Agile Medical Device Software Development

Introducing Agile Practices into MDevSPICE®

Fergal McCaffery, Marion Lepmets, Kitija Trektere
Regulated Software Research Centre & Lero,
Department of Computing & Mathematics
Dundalk Institute of Technology
Dundalk, Co. Louth, Ireland
e-mail:{fergal.mccaffery, marion.lepmets,
kitija.trektere}@dkit.ie

Özden Özcan-Top
Information Systems, Informatics Institute
Middle East Technical University
Ankara, Turkey
e-mail: ozdentop@gmail.com

Minna Pikkarainen
Centre of Health and Technology
University of Oulu, VTT, Technical Research Centre of Finland
Oulu, Finland
e-mail: minna.pikkarainen@oulu.fi

Abstract— **Medical device software is usually embedded within the overall system as one of the sub-systems. It needs to be integrated with other sub-systems such as the electrical and mechanical for a functional medical device to be developed. In order to develop a working medical device system through integrating its sub-systems, the sub-systems' requirements have to be derived from the overall medical device system requirements. The system requirements are continuously collected, analysed and built from the needs of different stakeholders such as patients, health professionals and other companies offering relevant devices, interfaces and software related to the medical device system under development. Various regulatory requirements have to be achieved for a medical device to be allowed market access. We have developed and piloted a medical device software process assessment framework called MDevSPICE® that integrates the regulatory requirements from the relevant medical device software standards. This paper describes how the MDevSPICE® framework has been designed to enable medical device software developers to produce software that will be safe and easily integrated with other sub-systems of the overall medical device. We also describe the lessons learned from piloting MDevSPICE® in the medical device industry and introduce an agile methodology together with its benefits and challenges. This paper outlines how MDevSPICE® can be extended to include agile practices to enable medical device software development to be performed in a more flexible manner.**

Keywords- medical device software; MDevSPICE®; medical device risks; medical device software development; agile methods; agile software development practices.

I. INTRODUCTION

Safety-critical software systems are increasingly affecting our lives and welfare as more and more software is embedded into safety critical systems such as hospital systems, medical devices, cars and airplanes. New approaches and international standards are being developed to ensure the safety of these systems before they are delivered. The integration of software into the complete medical device requires particular attention [1].

In order to market a medical device, the manufacturer has to satisfy a number of regional regulatory requirements commonly achieved by following international standards and guidance issued by international standardizing bodies and regional regulatory authorities. Additionally, in order for the solution to sell, the medical device also needs to fulfil the requirements of patients, health professionals and other medical system interface providers.

To help software companies in the medical device domain reach regulatory compliance, we have developed an integrated framework of medical device software development best practices called MDevSPICE®. This framework integrates generic software development best practices with medical device standards' requirements enabling robust software process assessments to be performed while preparing for a regulatory audit. The "SPICE" in MDevSPICE® reflects its foundation in the ISO/IEC 15504 (SPICE) [2] series of standards for process assessment. In this paper we describe the validation of the MDevSPICE® framework that provides evidence of the importance of traceability between the system and software levels of development. We also explain how the establishment of robust interface requirements for these two levels supports more effective software integration

In Section II, we provide an overview of the regulatory requirements medical device software development companies face before they are able to market their devices. In Section III, we describe the development of the MDevSPICE® framework. Section IV, outlines the lessons learned when validating the framework in expert reviews and in industry through MDevSPICE® pilot assessments. We also discuss the importance of traceability between system and software development processes when developing an embedded medical device software system as it increases the

safety and quality of the developed medical device. In Section V, we introduce an agile methodology by outlining its benefits, challenges and its suitability for medical device software development. The paper concludes in Section VI with areas of future research related to agile medical device software development.

II. MEDICAL DEVICE REGULATIONS

A medical device can consist entirely of software or have software as a component of the overall medical device system. In order to be able to market a medical device within a particular region it is necessary to comply with the associated regulatory demands of that region. Two of the largest global bodies responsible for issuing and managing medical device regulation belong to the central governing functions of the US and EU.

In the US, the Food and Drug Administration (FDA) issues the regulation through a series of official channels, including the Code of Federal Regulation (CFR) Title 21, Chapter I, Subchapter H, Part 820 [3]. Under US regulation, there are three medical device safety classifications: Class I, Class II and Class III. The medical device safety classification is based on the clinical safety of the device. Class I devices are not intended to support or sustain human life, and may not present an unreasonable risk of harm. A thermometer is a Class I device. Class II devices could cause damage or harm to humans. An example of a Class II medical device is a powered wheelchair. Class III medical devices are usually those that support or sustain human life, and are of significant importance in the prevention of human health impairment. An example of a Class III device is an implantable pacemaker. All implantable devices are Class III medical devices as the surgery required carries with itself additional high risks from anaesthesia and possible infections that go beyond the safety risks of the medical device.

In the EU, the corresponding regulation is outlined in the general Medical Device Directive (MDD) 93/42/EEC [4], the Active Implantable Medical Device Directive (AIMDD) 90/385/EEC [5], and the *In-vitro* Diagnostic (IVD) Medical Device Directive 98/79/EC [6] - all three of which have been amended by 2007/47/EC [7]. Similarly to the US, the EU device safety is also based on the clinical safety of the device embodying similar classifications and limitations, where Class I in the EU corresponds to Class I in the US, Class IIa and IIb to Class II, and Class III to Class III.

