Conformance to Medical Device Software Development Requirements with XP and Scrum Implementation

Ö. Özcan-Top[1] and F. McCaffery[1,2]
[1]Regulated Software Research Centre, Dundalk Institute of Technology & Lero, Dundalk, Ireland
[2] STATSports Group, Newry, Ireland

Abstract – *A key challenge of medical device software development companies is to maintain both conformance to the strict regulatory requirements enforced by the safety critical nature of the domain and achieve efficiency in software development. Agile software development methods provide promising solutions to overcome the efficiency issues and the challenges of traditional software development approaches. Even though Agile practices are being welcomed by the medical domain, their suitability for conformance to the regulatory requirements is still being questioned by the medical industry. In our previous work, we investigated to what extent the regulatory requirements defined in MDevSPICE® (the software process assessment framework for medical device software development) are met through using a Scrum implementation and what additional practices have to be performed to ensure safety and regulatory compliance in the medical domain. In this paper, we extended the research to include the XP method and provide a comprehensive and quantitative analysis of its suitability for medical device software development.*

Keywords: MDevSPICE®, Scrum, XP, Safety Critical Domain, Medical Domain, Agile Software Development

1 Introduction

Due to the potential risk of Medical Devices (MD) harming patients', strict regulations need to be in place in development to ensure the safety of these devices. Depending on the region that a MD is to be marketed, different standards or guidance have to be followed. 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 [1]. In the EU, the corresponding regulation is outlined in the general Medical Device Directive (MDD) 93/42/EEC [2], the Active Implantable Medical Device Directive (AIMDD) 90/385/EEC [3], and the In-vitro Diagnostic (IVD) Medical Device Directive 98/79/EC [4] - all three of which have been amended by 2007/47/EC.

The focus of this study is MD Software which is often an integral part of an overall medical device. IEC 62304:2006 [5] is the main medical device software development (MDSD) standard to establish the safety of medical device software by defining processes, activities and tasks for development so that software does not cause any unacceptable risks. Whether medical device software is marketed in the USA or EU, the challenges associated with the development remain the same. Some of them are listed below:

a) Adherence to a large number of regulatory requirements specified in various international standards [6];

b) Establishing a full traceability schema from stakeholder requirements to code [7, 8];

c) Performing changes to process artefacts (requirements, code, documents) in a traceable way [9, 10];

d) Being able to embrace change during development;

e) Producing development evidence for auditory purposes consistently and continuously and managing the documentation process in an effective way so that it is not overwhelming;

f) Ensuring reliability, safety and correctness of products;

g) Improving the quality of products and productivity of teams;

h) The necessity of clinical software validation to be done manually in some cases [10].

In relation to challenge (a), the MDevSPICE® framework [11], which was previously developed by our research group (RSRC), assists companies to efficiently prepare for the demanding and costly regulatory audits as it combines requirements from a wide number of medical software development and software engineering standards. To overcome the challenges listed from (b) to (h), usage of Agile Software Development (ASD) practices with a combination of traditional software development practices could provide significant improvements.

In one of our previous studies [12], we evaluated one of the most preferred agile methods, Scrum , to understand the level of regulatory compliance when it is fully implemented as described in the Scrum Guide™ [13]. We performed the evaluation by mapping the Scrum roles and events with MDevSPICE® base practices. The mapping results indicated that Scrum implementation would provide full or partial coverage in only five MDevSPICE® Processes: *Project Planning; Project Assessment and Control; Stakeholder Requirements Definition; System Requirements Analysis and*

Software Requirements Analysis. The significance of this study was that for the first time, the degree of compliance to the MDSD requirements for all of these processes was provided in a quantitative way.

In this study, we aimed to extend this evaluation by including eXtreme Programming (XP) [14], an agile method which evolved from the issues caused by long development cycles of plan-driven methods. Unlike Scrum, it focuses more on the technical side of software development. Considering that both XP and Scrum have been used in organizations, the evaluation of their combination would provide a more comprehensive perspective to organizations in terms of their compliance to medical requirements.

