SERNAZ NAEENA AHMED

AGILE SCRUM IN MEDICAL DEVICE SOFTWARE DEVELOPMENT PROCESS

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
JUNE 2021

ABSTRACT

Sernaz Naeena Ahmed: Agile Scrum in Medical Device Software Development Process
M.Sc. Thesis
Tampere University
Master's Degree Programme in Software Development
June 2021

Because of the advancement of information technology, the medical industry has been integrated with innovative technologies such as automated devices and software to control their functionalities. Therefore, the traditional medical industries now develop more lightweight software systems and complex regulatory systems. This new dimension creates a scope for implementing new design methods and different frameworks for the medical device industry.

Medical Device Software Industry has international quality assurance standards; authorized organizations regulate all the development works. Adopting agile practices is often a source of concern for the developers of safety-critical software where there is a high risk of financial cost uprise and loss of control. The number of regulatory standards that medical device software development companies must meet before selling their software is often overwhelming. Standards and guidelines have been established for various regions to help businesses comply with regulatory requirements. Nonetheless, medical device software development companies face a significant challenge in adhering to the stringent regulatory requirements imposed due to the domain's safety-critical nature and maintaining productivity in the software development process.

This research is initiated for exploring the implementation of Scrum as a framework of the Medical Device Software Development in compliance with the regulatory environment. The purpose of this study is to identify the complications of implementing Scrum for developing Medical Device Software and proposing feasible solutions to mitigate the complications. A literature review is performed, and interviews are conducted to understand the current practices. Thesis findings are reviewed by comparing the results of the interviews, and the results are reported in conjunction with feedback from practitioners.

The research outcome shows that adopting Scrum in the medical device domain may have complications. Those complications can be mitigated by integrating the regulatory concerns into the development lifecycle of medical device software. By taking the necessary measures such as defining products considering regulatory compliance, balancing the flexibility and restrictions to make changes, integrating risk management activities, introducing roles to monitor the development from a regulatory perspective, it is possible to implement Scrum and comply with the regulatory requirements successfully.

Key words and terms: Software development, medical device, agile, scrum, software process improvement, medical device software development, safety critical system, regulatory systems, methods, traditional model, plan-driven.

Contents

1 Introduction

Agile principles have become an accepted methodology for developing various complex or multiplex systems and software for different fields. The term scrum comes from the word scrummage, which refers to a rugby player squad (Vogelzang, 2019). Scrum is an agile project management methodology applicable for complex projects with aggressive deadlines and complicated requirements, which create a degree of uniqueness and reduce design and development timeframes employing a collaborative approach.

Medical device manufacturing has to adhere to specific regulatory standards governed by national and international authorities to confirm the safety issues of the patients for whom it is used for. According to the European Commission, medical device software is also considered to be a medical device since these software systems are synchronized with medical devices or used for medical health care activities (Regulation (EU) 2017/745).

Agile methods and medical device software development have been studied and reported in many research publications since 2000. But, particularly for Scrum, there has not been much research on how to implement Scrum in the field of medical devices feasibly. It is still under research whether agile Scrum falls in line with the safety-critical requirements of traditional regulatory standards.

This research aims to find possible solutions for adopting agile Scrum in Medical Device Software Development (MDSD). A literature review studies the current state of the practices for adopting Scrum in the medical device domain to determine the complications of adopting Scrum in Medical Device Software (MDS). Afterwards, possible solutions to adopt this methodology in practice on regulated software development are researched. Interviews are conducted with practitioners who currently work in companies that develop applications for medical devices. The practitioners review the findings of this research by participating in interviews. The analysis of their review provides a more comprehensive perspective to this research work regarding compliance with medical regulatory requirements

The research questions are as follows:

RQ1: What are the complications to adopt ”Scrum” in developing medical device software?
RQ2: Can Scrum mitigate complications to develop medical device software?

The objective of this research is to see how far regulatory requirements can be met by Scrum implementation, which processes of medical device development can be covered by implementing Scrum, the current practices, and additional practices required for complying with regulation. This research starts with a literature review to learn more about this subject and the challenges of incorporating Scrum into MDSD processes. As a part of this study, an interview is performed among the developers of MDS in Tampere, Finland. This interview aims to assess the results of the literature review and learn the complications to adopt scrum practices in the development lifecycle when developing MDS and find how to implement Scrum in the medical device domain successfully complying with the regulatory requirements.

Research steps are as follows:

1. Background study on Scrum, safety-critical systems, and other related topics.

2. Understand the current practices and barriers of implementing Scrum in MDSD.

3. Interview practitioners from companies about the barriers to adopting Scrum to comply with regulations.

4. Analyzing how Scrum practices can mitigate the complications while developing software systems for medical devices.

The expected outcome of the thesis work is to demonstrate the complications of adopting Scrum in MDSD and representing how Scrum can overcome these complications. This research also discusses how Scrum, as a development technique, can be used to develop MDS effectively.

The following is the paper's outline: Chapter 2 provides the most relevant of the key concepts from background studies are described briefly, then in Chapter 3, there is the literature review about medical device software development. Chapter 4 discusses the mitigation of complications of implementing Scrum in the medical device domain. Chapter 5 consists of the interview design, interview participants, interview questions, interview findings and analysis of the interview. Chapter 6 discusses the research findings. Chapter 7 contains a conclusion and future work of this research, and the last section contains all the references exercised for this research.

2 Key Concepts

2.1 Agile Software Development Methodology

Agile software development is a methodology that requires continuous iterations for breaking the product into individual elements called practices during the development of a product in the Software Development Lifecycle (Shore, 2007). Compared to agile, Waterfall or other plan-driven development model follows sequential steps. Analysis, design, implementation, testing, and maintenance are all steps in the process. Nothing in the middle of the development process is deliverable. Agile is an alternative to traditional models of software project development. Agile has four ideal values and twelve principles, which are described in the Agile Manifesto. Agile allows solving problems earlier than other models such as Waterfall. Agile involves collaboration, face-to-face interaction, communication among the developers, different stakeholders, clients, and other members of the team who develop the product (Shore, 2007).

Values of the agile mentioned in the Agile Manifesto (Beck et al., 2001):

“Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan”

Principles of agile stated in the Agile Manifesto (Beck et al., 2001) include the following:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Businesspeople and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

6.The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7.Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9.Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11.The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The development life cycle in agile development is broken down into smaller fragments called "iterations", as shown in Figure 1, rather than the single broad development process model used in traditional software development. Each of these small iterations consists of the phases of a conventional development process. After each iteration is complete, a working software build is delivered. All of the `builds` have an increment in terms of product features. The project's final build includes all of the functionality requested by the customer (Pawar, 2015).