A further safety classification applies to the software in medical devices as outlined in IEC 62304:2006 [8], where the safety classification is determined based on the worst possible consequence in the case of a software failure. In the case of failure of software that is of safety Class A, no injury or damage to the health of a patient can occur. When software of safety class B fails, injury may occur but it is not serious or life-threatening. Class C medical device software is the highest risk and in the case of failure of such software death or serious injury can happen. Depending on the functionality of software within the medical device, the software safety classification may vary from the overall medical device safety class. When software involves critical functionality of the medical device, it will carry the same classification as the device, i.e., Class C software in a Class III device. The safety classification of software may be lower but cannot be higher than the overall medical device safety class, e.g., software of safety Class B, may be embedded in Class III device but there cannot be software of safety Class C, in a Class I or Class II device.

Medical device manufacturers in the US as well as in EU must satisfy quality system requirements to market their developed devices. In the medical device domain, ISO 13485:2003 (ISO 13485 from hereon) [9] outlines the requirements for regulatory purposes from a Quality Management System (QMS) perspective in medical device domain. ISO 13485, which is based on ISO 9001 [10], can be used to assess an organization's ability to meet both customer and regulatory requirements in the medical device domain. ISO 13485 does not, however, include requirements for software development. IEC 62304, which can be used in conjunction with ISO 13485, does offer a framework for the lifecycle processes necessary for the safe design and maintenance of medical device software. As a basic foundation, IEC 62304 assumes that medical device software is developed and maintained within a QMS such as ISO 13485, but does not require an organization to be certified against ISO 13485. Therefore, IEC 62304 can be considered to be a software development specific standard supplement to ISO 13485, similar to ISO 90003 for ISO 9001.

IEC 62304 is based on ISO/IEC 12207:1995 [11], which although a comprehensive standard for software development lifecycle processes, has effectively been decommissioned following the publication of the more extensive ISO/IEC 12207:2008 [12]. Furthermore, other developments in the ISO and IEC communities for software development, such as ISO/IEC 15504 [13], have provided significant additional levels of software process detail to support ISO/IEC 12207:2008. IEC 62304 is a critical standard for medical device software developers as it is the only standard that provides recommendations for medical device software implementations based on the worst consequences in the case the software failure causing hazards. For general medical device risk management, IEC 62304 is used in conjunction with ISO 14971 [14] and IEC 80002-1 [15] that provides guidance on the application of ISO 14971 for software development.

Since IEC 62304 considers a medical device system to consist of software as a sub-system, the system or product level requirements are not included within IEC 62304 but instead within the medical device product standard of IEC 60601-1 [16]. Due to the increasing importance of usability of devices within the medical device industry, organizations should also adhere to the medical device usability engineering process requirements outlined in IEC 62366 [17]. When the Medical Device Directives were amended in 2007 [6], this defined standalone software to be a medical device in its own right. Previously, software had always been seen as a subsystem embedded in a medical device. This amendment revealed a gap in international standards as none of the published standards were addressing the concerns for standalone software as a medical device. Today, IEC CD 82304-1 [18] applies to the safety of healthcare software that

is designed to operate on general purpose IT platforms and that is intended to be placed on the market without dedicated hardware, e.g., iPad applications.

All companies planning to market a medical device in the United States need to register their product with the US FDA. Most Class I devices can be self-registered but most Class II devices require a 510(k) submission. For Class III devices, a Pre-Market (PMA) submission is needed. To support manufacturers in addressing the relevant guidance, the FDA has issued an overview of their guidance documents for medical device manufacturers and software developers [19]. The FDA Guidance on Premarket Submissions [20] provides guidance and recommendation for premarket submissions for software devices, including standalone software applications and hardware-based devices that incorporate software. Premarket submission includes requirements for software-related documentation that should be consistent with the intended use of the Software Device and the type of submission. The FDA Guidance on Off-The-Shelf Software Use in Medical Devices [21] was published in 1999 with the purpose of describing the information that should be provided in a medical device application that uses off-the-shelf (OTS) software. Many of the principles outlined in this guidance document may also be helpful to device manufacturers in establishing design controls and validation plans for use of off-the-shelf software in their devices. The FDA General Principles of Software Validation [22] outlines general validation principles that the FDA considers to be applicable to the validation of medical device software or the validation of software used to design, develop, or manufacture medical devices. This guidance describes how certain provisions of the medical device Quality System regulation apply to software. The scope of this guidance is somewhat broader than the scope of validation in the strictest definition of that term to support a final conclusion that software is validated.

The challenge that software development companies in the medical device domain face when they want to market a device is in the adherence to a large number of regulatory requirements specified in various international standards that can often become overwhelming. In order to help these companies better prepare for the demanding and costly regulatory audits, we developed the MDevSPICE® framework. MDevSPICE® includes requirements from all the previously mentioned standards and FDA guidance documents rendering the task of regulatory compliance much less complex. Following is a description of the development of the MDevSPICE® framework that integrates the requirements from various international medical device standards and guidance documents with the generic software development best practices while providing a possibility to assess processes.

III. MDevSPICE® Framework

This section describes the development of the MDevSPICE® process reference model, the MDevSPICE® process assessment model, the support MDevSPICE® provides for software and system integration, and the validation of the MDevSPICE® framework through pilot assessments in medical device industry.