The second purpose of this research is to reveal additional practices that have to be performed to ensure compliance when a combination of XP and Scrum are implemented.

The rest of the paper is structured as follows: In Section 2, we provide the background for this research which includes brief descriptions of MDevSPICE®, Scrum and XP. In Section 3, we describe the research methodology. In Section 4, we present the XP mapping analysis in great detail and discuss additional practices that have to be considered Additionally, we provide a summary of the Scrum mapping analysis which we published previously in [12] . Finally, in Section 5, we provide conclusions for this research.

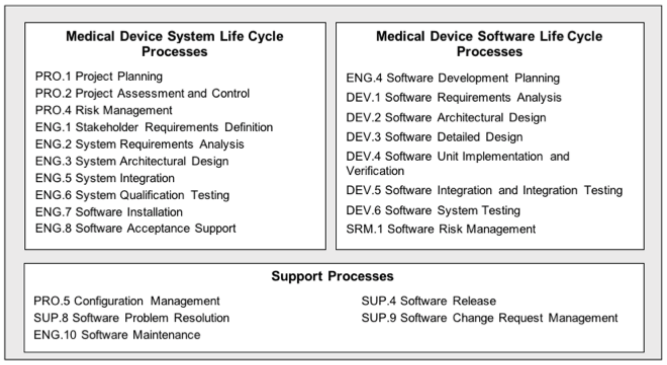
2 Background

2.1 MDevSPICE®

The challenge that medical software development companies 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 be overwhelming. In order to help companies better prepare for the demanding and costly regulatory audits, we previously developed the MDevSPICE® framework [11]. MDevSPICE® is an integrated framework of medical device software development best practices.

MDevSPICE® was based upon the ISO/IEC 15504/SPICE [15] (ISO/IEC 33000 series, now) process assessment standard and includes requirements from a wide number of medical software development and software engineering standards, some of which were mentioned in the introduction section. Requirements from different standards and guidance are reflected in the process reference model (PRM) which describes a set of processes in a structured manner through a process name, process purpose, process outcomes and base practices.

A base practice, which is one of the main components of this study, an activity that addresses the purpose of a particular process. Figure 1 shows the all processes of MDevSPICE®.


Medical Device System Life Cycle Processes

PRO.1 Project Planning
PRO.2 Project Assessment and Control
PRO.4 Risk Management
ENG.1 Stakeholder Requirements Definition
ENG.2 System Requirements Analysis
ENG.3 System Architectural Design
ENG.5 System Integration
ENG.6 System Qualification Testing
ENG.7 Software Installation
ENG.8 Software Acceptance Support

Medical Device Software Life Cycle Processes

ENG.4 Software Development Planning
DEV.1 Software Requirements Analysis
DEV.2 Software Architectural Design
DEV.3 Software Detailed Design
DEV.4 Software Unit Implementation and Verification
DEV.5 Software Integration and Integration Testing
DEV.6 Software System Testing
SRM.1 Software Risk Management

Support Processes

PRO.5 Configuration Management
SUP.8 Software Problem Resolution
ENG.10 Software Maintenance
SUP.4 Software Release
SUP.9 Software Change Request Management

Figure 1 MDevSPICE® processes

2.2 Scrum

Scrum is mainly a management model for software development, and was developed by Schwaber and Sutherland [13]. Although, use of technical practices was strongly supported by the creators of the model, it does not present any specific technical practices for implementation. The fundamental idea behind Scrum is to apply process control theory to software development to achieve flexibility, adaptability and productivity [16]. It relies on a set of values, principles and practices which can be adopted based on specific conditions. Scrum gives value on providing frequent feedback, embracing and leveraging variability, being adaptive, balancing upfront and just-in-time work, continuous learning, value-centric delivery and employing sufficient ceremony [17]. It offers effective solutions by providing specific roles, artifacts, activities and rules.

A Scrum Team consists of a number of roles: Product Owner; a Scrum Master; and the Development Team. Scrum Teams are self-organizing and cross-functional so that they may accomplish their work by themselves, rather than being directed by others outside of the team and without depending on others that are not part of the team [13]. There are special events in Scrum which have been developed to create regularity and to minimize the need for meetings and are time-boxed.