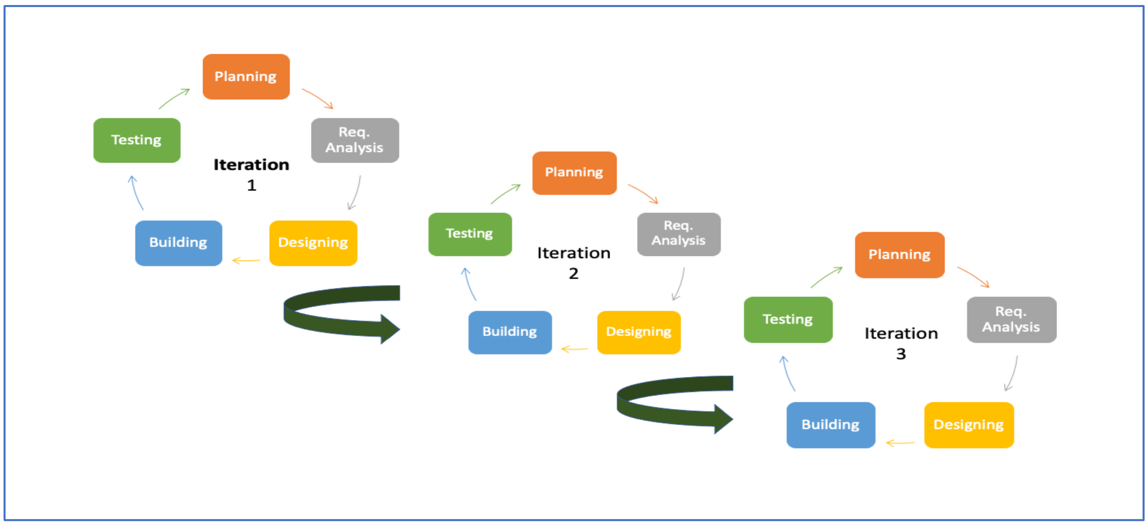

Figure 1. Agile Methodology, adapted from (Pawar 2015).

Agile software development has traditionally been known to be best suited for small groups of experts (Awad, 2005). It is based on flexibility and allows changes at any time. Such methods are lightweight, design and execution are always simple but effective in such approaches. It responds to changes quickly and efficiently. The key goal of agile is to keep the consumer satisfied by delivering a product on a regular basis. For that, it encourages feedback from the end-user and clients and adjusts teams' behaviour accordingly.

2.2 Scrum

Scrum is derived from a move in the sport of rugby, in which players must be in precise positions with a particular intent in order to accomplish a shared goal (Sutherland, 2014). Scrum is principally a management model for developing software proposed by Schwaber and Sutherland (Sutherland & Schwaber, 2020). The key concept of Scrum is to use process control theory to achieve flexibility, adaptability, and productivity when developing software (Abrahamsson et al., 2002). It acts upon a set of values and principles.

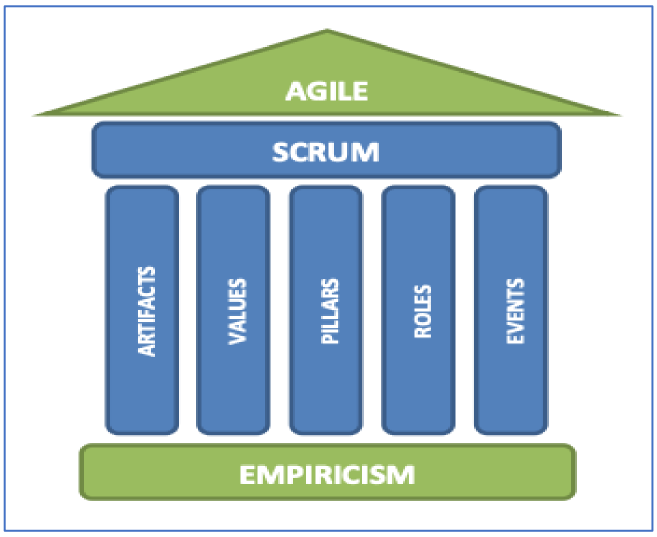

Figure 2. Base of Scrum (Bhavsar et al., 2020).

Figure 2 shows that Scrum is founded on an empiricism control theory that is controlled by Scrum's Artifacts, Values, Pillars, Roles, and Events (Bhavsar et al., 2020). Bhavsar discussed empiricism as a theory stating that information comes from experience and that decisions should be made based on what is experienced. The core pillars of Scrum that uphold this theory are Adaptation, Inspection and Transparency. Scrum optimizes predictability and controls risk by employing an iterative and incremental approach. Scrum engages groups of experts who collectively have all the skills to do the work and implement the empirical Scrum pillars in each event. Scrum values and principles strongly support technical practices, but there are no clear professional practices for Scrum implementation in a development project presented.

"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," by (Rubin, 2012). It assists effective development solutions by employing sufficient events. Scrum allows developers to apply various procedures and techniques to simple and complex software projects.

The Scrum framework consists of the following according to the Scrum Guide (Schwaber & Sutherland, 2020):

- Scrum Roles
- Scrum Artifacts and
- Scrum Events

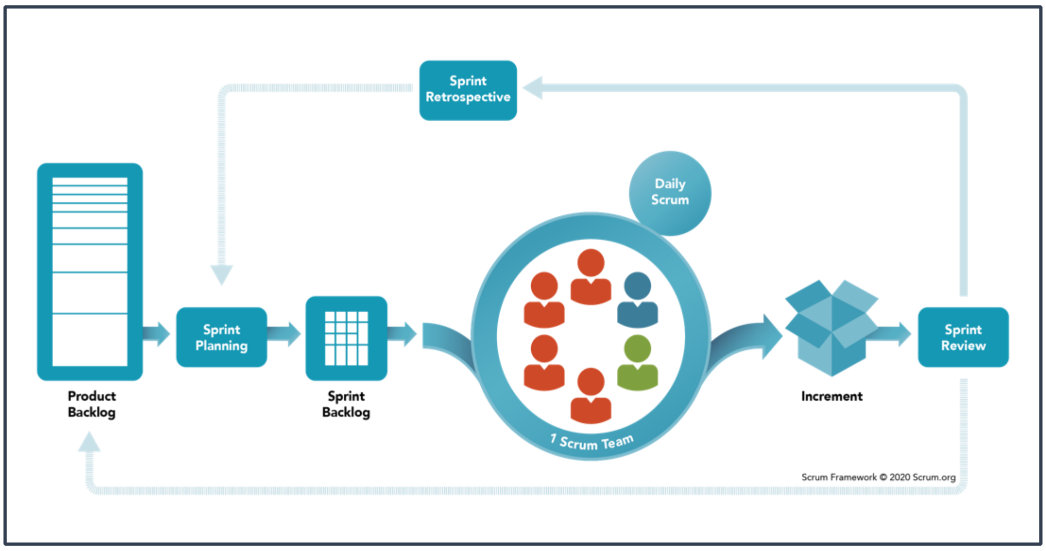

Figure 3. Scrum Framework (Schwaber & Sutherland, 2012).

