation, the residual risk is viewed from a broad perspective and it requires knowledge, experience and authority from the personnel performing the task. ISO 14971 does not define a specific method to be used in evaluation. Suitable methods are, for example, event tree analysis, review for conflicting requirements (of risk control measures), fault tree analysis, review of warnings, review of operating instructions, comparing risks (with other similar devices) and review by application experts. It is recommended to use application specialists that were not involved in the product development to get an unbiased view.

Finally, the risk management report of the release must be compiled. These discussed risk management requirements and defined project adaptations should be formalized to definition of done convention of the release. As a result, the definition of done is a verification plan of the release.

6.6.4. Iteration planning with iteration backlog

Incremental development cycle is one of the most important practices of agile development. The development work of adapting and extending the software based on new requirements is done within short-duration iteration. Normally the iteration lasts between 1 to 4 weeks and the process is repeated multiple times during the single project. The short period improves team’s ability to respond to change, thus it is not generally recommended to use longer than one month cycles. Furthermore, strict timeboxing establishes an effective work in process (WIP) limit and thus reduces the amount of unfinished work. As a general approach and mindset, agile does not provide clear definition or strict rules for iteration execution. However, certain practices and principles from Scrum and XP are commonly used.

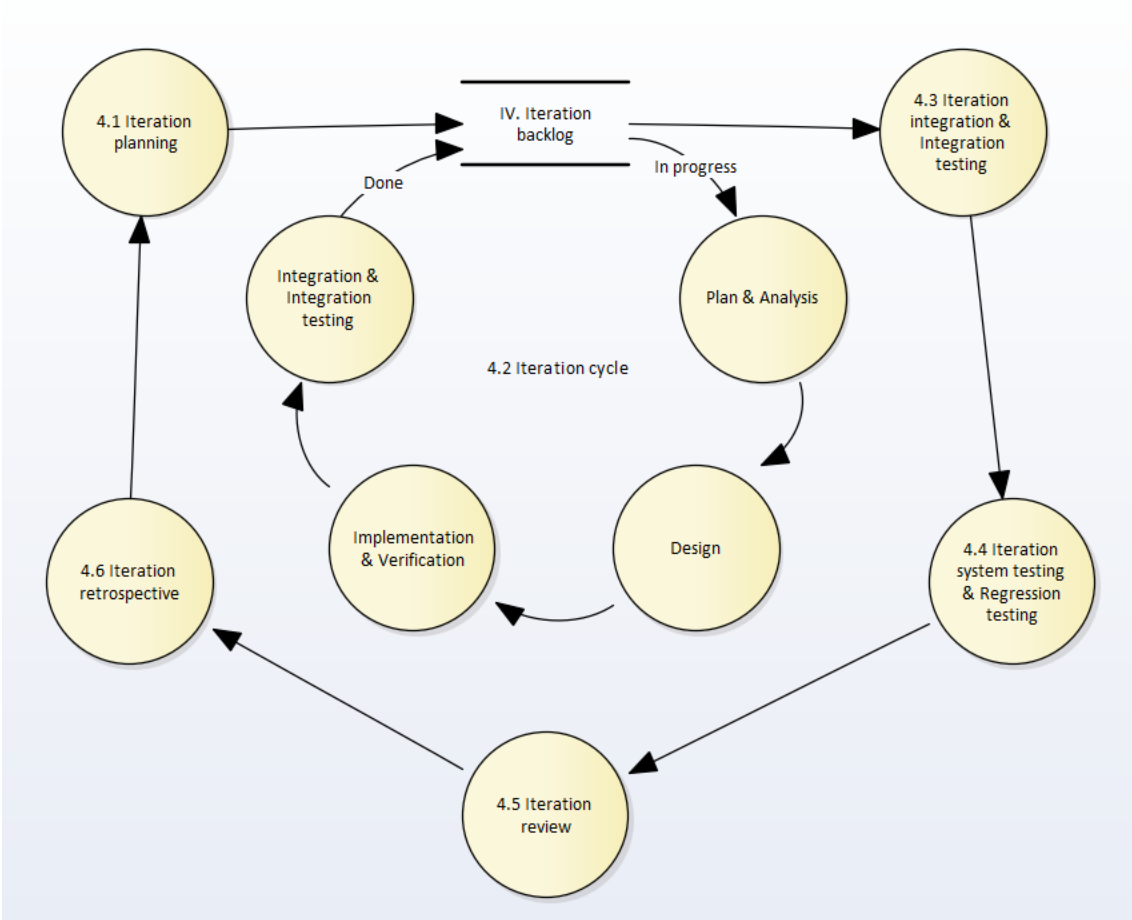


Figure 11. Reference iteration cycle.