2.3 eXtreme Programming (XP)

XP was developed by Kent Beck in 1999, it provides a collection of software engineering practices [14]. Even though the practices are not novel, XP brings them together to facilitate change and to produce higher quality software at a sustainable pace. XP is defined by values, principles and roles. Some of the fundamental practices of XP are *planning game, small releases, metaphors, simple design, continuous unit testing, refactoring, pair programming, collective code ownership, continuous integration, work 40-hour-a-week, on-site customer and coding standards*. XP does not provide much support for software project management activities [16]. The details of the XP practices are provided in the mapping section below.

3 Research Approach

We applied the same research approach that we followed in [12]. The purpose of this research is to reveal to what extent the regulatory requirements defined in MDevSPICE® are met when highly adapted agile software development methods, XP and Scrum are implemented. We defined the following research questions in relation to this purpose:

RQ1: Which processes of MDevSPICE® are covered by implementation of XP and Scrum? ***RQ2:*** Which base practices MDevSPICE® are covered by implementation of Scrum or XP? ***RQ3:*** What additional practices regarding those processes specified need to be performed in order to fully achieve regulatory compliance in the medical domain?

Research steps

1. Listing XP and Scrum practices/events/roles and their descriptions
2. Mapping the MDevSPICE® base practices with XP and Scrum practices/events/roles.
3. Identifying which processes were affected from the mapping.
4. Identifying the coverage ratio and deciding which MDevSPICE® base practices need to be included for those processes to satisfy a fully-achieved level.

Abrahamsson *et al*. provide a comparison of different agile software development methods in [16] and specify which phases of software development life cycle were supported by these methods. Based on this research, Scrum covers project management, requirements specification, integration test and system test stages/activities. XP, on the other hand, presents solutions for requirements specification, design, code, unit test, integration test and system test stages/activities.

For the mapping we performed, instead of initially selecting the processes mentioned above, and then checking their coverage within MDevSPICE®, we performed the mapping the other way around. We first listed the XP and Scrum practices and then mapped them to the MDevSPICE® base practices. With this approach we were able to identify which MDevSPICE® processes were satisfied through adopting XP and Scrum implementations.

Limitations of the Research

With the given descriptions of XP, we note it as a descriptive method in which the practices are described from a high abstraction level. Compared to XP, Scrum could be taken as a prescriptive method with the descriptions of how the Scrum events will be performed and how the artifacts will be developed. However, both of them were not at the practice description level provided by MDevSPICE®. Mappings of the methods were limited to the information in the following resources: The Scrum Guide™ by Ken Schwaber and Jeff Sutherland [13] and the book: Extreme Programming Explained: Embrace Change by Kent Beck [14].

Although twenty-two XP practices were described to the same level of detail in the XP book [14], they show significant differences in terms of their characteristics. For example, "sit together" vs "continuous integration" or "customer involvement" vs "incremental deployment" practices. As the definitions of the practices are limited, we needed to make some assumptions during the mapping.

4 The Systematic Mapping Process

In [12], we provided a very detailed analysis of the mapping of Scrum's activities, roles and MDevSPICE®'s processes and base practices. Due to space limitations, we will summarize the analysis of the Scrum mapping and detail the XP mapping.

For both of the XP and Scrum mappings and the coverage evaluation, MDevSPICE® Class B requirements were taken into account. Due to the descriptive characteristics of the methods mentioned above, we assumed that process artifacts such as project plans or project monitoring reports would be developed during XP and Scrum implementation, as the evidence for audits need to be collected. Although it is very likely that some base practices would be performed during software development with Scrum or XP, we couldn't rate a 100% coverage for them, as they might not be performed at the level of the detail that is required by MDevSPICE®.

The coverage ratio (CR) is calculated based on the formula of:

$$CR = \frac{\sum of\ the\ achieved\ base\ practices\ in\ a\ process}{total\ number\ of\ the\ base\ practices\ in\ a\ process} \qquad (1)$$

In this evaluation, the base practices (BPs) are considered either partially or fully achieved. For the calculation of *the* $\sum$ *of the achieved base practices in a process*, partially achieved (PA) practices were weighted by 0.5, while fully achieved (FA) BPs were weighted by 1.