Figure 3 shows actions from planning to software delivery in Scrum as outlined by Schwaber and Sutherland (Schwaber & Sutherland, 2012). Here, Scrum artifacts consist of the product backlog and the Sprint backlog and increment. Scrum events are labelled as Sprint planning, Sprint, Daily Scrum, Sprint Review, Sprint Retrospective. Scrum roles are indicated as a team

Scrum Team consists of a number of roles as following:

- Product Owner
- Scrum Master and
- Development Team

Scrum Teams are self-organizing and multi-functional, allowing them to complete their tasks independently. They have the freedom to decide on strategies and tasks to achieve the Sprint Goals instead of being controlled or being managed by others who are not members of the team (Sutherland & Schwaber, 2020). Scrum is intended for small collaborative development teams of about six to nine members. The success of Scrum relies on team members becoming more proficient in living five specific values as Courage, Focus, Commitment, Respect, and Openness (Sutherland & Schwaber, 2013). The Scrum Team is driven by these principles in their work, acts, and conduct.

Scrum defines three artifacts (Schwaber & Sutherland, 2020) such as product backlog, Sprint backlog and increment. Scrum artifacts are designed to maximize the transparency of key information. Each artifact contains a commitment ensuring that it provides information and enhances transparency. In addition, Scrum artifacts reinforce empiricism.

The product backlog is an ordered list of improvements that must be made to the product. Product backlog items that a Scrum team can complete within one Sprint are deemed ready for selection in a Sprint Planning event. The Scrum team's long-term goal is called the Product Goal. The Product Goal is the commitment for the product backlog.

The Sprint Goal is the single objective for the Sprint. The Sprint backlog is a plan by and for the Developers to accomplish during the Sprint in order to achieve the Sprint Goal. The Sprint backlog is updated throughout the entire Sprint. It should have enough detail that they can inspect their progress in the Daily Scrum.

The Definition of Done (DoD) is a description of the status of an Increment when it meets the quality measures required for the product. The moment a product backlog item meets the Definition of Done, an Increment is born, which is a concrete stepping stone toward the Product Goal. Each Increment of the Sprints is additive to the last Increment and thoroughly verified, ensuring that all Increments work together. Increments are presented at the Sprint Review, thus supporting empiricism. However, an Increment may be delivered to stakeholders prior to the end of the Sprint.

There are special events in Scrum as following (Sutherland & Schwaber, 2020):

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

These events have been created to establish regularity and enable transparency, inspect, and adapt Scrum artifacts continually. The Sprint is the main skeleton that holds the other events within this.

The development process is split into four-week iterations in Scrum, as illustrated in Figure 4. This iteration is called Sprint. At the end of each Sprint, the team builds a deliverable working increment of the product, allowing developers to forecast and measure progress and provide opportunities to the customer to make changes such as adding or redo a specific requirement. Before each of the Sprint, a Sprint-planning meeting is held, allowing the developers to select the tasks for the Sprint in collaboration with other stakeholders. After each Sprint, there is a piece of the project ready for the customer, reflecting the increment of the project development.

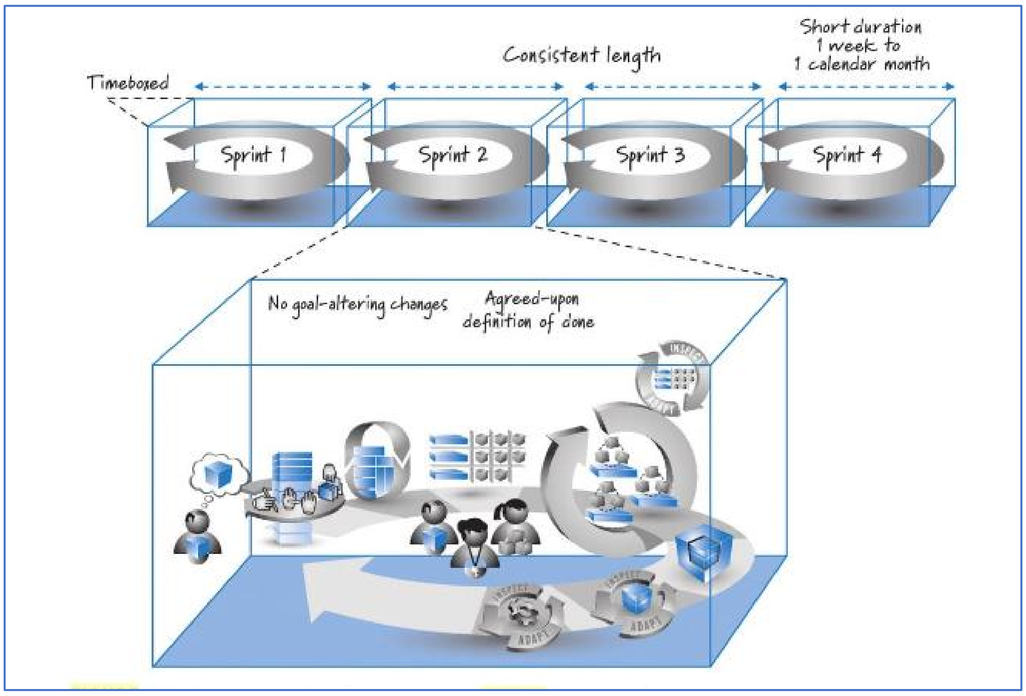

Figure 4. Sprint is the main skeleton of Scrum Framework (Rubin, 2012).

The customer reviews each increment of a Sprint for feedback gathering in Sprint review. After this, the deliverable is released to the customer. Sometimes the customer plays the role of a Product Owner in Scrum. User stories are created to encapsulate customer requirements, and then they are compiled into a prioritized product backlog. The product backlog is a “living” document since it is updated regularly and reflects the current understanding of customer requirements (Hron, 2018).

Every Scrum team needs a Scrum Master to oversee everyday work and ensure that the Scrum process is followed. Daily standup meetings named Daily Scrum, at which team members update each other on their progress and assignments for the following day, keep up the work speed. Learning is aided by retrospectives, which take place after each Sprint and allow the team to reflect on the work practices of the concluded Sprint.

