

# ULRR

## To what extent the medical device software regulations can be achieved with agile software development methods?

Item Type	Article
Authors	Özcan-Top, Özden;McCaffery, Fergal
Citation	The Journal of Supercomputing, 2019,75, 5227–5260
Publisher	Springer
Download date	2026-06-17 01:11:35
Item License	<a href="https://www.springernature.com/gp/open-research/policies/book-policies">https://www.springernature.com/gp/open-research/policies/book-policies</a>
Link to Item	<a href="https://doi.org/10.34961/researchrepository-ul.25498012">https://doi.org/10.34961/researchrepository-ul.25498012</a>

# To what extent the medical device software regulations can be achieved with agile software development methods?

XP – DSDM – Scrum

Özden Özcan-Top<sup>1</sup> and Fergal McCaffery<sup>1,2</sup>

<sup>1</sup>Regulated Software Research Centre, Dundalk Institute of Technology & Lero, Dundalk, Ireland

Tel: +353 42 9370200 Fax: +353 42 9370201

<sup>2</sup>STATSports Group, Newry, Ireland

[ozdentop@gmail.com](mailto:ozdentop@gmail.com), [fergal.mccaffery@dkit.ie](mailto:fergal.mccaffery@dkit.ie)

**Abstract** – For medical device software development organisations, it is very challenging 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 in the domain. Previously, we investigated to what extent the regulatory requirements defined in MDevSPICE<sup>®</sup> (the software process assessment framework for medical device software development) are met through using XP (eXtreme Programming) and Scrum and what additional practices have to be performed to ensure safety and regulatory compliance in the medical device software development domain. In this paper, we extended the research to include the Dynamic Systems Development Method (DSDM) which covers the whole software development life cycle. Here, we provide a comprehensive and quantitative analysis of XP's and DSDM's suitability for medical device software development and briefly discuss Scrum from the same perspective. We provide the coverage ratios of processes and practices defined in MDevSPICE<sup>®</sup> when these agile software development methods are implemented.

**Keywords:** Medical Domain, Medical Software, MDevSPICE<sup>®</sup>, XP, DSDM, Scrum, Safety Critical 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.

MD software which is the focus of this research, can be either of an integral part of an overall medical device such as in [5] or a mobile medical application as in [6]. IEC 62304:2006 [7] is the main medical device software development (MDS) 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 [8];
- b) Establishing a full traceability schema from stakeholder requirements to code [9, 10];

- c) Performing changes to process artefacts (requirements, code, documents) in a traceable way [11, 12];
- 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 [12].

In relation to challenge (a), the MDevSPICE<sup>®</sup> framework [13], 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 [14], 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<sup>™</sup> [15]. We performed the evaluation by mapping the Scrum roles and events with MDevSPICE<sup>®</sup> base practices. The mapping results indicated that Scrum implementation would provide full or partial coverage in only five MDevSPICE<sup>®</sup> Processes: *Project Planning*; *Project Assessment and Control*; *Stakeholder Requirements Definition*; *System Requirements Analysis and Software Requirements Analysis*. In another study of ours [16], we evaluated eXtreme Programming (XP) [17] from the same perspective. The XP method focuses more on the technical side of software development. The mapping of the XP practices into the MDevSPICE<sup>®</sup> processes and practices has indicated that XP implementation provides a very limited coverage of the following processes: *Project Planning*, *Project Assessment and Control*, *Software Requirements Analysis*, *Software Unit Implementation and Verification*, *Software Integration and Integration Testing*, *Software Release and Software Problem Resolution*.

Incorporating XP, in addition to Scrum has provided us with information on what could be achieved from a management and technical processes' perspective. The degree of compliance to the MDSO requirements for the 23 processes of MDevSPICE<sup>®</sup> was specified in a quantitative way. However, the mapping has also revealed that implementation of a combination of these two methods would not be sufficient to meet the needs of medical requirements. Implementation of several additional practices have to be included in the software development life cycle to ensure the compliance when a combination of XP and Scrum are performed.

In the light of these two studies, we performed a new mapping with the Dynamic Systems Development Method (DSDM) [18] which provides practices and the work products to cover the full software development life-cycle. With the inclusion of DSDM, we had a wider coverage of medical device software development regulations.

This study extends the previous mapping studies we have done with XP and Scrum to include DSDM analysis. It reveals in what ways XP, Scrum and DSDM can be combined, how these agile models can support each other and which additional system and software processes and practices should be considered to ensure medical regulatory compliance.

It should also be noted that neither regulatory compliance or “doing agile” should be the focus of organisations. The success comes with keeping the product quality and organisational excellence at the centre of all actions. Here in this paper, we aim to shed light into what additionally should be done for regulatory compliance and also show agile practices' applicability to the domain.

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<sup>®</sup>, XP, DSDM and Scrum. We provide literature

review results on the usage of XP, DSDM and Scrum in the medical domain. In Section 3, we describe the research methodology. In Section 4, we present the XP and DSDM mapping analyses in great detail and discuss the additional practices that have to be considered. We also provide a brief discussion of the Scrum mapping. In Section 5, we discuss the differences of these three methods from regulatory perspective and their contributions to the domain. Finally, in Section 6, 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 [13]. MDevSPICE® is an integrated framework of medical device software development best practices.

MDevSPICE® was based upon the ISO/IEC 15504/SPICE [19] (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®.

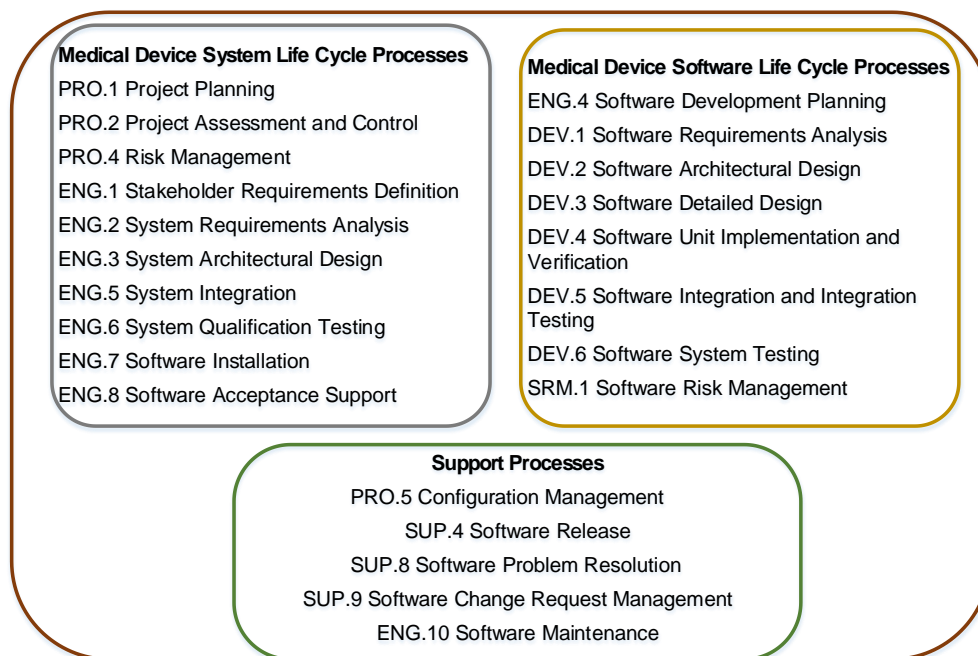


Figure 1 MDevSPICE® processes

### 2.2 eXtreme Programming (XP)

XP was developed by Kent Beck in 1999, it provides a collection of software engineering practices [17]. 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 with a set of 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 [20]. The details of the XP practices are provided in the mapping section below.