A. Development of the MDevSPICE® Process Reference Model

A process reference model (PRM) describes a set of processes in a structured manner through a process name, process purpose and process outcomes where the process outcomes are the normative requirements the process should satisfy to achieve the purpose of the process. In order to develop a PRM that integrates requirements from various standards allowing the processes to be evaluated in terms of their achievement of their purpose statements, we followed the format of the process description illustrated in ISO/IEC 24774 [23]. With that in mind, we first mapped and integrated the requirements from ISO/IEC 12207:2008 and IEC 62304 into what today is called the PRM for IEC 62304 that also reflects the updates to ISO/IEC 12207 from the 1995 to the 2007 version. A systematic approach of memoing and constant comparison, which is based on the principles of Grounded Theory [24] was followed when developing the PRM, further details of which are to be found in [25. The Process Reference Model of IEC 62304 was published in June 2014 as IEC TR 80002-3 [26].

While IEC 62304 describes only the software lifecycle processes, additional processes should be in place for system development in the case where software is not embedded as part of an overall medical device. These additional processes were derived from ISO/IEC 12207:2008. Design and development related requirements from ISO 13485 and ISO 14971 were also added to the MDevSPICE® Process Reference Model. Both ISO 13485 and ISO 14971 are *de facto* standards for medical device software organizations. ISO 13485 requirements are primarily related to system level processes and ISO 14971 is concerned with risk management (and therefore aligned with the Software Risk Management process of the PRM.

The final MDevSPICE® PRM consists of 23 processes of which 10 are system lifecycle processes, 8 are software lifecycle processes and the remaining 5 support both the system and lifecycle processes as can be seen in Figure 1.

The MDevSPICE® PRM was then extended with additional elements to create a process assessment model (PAM). The aim of the MDevSPICE® PAM is to provide a comprehensive model for assessing the software and systems development processes against the widely recognized medical device regulations, standards and guidelines that a software development organization in the medical device domain has to adhere to. The MDevSPICE® PAM, similar to ISO/IEC 15504-5 (SPICE) [26], has two dimensions – a process dimension and a capability dimension. The process dimension lists three groups of processes from various models and standards, i.e., systems lifecycle processes, software lifecycle processes and support processes. Each process is described in terms of a Process Name, Process Purpose, Process Outcomes, Base Practices, Work Products and Work Product Characteristics.

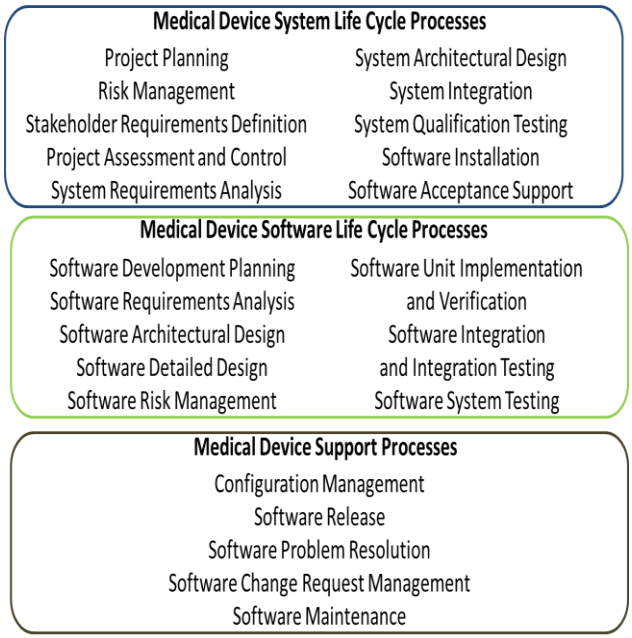

Figure 1. Processes of MDevSPICE® PRM

B. *Development of the MDevSPICE® Process Assessment Model*

The MDevSPICE® PRM is based on IEC 62304, ISO/IEC 12207:2008, ISO 14971 and ISO 13485. The MDevSPICE® PAM then extends this PRM with base practices and work products, some of the latter also being normative as they are described in IEC 62304, ISO 14971 or ISO 13485 as requirements. Where process outcomes are derived from ISO/IEC 12207:2008, their corresponding base practices and work products are derived from ISO/IEC 15504-5. Where process outcomes are derived from ISO 14971, their corresponding base practices are derived from IEC 80002-1. In addition to these sources, FDA guidance on premarket submissions, software validation and off-the-shelf software have been added to the informative base practices where the base practice did not already address the requirements of the corresponding FDA guidance. Product safety requirements have been added to the MDevSPICE® PAM from both IEC 60601-1 and IEC CD 82304-1, while the usability engineering requirements have been incorporated from IEC 62366.

The capability dimension of the MDevSPICE® PAM is derived directly from ISO/IEC 15504 together with the Capability Levels, Process Attributes, Generic Practices, Generic Resources and Generic Work Products.

While integrating processes from different standards and guidance documents for the MDevSPICE® PRM and PAM , a focus on the traceability between and within system and software lifecycle processes was maintained [27]. Both the FDA General Principles of Software Validation [22] and ISO/IEC 12207 [12] incorporate traceability of risks, changes and requirements throughout the development lifecycle. This interaction and traceability of requirements is a key enabler of subsequent integration, and it has a vital role to play in raising the safety of medical device software.

C. *MDevSPICE® Framework's support for integration*

The MDevSPICE® framework contains key facilities for integrating medical device software. Since MDevSPICE® is grounded in IEC 62304, the software sub system decomposition is consistent with the requirements of IEC 62304, meaning that the language of a *software unit*, a *software item* and a *software system* is adopted.