2.3 Safety Critical Software Development

A safety-critical system is a complex, sensitive system that may result in death or injury of people, environmental damage, or significant financial loss, if it fails (Bozzano, 2011). According to the experts, “a safety-critical system is a system that poses a physical hazard to human life. If a failure occurs and exposes a hazard, it might cause physical harm to users, patients, practitioners, doctors, nurses, bystanders, and even people in the proximity of an accident. Examples of safety-critical systems include medical instruments, antilock brakes in motor vehicles, power hand tools, aircraft, and control systems in chemical plants” (Fowler, 2004). Fowler addresses the concern in safety-critical systems is that these must be developed thoughtfully with carefulness. They also require traceability for every detail of the development process.

According to Ericson (2011), such systems are typically subjected to a stringent safety assurance procedure such as safety certification. He also stated that the certification aims is to ensure conformity of the system so that the system is proved safe for use in a particular environment under certain conditions. For software-intensive applications, product and process certification is generally the most difficult step (Kornecki et al., 2009). He added that a common way to gain confidence in safety is to set and achieve safety targets that reduce the potential safety risks that a system can pose while in operation. These goals are typically based on safety criteria that apply to the domain where the device intends to be used. Complying with a standard entail collecting persuasive evidence during the system’s lifecycle to endorse the standard’s protection objectives (Nair et al., 2013).

Since electronic devices are becoming increasingly complex, software development processes are now much more interlinked with hardware devices used in the medical industry. The combination of hardware and software systems makes a vital contribution to the medical device domain to be used for health services. Therefore, in the medical industry, the number of such safety-critical software systems is growing.

Software reliability associated with design flaws, cannot be calculated employing statistical reliability growth models for safety-critical systems. The number of failure cases observed is insufficient to allow any statistical analysis of the results, as discussed in (Bouissou et al., 1999). Consequently, the traditional method based on statistical dependability models cannot be used for the assessment of such safety-critical software systems.

2.4 Regulations for Safety Critical Software Development

Regulatory software systems are designed and manufactured following customer's requirements along with national or international regulatory standardization, or even with both national and international in some cases (Mc Hugh, 2019). Usually, these regulations are specific for the area in which the software is intended to be marketed and used. There are different sets of standards and regulations to serve as a roadmap for the implementation of safety-critical systems (FDA, 1997). For safety-critical systems such as medical devices, regulatory standards give frameworks for developing such systems, but it does not specify how each step of the development process occurs.

In the EU region, currently three directives enforced by the European Commission govern the manufacture of medical devices. The EU countries must implement these directives in national legislation, according to the European Commission. The EU legislative framework for medical devices consists of three existing Directives and two additional Regulations (MDCG, 2021):

- Council Directive 90/385/EEC on Active Implantable Medical Devices (AIMDD),
- Council Directive 93/42/EEC on Medical Devices (MDD),
- Council Directive 98/79/EC of the European Parliament and of the Council on vitro Diagnostic Medical Devices (IVDMD),
- Regulation (EU) 2017/745 on medical devices (MDR), fully applicable from 26 May 2021 and

- Regulation (EU) 2017/746 on in vitro diagnostic medical devices (IVDR), fully applicable from 26 May 2022.

Regulation (EU) 2020/561 amending Regulation (EU) 2017/745 on medical devices is adopted by the Council due to the covid-19 outbreak.

The principal directive for medical devices is Council Directive 93/42/EEC on Medical Devices (MDD, 1993). Council Directive 90/385/EEC (AIMDD) and Council Directive 98/79/EC (IVDMD) may be required to follow depending on the product type. In order to apply similar regulations in all the countries, harmonization work is done by authorities such as Global Harmonization Task Force (GHTF) and the International Medical Device Regulators Forum (IMDRF) (Granlund, 2016). International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) are organizations to produce international standards harmonized in the EU region. These are referred to as `harmonized standards´. The directives and the regulations are significant part of European Union´s harmonization legislation on health, protection and safety, and efficiency of products in the internal market. These legislations establish the fundamental criteria of products intended to be marketed in the EU. Technical information and solutions to support those requirements are stated in European standards that have been harmonized. The regulatory medical products are developed following the applicable specifications of the relevant harmonized European standards. These standards provide solutions for implementing Medical Device Directives (MDD). The main objective of these standards is to ensure safe and effective products in the medical device domain.

Valvira (2009) is the National Supervisory Authority for Welfare and Health. It regulates medical device manufacturing in Finland (Granlund, 2016). All medical devices eligible to be marketed in Finland must conform to existing EU regulations and be safe to use. Manufacturers of medical devices in the EU must prove the performance and reliability of their products and their suitability for the intended use; they must prove compliance with the Medical Device Regulations, MDR (Regulation (EU) 2017/745).

The main objective of imposing these regulations is to provide a general framework for the manufacturer in order to protect the users by guarding against unsafe medical products. MDR does not differentiate between the physical medical devices and medical device software used for patients directly or with other physical medical devices. Regulation 2020/561 amending Regulation (EU) 2017/745, Article 2 states in the definition section that,

" ...'medical device' means any instrument, apparatus, appliance, software, implant, reagent, material or other article intended by the manufacturer to be used, alone or in combination, for human beings for one or more of the following specific medical purposes: diagnosis, prevention, monitoring, prediction, prognosis, treatment or alleviation of disease..... and which does not achieve its principal intended action by pharmacological, immunological or metabolic means, in or on the human body, but which may be assisted in its function by such means."

The MDR regulations guarantee that medical devices run smoothly by setting high quality and safety standards to address and resolve common safety concerns. MDR defines general guidelines for the placement on the market, rendering available on the market, or bringing into operation in the Union of medical devices and accessories for human use, which is applied to clinical trials involving such medical devices and accessories. These rules are in place to ensure the protection and efficacy of devices designed for human use.

According to Fowler (2004), these regulations do not prescribe particular practices to be used during such software development. In preference, such regulations provide a framework that enables manufacturers to create design controls consistent with the rules and appropriate for their system development. Henceforth, the regulatory guidelines and harmonized standards provide frameworks for developing MDS where manufacturers can choose their development processes. In addition, they can tailor the details of the development lifecycle according to their convenience.

3 Medical Device Software Development

3.1 IEC 62304 for Medical Device Software Lifecycle

International harmonized standards are discussed in Section 2.4. European standards that are harmonized for developing MDSD are as follows (Commission communication in the framework of the implementation of the council directive 93/ 42/EEC concerning medical devices, 2017):

- ISO 13485:2016 for a quality management system,
- IEC 62304 for software development lifecycle activities,
- ISO 14971 for risk management,
- IEC 62366-1 for usability engineering,
- IEC 82304-1 for medical device software.

These are harmonized against the old legislation acts MDD and IVDD. No harmonized standards for the new legislation act MDR and IVDR are in action till May 2021. IEC 62304 is the main standard that provides a guideline for software development lifecycle activities. Other standards may also be applicable for an MDS depending on the class and type of the software.