### 2.3 Dynamic Systems Development Method (DSDM)

DSDM is an agile project framework, which provides practices and products to cover the full software development life-cycle [18]. It was initially developed in 1994 by a large group of practitioners for rapid application development. The DSDM content, which was previously available only to the DSDM consortium members, is now publicly available on [www.agilebusiness.org](http://www.agilebusiness.org). DSDM advocates key agile principles such as *Enough Design Upfront*, *Never Compromise Quality* and *Collaborate*. Some of its favoured practices are MoSCoW prioritisation, iterative and time-boxed development. DSDM uses roles such as *Business Analyst*, *Business Advisor*, *Technical Advisor* and *Business Ambassador*. Additionally, it prioritises timely development of essential documentation in a lean manner.

### 2.4 Scrum

Scrum is mainly a management model for software development, and was developed by Schwaber and Sutherland [15]. 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 [20]. 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 [21]. It offers effective solutions by providing specific roles, artefacts, 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 [15]. There are special events in Scrum which have been developed to create regularity and to minimize the need for meetings and are time-boxed.

### 2.5 Use of XP, DSDM and Scrum in the Literature

We performed a systematic literature review (SLR) to specify the agile methods used in the medical device software development domain. We performed the review on six major online libraries: *IEEE Xplore Digital Library*, *ACM Digital Library*, *Springer Link Online Library*, *Wiley Online Library*, *Elsevier Science Direct* and *Google Scholar*. The review included a six-staged evaluation process as shown in Figure 2. We included the papers which describe agile methods'/practices' implementation in the medical device software development (MDS) domain. Journal papers, conference papers, experience reports, technical reports were included. The keywords used for the search are: "agile AND (medical device software development OR (medical AND agile)). Based on the result of iterative paper screening process, 33 papers from 2005 to 2018 were included in the SLR.



Figure 2 Paper Screening Process

We looked at which agile methods were used in these studies. The analysis has revealed that Scrum, eXtreme Programming and Test-Driven Development are the most used agile methods in the MDSO domain. Lean Development, Feature Driven Development, and DevOps are the other agile methods that were referred in the papers. However, there was no evidence on utilisation of DSDM in the MDSO domain. It was also interesting to see that none of the scaled agile models (e.g. Disciplined Agile Delivery [22] and Scaled Agile Framework [23]), which provide specific solutions for safety critical projects, were referred in the papers.

The studies emphasize the benefits of using agile methods in medical device software development. For instance, performing daily stand-up meetings and using burn-down charts enable teams to uncover issues earlier, monitor the effect of changes, and estimate better [24]. Iterative development enables accommodating changes easier [9] [25], and getting early feedback from the customer and managing scope creep [26]. Alternatively, there are studies which discuss the unsuitability of agile methods to the MDSO domain and present hybrid models which integrate Scrum, XP and traditional development practices to ensure regulatory requirements. **R-Scrum** [9] augments the Scrum method with specific roles and artefacts. In addition to the Scrum roles, R-Scrum defines a “Vice President (VP) of Quality” role who verifies that the output(s) from each sprint adhere to the required procedures and a “Quality Control” role who produces system test documentation and executes system test scripts in line with required standards and product specification and documents all test results for release review. A role with responsibility for ensuring the links between development and documentation is also a member of the Sprint team. The **TXM** hybrid method [12] combines Scrum and XP practices with the other practices such as platform-based design. Process groups in the model such as functionality implementation, task integration, system refactoring, and system optimization provide solutions for both platform and product development. Another hybrid method [27] extends XP for the missing FDA requirements by adding necessary sub-roles and practices such as a process to support Human Factors Engineering practices.

It is evident that agile methods need to be tailored to meet the MDSO regulations. Tailored processes need to be evaluated to ensure they still meet the regulatory requirements. Because, improper process tailoring may lead to unwanted consequences in terms of product safety. However, the adoptions described in the studies given above are not based on a standard. Therefore, our mapping of agile practices and the MDevSPICE® practices brings a significant value to the domain considering that Scrum and XP are the most widely methods used. Although no studies were found in the literature regarding DSDM’s utilisation in the MDSO domain, we would like to highlight the usability of the DSDM method in the MDSO domain due to its emphasis on governance, effective documentation and specific roles.

### 3 Research Approach

In this extended version, we applied the same research approach that we followed both in [14] and [16]. The purpose of this research is to reveal to what extent the regulatory requirements defined in MDevSPICE® are met when agile software development methods, XP, Scrum and DSDM are implemented. We defined the following research questions in relation to this purpose:

**RQ1:** Which processes of MDevSPICE® are covered by implementation of XP, Scrum and DSDM?

**RQ2:** Which base practices MDevSPICE® are covered by implementation of XP, Scrum and DSDM?

**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, Scrum and DSDM practices/events/roles/products and their descriptions
2. Mapping the MDevSPICE® base practices with XP, Scrum and DSDM 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 [20] 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 presents solutions for requirements specification, design, code, unit test, integration test and system test stages/activities. DSDM, however, covers the entire life cycle. While Scrum and DSDM provide practices for project management, XP does not. Figure 3, which was reproduced from [20], shows the supported life cycle phases by XP, Scrum and DSDM.

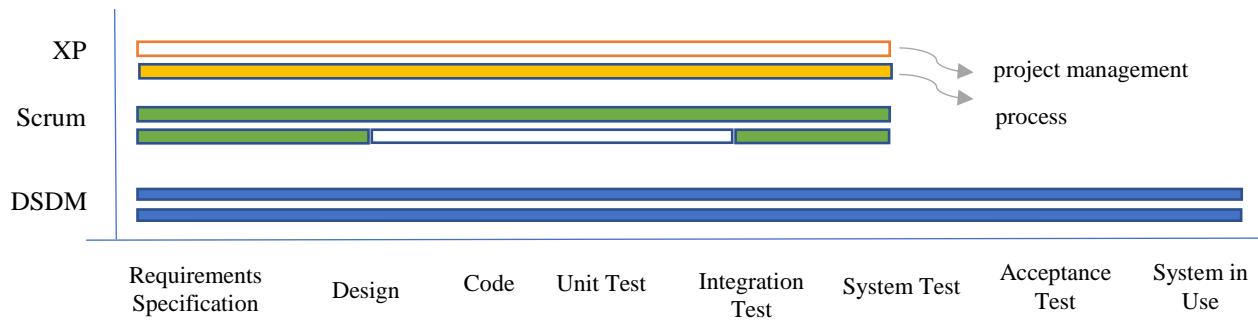


Figure 3 Software Development Life Cycle Support – reproduced from [20]

For the mapping we first listed the XP, Scrum and DSDM practices/work products/roles 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, Scrum and DSDM 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 artefacts will be developed. However, both of them were not at the practice description level provided by MDevSPICE®. Although twenty-two XP practices were described to the same level of detail in the XP book [17], 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.

DSDM on the other hand, gives detailed descriptions on the practices but limited descriptions on the work products. Mappings of the methods were limited to the information in the following resources: The Scrum Guide™ by Ken Schwaber and Jeff Sutherland [15], the book: Extreme Programming Explained: Embrace Change by Kent Beck [17] and DSDM Handbook provided by Agile Business Consortium ([www.agilebusiness.com](http://www.agilebusiness.com)).