According to The Scrum Guide [2016], the iteration has three important principles: 1) the general goal of the iteration cannot change during the iteration, 2) quality goals of the project cannot decrease during the iteration and 3) the scope of the iteration can be clarified and re-negotiated as knowledge is increased during the iteration. It is worth to notice that mentioned third principle is not shared between all agile and lean methodologies, although it is practical and realistic practice. In addition to these principles, iteration should produce largest possible amount of real value for customer or user.

The product backlog is the most important input when planning the iteration. Only realistic amount of work should be selected to be placed in to the iteration backlog. The iteration planning is done just-in-time at the beginning of iteration when the best possible information is available to decide the most valuable new features [Rubin, 2012]. With this timing the ability to respond to change increases.

One emerging trend in agile field is to avoid estimating. Jeffries [2016] states: “Estimates are likely to be wrong and they focus attention on cost of things rather than value”. However, a little amount of estimation is inevitably needed in order to decide the appropriate workload for iteration. Beck & Fowler [2000] suggest a simple practice called

Yestardays Weather where planned workload is based on the amount of work done within last iteration. Nevertheless, this practice cannot work if iteration length is not fixed.

Before placed in to the iteration backlog, the product backlog use case should have enough details that it can be prioritized and estimated, while the design details can still become more precise during the iteration. Kniberg [2011] suggest that special label "ready for development" could be used to indicate that the use case is ready to be added to the iteration. This concept has additional benefits when mapped to risk management process. For example, "ready for development" could indicate that risk analysis task related to that specific use case is started. This practice verifies that use case can not enter to iteration without being a subject of formal risk management analysis activity. Similarly, the importance of prioritization should be emphasized since it assures that the iteration produces the best possible value.

Jeffries [2016] argues that agile project needs to have a high quality architectural design continually throughout the whole project. Therefore, architectural evaluation, design and refactoring activities must be carried out in every iteration – at the very least to ensure that new functionality implemented in the iteration is compatible with the existing high level architecture vision. During the iteration execution, the architecture vision guides the way how specific use cases are being implemented [AAMI TIR45, 2012]. At the same time, the implementation details of a certain issue might influence the architecture. Consequently, the architecture evolves in every iteration as necessary since high quality of the architecture is solid foundation of an agile project. The architectural design must be documented as required by the regulatory framework, including the details how the detailed design activities are performed during the development process.

When multiple teams are working in the same project, certain amount of coordination is needed to ensure that the items crossing the team boundaries are well synchronized. The best and most secure way to coordinate between the teams is to use automated tests [Jeffries, 2016].

The design in agile project should be simple enough just to produce the functionality that is needed in the current iteration artifact without any additional and speculative features that might or might not be needed in the future. This practice ensures that the team is always working with small and well-focused features that will bring largest possible amount of value for the current product. Furthermore, the team is able to break down the system to small pieces with robust and well-defined interfaces – it is important to notice that the interfaces are also design artifacts and must be documented accordingly. Verification of emerging design is done with automated tests and continuous integration.

As the design evolves, additional potential causes for hazards might emerge and the existing ones can change, resulting the implemented risk control measures to expire. Therefore, the risk analysis must be reconsidered when planning the iteration. When

identifying the hazards, it is useful to consider data, coding and transmission errors with hardware failures.

Overall, iteration and focus to working software are very practical and effective concepts to meet the regulatory requirement of integration strategy. In agile development, the integration and integration test activities are practically built in to daily development activities.

6.6.5. Iteration execution

The design and implementation of iteration backlog items must be verified during and after the iterations. The recommended agile practices to verify software units are automated unit, integration and acceptance tests in conjunction with peer review and continuous integration. The final requirements and the design of a use case are completed at the latest when the use case is done. The definition of done should be applied to ensure that all regulatory requirements are fulfilled, including the risk management activities. Different kind of deliverables can have different convention for definition of done and also item specific acceptance criteria can be used. Final requirements must be documented, approved and controlled.

Active involvement of the customer during the iteration ensures that the requirements are correctly understood by the team. In practice, it is usual that the customer access can be limited, so special attention should be paid to reduce the risk of incorrect requirements and to ensure the effectiveness of requirements elaboration.