A software system is the integrated collection of software items to accomplish a specific function or set of functions; a software item is any identifiable part of a computer program; and a software unit is a software item that is not subdivided into other items. This software system hierarchy has an important role to play when a software developer wishes to decompose a system into parts of varying software safety classification. A benefit of such decomposition is that those parts of the software subsystem that are vital for safety (and which require additional safety activities when under development) can be isolated until they are later integrated with the other software components. It is also important that when the components are integrated that the safety implications are reflected in test cases that are pre-defined, then tested and the results are checked to ensure that they match the expected results. Otherwise sign-off cannot take place at the various levels – unit tests, integration tests and system tests.

Integration activities in the MDevSPICE® framework start by integrating software units into software items, and thereafter software items are further integrated with each other (and possibly with other units as well) into the software subsystem (which in turn is integrated into the overall medical device system). In other words, there are several levels of integration and they must take into consideration the safety implications at each step. It is further the case that the bi-directional traceability of requirements (including requirements related to safety) from the product level right down to the individual software unit level is supported in MDevSPICE® thus further enhancing medical device software safety at the integration stage and beyond.

D. *Piloting the MDevSPICE® Framework*

The MDevSPICE® framework has been validated in various stages of its development by different parties through both international expert reviews and industrial trials. The foundation of the MDevSPICE® PAM, IEC TR 80002-3 (the development of which was led by the authors), was published after several iterations of development and analysis by the standardization working group responsible for the publication of IEC 62304 (i.e., ISO/IEC Sub-Committee 62A, Joint Working Group 3). An international standard is published only after the national delegates of the standard's working group have agreed on every detail of that standard.

In addition to working with the international medical device standards community, the MDevSPICE® PAM has also been developed together with and analysed by experts of the Working Group 10 of ISO/IEC Joint Technical Committee 1, Sub-Committee 7, responsible for the development and maintenance of the series of process assessment standards. These standards are currently being revised from ISO/IEC 15504 series to ISO/IEC 330xx series of standards. MDevSPICE® framework keeps abreast of these updates as well as with the updates of any other standard and guidance document information, which is contained in the MDevSPICE® framework.

Upon successful completion of international expert reviews, the MDevSPICE® process assessment framework was then validated in the medical device software industry through pilot assessments over the past two years. MDevSPICE® process assessments were conducted in different types of organizations: (1) a small software company wishing to supply software to a large medical device manufacturer who wants them to demonstrate that they are capable of developing safe medical device software and provide the medical device manufacturer with a feeling that they will not jeopardize the safety of their overall medical device or the reputation of their organization; (2) three different assessments (across a 2 year period) were performed in two different international sites of a multinational medical device manufacturer who wants to ensure that they are incorporating best practices within their software development processes to not only achieve regulatory compliance but also reduce the likelihood of recalls through developing better quality and more robust software; (3) a software development company seeking to achieve regulatory compliance against IEC 62304 so that they can become medical device software suppliers; and (4) a large automotive manufacturer experienced in developing safety-critical embedded automotive software now wishing to also develop embedded medical device software.

IV. LESSONS LEARNED FROM PILOTING MDEVSPICE®

As a result of the MDevSPICE® pilot assessments we have witnessed different types of needs and challenges that companies face in medical device software development.

Companies that manufacture medical devices as well as develop embedded software for their devices, manage traceability and integration between systems and software lifecycle processes well. This might be due to systems and software engineers working closely together in building a safe medical device where the software developers are aware of the system risks and requirements.

For companies that develop and supply software to large medical device manufacturers it can be very difficult to become aware of the overall system level requirements including the requirements of end users, e.g., patients, health professionals and related interfaces, as well as the risks before the software development project commences. Medical device manufacturers working on innovative devices are sometimes reluctant to provide their software subcontractors with the details of their device design and end user requirements as this could jeopardize device novelty or competitive advantage. Yet, the safety risks related to the performance of medical devices can outweigh such business risks when the medical device manufacturer has a proper legal know-how and proficiency about the market needs. When the system requirements are not provided to the software developers, the traceability engineering and integration of the sub-systems of the medical device will be hindered. Therefore, we would recommend medical device manufacturers to communicate with their software subcontractors more openly in order to best support risk and requirements management throughout their device design – even if this only encompasses those product requirements, which are related to the software requirements (and especially those, which are safety related). Although there is a potential issue in capturing, managing and changing requirements throughout the development of a medical device, the ultimate goal for all device manufacturers is to have a safe medical device on the market and not risk liability or damage of their brand as a result of a recall of a faulty device.

V. AGILE FOR MEDICAL DEVICE SOFTWARE DEVELOPMENT

It is generally believed that technology will automatically improve health care efficiency, quality, safety, and cost, however, few people consider that technologies may also introduce errors and adverse events. Nearly 5,000 types of medical devices are used by millions of health care providers around the world [28]. While this technology holds much promise, the benefits of the technology are not always realized due to poor technology design that does not adhere to human factors, a poor technology interface with the patient or environment and an inadequate plan for implementing a new technology into practice [29].

Future trends indicate that medical software and devices in which clinical decisions will be guided by individual patient preferences, combined personal and medical data as well as specific needs and values [30]. In this case, continuous requirements collection and involvement of different stakeholders such as patients, health professionals and interface providers can be seen as essential for the future success of medical device software development. However, the development of medical devices that target the needs of either the patient or the health professional can be difficult when adopting a traditional, plan-driven software development approach where all system requirements should be known at the beginning of the development process. We believe that agile software development methodologies could provide support in achieving this challenge when delivering medical device software. In the next sub-sections we describe agile software development together with its benefits, the challenges it presents when adopted in the medical device domain and a brief justification as to why agile practices should be integrated into the MDevSPICE® framework.