## 4 The Systematic Mapping Process

In [14], we provided a very detailed analysis of the mapping of Scrum’s activities, roles and MDevSPICE®’s processes and base practices. We mapped here the DSDM Products – DSDM Practices (some of these practices may have been released as part of another agile method. However, we refer them as the DSDM practices, as they were considered a part of the model)

For all of the XP, and Scrum and DSDM mappings and the coverage ratio specification, MDevSPICE® Class B requirements were considered. Due to the descriptive characteristics of the methods or limited information given in the available resources, we had to make some assumptions. Although it is very likely that some base practices would be performed during software development with XP, Scrum or DSDM, 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:

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

In this evaluation, the MDevSPICE base practices (BPs) are considered either partially or fully achieved. For the calculation of *the  $\Sigma$  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 [17] 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’ behaviour 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 [17], 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 2<sup>nd</sup> 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

<b>Primary Practices</b>	<b>MDevSPICE® Processes and Base Practices</b>
Weekly Cycle	<b>PRO.1 Project Planning</b> PRO.1.BP4: Define and maintain estimates for project attributes PRO.1.BP5: Define project activities and tasks <b>PRO.2 Project Assessment and Control</b> PRO.2.BP3: Report progress of the project PRO.2.BP4: Perform project review PRO.2.BP5: Act to correct deviations. <b>DEV.1 Software Requirements Analysis</b> DEV.1.BP2: Prioritize requirements. DEV.1.BP7: Baseline and communicate software requirements.
Quarterly Cycle	<b>PRO.2 Project Assessment and Control</b> PRO.2.BP3: Report progress of the project PRO.2.BP4: Perform project review PRO.2.BP5: Act to correct deviations.
Whole Team	<b>PRO.1 Project Planning</b> 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	<b>PRO.1 Project Planning</b> PRO.1.BP6: Define needs for experience, knowledge and skills.
Pair Programming	<b>DEV.4 Software Unit Implementation and Verification</b> DEV.4.BP1: Implement the software units.

Slack	<p><b>PRO.1 Project Planning</b> PRO.1.BP8: Define project schedule.</p> <p><b>PRO.2 Project Assessment and Control</b> PRO.2.BP5: Act to correct deviations.</p>
Ten Minute Build	<p><b>DEV.5 Software Integration and Integration Testing</b> 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.</p>
Continuous Integration	<p><b>DEV.4 Software Unit Implementation and Verification</b> DEV.4.BP4: Verify software units.</p> <p><b>DEV.5 Software Integration and Integration Testing</b> DEV.5.BP1: Integrate software units into software items.</p> <p><b>SUP.4 Software Release</b> SUP.4.BP1: Ensure the completeness of software verification</p>
Test First Programming / Continuous Testing	<p><b>DEV.4 Software Unit Implementation and Verification</b> DEV.4.BP4: Verify software units.</p> <p><b>DEV.5 Software Integration and Integration Testing</b> 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.</p> <p><b>SUP.4 Software Release</b> SUP.4.BP1: Ensure the completeness of software verification</p>
Incremental Design	Excluded from the analysis, as the definition of this process was not clear
Story	<p><b>DEV.1 Software Requirements Analysis</b> DEV.1.BP1: Define and document all software requirements.</p>

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

<b>Corollary Practices</b>	<b>MDevSPICE® Processes and Base Practices</b>
Real Customer Involvement	<p><b>PRO.1 Project Planning</b> PRO.1.BP6: Define needs for experience, knowledge and skills.</p>
Incremental Deployment	<p><b>SUP.4 Software Release</b> 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.</p>
Team Continuity	<p><b>PRO.1 Project Planning</b> PRO.1.BP6: Define needs for experience, knowledge and skills.</p>
Shrinking Teams	<p><b>PRO.1 Project Planning</b> PRO.1.BP6: Define needs for experience, knowledge and skills.</p>
Root-Cause Analysis	<p><b>SUP.8 Software Problem Resolution</b> 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.</p>
Shared Code/ Collective Code Ownership	<p><b>DEV.4 Software Unit Implementation and Verification</b> DEV.4.BP1: Implement the software units.</p>
Code and Tests	Contradicts with MDevSPICE®
Single Code Base	<p><b>DEV.5 Software Integration and Integration Testing</b> DEV.5.BP1: Integrate software units into software items. DEV.5.BP2: Verify that software integration follows integration strategy.</p>

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 artefacts 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 [17] 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 Process

(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 be 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 be 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® emphasizes 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, analysed, 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 behaviour 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 DSDM Mapping**

The DSDM Method is one of the early published agile methods, along with XP and Scrum. The method is described in terms of principles, a process (refers to life cycle description), people, products and practices. It adopts a *common-sense* idea which is described as “using sound practical judgement independent of specialised knowledge or training” and also *pragmatism*, a philosophy that favours actions taken based on evaluation of immediate consequences rather than theories and dogmas. The key principle of DSDM for keeping product quality at a desirable level is to allow scope and features to vary while the project cost and budget are constant. From the practices' perspective, DSDM also provides alternative options such as whether choosing a “structured timebox” or a “free format timebox” for managing the iterations. This makes it a good fit for adoptions that have to be made in regulatory domains.

In the mapping, we used the products and the practices of DSDM. The responsibilities of the DSDM roles were also considered in mapping inherently. We focused on actions/activities and outcomes rather

than abstract descriptions. Therefore, the principles of the method were not included in the mapping. We followed details provided in the DSDM Handbook on “<https://www.agilebusiness.org/what-is-dsdm>”

In Table 4 and Table 5, the mapping between the DSDM products, the DSDM practices and the processes and base practices of MDevSPICE® are provided (RQ1-RQ2). Considering there was no DSDM related publications in the MDSD, we added brief descriptions of products and practices in the second column of the tables. The bold text in the 3<sup>rd</sup> columns of Table 4 and Table 5 show the mapped processes. The other text in the same column cell refers to the mapped base practices (BPs).

**Table 4** Mapping of the DSDM Products and the MDevSPICE® Processes and Base Practices

<b>DSDM Products: either an Evolutionary (E) or a Milestone product (M)</b>	<b>Description of the product</b>	<b>MDevSPICE® Processes and Base Practices</b>
Terms of Reference - M	The Terms of Reference product defines the high-level objectives of the project. It provides a basis for feasibility studies.	<b><i>PRO.1 Project Planning</i></b> PRO.1.BP1: Define the scope of work.
Business Case - E	The Business Case product provides the vision and the justification of the project from a business perspective. The product is updated incrementally.	<b><i>PRO.1 Project Planning</i></b> PRO.1.BP1: Define the scope of work. PRO.1.BP3: Evaluate feasibility of the project.
Prioritised Requirements List - E	The Prioritised Requirement List (PRL) describes the project requirements from a high-level perspective, and indicates their priority with respect to the business objectives.	<b><i>PRO.1 Project Planning</i></b> PRO.1.BP1: Define the scope of work. <b><i>ENG.1 Stakeholder Requirements Definition</i></b> ENG.1.BP2: Obtain requirements. ENG.1.BP5: Identify critical requirements. ENG.1.BP6: Evaluate requirements. <b><i>DEV.1 Software Requirements Analysis</i></b> DEV.1.BP7: Baseline and communicate software requirements. <b><i>ENG.2 System Requirements Analysis</i></b> ENG.2.BP1: Establish system requirements.
Solution Architecture Definition - E	This product provides a high-level design framework for the solution. Both business and technical aspects of the solution are covered to a level of detail that makes the scope of the solution clear. This definition does not constrain evolutionary development.	<b><i>ENG.3 System Architectural Design</i></b> ENG.3.BP2: Describe system architecture. ENG.3.BP5: Verify system architecture. ENG.3.BP7: Communicate system architecture design.
Development Approach Definition - E	The Development Approach product provides a high-level definition of the tools, techniques, practices and standards that will be applied to the evolutionary development of the solution. It also describes the testing strategy and how quality of the solution will be assured.	<b><i>ENG.4 Software Development Planning</i></b> ENG.4.BP4: Plan the software development methods, tools and standards to be used in the development of Class C software items. ENG.4.BP5: Include or reference a software integration test plan in the software development plan. ENG.4.BP6: Include or reference a verification plan in the software development plan.
Delivery Plan - E	The Delivery Plan product is a high-level schedule of the project increments. The schedule rarely	<b><i>PRO.1 Project Planning</i></b> PRO.1.BP3: Evaluate feasibility of the project.