The practice of daily builds is especially useful since it reduces the feedback cycle. Each change in the systems is verified with automated tests that are executed continuously in integration process. The intention of the tests is to verify the functionality and performance of the use cases. The executable specifications can be used to verify that business goals are delivered as defined. Depending on the complexity of the system, it is not always practicable to run all the integration tests locally several times during a day since running the tests can be time consuming activity. When this is the case, the continuous integration process should be placed to run in external server and to be triggered to start from version control system check-ins. This practice allows fastest possible integration feedback while not interfering the development of the next feature. Small size unit tests should also be used to verify the correctness of software units and functionality.

The tests that fail at some point in time after the implementation had already being accepted should be a subject of risk analysis. The similar code implementations should be searched and evaluated if found. The implemented risk control measures should be verified in all possible range of conditions and platforms [IEC, 2009].

Daily meeting is important inspect and adapt activity. The intention is to share common knowledge about the project and status of the current iteration and to plan immediate development activities. Besides focusing to the wider picture of the situation,

the synchronization helps team to avoid blocking and disrupting problems. Daily meeting can follow a practice familiar from Scrum where participants answer three questions: what did I accomplish yesterday, what do I plan to do today and are there any obstacles preventing me from making progress [Rubin, 2013]. However, it is also possible to use less formal approach if the objectives of the practice realize.

In addition to daily meetings, the common knowledge is increased with collective plan and code reviews. These practices ensure that mistakes are reduced and thus improve the quality of the product. These reviews should be organized in a manner that satisfies the regulatory requirements and they should be documented accordingly.

6.6.6. Iteration review with working software

The iteration review is an inspect-and-adapt activity that focuses to the product itself by presenting the increment done during the iteration, thus the timing of iteration review should be near the end of the iteration cycle. Iteration review acts as a frequent checkpoint that offers a transparent view to state of the project. The focus is strictly limited to the product since the main measure of the agile project progress is working software [Medinilla, 2012]. The practice to demonstrate the working version of the software at the end of every iteration forces the development team to finish open issues in time and clearly makes visible which tasks are finished and which are not, as only completed work should be presented.

The participants of the review should include all the interested parties of the project. In this way, the team is able to gather feedback about the product from the persons which are not usually available on a daily basis [Rubin, 2013]. The information flow in the review should be bidirectional – from development team to other participants as a demo of the software and back from stakeholders to the team as a feedback. The important aspect of the review is conversation and cooperation.

At the beginning of the iteration, the items in iteration backlog can be incomplete and at the relatively high on abstraction level. The detail level of the item elaborates throughout the iteration and the requirements should be clear and precisely defined at the end. As a result, the item can be used as a formal requirement for the product. With this practice, the iteration backlog use cases can be validated and verified at iteration review as required by the regulatory framework and they can act as an input for final requirements documentation. As incorrectly implemented features can result a considerable safety problem, the importance of the verification cannot be overly emphasized. Furthermore, the iteration review is natural development phase where all project deliverables are verified and product progression gets a formal approval.

In order to close and verify the risk management activities of the iteration, the complete traceability and coverage analysis should be performed [IEC, 2009].

6.6.7. Retrospectives to seek continuous improvement

The effectiveness of the adapted agile development method should be evaluated frequently in order to enable the continuous improvement of the model. This principle is defined in the Agile Manifesto. One common way to do the evaluation is to facilitate a team retrospective between the iterations. The relatively short time between retrospectives is recommended since it shortens the feedback cycle and the time in which the team is not operating in optimal level.

Furthermore, the effectiveness of the regulatory requirement implementation must be validated regularly. The quality management system is monitored by audits: internal audits which are performed by organization's quality management representative and external audits which are performed by a regulatory-approval body. Naturally, it is reasonable to periodically evaluate the effectiveness of the quality management system also in team retrospectives to verify that the process is compliant with the regulatory expectations.

The format of the retrospective can be formal or less formal. There exist several different retrospective techniques in agile methodologies to choose from, thus the practice can be tailored based on the unique needs of the team. The intention of the retrospective is to 1) inspect how the last iteration went (from the viewpoint of development process), 2) recognize successes and issues to improve and 3) create a plan for improvements [The Scrum Guide, 2016]. The overall goal is to improve the quality of the product and allow the team to become more effective and to get more pleasure from work.