A. What is Agile?

In recent years agile software development methodologies have gained significant interest in the IT

community with proposed solutions to the problems of traditional, plan-driven software development approaches. "It's a framework, attitude, and approach to software delivery that is lean, fast, and pragmatic. It's no silver bullet, but it dramatically increases your chances of success while bringing out the best your team has to offer" [31].

Agile software development is a set of principles and practices used by self-organizing teams to rapidly and frequently deliver customer-valued software. It follows an Incremental and Evolutionary lifecycle, emphasizing close collaboration between the software development team, the customer, and other stakeholders. It is adaptable, emphasizing the need to adjust the principles and practices to fit the context and environment in which the software is being created [32].

B. Benefits of Agile

A priority of an agile methodology is to "satisfy the customer through early and continuous delivery of valuable software" [33]. There are many benefits that an agile methodology promotes, such as, improved quality, sustainable development, continuous attention to technical excellence as well as changing to this sort of approach is welcomed even late in the development. By adopting agile practices, the speed to market is improved, supporting the achievement of competitive advantage in the market.

Agile principles also dictate that "working software is the primary measure of progress" [33]. Unlike the extensive upfront planning and heavyweight processes and bureaucracy required for plan-driven development; agile development focuses on delivering highest business value to the customer through: short time-boxed iterations; receiving and providing fast feedback; collaborating with stakeholders, making use of self-organizing teams, embracing requirements changes, balancing up-front and just-in-time work, and favouring adaptive and exploratory development approaches [33]. A key factor in the agile process of system delivery is the close collaboration between clients (i.e., clinicians, patients and related interface providers) and developers, which assists decision making and optimizes the market value of the developed solution. The client–developer collaboration and the continuous requirements prioritization are also important parts of a typical agile requirements engineering (RE) approach [34].

C. Challenges with Agile Adoption

Despite the abovementioned benefits of agile methods in software development, the experience reports and case studies indicate that there are several challenges to adopting agile methods in the medical device development domain. These challenges can broadly be grouped into the following three groups:

1) Challenges in relation to the perceived unsuitability of agile software development approaches for safety critical domains because of the conflicts with satisfying regulatory requirements;

2) Challenges in relation to the tailoring of agile practices to conform with the regulatory requirements;

3) Challenges in relation to the acceptability of agile adoption when conflicts occur between executives/high level managers and development teams.

In this subsection, we are going to address the problems related to the perception of adopting agile practices for medical software development and we aim to change this perception through providing empirical support. We title each section with a different misconception:

1) Undisciplined nature of agile software development

2) Approaches vs the demands of a highly regulated medical device development domain:

"Discipline" against "agility" was first used by Boehm and Turner in 2003 in their book titled "Balancing Agility and Discipline: A Guide for the Perplexed" [3]. The traditional software development methods and quality standards (SW-CMM at that time) were defined as the disciplined side of the contradiction. They clearly stated that "*agility is the counterpart of discipline. Where discipline ingrains and strengthens, agility releases and invents*" [36]. Ambler, the author of the disciplined agile delivery approach, states that when "properly executed, agile is not an excuse to be undisciplined" [37]. High ceremony procedures of traditional approaches such as formal document reviews or formal document approval are a sign of bureaucracy rather than discipline [37]. The misperception of agile being an undisciplined approach could be due to its empirical nature, self-organizing teams and an emphasis on less documentation. On the contrary, agile software development methods have to focus more on establishing discipline than other approaches to achieve built-in quality and to remove the *cost of non-conformance* in the first instance. Ambler states that discipline in agile projects is what makes the difference between successful and unsuccessful agile adoption [37].

The discipline, which also means consistency is established in agile software development by people applying a set of rules and practices. One of the significant practices of agile software development such as continuous integration brings commitment and discipline to development teams. It was stated by Humble and Farley that when the necessary discipline for this practice is not adopted, the improvements in quality will not be as expected [38]. Continuous integration requires being disciplined in; refactoring, ensuring that the mainline is never broken, coding automated tests, and maintaining acceptance tests over a long term [38]. Having focused people, trusting and respecting each other in a safe environment where there is no hesitation to share ideas or no fear to fail is the start of the disciplined agile environments [37]. Therefore, *the undisciplined nature of an agile software development approach* is an expression of belief, not an expression of fact.

3) Documenting the evidence required by regulatory standards vs little emphasis of agile on documentation:

In medical device development projects, evidence is required in order to prove that the executed process ensures a

safe and reliable product. Essential phases of software development process including requirements, architecture, design and test phases are recorded in a traceable way from initiation to release of the product. In the CFR 21:820 Quality System Regulation of FDA, it has been stated that design and development planning, design inputs, design outputs, design reviews, design verification, design validation, process validation, design changes, traceability and much more have to be established, which means "defined, documented (in writing or electronically), and implemented" [3].

One of the four values of the agile manifesto states that *working software is preferred over comprehensive documentation* [33]. This would suggest that following an agile software development method would not support the development of sufficient documentation necessary to achieve regulatory approval or it could be misinterpreted that an agile approach emphasises "no documentation". From either of the perspectives, it would be reasonable to accept that the documentation required for regulatory purposes needs to be developed regardless of the SDLC (Software Development Life Cycle) adopted in the company [32].