	includes items at the task level.	PRO.1.BP4: Define and maintain estimates for project attributes. PRO.1.BP8: Define project schedule. PRO.1.BP9: Allocate resources and responsibilities. PRO.1.BP11: Implement the project plan.
Management Approach Definition - E	The Management Approach Definition product defines how the project will be planned, stakeholder interactions and reporting.	<b>PRO.1 Project Planning</b> PRO.1.BP2: Define life cycle model for the Project PRO.1.BP7: Identify and monitor project interfaces. PRO.1.BP10: Establish project plan.
Feasibility Assessment -M	The Feasibility Assessment product defines whether the product is feasible to develop or not. Prototypes are used for feasibility analysis.	<b>PRO.1 Project Planning</b> PRO.1.BP3: Evaluate feasibility of the project.
Foundation Summary -M	The Foundation Summary product is the baselined collection of the products mentioned above. It can be considered as an executive summary covering the key aspect of each of the products given above.	<b>ENG.1 Stakeholder Requirements Definition</b> ENG.1.BP8: Establish stakeholder requirements baseline.
Evolving Solution - E	The Evolving Solution is described as a product that consists of the components of the developed solution at any time. These components can be models, prototypes, testing and review artefacts.  <i>The given definition is not enough to make a judgement about the work product</i>	<i>Unknown</i>
Timebox Plan - E	Timebox is defined as a fixed time period by DSDM. The progress and success are evaluated at the end of a timebox. Suggested duration for a timebox is two to four weeks.  In a structured DSDM timebox, specific planning and review sessions are followed, like daily stand-ups. Three steps of it are Investigation, Refinement and Consolidation, each of these stages are followed by a review session. The Timebox Plan is the detailed version of the Delivery Plan product. It is developed by the solution development team. Project work is displayed in the form of: work to do, in progress, and done. It is updated on a daily basis at the Daily Stand meetings.	<b>PRO2. Project Assessment and Control</b> PRO.2.BP3: Report progress of the project. PRO.2.BP4: Perform project review. <b>PRO.1 Project Planning</b> PRO.1.BP5: Define project activities and tasks.
Timebox Review Record - E	The Timebox Review Record includes the review of the work done during a timebox (iteration)	<b>PRO2. Project Assessment and Control</b> PRO.2.BP1: Monitor project attributes. PRO.2.BP2: Monitor project interfaces.

		PRO.2.BP3: Report progress of the project. PRO.2.BP4: Perform project review.
Project Review Report -M	The Project Review Report includes information on delivered and not delivered items.	<b>PRO2. Project Assessment and Control</b> PRO.2.BP1: Monitor project attributes. PRO.2.BP6: Collect project experiences
Benefits Assessment - M	The Benefits Assessment product is a document that describes the business benefits for the products delivered. It is a justification of the investment.	This product does not have a corresponding process or practice.

**Table 5** Mapping of the DSDM Practices and the MDevSPICE® Processes and Base Practices

<b>DSDM Practices</b>	<b>Description</b>	<b>MDevSPICE® Processes and Base Practices</b>
MoSCoW Prioritisation	<p>The MoSCoW Prioritisation is an approach that is used to prioritise any work item such as: requirements, user stories, tasks, products, use cases, acceptance criteria and tests. The letters stand for: “Must Have, Should Have, Could Have and Won’t Have this time”.</p> <p>Requirements (Prioritized Requirements List) are a kind of backlog at different levels of detail.</p>	<p><b>PRO.1 Project Planning</b> PRO.1.BP4: Define and maintain estimates for project attributes.</p> <p><b>DEV.1 Software Requirements Analysis</b> DEV.1.BP2: Prioritize requirements.</p> <p><b>ENG.2 System Requirements Analysis</b> ENG.2.BP4: Analyse system requirements.</p>
Iterative Development	<p>In an iterative development cycle several processes and practices are addressed. They are listed below:</p> <ul style="list-style-type: none"> <li>- Initial informal collaborative planning that evolves to a formal documentation over time.</li> <li>- Agreed acceptance criteria (AC) for the solution/features. Conversions are run to agree on the AC.</li> <li>- Definition of functional, non-functional and usability requirements.</li> <li>- It is ensured that the details of the requirements are understood by team members.</li> <li>- Architectural design of the solution is developed. DSDM defines alternative options to develop architectural layers from planning perspective.</li> <li>- It is ensured that the code is under configuration control.</li> </ul>	<p><b>PRO.1 Project Planning</b> PRO.1.BP8: Define project schedule. PRO.1.BP10: Establish project plan.</p> <p><b>ENG.4 Software Development Planning</b> ENG.4.BP6: Include or reference a verification plan in the software development plan.</p> <p><b>DEV.1 Software Requirements Analysis</b> DEV.1.BP1: Define and document all software requirements. DEV.1.BP4: Develop acceptance criteria for software testing based on the software requirements.</p> <p><b>ENG.1 Stakeholder Requirements Definition</b> ENG.1.BP4: Define user interaction. ENG.1.BP7: Agree on requirements.</p> <p><b>DEV.2 Software Architectural Design</b> DEV.2.BP1: Describe software architecture.</p> <p><b>ENG.3 System Architectural Design</b> ENG.3.BP1: Evaluate alternative system architectures. ENG.3.BP2: Describe system architecture.</p> <p><b>PRO.5 Configuration Management</b> PRO.5.BP4: Establish baselines. PRO.5.BP5: Control modifications and releases</p>

Reviews – Static Verification	<p>Informal and formal reviews are the types of the static verification process. Evidence of the review is kept. Formal reviews are planned beforehand.</p>	<p><b>DEV.4 Software Unit Implementation and Verification</b> DEV.4.BP4: Verify software units.</p>
Testing – Dynamic Verification	<p>Validation activities are performed as part of the evolving solution with the involvement of business ambassador and business advisor.</p> <p>Verification is fully integrated in Iterative Development cycle. The Development Approach definition product includes information on how the verification will be done.</p> <p>An independent tester also verifies the testing process/results.</p> <p>Tests are run to ensure that all acceptance criteria for the features are met.</p> <p>Positive, negative and unusual paths are tested in dynamic verification. Tests to run are prioritised based on the MoSCoW approach. Test classes are created for each test showing the action steps and possible outcomes. Exploratory tests are also run in addition to planned ones. A mix of manual and automated testing is suggested. Evidence of the testing activities are kept. Unit tests and business tests are the tests that are run as part of the dynamic verification activities.</p>	<p><b>DEV.4 Software Unit Implementation and Verification</b> DEV.4.BP2: Establish unit verification procedures. Fa DEV.4.BP3: Establish software unit acceptance criteria and ensure that software units meet the defined criteria. DEV.4.BP4: Verify software units.</p> <p><b>DEV.5 Software Integration and Integration Testing</b> DEV.5.BP3: Develop tests for integrated software items. DEV.5.BP5: Regression test the integrated software items after changes are made to software units, designs or requirements. DEV.5.BP6: Manage the anomalies found in the software integration testing.</p> <p><b>DEV.6 Software System Testing</b> DEV.6.BP1: Develop tests for integrated software product. DEV.6.BP2: Test integrated software product and record the results. DEV.6.BP3: Regression test integrated software when changes are made to software items. DEV.6.BP4: Manage the anomalies found in the software system testing. DEV.6.BP6: Evaluate software system testing. DEV.6.BP7: Record the software system tests.</p>
Modelling	<p>Several modelling techniques are used to improve traceability, understanding, clarity to check consistency. Storyboards, process flow diagrams, network diagrams are some of the examples of modelling. Prototyping is also used.</p> <p>User story mapping is the model type that is used for requirements analysis.</p>	<p><b>ENG.1 Stakeholder Requirements Definition</b> ENG.1.BP2: Obtain requirements. ENG.1.BP5: Identify critical requirements.</p> <p><b>ENG.2 System Requirements Analysis</b> ENG.2.BP1: Establish system requirements.</p>
Kick-off session	<p>Kick-off session is performed to understand the timebox objectives. It is performed at the beginning of each timebox.</p>	<p><b>PRO.1 Project Planning</b> PRO.1.BP11: Implement the project plan.</p>
Investigation	<p>Requirements are detailed and confirmation are received for them. Deliverables and acceptance criteria are decided and measures for success of the timebox were decided.</p> <p>Project dependencies are understood</p>	<p><b>DEV.1 Software Requirements Analysis</b> DEV.1.BP1: Define and document all software requirements. DEV.1.BP2: Prioritize requirements. DEV.1.BP4: Develop acceptance criteria for software testing based on the software requirements.</p>