The default focus of retrospective is to consider the whole development process during the latest iteration. However, at certain times it can be useful to limit the scope to specific and important issue team is currently facing. Special attention should be paid to find out the root cause behind the improvement need. The outputs of the retrospective should be concrete improvement actions which the team is committed to perform during the next iteration. Furthermore, the improvement actions should follow the common agile practice of plan-do-check-act (PDCA). When the root cause has been found, the corrective action and concrete improvement goal is planned (plan). The plan is executed during the next iteration and data about the effects of the action is collected (do). After the iteration, the data is analyzed to see if the process has improved and goal is reached (check). The final phase is to change the process if the plan was successful or to reject the changes if the goal was not met (act). From the regulatory perspective, the check activity should be used to formally validate the results and the act activity defines the new routine practice.

The quality management system requires the use of corrective and preventive action (CAPA) process as defined in ISO 13485. Therefore, all the development process anomalies should be documented and used as input to CAPA process.

6.7. Risk management activities following the release

Medical device risk management process continues throughout the whole lifecycle of the device. After releasing the product, it is essential to monitor the field performance to ensure it correlates with expectations [Schmuland, 2005]. Furthermore, IEC 62304 describes requirements for software maintenance and problem resolution processes and the risk management process should be integrated with them.

When considering software product, there are several technical solutions to perform active monitoring. In addition, the system to collect customer feedback is needed as a part of problem resolution process. From the viewpoint of risk management, the customer feedback is analyzed to re-evaluate the overall residual risk of the product. If new or changed safety issues arise, the information provided by the feedback system must be fed back as an input to risk management process and the review must be conducted and documented [ISO, 2012].

6.8. Potential software related pitfalls of risk management process

IEC/TR 80002-1 [2009, Annex C] provides a list of potential pitfalls related to medical device software risk management process. The list is relatively comprehensive and can be used as a checklist when evaluating the effectiveness of the risk management process.

The common pitfall related to software development lifecycle is that software related risk management activities are performed only late in the development. This is a worrying sign of the possibility that software features have been added to the product without properly considering the related hazards and hazardous situations. Equally problematic situations can occur if certain relevant misuse scenarios or use environments are not recognized.

As discussed earlier in Section 6.4, software product related risk probability estimation can be difficult. As a result, unrealistic low probability estimates can be applied, resulting unrealistic risk ratings and correspondingly inappropriate risk control measures. It should be noticed that as testing can never be exhaustive, it cannot eliminate the risk completely and reduce the related risk probability to zero. Furthermore, clinical knowledge is needed in order to be able to adequately estimate the severity of the risk.

The risk control measures implemented as software items can make the design more complex. It is obvious that the increased software complexity also increases the probability of software defects. Furthermore, all the risk control measures should be validated under a wide range of abnormal and stress conditions. A common characteristic of software is that certain defects are particularly difficult to produce in test environment. However, these risk control measures must also be verified as required by ISO 14971.

7. Conclusions

The aim of this thesis was to research similarities and differences between the ISO 14971 risk management process and agile principles. Therefore, the International Standard ISO 14971 was carefully analyzed and the requirements were gathered. Furthermore, the agile philosophy was examined through the value statements and principles defined originally in the Agile Manifesto.

Besides studying the theory behind these two concepts, the aim was to provide guidance and produce practical ideas for implementation of the risk management process that meets the regulatory requirements and, at the same time, follows agile values and principles. Accordingly, the example development process model was designed to be used as a reference implementation. However, the actual process implementation must always be designed based on the unique set of conditions that apply to the organization and circumstances.

By following the given guidance and instructions the medical device software manufacturers should be able to create the applicable risk management process and to claim conformity to ISO 19471. While it is possible to be adaptive with agile practices, the regulatory requirements must be followed in detail. For the organizations, it is essential to know and truly understand all the relevant regulatory requirements to ensure that the full conformity assessment can be made.

7.1. Agile and risk management process of ISO 14971

The intention of agile philosophy is not to completely ban processes and project documentation. Instead, the importance of personal interaction and communication is emphasised for knowledge sharing activity. Therefore, the agile processes should be designed to support direct communication between the participants and, from the regulatory viewpoint, there are no barriers for a communication-centric approach.

From agile perspective, the project documentation does not produce direct value to the end user and should be therefore limited to the minimum. It is not difficult to see that regulatory process requirements for documentation can be inefficient to a certain degree. Nevertheless, they do bring business value to the product since only few patients or healthcare professionals would want to use a medical device that is manufactured unfettered by the regulatory framework.