It should be noted that plan-driven methods such as the V-model and waterfall model are well-suited to the addressing the documentation needs as the SDLC phases and process outputs are in correlation. Agile software development methods do not undermine the value of necessary documentation. In a user story mapping approach, it is stated that a story-driven process needs lots of documents to work but those documents don't always look like traditional requirements documents [39]. Furthermore, there is also no clear emphasis on traceability of the software development process either in the agile manifesto or in agile principles. Evidence obtained from the literature suggests that documentation and traceability concerns in agile projects are resolved by managing the artefacts with appropriate tailoring and using software tools effectively [40], [41], [42]. For example, to ensure that all the necessary documentation is achieved within a sprint, a person who is responsible for documentation and support was established as a permanent member of an agile project at the QUMAS medical company [41]. This enables both adherence to regulations and standards without slowing down the process. Also, *living traceability* was achieved with the support of the integrated tools of *Atlassian* in the same company, which enabled an accurate snapshot to be provide of the system in real-time.

Rottier and Rodrigues report that a Use Case document can be used for the validation of the medical product in an agile project with a supplement to a Software Requirements Specification (SRS) document, which details non-functional requirements [43]. Obviously, use of use cases instead of traditional requirement specifications makes a difference for *Cochlear* in terms of agility as it was mentioned. At this point, the agility level that was achieved with use cases needs to be evaluated. Manjunath, Jagadeesh and Yogeesh mention that user stories acquired from customers were documented in the SRS in another agile medical project [42]. The founders of a *user story mapping* approach, Patton and Economy, make an important statement for the use of user stories as software requirements: "Stories in agile software development get their name from how they should be used, not what you write down. If you are using stories in development and you are not talking together using words and pictures, you are doing it wrong" [39].

D. Agile in Medical Domain

Based upon the Chaos Report of Standish Group, among 1500 software projects developed between 2011 and 2015, 39% of all the software projects that were developed using agile software development methods were successful [44]. While agile software development projects were 3 times more successful than waterfall projects, the ratio of the "challenged projects" (52% and 60% for agile and waterfall projects, respectively) cannot be underestimated. Those challenged projects refer to projects that were delivered with incomplete functionality, or exceeded the planned budget or schedule. The ratios present that there are challenges in relation to adoption and adaptation of the agile practices, interpretation of the agility principles and mindsets in the IT and medical community.

The regulatory requirements and audits, that safety critical projects are subject to, bring more concerns about the applicability of the agile approaches in the field and increase the challenges. For Class II and Class III type projects and some of Class I type projects, the FDA requires formal approval of most of the steps and items in the SDLC. The reason why traditional approaches like waterfall or V-model are being used in medical device domain could be explained with the rigid predictability and linear flow that the models present. On the other hand, Sutherland states that according to the leading research and analysis firms, such as Gartner, Forrester Research and Standish Group [45], the old style work characterized by command and control and rigid predictability is *obsolete*.

Regulatory issues are not a barrier for the implementation of agile approaches [46]. It is indicated in a mapping study [46] that SCRUM practices could be successfully used in medical device software development. Similarly, Perline [47] recommends agile methods like SCRUM for lightweight and proven framework for managing work in the complex software development projects like those in the safety-critical domains.

The Association for the Advancement of Medical Instrumentation (AAMI) published a guidance for the use of agile practices in the development of medical device software [32]. The report (AAMI TIR45:2012) provides high level guidance of agile practices, which have been found useful and appropriate for medical device software development. The report is a good resource that states major challenges for the agility implementation such as review and verification activities, use of documentation, managing the change, risk and traceability. However, the guidance was kept at an abstract level.

Evidence shows that agile development approaches in medical domain are being widely used with proper adoptions and tailoring [1], [42], [46], [48], [49], [50], [51], [52]. For instance, it has been indicated that user stories can be used as an up-front planning technique; iterative testing

and test driven development to assure that all the software will be fully tested before releasing; and configuration management is used to perform necessary traceability between initial requirements and released solutions [46].

E. How can MDevSPICE® be improved by using Agile practices

The sequential flow of development processes in the MDevSPICE® framework might suggest that the V-model could be best suited for medical device software development, making the software development process long and tough on budget, especially when requirements changes are introduced later in the lifecycle. When integrating agile practices into the MDevSPICE® framework, the overall medical device software development process could become more flexible and faster. It will take some time to gather the best agile practices that would be most suitable for MDevSPICE®, but once done – this framework will be a comprehensive guide to all medical device software companies.

MDevSPICE®, similarly to international medical device standards and FDA guidance documents, does not dictate or recommend the use of any specific software development lifecycle approach. MDevSPICE® is an integrated set of regulatory requirements, practices to achieve these requirements and work products that need to be delivered in order to be allowed to sell the software on the market. With the reported and abovementioned benefits of agile methods in safety-critical software development, a medical device software development organization needs to select the most appropriate agile practices that their organization should follow and integrate them into the development lifecycle model applied in their organization [53]. It is important for the medical device software organizations to realize that the key values of the agile manifesto [33] are not contradictory but can be aligned to be complimentary to the development of medical device software, resulting in a quality management system that produces high-quality medical device software [32].

VI. CONCLUSION

Safety-critical domains are characterized by heavy regulatory demands that companies have to adhere to before they can place their devices on the market. Regulatory audits are conducted regularly to evaluate these companies and the safety of their devices. In order to pass these audits, medical device manufacturers have to ensure that all regulatory requirements have been adhered to in the design and development of each of the medical device subsystems.