	<p>Plan is reviewed for the work that needs to be done in future timeboxes.</p> <p>Risks are recorded into the Risk Log, analysed with solution development team</p>	<p>DEV.1.BP5: Verify all software requirements.</p> <p>DEV.1.BP6: Evaluate and update requirements</p> <p>DEV.1.BP8: Establish and maintain risk control measures in software requirements.</p> <p>DEV.1.BP.9: Re-evaluate and maintain medical device risk analysis.</p> <p><b>ENG.1 Stakeholder Requirements Definition</b></p> <p>ENG.1.BP4: Define user interaction.</p>
Refinement	<p>Includes development, testing and review activities</p>	<p><b>DEV.4 Software Unit Implementation and Verification</b></p> <p>DEV.4.BP1: Implement the software units.</p> <p>DEV.4.BP2: Establish unit verification procedures.</p> <p><b>DEV.5 Software Integration and Integration Testing</b></p> <p>DEV.5.BP1: Integrate software units into software items</p> <p>DEV.5.BP3: Develop tests for integrated software items.</p> <p>DEV.5.BP4: Test integrated software items in accordance with the integration plan and document the results.</p> <p><b>DEV.6 Software System Testing</b></p> <p>DEV.6.BP1: Develop tests for integrated software product.</p> <p>DEV.6.BP7: Record the software system tests.</p>
Consolidation	<p>At this stage it is ensured that all requirements are met based on the acceptance criteria.</p>	<p><b>DEV.4 Software Unit Implementation and Verification</b></p> <p>DEV.4.BP3: Establish software unit acceptance criteria and ensure that software units meet the defined criteria.</p> <p>DEV.4.BP4: Verify software units.</p>
Timebox Retrospective	<p>Retrospective analysis of the timebox is performed, and lessons learned are discussed, actions that should be taken to improve processes are decided.</p>	<p><b>PRO2. Project Assessment and Control</b></p> <p>PRO.2.BP6: Collect project experiences</p>
Daily Stand-up meeting	<p>A 15-minute meeting performed with development team to discuss the work done, planned work and issues.</p>	<p><b>PRO2. Project Assessment and Control</b></p> <p>PRO.2.BP3: Report progress of the project.</p> <p>PRO.2.BP4: Perform project review.</p>

Change Management	A formal change control process is not followed all the time, instead it is ensured that each team member is empowered to handle any change. Solution development team discuss the details of the change and escalates when needed. The key thing is to handle the change request as quickly as possible without any delays.	<p><b>DEV.1 Software Requirements Analysis</b> DEV.1.BP6: Evaluate and update requirements.</p> <p><b>SUP.9 Software Change Request Management</b> SUP.9.BP5: Define the criteria for confirming change requests. SUP.9.BP6: Identify the verification and validation activities to be performed for implemented changes. SUP.9.BP7: Evaluate and approve changes. SUP.9.BP8: Approved change requests are communicated to users and regulators. SUP.9.BP9: Implement the approved changes and review the implemented change.</p>
Requirements and User Stories	DSDM avoids defining requirements upfront in a detailed way. It uses the themes, epics, features, and user stories. It is also suggested to you INVEST criteria for better definition of requirements. There is also emphasis on requirements traceability in the model. Business needs - requirements	<p><b>DEV.1 Software Requirements Analysis</b> DEV.1.BP1: Define and document all software requirements. DEV.1.BP2: Prioritize requirements. DEV.1.BP4: Develop acceptance criteria for software testing based on the software requirements. DEV.1.BP5: Verify all software requirements. DEV.1.BP6: Evaluate and update requirements. DEV.1.BP7: Baseline and communicate software requirements.</p> <p><b>ENG.1 Stakeholder Requirements Definition</b> ENG.1.BP6: Evaluate requirements.</p>

According to the mappings shown in Table 4 and Table 5, the DSDM method, when implemented fully, is related to 13 processes of MDevSPICE®. Table 6 shows these processes and the coverage ratio of each process.

**Table 6** Coverage Ratios of the Mapped MDevSPICE® Processes from DSDM Perspective

	<b>Mapped MDevSPICE® Processes</b>	<b>CR</b>
1.	PRO.1 Project Planning	1.0
2.	PRO.2 Project Assessment and Control	1.0
3.	DEV.1 Software Requirements Analysis	0.78
4.	ENG.2 System Requirements Analysis	0.71
5.	DEV.2 Software Architectural Design	0.33
6.	ENG.3 System Architectural Design	0.43
7.	DEV.4 Software Unit Implementation and Verification	1.0
8.	DEV.5 Software Integration and Integration Testing	0.67
9.	DEV.6 Software System Testing	0.79
10.	ENG.1 Stakeholder Requirements Definition	0,75
11	ENG.4 Software Development Planning	0.25
12	PRO.5 Configuration Management	0.21
13	SUP.9 Software Change Request Management	0.39

Below, we discuss the coverage of each process given in Table 6, and what additional practices are required to achieve compliance with the medical requirements (**RQ3**). The DSDM practices are written in italics and underlined form to distinguish them with MDevSPICE® base practices, which are just given in the italic form. It should be noted that the practice concept of DSDM was different from the ones given in XP and Scrum. For example, under the description of Iterative Development practice, there were several sub-practices, such as definition of software requirements and acceptance criteria. As our focus is on the process coverage, this structure of DSDM did not affect the coverage results. We did the mapping for each description given in DSDM.

### **#1 PRO.1 Project Planning Process:**

**(CR of PRO.1 = 11 BP/11 BP = 1.0)**

In DSDM, the project planning process has a specific importance. Several products and practices are associated with the project planning process. For instance, *the Management Approach Definition* product is the master plan, along with which relevant plans, such as configuration management plan and software risk management plan, are needed to be established looking from the regulatory perspective. As DSDM provides a flexible structure to create necessary documentation, we consider that the base practices of project planning process are fully covered by the DSDM practices and products.

### **#2 PRO.2 Project Assessment and Control Process:**

**(CR of PRO.2 = 6 BP/ 6 BP = 1.0)**

The base practices of the project assessment and control process are fully covered by the DSDM practices and products. Project status and changes in the project are followed within a timebox. DSDM allows actions to be performed when project goals are not achieved. Deviations from the plan are corrected and preventive actions are taken in order to prevent recurrence of problems.

### **#3 DEV.1 Software Requirements Analysis Process:**

**(CR of DEV.1 = 7 BP/ 9 BP = 0.78)**

Although the products and practices enable definition of software requirements in DSDM, there are specific practices that need to be addressed for *DEV.1.BP1*. For example, as medical device software includes third party items, all functional and performance requirements including hardware and software requirements, e.g. processor type and speed, memory type and size, system software type, communication and display software requirements have to be specified. Therefore, *DEV.1.BP1* is considered partially achieved. Software requirements are prioritised using *the MoSCoW prioritisation* approach and a *Prioritised Requirement List (PRL)* is established afterwards. It is also stated that after the approval of PRL, any change to the requirements are addressed formally. From regulatory perspective, this means that the requirements are baselined and changes to them will be controlled (*DEV.1.BP7*). It was mentioned that the products are baselined and then called as *foundation summary*. It was stated in the model that software requirements are traced back to its origin. We assume that these would be stakeholder/business requirements. However, a full traceability is needed in the MDS and bilateral traceability links are needed to be established between system requirements and software requirements.