The ability to respond to change is one of the four value statements of the Agile Manifesto. This value is clearly contradicting with the regulatory perspective, where change is seen as a sign of process and design flaw. This raises a serious question whether agile practices are too undisciplined to be used in the regulated environment. However, this research indicates that there are no actual barriers to use agile planning practices in medical device software development. While the regulatory requirements do dictate certain additional subjects to be addressed in planning documentations, these

requirements can be fulfilled by extending related agile tools and techniques. The recommended way to address this issue is to focus on high-level regulatory development process documentation and to use suitable agile practices to generate plans for more detailed tasks. The overall software development plan should cover all regulatory requirements.

The primary goal of the regulatory framework is to ensure the safety of medical devices. Similarly, also the agile approach has a client focus - the aim is to maximize the value from the client's perspective. These objectives are well aligned as the safety is one of the most valuable features of medical devices. Using agile practices also brings certain risk management related advantages: collaboration and communication increase the overall quality and allow more efficient risk detection, continuous integration produces feedback and visibility to the effectiveness of risk management, and finally, continuous feedback and validation also increase safety and usability related knowledge.

7.2. Usefulness of the research

Agile practices do not directly address subjects related to risk planning or risk assessment. The previous research in the field has focused either on agile concepts or on the risk management process of ISO 14971. However, there is a lack of research on the integration. This research provides ideas and guidelines for the integration and proves that agile practices integrate well with risk management activities. Therefore, this thesis clearly makes a contribution to the field.

There is no end state in agile implementation process. In agile approach, the continuous process improvement is cherished in order to become more efficient with the development activities. Similarly, the quality management system and the risk management process are always under transition to become more effective.

7.3. Limitations and further research

The scope of this thesis is limited to the ISO 14971 risk management process. As a result, not all the requirements of the regulatory framework and the standards within have been taken into account in this research. For example, IEC 62304 includes a great number of requirements for design documentation and configuration management that have been scoped out. However, the risk management implementations based on the ideas of this research should be compliant with other related regulatory requirement standards as there are no known or recognized conflicts; requirements from other standards can be added where applicable. Therefore, the requirements of IEC 62304 and IEC 62366-1 would be relevant and interesting subjects to be addressed in further research.

7.4. Recommendations

When designing the process implementation, it is essential to thoroughly understand the goals and principles of the regulatory framework. The formal language and unique terms used in the standards can be difficult to understand at the beginning, although the

understanding of the complete context deepens with more experience with the concepts. The baseline is set with the accurate safety classification of the device and correctly selected set of applicable standards.

It is important to notice that the regulatory framework is constantly evolving and manufacturers are responsible for keeping their processes aligned with the most recent requirements. This is considerable managemental issue in organizations; thus, participating in the work of related associations in the field is strongly recommended. For example, the Finnish Health Technology Association (FiHTA) provides the most recent news and information concerning international medical device regulations.

References

AAMI TIR45:2012. 2012. Guidance on the use of AGILE practices in the development of medical device software.

Gojko Adzic, 2011. *Specification By Example. How successful teams deliver the right software*. Manning.

ANSI/AAMI/IEC 62304:2006. 2006. Medical device software - Software life cycle process.

Kent Beck & Martin Fowler, 2000. *Planning Extreme Programming*. Addison Wesley.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas, 2001. Manifesto for Agile Software Development. www.agilemanifesto.org

Alistair Cockburn, 2001. *Writing Effective Use Cases*. Addison Wesley.

Mike Cohn, 2004. *User Stories Applied: For Agile Software Development*. Addison Wesley.

European Commission. 1990. Council Directive 90/385/EEC on Active Implantable Medical Devices (AIMDD). eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:01990L0385-20071011

European Commission. 1993. Council Directive 93/42/EEC on Medical Devices (MDD). eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:01993L0042-20071011

European Commission. 1998. Council Directive 98/79/EC on In Vitro Diagnostic Medical Devices (IVDMD). eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:01998L0079-20120111&qid=1413308118275&from=EN

European Commission. 2011. Directive 2011/24/EU of the European Parliament and of the Council of 9 March 2011 on the application of patients' rights in cross-border healthcare

http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32011L0024

European Commission. 2016. Summary list of titles and references harmonised standards under Directive 93/42/EEC for Medical devices. http://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/medical-devices/

European Commission. 2016. Regulatory framework. ec.europa.eu/growth/sectors/medical-devices/regulatory-framework/index_en.htm

European Union. 2016. Regulations, Directives and other acts. https://europa.eu/european-union/law/legal-acts_en

Thomas H. Faris, 2006. *Safe and Sound Software: Creating an Efficient and Effective Quality System for Software Medical Device Organizations*. ASQ Quality Press.