In this paper, we have explained the medical device regulatory requirements and the related standards and guidance documents. We have described how MDevSPICE® addresses all concerns regarding regulatory requirements in a single medical device software framework. The key to developing this framework was an acknowledgement that the overall medical device requirements have a direct impact on the safety of the device, and it is therefore critical that top level product requirements are fully realized in the software system and its related requirements. This can be especially difficult to achieve in environments where device manufacturers decide to outsource software development without necessarily sharing all top level product requirements with the subcontractors. To address this critical interface, the MDevSPICE® framework incorporates not just software development lifecycle processes but also the system level processes. Hence, system requirements that have an impact on software requirements are identified in MDevSPICE®, and through the implementation of bilateral requirements traceability, decisions taken during the software subsystem development are fed back to the top level system requirements – thus providing a closed loop for requirements management, which can help to increase the overall safety of the device.

In this paper we have argued that agile practices such as iterative development cycles, continuous integration, sprint planning meetings and continuous requirements prioritization should be tested when assuring the development of better technology design and technology interfaces. We have also illustrated the benefits and the challenges of agile practice integration into traditional medical device software development. Through providing empirical support to these challenges we have established the basis for our future research work in which we will decide upon the most appropriate agile practices that will be integrated into the MDevSPICE® framework. We will then integrate the selected agile best practices into the MDevSPICE® framework to shorten the medical device software development lifecycle as well as the time to market for the resulting medical devices.

ACKNOWLEDGMENT

This research is supported by the Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund and by Lero - the Irish Software Research Centre (http://www.lero.ie) grant 10/CE/I1855 & 13/RC/20194. The research is also supported by Digital Health Revolution project and Tekes, Finnish Funding Agency for Innovation.

REFERENCES

[1] F. McCaffery, M. Lepmets, and P. Clarke, “Medical device software as a subsystem of an overall medical device: The MDevSPICE experience,” The First International Conference on Fundamentals and Advances in Software Systems Integration FASSI, Aug. 2015, Venice, Italy, doi:10.1002/smr.1731

.[2] ISO/IEC 15504-5. Information technology - process assessment - Part 5: an exemplar process assessment model. 2012. p. 211.

[3] FDA. Chapter I - Food and drug administration, department of health and human services subchapter H - Medical devices, Part 820 - Quality system regulation. Available from: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820. Last date accessed - 26.05.2016.

[4] Directive 93/42/EEC of the European Parliament and of the Council concerning medical devices. 1993. European Commission, Brussels, Belgium. p. 43.

[5] Council directive 90/385/EEC on active implantable medical devices (AIMDD). 1990. Brussels, Belgium. p. 35.

[6] Directive 98/79/EC of the European Parliament and of the council of 27 October 1998 on in vitro diagnostic medical devices. 1998. Brussels, Belgium. p. 43.

[7] Directive 2007/47/EC of the European Parliament and of the Council concerning medical devices. 2007. EC: Brussels, Belgium. p. 35.

[8] IEC 62304: Medical device software - software life-cycle processes. 2006. IEC: Geneva, Switzerland. p. 151.

[9] ISO 13485: Medical devices - quality management systems - requirements for regulatory purposes. 2003. ISO: Geneva, Switzerland. p. 57.

[10] ISO 9001:2000 - Quality management systems - requirements. 2000. Geneva, Switzerland. p. 27.

[11] ISO/IEC 12207:1995 - Information technology - software life-cycle processes. 1995. ISO/IEC: Geneva, Switzerland. p. 106.

[12] ISO/IEC 12207:2008 - Systems and software engineering - Software life cycle processes. 2008. ISO/IEC: Geneva, Switzerland. p. 138.

[13] ISO/IEC 15504-2:2003, Software engineering - Process assessment - Part 2: Performing an assessment. ISO: Geneva, Switzerland. 2003.

[14] ISO 14971 - Medical devices - Application of risk management to medical devices 2009. ISO: Geneva, Switzerland. p. 82.

[15] IEC TR 80002-1 - Medical device software - Part 1: Guidance on the application of ISO 14971 to medical device software. 2009. IEC: Geneva, Switzerland. p. 58.

[16] IEC 60601-1 - Medical electrical equipment – Part 1: General requirements for basic safety and essential performance 2005. IEC: Geneva, Switzerland. p. 20.

[17] IEC 62366 - Medical devices - Application of usability engineering to medical devices. 2007. IEC: Geneva, Switzerland. p. 104.

[18] IEC 82304-1: Health software -- Part 1: General requirements for product safety. 2012. IEC: Geneva, Switzerland. p. 30.

[19] FDA Guidance documents (medical devices and radiation-emitting products), 2015. FDA: USA.

[20] FDA Guidance for the content of premarket submissions for software contained in medical devices. 2005. FDA: USA. p. 20.

[21] FDA's Guidance for industry, FDA reviewers and compliance on - Off-the-shelf software use in medical devices. 1999. FDA: USA. p. 26.

[22] FDA's General principles of software validation; final guidance for industry and FDA Staff. 2002. FDA: USA. p. 43.

[23] ISO/IEC 24774 - Systems and software engineering - Life cycle management - Guidelines for process description. 2010. Geneva, Switzerland. p. 15.