In medical device software development, special attention is given to risk analysis and mitigation. In DSDM process, risks are recorded into the *Risk Log* and analysed with solution development team. We assume that *DEV.1.BP8 (Establish and maintain risk control measures in software requirements)* and *DEV.1.BP.9 (Re-evaluate and maintain medical device risk analysis)* base practices are achieved with the activities performed at the Investigation Stage. However, more detailed risk practices which are listed in the *SRM.1 Software Risk Management Process* need special attention. However, we could not find a specific practice in DSDM for determining the impact the requirements on the operating environment (*DEV.1.BP3*).

### **#4 ENG.2 System Requirements Analysis Process:**

**(CR of ENG.2 = 5 BP/ 7 BP = 0.71)**

In the medical device industry, a medical device system may be composed of different types of requirements: hardware, software, electronic etc... When it is just a software product, we only define

software requirements. Although, there is no emphasis on definition of system requirements in DSDM, the mechanisms to specify and analyse software requirements work for system requirements as well. From this perspective, the *ENG.2.BP1: Establish system requirements* base practice are considered as partially achieved. The reason for the partial achievement is that specific medical device software requirements are needed when developing system requirements. On the other hand, *ENG.2.BP3: Optimize project solution*, *ENG.2.BP5: Evaluate and update system requirements*, *ENG.2.BP6: Ensure consistency*, and *ENG.2.BP7: Communicate system requirements* practices are considered as fully achieved.

The *analyse system requirements base practice* (ENG.2.BP.4) are considered as partially achieved. The achieved part of ENG.2.BP.4 is that system requirements are prioritized and the prioritized requirements are analysed for correctness, completeness, consistency, feasibility and testability. Additionally, the necessary elements of the system should be identified with the any changes to the operating environment as part *ENG.2.BP.4*. However, *ENG.2.BP2: Assign a safety class to the medical device based on the regional regulations* is Not Achieved.

#### **#5 DEV.2 Software Architectural Design Process:**

**(CR of DEV.2 = 2 BP/ 6 BP = 0.33)**

As part of the iterative development practice defined in DSDM, architectural design of the solution is developed. Therefore, we consider that *DEV.2.BP.1* for describing software architecture and *DEV.2.BP.2* for defining interfaces between software items practices are achieved. However, the other base practices of DEV.2 which are *analyzing the architectural design for correctness and testability*, *defining the software safety class*, *identifying and ensuring the effectiveness of segregation for risk management and ensuring consistency of software requirements analysis to software design*, are not achieved.

#### **#6 ENG.3 System Architectural Design Process:**

**(CR of ENG.3 = 3 BP/ 7 BP = 0.43)**

The *Solution Architecture Definition* product provides a high-level design framework for the solution. In this product, both business and technical aspects of the solution are covered to a level of detail that makes the scope of the solution clear. This product does not constrain evolutionary development. Alternative options to develop architectural layers are given in this product. From this perspective, we think that *ENG.3.BP1: Evaluate alternative system architectures* and *ENG.3.BP2: Describe system architecture* base practices are achieved. The DSDM teams and stakeholders frequently come together and communicate on the work in progress. Therefore, we assume that the system architecture design is disseminated to all parties who will be using it (*ENG.3.BP.7*)

#### **#7 DEV.4 Software Unit Implementation and Verification Process:**

**(CR of DEV.4 = 4 BP/ 4 BP = 1.0)**

The base practices of *DEV.4* process are achieved by several products and practices of DSDM. The *Refinement Phase* of DSDM includes development, testing and reviewing activities (that means achievement of *DEV.4.BP1* and *DEV.4.BP2*). In DSDM, both static and dynamic verification processes are applied. Informal and formal reviews are the types of the static verification process. On the other hand, automated and manual tests are run to ensure that all acceptance criteria for the features are met. These tests are run at the *Consolidation Phase* that means achievement of *DEV.4.BP3* and *DEV.4.BP4*.

#### **#8 DEV.5 Software Integration and Integration Testing Process:**

**(CR of DEV.5 = 4 BP/ 6 BP = 0.67)**

Even though DSDM does not specifically mention integration of software items and integration testing, its structure allows running integration testing activities.

Integration tests can be developed and run in the *Refinement and Consolidation Stages* (that is a full achievement for *DEV.5.BP1: Integrate software units into software items* and *DEV.5.BP3: Develop tests for integrated software items* base practices). From regulatory perspective, in addition to running

integration tests, we need to verify that software integration follows an integration strategy. Therefore, we consider that *DEV.5.BP2: Verify that software integration follows integration strategy* base practice is Not Achieved.

A specific integration and integration test plan need to be developed from regulatory perspective (*DEV.5.BP4*). In DSDM, test cases are developed and evidence of each test is kept. *DEV.5.BP4* is Partially Achieved, as there is no emphasis on developing integration plans in DSDM. The anomalies found in the software integration testing can be managed during dynamic verification activities (a full achievement for *DEV.5.BP6*). We also assume that regression tests (*DEV.5.BP5*) are performed in dynamic testing activities. However, as part of *DEV.5.BP5*, a special analysis needs to be performed for third party software updates that might be used medical device software.

### **#9 DEV.6 Software System Testing (CR of DEV.6 = 5.5 BP/ 7 BP = 0.79)**

The assumptions we have made for the *software unit verification and software integration testing* processes, are also valid for the software system testing process (*DEV.6*). The following base practices can be fully performed at the Refinement and Consolidation Stages: *Develop tests for integrated software product* (*DEV.6.BP1*), *Test integrated software product and record the results.* (*DEV.6.BP2*), *Regression test integrated software when changes are made to software items* (*DEV.6.BP3*), *Manage the anomalies found in the software system testing* (*DEV.6.BP4*) and *Record the software system tests* (*DEV.6.BP7*). Dynamic and Static test practices of DSDM are suitable for the *DEV.6* process.

However, the sixth base practice of *DEV.6* process which is Evaluating software system testing, is considered as Partially Achieved, due to the traceability links that need to be established between software requirements and tests that are run. Additionally, in medical device software development, when changes to software items are made during software system testing, relevant risk management activities have to be performed. The base practice related to this requirement (*DEV.6.BP5*) is considered Not Achieved as there is no risk evaluation/management emphasis at the system testing stage in DSDM.

### **#10 ENG.1 Stakeholder Requirements Definition (CR of ENG.1 = 6 BP/ 8 BP = 0.75)**

The Prioritised Requirement List (PRL) of DSDM describes the project requirements from a high-level perspective, and indicates their priority with respect to the business objectives. These high-level requirements both refer to stakeholder requirements and system requirements. Storyboards, user story mapping process flow diagrams and network diagrams are used for *obtaining requirements* (*ENG.1.BP2*;) and *identification of critical requirements* (*ENG.1.BP5*). In DSDM, it is suggested to use the INVEST criteria for evaluation of requirements which refers to *ENG.1.BP6: Evaluate requirements* in *ENG.1* process.

At the investigation phase, when requirements are detailed, the interaction between users and the system, taking into the account human capabilities and skills limitations are defined iteratively (full achievement for *ENG.1.BP4*). These refer to the usability requirements of the system. The Foundation Summary product keeps the baselined collection of requirements. This refers to the 8<sup>th</sup> base practice of *ENG.1* process which is *Establish stakeholder requirements baseline*. Acceptance criteria discussions performed in each iteration provides a means for communication on stakeholder requirements (*ENG.1.BP7: Agree on requirements*).

In MDSD, the constraints on a system solution which would affect management decisions and technical decisions need to be defined (*ENG.1.BP3: Define constraints*). Such constraints can be defined in the Terms of Reference (ToR) and Business Case (BC) products. In the current structure of the ToR and BC products, the constraints are not taken-into-account. Therefore, it is considered as Not Achieved.

Additionally, as part of *ENG.1.BP.1*, the stakeholders who have a legitimate interest in the system need to be identified (Not Achieved from MDevSPICE® perspective).