4.1 XP Mapping

The XP method [14] is described in terms of roles, values, principles and practices. The **roles** in an XP team are testers, interaction designers, architects, project managers, product managers, executives, technical writers, users, programmers, human resources. They are suggested to have a flexible structure rather than being fixed and rigid.

The **values** which are "communication, simplicity, feedback, courage, respect, safety, security, predictability, and quality-of-life", shape the teams' behavior and the development environment. But, they don't provide concrete guidance on software development. The **principles** play a bridge role between the values and the practices. Some of the XP principles are "humanity economics, mutual benefit, self-similarity, improvement and diversity". As could be seen, they are also at a very abstract level and do not provide advice for software development. The XP **practices**, which are the main component of this mapping, are presented in two categories: the primary practices and the corollary practices. Based on Kent [14], the primary practices aim at immediate improvement and the corollary practices are difficult without mastering the primary practices.

In Table 1 and Table 2, the mapping between the primary and corollary practices of XP and the processes and base practices of MDevSPICE® are provided (**RQ1-RQ2**). Due to space limitation, the XP practice descriptions cannot be provided in the below tables. The bold text in the 2nd columns of Table 1 and Table 2 show the mapped processes. The other text in the same column cell refer to the mapped base practices (BPs).

Table 1 Mapping of the XP Primary Practices and the MDevSPICE® Processes and Base Practices

Primary Practices	**MDevSPICE® Processes and Base Practices**
Weekly Cycle	***PRO.1 Project Planning*** PRO.1.BP4: Define and maintain estimates for project attributes PRO.1.BP5: Define project activities and tasks ***PRO.2 Project Assessment and Control*** PRO.2.BP3: Report progress of the project PRO.2.BP4: Perform project review PRO.2.BP5: Act to correct deviations. ***DEV.1 Software Requirements Analysis*** DEV.1.BP2: Prioritize requirements. DEV.1.BP7: Baseline and communicate software requirements.
Quarterly Cycle	***PRO.2 Project Assessment and Control*** PRO.2.BP3: Report progress of the project PRO.2.BP4: Perform project review PRO.2.BP5: Act to correct deviations.
Whole Team	***PRO.1 Project Planning*** PRO.1.BP6: Define needs for experience, knowledge and skills.
Informative Workspace	No Corresponding Practice
Energized Work	No Corresponding Practice
Sit Together	No Corresponding Practice
Pairing and Personal Space	***PRO.1 Project Planning*** PRO.1.BP6: Define needs for experience, knowledge and skills.
Pair Programming	***DEV.4 Software Unit Implementation and Verification*** DEV.4.BP1: Implement the software units.
Slack	***PRO1.Project Planning*** PRO.1.BP8: Define project schedule. ***PRO.2 Project Assessment and Control*** PRO.2.BP5: Act to correct deviations.
Ten Minute Build	***DEV.5 Software Integration and Integration Testing*** DEV.5.BP1: Integrate software units into software items. DEV.5.BP2: Verify that software integration follows integration strategy. DEV.5.BP3: Develop tests for integrated software items.
Continuous Integration	***DEV.4 Software Unit Implementation and Verification*** DEV.4.BP4: Verify software units. ***DEV.5 Software Integration and Integration Testing*** DEV.5.BP1: Integrate software units into software items. ***SUP.4 Software Release*** SUP.4.BP1: Ensure the completeness of software verification
Test First Programming / Continuous Testing	***DEV.4 Software Unit Implementation and Verification*** DEV.4.BP4: Verify software units. ***DEV.5 Software Integration and Integration Testing*** DEV.5.BP3: Develop tests for integrated software items. DEV.5.BP4: Test integrated software items in accordance with the integration plan and document the results. ***SUP.4 Software Release*** SUP.4.BP1: Ensure the completeness of software verification
Incremental Design	Excluded from the analysis, as the definition of this process was not clear
Story	***DEV.1 Software Requirements Analysis*** DEV.1.BP1: Define and document all software requirements.