[24] B. Glaser, and A. Strauss, "The discovery of grounded theory: strategies for qualitative research", Ed: A.d. Gruyter. Hawthorne, NY, USA, 1976.

[25] M. Lepmets,P. Clarke, F. McCaffery, A. Finnegan, and A. Dorling, "Development of a process assessment model for medical device software development," in Industrial Proceedings of the 21st EuroSPI Conference, 2014, Luxembourg, pp. 2.25-2.35.

[26] IEC TR 80002-3: Medical device software -- Part 3: Process reference model of medical device software life cycle processes (IEC 62304). 2014. IEC: Geneva, Switzerland. p. 23.

[27] G. Regan, M. Biro, F. McCaffery, K. McDaid, and D. Flood, "A traceability process assessment model for the medical device domain, " in EuroSPI, Luxembourg, 2014, pp. 206-216, doi: 10.1007/978-3-662-43896-1_18.

[28] J. Gaev,"Technology in healthcare. in: clinical engineering handbook, " Ed: Dyro, Joseph F., p.342, 2004.

[29] G. Powell-Cope, A. Nelson, and E. Patterson, "Patient safety and quality: an evidence-based handbook for nurses," Chapter 50. Hughes RG, Ed. Rockville (MD): AORN Journalvol.90(4), pp.601-602, 2008.

[30] M. Dulin, C. Hugh-Jones, M. Pitts, and G. Hughes, "Applying data to improve patient-centric and personalized medicine," Conclusions Paper, 10th Annual Health Care & Life Sciences Executive Conference, SAS, 2013.

[31] J. Rasmusson, The Agile samurai: how agile masters deliver great software, Pragmatic Bookshelf, 2010.

[32] AAMI, AAMI TIR45:2012 -- "Guidance on the use of agile practices in the development of medical device software," 2012.

[33] (2001). Agile Manifesto. Available from: www.agilemanifesto.org. Last date accessed: 26.05.2016.

[34] M. Daneva, E. van der Veen. C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, and R.Wieringa, "Agile requirements prioritization in large-scale outsourced system projects: An empirical study, " The Journal of Systems and Software, vol. 86(5) , pp.1333– 1353, 2013.

[35] B. Boehm and R Turner. "Balancing agility and discipline: A guide for the perplexed," Addison-Wesley Professional, 2003.

[36] B. Boehm and R. Turner. "Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods," 26th International Conference in Software Engineering (ICSE), 2004, pp. 718-719.

[37] A. Scott W. and M. Lines, Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise, IBM Press, 2012.

[38] J. Humble and D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation, Pearson Education, 2010.

[39] J. Patton and P. Economy, User story mapping: Discover the whole story, build the right product, O'Reilly Media, Inc., 2014.

[40] U. Upender, "Staying agile in government software projects," in Agile Conference Proceedings, July 2005, pp. 153-159, doi: 10.1109/ADC.2005.41.

[41] B. Fitzgerald, K.-J.Stol, R. O'Sullivan, and D. O'Brien, "Scaling agile methods to regulated environments: An industry case study," 35th International Conference in Software Engineering (ICSE), May 2013, pp. 863-872.

[42] K.N. Manjunath, J. Jagadeesh, and M. Yogeesh, "Achieving quality product in a long term software product development in healthcare application using Lean and Agile principles: Software engineering and software development," International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), March 2013, pp. 26-34, doi: 10.1109/iMac4s.2013.6526379.

[43] P.A. Rottier, and V. Rodrigues "Agile Development in a Medical Device Company," in Agile Conference Proceedings, pp.218-223, Aug. 2008, doi: 10.1109/Agile.2008.52

[44] Chaos Report, Standish Group, based upon Jeff Sutherland Scaling Agile course notes, 2015, Boston.

[45] J. Sutherland, Scrum: The Art of Doing Twice the Work in Half the Time, Crown Business, 2014.

[46] M. McHugh, F. McCaffery, and V. Casey, "Barriers to adopting agile practices when developing medical device software," in Software Process Improvement and Capability Determination Proceedings, vol.290, pp. 141-147, 2012, doi: 10.1007/978-3-642-30439-2_13.

[47] J. Pelrine, "On understanding software agility - A social complexity point of view," E:CO. vol.13(1-2), pp. 26-37, 2011

[48] Z.R. Stephenson, J.A. McDermid, and A.G Ward, "Health modelling for agility in safety-critical systems development," in the 1st Institution of Engineering and Technology International Conference Proceedings, June 2006, pp. 260-265, doi: 10.1049/cp:20060225.

[49] J. Sutherland."Future of scrum: parallel pipelining of sprints in complex projects," in Agile Conference Proceedings, July 2005, pp. 90-99, 24-29.

[50] M. Sumrell, "From waterfall to agile - how does a qa team transition?," in Agile Conference Proceedings, Aug. 2007, pp. 291-295, doi: 10.1109/AGILE.2007.29.

[51] J. W. Spence "There has to be a better way! [software development]," in Agile Conference Proceedings, July 2005, pp. 272-278, doi: 10.1109/ADC.2005.47.

[52] K. Weyrauch, "What are we arguing about? A framework for defining agile in our organization," in Agile Conference Proceedings, July 2006, pp. 213-220, doi: 10.1109/AGILE.2006.62.

[53] M. McHugh, F. McCaffery, and V. Casey. "Adopting agile practices when developing software for use in the medical domain," Journal of Software: Evolution and Process, vol.26(5), pp. 504-512, 2014, doi: 10.1002/smr.1608.