Currently, the emphasis has increased on the medical device's software system's reliability and the risks associated with it at all levels of use since the embedded software systems have become more demanding in the medical device domain nowadays. IEC 62304 is a harmonized standard for software development in the medical device domain adopted by the European Union (EU) that has become a global international standard for software development lifecycle management. Medical device manufacturers implying this standard will ultimately comply with the MDD. Certainly, it ensures the production of high-quality software through a well-defined and well-controlled software development process, including a collection of specifications based on the software's safety class assigned to the software system considering the potential risks associated with the usage of the system. The production of a safety-critical software system can be split into items if the constituent parts of an MDS possess different risks and non-identical safety classes. It is a crucial necessity that the software splitting is appropriate to confirm the highest quality possible and the safety of the end-user. Software development processes must include verification processes, integration testing and system testing, a well-established risk management system and documentation for all software regardless of the safety classes in MDSD, according to the IEC 62304 (2006).The safety-classes the manufacturers must assign to software based on the potential to create hazardous situation are as follows (IEC 62304, 2006); the higher the safety class, the more effort is required to ensure safety:

- Safety Class A: No injury or damage to health is possible.
- Safety Class B: Non serious injury is possible.
- Safety Class C: Death or serious injury is possible.

IEC 62304 (2006) demands process, activity, and task to be applied in the software development lifecycle to incorporate the compliance. This standard has nine sections, where the first four sections (1 to 4) are: scope, normative references, terms and definitions, general requirements. Then it defines five processes (5-9) such as:

- Software Development Process, the main process including all other processes from planning to release,
- Software Maintenance Process to form of the software development process for handling risks and bugs,
- Software Risk Management Process for risk assessment,
- Software Configuration Management Process to control code and system development environment, to manage builds and releases and
- Software Problem Resolution Process for bug tracking and resolving.

The standard defines activities in software development process (section 5) as the following:

- Software Development Planning (define the project's scope of work),
- Software Requirements Analysis (convert requirements into software requirements and define the features that must be incorporated),
- Software Architectural Design (high level design of the software, including partitioning the constituent parts of the software with different risks),
- Software Detailed Design (define the software units and their interfaces such as state diagrams, data structures, and risk controls),
- Software Unit Implementation & Verification (verify the coding and testing,
- Software Integration & Integration Testing (test to merge software components ensuring that all of them work properly together and with the associated hardware system, in cases),

- Software System Testing (verification of entire software system against the requirements) and
- Software Release (deploy a verified version of the software).

The software maintenance process is described in section 6 of IEC 62304 (2006), which includes: establish a software maintenance plan, problem and modification analysis and implementation of modification.

The software risk management process in section 7 includes activities and tasks (IEC 62304, 2006) such as analysis of software contributing to a hazardous situation, risk control measures, verification of risk control measures and risk management of software changes.

Hence, IEC 62304 legitimizes MDS development, providing general guidelines to follow during the software lifecycle and safety requirements to implement within the medical device domain to comply with the regulation. However, this standard does not specify the manufacturer's organizational structure and does not dictate which parts of the organization are responsible for the process, activity, or task. For example, it includes tasks documentation, but it is up to the standard's user or the manufacturer to decide how to produce this documentation. This standard does not mandate a particular life cycle model; the medical device manufacturers are responsible for choosing a life cycle model for the software to map the required processes, activities, and tasks.

3.2 Plan-Driven Sequential Software Development

MDS is often a part of physical medical devices or hardware systems. It must be developed under the regulatory requirements to ensure that the system is safe, reliable, and secure to be operated. System developers usually use a plan-driven sequential Software Development Lifecycle (McCaffery, 2011). According to Boehm and Turner (2005), plan-driven methods (such as Waterfall, Spiral, and V-model) are common approaches for software development based on quality disciplines. They are characterized by strong documentation and traceability throughout the life of the system. These models focus on predictability and validation. A vital part of the development process is the risk management process.

The Waterfall Model or V-Model software development methodologies are commonly used when developing software for safety-critical domains (Xiaocheng et al., 2010; Bulska, 2011)]. These life cycles are characterized by upfront design, which places a great deal of emphasis on the production of documentation (Xiaocheng et al., 2010).

According to Zema et al. (2015), the Waterfall model is a linear sequential plan-driven method in which software implementation progress is viewed as streaming steadily downwards like a waterfall through the development phases, as shown in Figure 6. In the Waterfall model, the client's specifications must be specified at the requirement elicitation phase before moving on to the next step of the development life cycle, as the requirement phase must be completed before the design phase can begin.

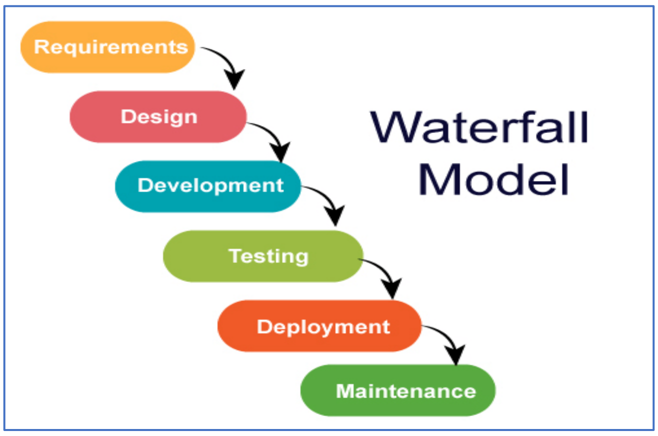

Figure 6. Waterfall Model (Balaji & Sundararajan, 2012).

In this model, risk management activities or safety requirements can be included in the requirement elicitation phase. Then other phases such as development, testing, deployment is followed one by one. Considering the IEC 62304, these activities are required to fulfil the requirements of the assigned safety class defined by the regulatory bodies. If the client wants any changes in requirements during the development phase, it will not be considered.

Consequently, the issues with one process are never fully resolved during that phase. Many issues with a process occur after the phase has been fully closed. These cause inflexibility and rigidness (Balaji & Sundararajan, 2012).

The V-model, also known as the validation and verification model, is a modified version of the Waterfall method (Balaji & Sundararajan, 2012). This balanced developmental model necessitates verification from the previous process before moving on to the next. MDS developers widely use the V-Model (Bulska, 2011) because it provides critical deliverables such as requirement traceability at all levels of the software development lifecycle, which is a vital point for regulatory approval (McCaffery, 2011).

It is necessary to have clear linkages among different development phases and maintain traceability to confirm regulatory compliance, including risks or safety, throughout the various stages of the software development life cycle. Clear demonstration is required on how the development life cycle is followed (McCaffery, 2011).