Table 2 Mapping of the XP Corollary Practices and the MDevSPICE® Processes and Base Practices

Corollary Practices	**MDevSPICE® Processes and Base Practices**
Real Customer Involvement	***PRO.1 Project Planning*** PRO.1.BP6: Define needs for experience, knowledge and skills.
Incremental Deployment	***SUP.4 Software Release*** SUP.4.BP2: Define the software product for release SUP.4.BP3: Assemble product for release. SUP.4.BP5: Deliver the release to the acquirer and obtain a confirmation of release.
Team Continuity	***PRO.1 Project Planning*** PRO.1.BP6: Define needs for experience, knowledge and skills.
Shrinking Teams	***PRO.1 Project Planning*** PRO.1.BP6: Define needs for experience, knowledge and skills.
Root-Cause Analysis	***SUP.8 Software Problem Resolution*** SUP.8.BP1: Identify and record each problem in a problem report. SUP.8.BP2: Provide initial support to reported problems and classify problems. SUP.8.BP3: Investigate and identify the cause of the problem. SUP.8.BP4: Assess the problem to determine solution and document the outcome of the assessment. SUP.8.BP7: Implement problem resolution.
Shared Code/ Collective Code Ownership	***DEV.4 Software Unit Implementation and Verification*** DEV.4.BP1: Implement the software units.
Code and Tests	Contradicts with MDevSPICE®
Single Code Base	***DEV.5 Software Integration and Integration Testing*** DEV.5.BP1: Integrate software units into software items. DEV.5.BP2: Verify that software integration

	follows integration strategy.

According to the mappings shown in Table 1 and Table 2, the XP method, when implemented fully, is related to seven processes of MDevSPICE®. Table 3 shows these processes and the coverage ratio of each process.

Table 3 Coverage Ratios of the Mapped MDevSPICE® Processes from XP Perspective

	Mapped MDevSPICE® Processes	**CR**
1.	PRO.1 Project Planning	0.32
2.	PRO.2 Project Assessment and Control	0.50
3.	DEV.1 Software Requirements Analysis	0.22
4.	DEV.4 Software Unit Implementation and Verification	0.375
5.	DEV.5 Software Integration and Integration Testing	0.80
6.	SUP.4 Software Release	0.43
7.	SUP.8 Software Problem Resolution	0.80

Below, we discuss why these processes in Table 3 did not have a full coverage ratio and what additional practices are required in order to achieve compliance to the medical requirements (**RQ3**). XP practices are shown in italics and underlined not to be confused with the MDevSPICE® base practices.

The mapping illustrated that some of the primary XP practices which are *Informative Workspace, Energized Work and Sit Together* do not have specific correspondence at the MDevSPICE® side. The practice called *Code and Test* favours maintaining only the code and the tests as permanent artifacts and generating other documents from the code and tests when necessary. It is suggested to rely on social mechanisms to keep alive important historical parts of the project. However, this approach will not be acceptable in a safety critical software project for traceability and auditory reasons. We excluded the *Incremental Design* practice of the mapping, as the definition provided in [14] was not clear to associate the practice either with the Architectural Design or Software Detailed Design processes.

Below, we discuss the mapped processes and practices in terms of coverage analysis.

#1 PRO.1 Project Planning Process
(CR of PRO.1 = 3.5 BP / 11 BP = 0.318)
The sixth base practice of *PRO.1, Define needs for experience, knowledge and skills,* could be achieved with the implementation of *Whole Team*, Real Customer Involvement, Team Continuity and Shrinking Teams practices. The strong emphasis on the team structure of XP could be deduced with these four practices. *PRO.1.BP8: Define project schedule* base practice requires determining the sequence and schedule of performance of activities within the project. *Slack* is a practice which suggests having flexibility on project schedule by allowing tasks to be added, changed or dropped and discusses the commitments in terms of honesty with the stakeholders. However, this BP is assumed to be partially achieved (PA), as it is not enough just by itself to establish a project schedule for a medical device software project. The *Weekly Cycle* practice of XP advises customers to decide stories to be implemented for the following week, breaking the stories into tasks and team members sign up for tasks and estimate them. Therefore, *PRO.1.BP4: Define and maintain estimates for project attributes* and *PRO.1.BP5: Define project activities and tasks* BPs may be achieved with proper implementation of the Weekly Cycle practice.
Based on this analysis, of the XP implementation, three BPs of PRO.1 (BP4-BP5-BP6) are fully achieved and one BP (BP8) is partially achieved. Additionally, for the PRO.1 process to be fully achieved, the project scope, the project life cycle model, the need for experience, knowledge and skills, and major project interfaces have to be defined with a project plan, including all this information being established and implemented.