### **#11 ENG.4 Software Development Planning (CR of ENG.4 = 3 BP/ 12 BP = 0.25)**

In MDevSPICE®, a special emphasis is given to software development planning that require establishing a detailed plan for software development activities. Therefore, in addition to the *PRO.1 Project Planning* process, the *ENG.4 Software Development Planning* process is defined in MDevSPICE®. The *ENG.4* process starts with *assigning a software safety class to the software system (ENG.4.BP.1- Not Achieved)*. Because, depending on the safety class, processes to be performed, risk control measures to be implemented and evidences to be collected would change.

The Development Approach Definition (DAD) product of DSDM provides a high-level definition of the tools, techniques, practices and standards that will be applied to the evolutionary development of the solution. It also describes the testing strategy and how quality of the solution will be assured. The activities performed for the DAD product are associated with three base practices which are *ENG.4.BP4: Plan the software development methods, tools and standards to be used in the development of Class C software items (Fully Achieved)*, *ENG.4.BP5: Include or reference a software integration test plan in the software development plan (Fully Achieved)*, and *ENG.4.BP6: Include or reference a verification plan in the software development plan (Fully Achieved)*.

Although the base practices #4, #5 and #6 are performed as part of a DAD product, a separate software development plan has to be established and maintained for the medical device software to be developed (*ENG.4.BP2-Not Achieved*). A bilateral full-traceability strategy has to be established as part of the plan (*ENG.4.BP3- Not Achieved*). Additionally, as part software development planning activities; a *software risk management plan, a documentation plan, a configuration management plan, a plan to control supporting items (such as compiler/assembler versions, batch files, etc..) and a change management plan* have to be developed (*ENG.4.BP7, ENG.4.BP8, ENG.4.BP9, ENG.4.BP10, ENG.4.BP11, ENG.4.BP12 – Not Achieved for all*).

## **#12 PRO.5 Configuration Management**

**(CR of PRO.5 = 1.5 BP/ 7 BP = 0.21)**

In DSDM, there is not much information given regarding the configuration management process. However, in the DSDM handbook, it was stated that code is kept under configuration control. This practice refers to the *PRO.5.BP5: Control modifications and releases* base practice. For *PRO.5.BP5*, we consider a partial achievement, because along with code configuration, other configuration items such as documents or third-party software should be under configuration control. Additionally, the Foundation Summary product keeps the baselined collection of products (*PRO.5.BP4: Establish baselines- Fully Achieved*).

With this limited information, among seven base practices of the *PRO.5* process, we consider that only the *PRO.5.BP4 is fully and the PRO.5.BP5 is partially achieved*.

## **#13 SUP.9 Software Change Request Management**

**(CR of SUP.9 = 3.5 BP/ 9 BP = 0.39)**

DSDM has a specific process for change management which could either be run in a formal or informal way depending on the circumstances. It was stated that “*change control in a DSDM project tends to be more formal at the project level than it is at the Solution Development Team level*”. In DSDM, it is ensured that each solution development team member is empowered to handle any change rather than following a formal change control process all the time. Team members discuss the details of the change and escalates when needed. The purpose is to handle the change requests as quickly as possible without any delays. However, from the regulatory perspective, it is mandatory to manage, track and control the software changes. No practices were found in DSDM regarding *developing a change management plan (SUP.9.BP1-Not Achieved)*. We also do not know if *change requests are uniquely identified and recorded in DSDM tracking system (SUP.9.BP2 – Not Achieved)*. In DSDM, requirement changes are mainly managed in the Prioritised Requirement List. This structure would not be sufficient for *recording the status of change requests and maintain these records (SUP.9.BP3-Not Achieved)*. *The relationships and dependencies between the recorded change requests need to be established (SUP.9.BP4-Not Achieved)*.

The criteria which specify when to get the change requests approved are defined in DSDM. However, these criteria are not as strict as required by medical device software development requirements (*SUP.9.BP5-Partially Achieved*). Due to static and dynamic verification practices applied in DSDM, we consider that *SUP.9.BP6: Identify the verification and validation activities to be performed for implemented changes* is Fully Achieved. Before implementation, *the change requests need to be analysed to determine whether: a) additional potential causes are introduced contributing to a hazardous situation; and b) additional software risk control measures are required (SUP.9.BP7-Partially Achieved)*. After that the users of medical device software and regulators are required to be informed about *the changes (SUP.9.BP8 – Partially Achieved)*. The final base practice which is *SUP.9.BP9: Implement the approved changes and review the implemented change* is achieved by DSDM.

### 4.3 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®. The mapping has revealed that Scrum provides coverage for five processes at different levels. Table 7 below shows these processes and the coverage ratio of each process. The table showing the Scrum-MDevSPICE® mapping were provided in the publication titled with “How does Scrum Conform to the Regulatory Requirements Defined in MDevSPICE®?” [14].

**Table 7** CRs of Mapped MDevSPICE® Processes from Scrum Perspective

	<b>Mapped MDevSPICE® Processes</b>	<b>CR</b>
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

Below, we discuss the processes which have low coverage ratios with Scrum implementation in medical device software development.

#### **#3 ENG.1 Stakeholder Requirements Definition Process: (CR of ENG.1 = 5 BP/ 9 BP = 0.55)**

The following base practices of *ENG.1* are assumed to be achieved by the product owner and the development team in product backlog grooming sessions: *ENG.1.BP1: Identify stakeholders, ENG.1.BP2: Obtain requirements, ENG.1.BP3: Define constraints, ENG.1.BP6: Evaluate requirements, ENG.1.BP7: Agree on requirements*. However, the other base practices of this process need special attention which are not addressed in Scrum.

For an IEC 62304 Class B type medical software, user interaction has to be defined and evidence has to be provided. Based on the *ENG.1.BP4: Define user interaction* base practice the following information has to be defined for a medical device. In a product backlog grooming session, we may assume that all stakeholder requirements are specified. However, as part of the *ENG.1.BP5: Identify critical requirements* practice of MDevSPICE®; it has to be ensured that *health, safety, security, environment and other stakeholder requirements and functions that relate to critical qualities and shall address possible adverse effects of use of the system on human health and safety* are identified.

In medical device software development, every change on the product, whether it is on the artefacts or the code has to be made in a controlled way. This is one of the major contradictions between agile and the regulated worlds. For a change to be controlled, a version control system should be in place and baselines established. This is referred to in *ENG.1.BP8: Establish stakeholder requirements baseline*

base practice. However, a product backlog is a dynamic list which is continuously changing and no baselines are taken over it.

In Scrum, there is no specific emphasis on the development of a traceability schema. Therefore *ENG.1.BP9: Manage stakeholder requirements changes* is considered as Not Achieved.

#### **#4 ENG.2 System Requirements Analysis Process: (CR of ENG.2 = 5 BP/ 7 BP = 0.71)**

We may assume that base practices: *ENG.2.BP1: Establish system requirements*, *ENG.2.BP3: Optimize project solution*, *ENG.2.BP4: Analyze system requirements*, *ENG.2.BP5: Evaluate and update system requirements*, *ENG.2.BP7: Communicate system requirements* are performed in product backlog grooming sessions, as there are mechanisms to achieve them. However, the following two base practices need to be handled separately.

As part of *ENG.2.BP2: Assign a safety class to the medical device based on the regional regulations* process, at the system requirements analysis phase, a safety class has to be assigned to the product as the specific regulations apply based on the safety class in order to prevent potential harm to human life. As mentioned before, *bilateral traceability between the stakeholder requirements and the system requirements needs to be established* as part of *ENG.2.BP6: Ensure consistency* base practice.

#### **#5 DEV.1 Software Requirements Analysis Process: (CR of DEV.1 = 3 BP/ 9 BP = 0.33)**

We assumed that base practice, *DEV.1.BP1: Define and document all software requirements* is partially achieved, as there are specific issues that needs to be addressed for this BP. Based on FDA rules, software requirements have to be documented in a software requirements specification document and this document should contain details of the software functions.