It is also possible to incorporate the risk management practices into the typical stages of the development process according to the classical V-model (Scholtes et al., 2018). Hence, the V-model is remarkably appropriate for regulatory requirements (McCaffery, 2013) from international standards such as IEC 62304.

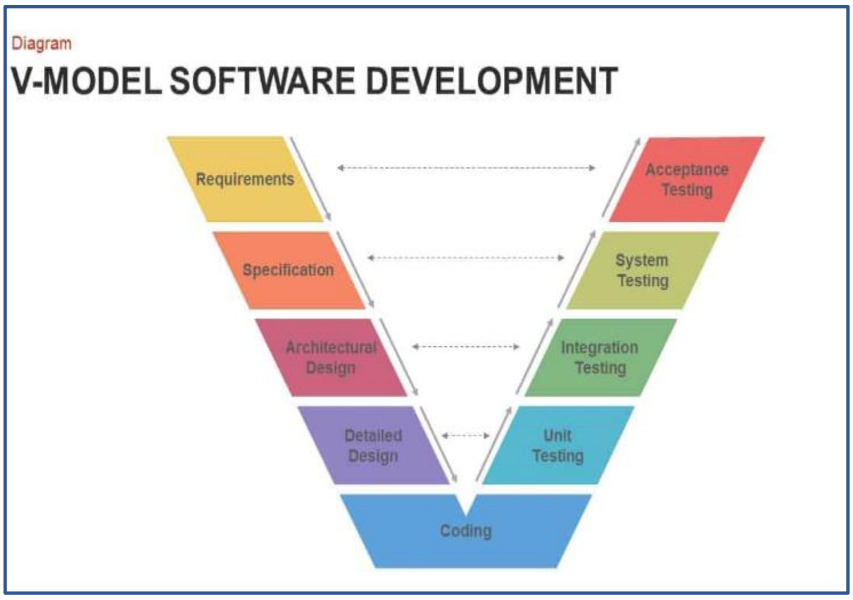

Figure 7. V-model (Balaji & Sundararajan, 2012).

Balaji and Sundararajan (2012) stated that V-model relates each specification phases (left tail of the V) to the testing phases (right tail of the V) as shown in Figure 7, where tails meet represents development phases. The System Test cases are created based on the requirements. High-Level Design and Low-Level Design are used to describe integration test cases. The coding is then completed. Unit-testing, integration testing and system testing are performed in an orderly sequence after the coding is finished. If any changes happen middle of the development, phases such as functional specification, high-level design, low-level design, unit testing, system testing, integration testing are affected even though requirement changes are said to be embraced at in any time during the development in this model. Documentation prepared from different phases needs to be updated as well. Though it creates scope to integrate regulatory requirements, it may also be considered as being rigid and inflexible if there is a change after the development process has started.

Although traditional models such as Waterfall or V-model are rigid as described above and have several limitations, these are still valid today. Özcan-Top and McCaffery (2017) listed some reasons why these are still valid as follows:

(a) With these models, producing the required deliverables to meet regulatory audit requirements is relatively simple (since, these models follow predefined requirements as discussed above).

(b) In the development of MDS, verification, validation, and risk assessments are especially significant, and these procedures are designed and performed following the V-Model's development phases (as mentioned above, risk management activities are integrated into the typical development phases). (c) Each process must be completed before moving on to the next in these models.

According to them, these models work appropriately when the requirements are specified with high confidence. However, even though these models seem to be structured in a way that enables regulatory compliance, consistently confirming the regulatory requirements is an inevitable difficulty that MDS manufacturers encounter who adopt such methods for their project development (McCaffery, 2017). Because continuous requirements change is not a problem to be solved; rather, it is the nature of the software projects (Scholtes et al., 2018).

Some other companies surpass the change throughout the development process by being well-timed to market, guaranteeing high quality, safety, and high productive capacity (McCaffery, 2017). When it comes to a focus of overcoming rigidity and inflexibility, the agile software development approach has shown to be successful (Abrahamsson et al., 2002) in different regulatory fields such as MDSD.

3.3 Agile Software Development

Agile software development techniques have gained universal recognition and use in all sectors worldwide. As discussed in Section 3.1, the use of a specific lifecycle for MDSD is not defined by regulatory criteria or development specifications provided by EU (ICE 62304, 2006). Instead, MDS can be built using a traditional or iterative approach according to the convenience of the manufacturers. Hence, developers of today´s critical systems have concerns about implementing agility instead of the conventional inflexible models. Therefore, medical device manufacturing companies are adopting modern methods to attain not only agility but also safety and reliability (McCaffery, 2017), even though there are issues to solve when doing so.

3.4 Complications of Adopting Agile

Agile methodology and safety-critical regulatory systems are considered incompatible with each other (Stålhane, 2012; Fitzgerald, 2013). This may be due to a disaffirmation between the Agile Manifesto (Fowler, 2001) and regulatory requirements since agile provides the developers with ease rather than formality.

Agile Manifesto depicts four fundamental value propositions for agile, as mentioned in Section 2.1. The agile framework advocates more value the left part of the statements, whereas the regulatory environments acknowledge more importance of the right parts (Fitzgerald, 2013). Hence, this assessment might reach out an agreement that agile approaches and regulated systems are not commensurable.

In a systematic literature review, Heegar and Nielsen (2018) clarified four concerns for adopting agile method to the safety-critical software development.

First one is “Light documentation”. Safety-critical software development processes rely on formalized processes focused on documentation, while agile processes rely on face-to-face communication and informal collaboration.

The next issue is “Flexible requirements in user stories”. Agile methodologies promote changeability of specifications and a less formal specification. Changes, on the other hand, pose challenges to documentation in a safety-critical production, and specifications documentation must be done formally. Here the term “formal” means field specific formal notations and methods such as VDM, LOTOS and so on (Bowen, 1993).

\`Iterative and incremental life cycle´ is another area of concern while implementing agile in safety-critical systems. To foster learning and adoption, the agile development approach implies an iterative life cycle. In contrast, in safety-critical systems, implementation reliability is prioritized, and development follows a strict life cycle.

The last issue found in their literature review is \`Test-first processes´. Test-driven development and iterative testing are central to agile methodologies. In safety-critical development, on the other hand, the testing is performed in the final stages of the project. Instead of using test-driven production, detailed test plans are used.

Mc Hugh et al. (2017) conducted a questionnaire-based survey of MDSD companies in Ireland, inquiring about the perceived barriers to agile adoption and the actual barriers to adopting agile practices. The survey revealed the following barriers:

Percieved Barriers

- 25% of respondents reported “Lack of Documentation”.
- 25% of respondents reported “Regulatory Compliance”.
- 16% of respondents reported “Lack of Up-Front planning”.
- 17% of respondents reported “Insufficient coverage of risk management activities.