#2 PRO.2 Project Assessment and Control
(CR of PRO.2 = (3 BP / 6 BP =0.50)
BPs, *PRO.2.BP3: Report progress of the project, PRO.2.BP4: Perform project review* and *PRO.2.BP5: Act to correct deviations* may be achieved through adopting the *Weekly Cycle* and *Quarterly Cycle* primary practices of XP. The Weekly cycle practice suggests reviewing progress to date, including monitoring how the actual progress for the previous week matches expected progress. *PRO.2.BP3* is fully achieved with this practice. The Quarterly Cycle practice suggests planning work on a quarterly basis from a broader perspective. During this planning activity it is suggested to identify bottlenecks, especially those controlled outside the team, initiate repairs, plan the theme or themes for the quarter and select a quarter's worth of stories to address those themes. Use of a combination of Weekly Cycle, Quarterly Cycle and Slack practices would be sufficient to enable *PRO.2.BP4* and *PRO.2.BP5* to be fully achieved.
Although these practices provide a good structure to monitor project activities and take corrective actions, the medical domain needs special focus on monitoring project attributes such as scope, budget, cost, resources; project interfaces and recording project experiences and data to be available for future projects and process improvement.

#3 DEV.1 Software Requirements Analysis Process:
(CR of DEV.1 = (2 BP / 9 BP = 0.22)
Beck introduces a new form of requirements in XP: *Stories* which are described using a short name and a graphical description on an index card. They are a place holder to initiate discussions around the requirements. With this definition of Stories in XP, we could say it is not a suitable format for a regulated domain. However, adaptations could be performed on stories to extend their usage. Briefly, in relation to *DEV.1.BP1: Define and document all software requirements* BP*;* all functional, performance, security and usability requirements including hardware and software requirements (for third party software as well), software system inputs and outputs, interfaces between the software system and other systems; software-driven alarms, and warnings need to be defined. With the *Weekly Cycle* practice, defined software requirements could be prioritized. Stories

and Weekly Cycle provide a means to communicate on software requirements. However, *DEV.1.BP7: Baseline and communicate software requirements* BP cannot be considered as fully achieved with the implementation of these two practices, as software requirements have to be baselined.

Based on this analysis, with XP implementation, two BPs of DEV.1 (BP1-BP7) are partially achieved and one BP (BP2) is fully achieved. Additionally, for the DEV.1 process to be fully achieved impact of the requirements on the operating environment needs to be determined and risk control measures in software requirements need to established and maintained. It is also essential that the consistency is achieved between system and software requirements. In the safety critical domain, the consistency is supported by establishing and maintaining bilateral traceability between the project artefacts. However, XP does not make distinction between requirement types and suggest establishing traceability.

#4 DEV.4 Software Unit Implementation and Verification Process: (CR of DEV.4 = (1.5 BP/ 4 BP = 0.375)

The Pair Programming, Test First Programming and Shared Code practices of XP were mapped to *DEV.4.BP1: Implement the software units* and *DEV.4.BP4: Verify software units* BPs. Additionally, the Continuous Integration practice was also mapped to *DEV.4.BP4*. As part of the *DEV.4.BP4* BP, each software unit implementations needs to verified and the verification results have to be documented. Because of the documentation requirement *DEV.4.BP4* BP was considered to be partially achieved (PA).