Derek Flood, Fergal McCaffery, Valentine Casey, Ruth McKeever and Peter Rust, 2015. A roadmap to ISO 14971 implementation. *Journal of Software: Evolution and Process*. 27, 319-336.

FDA, U.S. Food and Drug Administration. 2012. Guidance for Industry, Third Parties and Food and Drug Administration Staff - Medical Device ISO 13485:2003 Voluntary Audit Report Submission Pilot Program. www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm212795.htm

Jim Highsmith, 2002. *Agile Software Development Ecosystems*. Addison Wesley.

Jim Highsmith, 2004. *Agile Project Management*. Addison Wesley.

Andrew Hunt & David Thomas, 2000. *The Pragmatic Programmer. From Journeyman to Master*. Addison Wesley.

IEC 62304:2006, 2006. Medical device software - Software lifecycle processes

IEC 62366-1:2015, 2015. Medical devices - Part 1: Application of usability engineering to medical devices.

IEC/TR 80002-1, 2009. Medical device software – Part 1: Guidance on the application of ISO 14971 to medical device software.

ISO 13485:2003. 2003. Medical devices - Quality management systems - Requirements for regulatory purposes.

ISO 14971:2012. 2012. Medical devices - Application of risk management to medical devices.

ISO 9000:2005. 2005. Quality management systems - Fundamentals and vocabulary.

ISO 9001:2000. Quality management systems – Requirements.

Ivar Jacobson, Ian Spence and Brian Kerr, 2016. Use-Case 2.0. *Communications of the ACM*. No. 5. Vol 59, 61-69.

Ron Jeffries, 2016. *The Nature of Software Development*. The Pragmatic Bookshelf.

Henrik Kniberg, 2011. *Lean from the Trenches*. The Pragmatic Bookshelf.

Henrik Kniberg, 2015. *Scrum and XP from the Trenches*. 2nd edition. InfoQ.

Dean Leffingwell, 2011. *Agile Software Requirements: Lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley.

Martin McHugh, Fergal McCaffery and Valentice Casey, 2013. Adopting agile practices when developing software for use in the medical domain. *Journal of Software: Evolution and Process*. 26, 504-512.

Ángel Medinilla, 2012. *Agile Management. Leadership in an Agile Environment*. Springer.

Marcus Pinto & Dafydd Stuttard, 2011. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition*. John Wiley & Sons, Inc.

Mary Poppendieck & Tom Poppendieck, 2003. *Lean Software Development. An Agile Toolkit*. Addison-Wesley.

Winston W. Royce, 1970. Managing the development of large software systems. *Proceedings of IEEE WESCON*. 26, August, 1–9.

Kenneth S. Rubin, 2012. *Essential Scrum*. Addison-Wesley.

Carl Schmuland, 2005. Value-Added Medical-Device Risk Management. *IEEE Transactions on Device and Materials Reliability*. Vol 5, No. 3, 488-493.

Ken Schwaber, 2004. *Agile Project Management with Scrum*. Microsoft Press.

Ken Schwaber & Jeff Sutherland, 2016. The Scrum Guide. http://www.scrumguides.org/scrum-guide.html

James Shore & Shane Warden, 2007. *The Art of Agile Development*. O'Reilly Media.

Ralph D. Stacey, 2007. Strategic management and organisational dynamics – The challenge of complexity to ways of thinking about organisations. Prentice Hall.

Tom Ståhlberg, 2015. *Terveydenhuollon laitteiden lakisääteiset määräykset kansainvälisillä markkinoilla*. Tekes.

The Standish Group International, 1995. The Chaos Report. www.standishgroup.com

Valvira, 2009. Terveysteknologia. www.valvira.fi/terveydenhuolto/terveysteknologia

Vaughn Vernon, 2016. *Domain-Driven Design Distilled*. Addison-Wesley.

VersionOne, 2015. 10th Annual State of Agile Survey. http://stateofagile.versionone.com/