Actual Barriers

- 50% of respondents reported “Lack of Experience”.
- 33% of respondents reported “Having to change the existing lifecycle”.
- 16% of respondents reported “Management Opposed to Change”.
- 16% of respondents reported “Team size”.
- 17% of respondents reported that “Getting stakeholder buy-in” as a barrier.
- 17% of respondents reported “Level of retraining required” as another barrier to agile adoption.

100% of the respondents of the survey (Mc Hugh et al., 2017) were developing MDS for using in Europe. Among them, 79% were also developing MDS for the US market. The survey results clearly represent that, the issues arising from implementing agile methods relate to the lack of formal processes, the organizational structure, and the expertise level of the team.

In practice, several contradictions exist with implementing agile in safety-critical systems. As such, the informal evaluation technique (without any field-specific formal method) is one of the principal contradictions. This issue might lead to inadequacy in the quality of safety-critical systems (Turk et al., 2002). Boehm and Turner (2005) discovered that combining risk management activities with agile can be challenging. This issue can be an obstacle to implement agile practices in the safety domain because agile concepts do not include sufficient guidance about conducting the requisite risk management activities. Another challenge for adopting agile practices is ensuring the highest quality and efficiency. Boehm and Turner (2005) mentioned that software developed using agile practices has a lower quality assurance than software developed using traditional plan-driven life cycles. This challenge is one of the 40 perceived barriers to agile implementation identified by them in an annual workshop at the University of Southern California Center for Software Engineering. Since MDS is safety-critical, it must ensure the highest possible quality even in a limited or short release, which agile methodologies, such as Scrum, allow for (Abrahamsson et al., 2002). It is unreasonable to release unfinished software and wait for input when designing software for a medical device. The software must be thoroughly checked and functional before being used to treat patients (FDA, 2007).

Aside from these issues, the lack of management control is another possible major roadblock to adopting agile methods when designing safety-critical applications. Project teams should be self-organizing, according to agile methodology. This approach of self-organizing teams deprives management of any decision-making authority (Moe et al., 2008). Consequently, this can lead to a lack of management power in the organization (Salo & Abrahamsson, 2005). The paper also mentions the need for organizational resources for agile practices to succeed.

A regulatory system such as MDS development requires up-front planning for a complete design scheme to comply with the regulatory requirements. In contrast, in agile, the processes are incremental to encourage continuous learning.

Changes in requirements are welcomed in any phase because of the incremental approach. These changes may also raise an issue for regulatory compliance. There are two types of requirements considered when developing medical device software in practice. One is internal requirements including process and product requirements (usability, performance, code, UI design), the other is the external regulatory requirement (requirements from the standards and regulations). Requirement change is possible in the MDS development process, but some boundaries cannot be crossed. If the required change is a functional requirement that does not affect the regulatory requirements, then the change can be allowed. Otherwise, the change may not be possible.

For satisfying the regulations, it is mandatory to include risk management in the development lifecycle while the agile processes do not introduce such activities. Other concerns are existing between agile and MDS development.

Table 1: Complications of agile implementation in MDS development.

Agile Software Development	**Regulatory Medical Device Software Development**
Agile lacks up-front planning for a complete design.	Requires complete design scheme developed to comply safety requirements.
Agile allows flexible requirements and changes in any phase.	Requirements include both regulatory requirements and functional requirements in MDS development. Requirements related to functionality can be changed or altered if required. Regulatory requirements cannot be changed without the approval of notified body.
Agile possesses an incremental life cycle.	Follows a sequential flow of work for satisfying the regulatory requirements.
Agile prioritizes informal coordination over documentation, lacks formalized documentation.	Regulatory compliance requires detailed technical documentation in every phase of the development cycle from requirement elicitation to testing phase as per the regulatory bodies need for the compliance. Upon a request of the regulatory body, all documents must be available.
Agile lacks detailed test-plan.	Regulatory compliance demand high quality and safety, consequently, requires detailed test-plan to ensure safety measures.
Agile does not specify risk-management activities, thus possesses less quality assurance.	For regulatory compliance, risk management activities must be included in the development cycle, not only for functional risks related to code, but also for patient´s safety.
Agile teams are self-controlled, lacks management control.	Involvement of regulatory bodies outside the team is a must for regulatory compliance. Regulatory bodies are responsible for approving the release and other necessary verification.

In Table 1, all the possible complications of adopting agile, which are derived from the above discussion, are listed in the first column. The second column describes the actions required and measures to be considered to mitigate those complications regarding the regulatory medical device domain.

Despite recognizing these concerns, several researchers such as Rayside et al. (2009) and Black et al. (2009) argued that while there are challenges in implementing agile methods in safety-critical development, none of these predicaments will restrict agile methods in safety-critical development under any circumstances.

3.5 Complications of Adopting Scrum

The agile approach has some complications when implementing in regulatory environments such as MDS development, as discussed in Section 3.4. Similar problems are perceived for Scrum since Scrum is a software development methodology based on the agile philosophy. In addition to that, the Scrum approach has more specific rules and principles. Thereupon several issues arise when software manufacturers implement Scrum in the MDS development process. Regarding fulfilling the regulatory compliance, the following complications have been persuaded in this research: product definition, requirement changes, uncertainty and unpredictability, documentation and traceability, verification and validation, risk management, self-organized team, number of roles.

3.5.1 Product Definition & Quality Assurance

Scrum Team members must have a shared DoD as discussed in Section 2.3. However, it provides no concrete guideline on what should be in the DoD. Teams themselves define it. It is of high risk for regulatory systems if development is considered complete without proper testing, proper documentation, or other safety requirements that ensure that the product is ready to be used. It may be hard to satisfy the quality requirement of the MDR if the DoD is defined without considering the Regulations.

According to paragraph 4, article 10, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561, the requirements need to be demonstrated following the conformity assessment to allow the device's conformity with the requirements of this Regulation. Article 9 states that manufacturers must establish proper documentation, implementation, maintenance and need to improve quality management system to ensure regulatory compliance with this Regulation in the most effective manner.

3.5.2 Flexibility in Requirement Changes

Scrum welcomes requirement changes at any phase of the development to enable continuous learning. It does not have stability in the requirement specification. The requirements are listed in the product backlog; the team can make changes to them by consulting with the Product Owner.

On the contrary, regulatory requirements require prior specifications of the requirements because some changes may result in disaffirmation or contradiction to the regulations. According to paragraph 9, article 10, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561, manufacturers must ensure that processes are followed to ensure that series production meets the Regulation's requirements regarding conformity. Any changes in the design or characteristics of a device must be considered promptly. This confrontation may also be an issue while adopting Scrum in MDS development.