For the DEV.4 process to be fully achieved, software unit verification procedures needed to be established, acceptance criteria are needed to be defined for each software unit and it has to be ensured that the software units meet that criteria. From a FDA perspective, source code should be evaluated to verify its compliance with specified coding guidelines and additional code inspections need to be performed.

#5 DEV.5 Software Integration and Integration Testing Process: (CR of DEV.5 = 4 BP/ 5 BP= 0.80)

The purpose of the Ten-Minute Build practice of XP is to automatically build the whole system and run all of the tests in ten minutes. With a combination of Continuous Integration and Single Code Base, these practices were mapped to the first three BPs of the DEV.5 process. With Test First Programming practice, the *DEV.5.BP4: Test integrated software items in accordance with the integration plan and document the results* will enable the process to be achieved.

Additionally, MDevSPICE® emphases developing regression tests and providing evidence regarding the tests performed. There is no information found on XP regarding regression tests.

#6 SUP.4 Software Release Process:
(CR of SUP.4 = 3 BP/ 7 BP= 0.43)

Continuous Integration and Test First Programming practices could be used to achieve *SUP.4.BP1* which requires ensuring that the detected residual anomalies have been evaluated to ensure that they do not contribute to an unacceptable risk of a hazard. It is also essential that these anomalies were recorded and traced. *SUP.4.BP2* requires defining the products associated with the release *and* documenting the version of the released software. *SUP.4.BP3* requires preparing and assembling the deliverable product and establishing the baseline for the product including user documentation, designs and the product itself. *SUP.4.BP5* requires delivering the release to the acquirer and obtaining confirmation of the release. XP suggests the Incremental Deployment practice and *SUP.4.BP2, SUP.4.BP3 and SUP.4.BP5* are highly related to this practice*. However, we could only assume that SUP.4.BP5* is fully achieved and others partially achieved due to an emphasis on delivering documentation and baselines.

In addition to the above, for the SUP.4 process to be fully achieved, procedures to ensure that the released software product can reliably be delivered to the point of use without corruption and unauthorized change need to be established and all software development activities and tasks together with their associated documentation have been completed.

#7 SUP.8 Software Problem Resolution Process:
(CR of SUP.8 = 4 BP/ 5 BP= 0.80)

The purpose of the software problem resolution process is to ensure that all discovered problems (bugs, defects) are identified, analyzed, managed and controlled to resolution. Test First Programming and Continuous Integration practices are mainly related to the detection and resolution of problems. The *Root-Cause Analysis* practice has a good procedure for problem resolution. It suggests writing automated system-level tests that demonstrate the defect and the desired behavior of the system, writing unit tests with the smallest possible scope that also reproduces the defects and fixes the system so the unit tests work. It is also suggested that once the defect is resolved, to identify why the defect was created and wasn't caught in the first place and to initiate the necessary changes to prevent this kind of defect in the future.

With these practices/procedures, *SUP.8.BP1: Identify and record each problem in a problem report, SUP.8.BP2: Provide initial support to reported problems and classify problems, SUP.8.BP3: Investigate and identify the cause of the problem* and *SUP.8.BP7: Implement problem resolution* BPs are fully achieved, whereas *SUP.8.BP4: Assess the problem to determine solution and document the outcome of the assessment* BP is partially achieved.

Additionally, for the SUB.8 process to be fully achieved, problem reports need to be developed to include a statement of criticality and potentially adverse events. A problem's relevance to safety needs to be evaluated. The outcome of the evaluation needs to be documented, relevant parties of the existence of the problem need to be informed. Records of problem reports, problem resolutions and their verification are maintained.

4.2 Summary of the Scrum Mapping

With the same approach described and followed above, we evaluated Scrum to learn how it meets the regulatory requirements defined in MDevSPICE®. **Table 4** below shows these processes and the coverage ratio of each process.

Table 4 CRs of Mapped MDevSPICE® Processes from Scrum Perspective

	Mapped MDevSPICE® Processes	CR
1.	PRO.1 Project Planning	1
2.	PRO.2 Project Assessment and Control	0.9
3.	ENG.1 Stakeholder Requirements Definition	0.55
4.	ENG.2 System Requirements Analysis	0.71
5.	DEV.1 Software Requirements Analysis	0.33

The details of this evaluation were provided in “How does Scrum Conform to the Regulatory Requirements Defined in MDevSPICE®?” publication [12].