It is important to determine the interfaces between the software requirements and other elements of the operating environment such as third-party software. This is achieved as part of base practice, *DEV.1.BP3: Determine the impact the requirements have on the operating environment*. At this stage, it is expected that the acceptance criteria for the software tests are defined from software requirements (*DEV.1.BP4: Develop acceptance criteria for software testing based on the software requirements*.) Scrum does not have such a rule.

As mentioned above, consistency of system requirements to software requirements has to be ensured. This is achieved through establishing and maintaining bilateral traceability between system requirements and the software requirements (*DEV.1.BP5: Verify all software requirements – Not Achieved*.)

The 7<sup>th</sup> base practice of DEV.1 requires establishing a baseline of software requirements and also providing communication of the software requirements. Due to use of communication channels in Scrum, we feel that the second part of this base practice can be achieved. However, the baseline of software requirements should also be added.

With the base practices, *DEV.1.BP.9: Re-evaluate and maintain medical device risk analysis* and *DEV.1.BP8: Establish and maintain risk control measures in software requirements*, it is ensured that risks regarding the software requirements are identified and risk control measures are defined. Risk management should be a part of daily or weekly Scrum review meetings. Finally, although Scrum proposes effective ways to manage projects, the special plans such as *software integration test plan*, *verification plan* and *software risk management plan* plans are not part of a basic Scrum method. Therefore, we assumed that *ENG.4 Software Development Planning* is not covered with Scrum, even though it is a “planning” process.

## **5 Discussion**

In this study, we performed the mapping of the XP, DSDM and Scrum methods’ processes, practices and work products with the processes and base practices of MDevSPICE®, an integrated framework of medical device software development. The mapping has revealed that XP, DSDM and Scrum methods fully or partially cover seven, thirteen, and five MDevSPICE processes, respectively, when they are

implemented in a medical device software development organization. Table 8 lists the covered processes together with their respective coverage ratios.

It was stated before that DSDM has been developed as a method to cover the entire software development life cycle. Therefore, it was not a surprise to get more coverage in terms of the processes and practices with DSDM. Among 23 processes of MDevSPICE®, *Project Planning*, *Project Assessment and Control*, and *Software Unit Implementation and Verification* are the processes that can be fully covered with DSDM and Scrum implementation. DSDM has been the single method that provides solutions for Configuration Management, Software Development Planning, Software Architectural Design and Software System Testing processes. On the other hand, Software Problem Resolution and Software Release processes are the ones, which are covered only with XP implementation. 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.

Risk Management, System Integration, System Qualification Testing, Software Installation, Software Acceptance Support, Software Maintenance, Software Detailed Design and Software Risk Management are the processes that none of these three agile methods could be linked.

None of the methods provide full coverage for medical device software development. This is expected as there are several special requirements for MDSD. However, a combination of these methods with appropriate tailoring would provide significant improvements in product quality, customer and employee satisfaction, and effective development.

**Table 8** The processes and coverage ratios for each method

No	MDevSPICE® Process	XP Coverage	DSDM Coverage	Scrum Coverage
1	PRO.1 Project Planning	0.32	1.0	1.0
2	PRO.2 Project Assessment and Control	0.5	1.0	0.9
3	PRO.4 Risk Management			
4	PRO.5 Configuration Management		0.21	
5	ENG.1 Stakeholder Requirements Definition		0.75	0.55
6	ENG.2 System Requirements Analysis		0.71	0.71
7	ENG.3 System Architectural Design		0.43	
8	ENG.4 Software Development Planning		0.25	
9	ENG.5 System Integration			
10	ENG.6 System Qualification Testing			
11	ENG.7 Software Installation			
12	ENG.8 Software Acceptance Support			
13	ENG.10 Software Maintenance			
14	DEV.1 Software Requirements Analysis	0.22	0.78	0.33
15	DEV.2 Software Architectural Design		0.33	
16	DEV.3 Software Detailed Design			
17	DEV.4 Software Unit Implementation and Verification	0.37	1.0	
18	DEV.5 Software Integration and Integration Testing	0.8	0.67	
19	DEV.6 Software System Testing		0.79	
20	SRM.1 Software Risk Management			
21	SUP.4 Software Release	0.43		
22	SUP.8 Software Problem Resolution	0.8		

## 6 Conclusions

In this paper, we analysed how well the medical device software development requirements are met by the implementation of XP, DSDM and Scrum and provided the coverage ratios of MDevSPICE® processes when these methods are implemented. The purpose of providing coverage ratios is to provide readers and practitioners with an indication of how much value is achieved with the XP, Scrum and DSDM implementation in medical device software development domain and what needs to be done more from a regulatory perspective.

Although full process coverages have only been provided in three MDevSPICE® processes, this study indicates how agile practices could be incorporated with MDSR requirements. This mapping has illustrated that the level of XP's support for project management processes/practices is very limited. Scrum is a good complement to XP for planning and assessment practices in MDevSPICE®. The combination of XP and Scrum provide association in nine MDevSPICE® processes. Inclusion of the DSDM practices increases that number to fifteen unique MDevSPICE® processes and enables reaching better coverage ratios in the challenging practices such as baselining and establishing end-to-end bilateral traceability.

As a future research plan, we will continue extending the mapping process to include other agile software development methods to be able to provide a better 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.

## 7 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* [Online]. 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] S.-D. Min, C.-W. Wang, H.-M. Lee, and B.-K. Jung, "A low cost wearable wireless sensing system for parietic hand management after stroke," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 5231-5240, 2018.
- [6] J. D. Hemanth, U. Kose, O. Deperlioglu, and V. H. C. de Albuquerque, "An augmented reality-supported mobile application for diagnosis of heart diseases," *The Journal of Supercomputing*, pp. 1-26, 2018.
- [7] *IEC 2006. IEC 62304: Medical Device Software - Software Life-Cycle Processes*.
- [8] F. McCaffrey, M. Lepmets, K. Trektore, O. Ozcantop, and M. Pikkarainen, "Agile Medical Device Software Development: Introducing Agile Practices into MDevSPICE®," 2016.
- [9] 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: IEEE, pp. 863-872.
- [10] 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.
- [11] M. Mc Hugh, O. Cawley, F. McCaffery, 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: IEEE, pp. 12-19.
- [12] P. A. Rottier and V. Rodrigues, "Agile development in a medical device company," in *Agile, 2008. AGILE'08. Conference*, 2008: IEEE, pp. 218-223.

- [13] 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.
- [14] 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: Springer, pp. 257-268.
- [15] J. Sutherland and K. Schwaber, "The scrum guide," *The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2013.
- [16] Ö. Özcan-Top and F. McCaffery, "Conformance to Medical Device Software Development Requirements with XP and Scrum Implementation," in *the 16<sup>th</sup> International Conference on Software Engineering Research and Practice*, Las Vegas, USA, 2018, pp. 99-105.
- [17] K. Beck, *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.
- [18] J. Stapleton, *DSDM Dynamic Systems Development Method: the method in practice*. Cambridge University Press, 1997.
- [19] *ISO/IEC 15504-5:2012 Information technology -- Process assessment -- Part 5: An exemplar software life cycle process assessment model*, 2012.
- [20] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," ed: VTT Finland, 2002.
- [21] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional, 2012.
- [22] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press, 2012.
- [23] D. Leffingwell, *SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering*. Addison-Wesley Professional, 2016.
- [24] J. W. Spence, "There has to be a better way![software development]," in *Agile Conference, 2005. Proceedings*, 2005: IEEE, pp. 272-278.
- [25] A. f. t. A. o. M. I. (AAMI), "AAMI TIR45:2012 Guidance on the use of AGILE practices in the development of medical device software."
- [26] K. 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," in *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on*, 2013: IEEE, pp. 26-34.
- [27] H. Mehrfard, H. Pirzadeh, and A. Hamou-Lhadj, "Investigating the Capability of Agile Processes to Support Life-Science Regulations: The Case of XP and FDA Regulations with a Focus on Human Factor Requirements," in *SERA (selected papers)*, 2010: Springer, pp. 241-255.