5 Conclusions

In this paper, we analyzed how well the medical device software development requirements are met by the implementation of XP and provided a very detailed evaluation. For this evaluation, we listed XP practices and mapped them to MDevSPICE® base practices and calculated the coverage ratios for the associated MDevSPICE® processes. The purpose of providing these ratios is to provide readers and practitioners with an indication of how much value is achieved with the XP implementation and how much needs to be done more from a regulatory perspective.

Implementing the XP and Scrum practices in a medical device software organization may provide partial or full achievement in nine MDevSPICE® processes. Above, we provided what additional practices need to performed for conformance to medical regulations. However, the framework defines 14 more processes to be addressed. This shows that main agile software development methods could be implemented in a MDSD organization. However, they cannot be a complete solution just by themselves. It can be deduced that tailoring is essential for agile practices within the medical device software domain.

This mapping has illustrated the level of XP’s support for project management processes/practices, is very limited. Therefore, Scrum could be a good complement to XP for planning and assessment practices in MDevSPICE®. Besides, even though the technical practices are provided by XP, it was shown that they are also not sufficient to meet the needs of medical requirements.

We have captured association in nine MDevSPICE® processes, in research we will extend the mapping process to include other ASD methods to be able to provide a complete software development life cycle coverage.

Acknowledgement. This research is supported by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund through Lero - the Irish Software Research Centre (http://www.lero.ie) grant 13/RC/2094. This research is also partially supported by the EU Ambient Assisted Living project – Maestro.

6 References

[1] FDA. (15.05). *Chapter I - Food and drug administration, department of health and human services subchapter H - Medical devices, Part 820 - Quality system regulation*. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820

[2] *Directive 93/42/EEC of the European Parliament and of the Council concerning medical devices*, 1993.

[3] *Council directive 90/385/EEC on active implantable medical devices (AIMDD)*, 1990.

[4] *Directive 98/79/EC of the european parliament and of the council of 27 october 1998 on in vitro diagnostic medical devices*, 1998.

[5] *IEC 2006. IEC 62304: Medical Device Software - Software Life-Cycle Processes.*

[6] F. McCaffrey, M. Lepmets, K. Trektere, O. Ozcantop, and M. Pikkarainen, "Agile Medical Device Software Development: Introducing Agile Practices into MDevSPICE®," 2016.

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

[8] G. Regan, F. Mc Caffery, K. Mc Daid, and D. Flood, "Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model," *Computer Standards & Interfaces*, vol. 36, no. 1, pp. 3-9, 2013.

[9] M. Mc Hugh, O. Cawley, F. McCaffcry, I. Richardson, and X. Wang, "An agile v-model for medical device software development to overcome the challenges with plan-driven software development lifecycles," in *Software Engineering in Health Care (SEHC), 2013 5th International Workshop on*, 2013, pp. 12-19: IEEE.

[10] P. A. Rottier and V. Rodrigues, "Agile development in a medical device company," in *Agile, 2008. AGILE'08. Conference*, 2008, pp. 218-223: IEEE.

[11] M. Lepmets, F. McCaffery, and P. Clarke, "Development and benefits of MDevSPICE®, the medical device software process assessment framework," *Journal of Software: Evolution and Process*, vol. 28, no. 9, pp. 800-816, 2016.

[12] Ö. Özcan-Top and F. McCaffery, "How Does Scrum Conform to the Regulatory Requirements Defined in MDevSPICE®?," in *International Conference on Software Process Improvement and Capability Determination*, 2017, pp. 257-268: Springer.

[13] J. Sutherland and K. Schwaber, "The scrum guide," *The Definitive Guide to Scrum: The Rules of the Game. Scrum. org*, 2013.

[14] K. Beck, *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.

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

[16] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," ed: VTT Finland, 2002.

[17] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional, 2012.