3.5.3 Unpredictability and Variability

Scrum leverages variability and uncertainty (Rubin, 2012). In Scrum, all processes follow a well-defined set of steps. At the same time, Scrum supports the fact that some level of variability is required to create something innovative. Each Sprint allows the team to learn continuously to improve what they build and how they build. Hence, Scrum leverages unpredictability when dealing with complex projects. Even the project's design scheme does not include the complete design of the project since solutions are innovative and designs are creative in Scrum methodology as it follows an iterative and incremental approach.

On the contrary, regulatory compliance demand predictability to conform to the safety issues and require a complete design scheme, according to article 10, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561. Where there is unpredictability, there is a risk of unconformity.

3.5.4 Documentation & Traceability

Scrum principles include transparency, but it does not emphasize documentation. On top of that, changes are allowed in requirements anytime during the Sprint. If the developers do not trace changes and their effects on the development processes, it might be a big issue for regulatory compliance. Documentation of each phase is needed to trace the changes in regulatory MDS development.

According to paragraph 8, article 10, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561, upon request by a competent authority, the manufacturers should be able to provide required technical documentation. Examples of technical documentation are product requirement document, UX design document, software architecture document, user-manual, source code document and so on.

3.5.5 Validation and Verification

Scrum manages development-related risks by continuous learning and adaptation. It has no validation and verification processes imposed on the iterations rather than testing after each iteration when it releases an increment. After the iterations, the development team releases a deliverable to the customer by performing unit testing. Releases do not wait for the testing or validation of the increments after merging them with the complete project. Regulatory and safety requirements demand a detailed test plan to validate a product before it is released and verify that product's safety.

According to paragraph 3, article 10, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561, manufacturers must evaluate their development under the regulatory requirements. Consequently, Scrum in the medical device domain may require different additional approaches to satisfy regulatory compliance.

3.5.6 Risk Management

In general, a software development process risk is the functional risks mainly related to requirements, process, or environment, or coding; or related to the failure caused by associated hardware (Chittister & Haimes, 1993). MDSD focuses less on that type of risk, focusing on the clinical risk or patient´s risk of the product. Risk management activities must be planned before starting the development or coding to verify the quality aspects. Risk management activities include identifying potential risks, mitigating unacceptable risks and updating the risk management file continuously.

Scrum reduces operational risks by continuous learning but does not introduce any domain-specific risk management activity. In contrast, safety-critical MDS development requires field-specific risks related to the safety of the product´s user to be mitigated or managed since regulatory compliance enforces risk management measures.

Manufacturers need to maintain a quality management system that covers all processes during the development, including risk management according to paragraph 9, article 10, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561. In section 3 of Annex I, it is clearly stated that manufacturers shall establish, implement, document, and iteratively maintain a risk management system. Risk management activities include a risk management plan, identifying the foreseeable hazards associated with the device, estimation, and evaluation of the risks associated with and occurring during, along with controlling risks following the regulatory requirements. For this reason, the Scrum framework has a contrariety with medical device regulatory software system development.

3.5.7 Self-organized and Self-controlled Team

The Scrum software development method allows teams to be self-organized, and teams make decisions on their own. Oppositely for regulatory compliance, the decisions should be made considering the regulations imposed by the authorities, according to paragraph 4, Article 10 of Regulation (EU) 2020/561. The Scrum team usually does not involve anyone from outside in the development process. Consequently, incompetent or unsatisfactory decisions might contradict the regulatory requirements if the team does not take into account regulations while implementing Scrum. Less involvement of other parties in checking on quality measures in the project makes it less likely to maintain the highest quality.

3.5.8 Number of Roles

The Scrum team has a Product Owner, Scrum Master and Development Team. They employ themselves with enough workload during the development process. For implementing Scrum in MDS development, the developer team needs to communicate the safety requirements for regulatory compliance. If the Scrum Master handles the regulatory conformance, the workload will be way too high. To check on the requirements regarding the safety or regulatory issues regularly, a team of at least one person other than the Scrum Master must focus on this particular area. Hence, Scrum advocates the insufficient number of roles for regulatory compliance.

According to paragraph 1, article 15, Regulation (EU) 2017/745, amended by Regulation (EU) 2020/561, manufacturers must have at least one person responsible for regulatory compliance. Furthermore, this person must possess the required expertise in medical devices with proper evidence (diploma, certificate, degree, or four years of professional experience in regulatory affairs). Hence, Scrum advocates an insufficient number of roles for regulatory compliance.

However, Complications of Scrum in MDS development do not stop organizations from implementing it because Scrum has the scope to mitigate all complications with the necessary extension added to the typical Scrum development lifecycle.

4 Mitigating the Complications

4.1 Agile Mitigating the Complications

Stephenson et al. (2006) have also mentioned four issues in their study on the application of agile software development in safety-critical health systems. They are discussed below:

1. Since agile is a communication-based approach, safety concerns must be communicated adequately to share the understanding while implementing agile in the safety domain to improve communication.
2. The safety engineer and the system engineer may have different reasoning for choosing the design schemes when integrating agile in the safety domain. It requires capturing a design that is the justifiable and reasonable from multiple points of view.
3. Agile development is incremental, and changes have effects. In safety-critical systems, there is no expectation of change. To adopt agility in such systems, it is mandatory to trace the changes and their effects automatically.
4. In general, agility accomplishes one area of functionality and gradually improves to achieve a complete design. On the other hand, safety analysis focuses on a depiction of a complete system. To implement agile in safety models, the practitioners must include assumed details about the complete design in the automatic traceability to fill in the "blanks" in the design. So that further deployment in all those areas can be evaluated to see how well it adheres to the current assumptions (Stephenson et al., 2006).

In practice, manufacturers can develop safety-critical systems by meshing, which is referred to as the "mixing, balancing, and merging of software processes, parts of processes, and process components" (Turk et al., 2005). Boehm and Turner (2005) developed a mesh approach called "Five-Step Method for Balancing Agile and Plan-Driven Methods" and described multiple aspects, all of which are of significance for software development in general. The steps are as follows (Boehm & Turner, 2005):

Step 1. Identify and rate the environmental, agile, plan-driven risks and uncertain risks by prototyping or other data collection methods; visualizing the data using the polar chart.

Step 2. Use a risk-based plan-driven development method or risk-based agile development method; if the risks with agility dominate, the plan-driven risks or plan-driven risks dominate agile risks.

Step 3. Create a project-specific hybrid development method when the risks are mixed by architecting the application to encapsulate the agile parts.

Step 4. Create a project management and development plan which integrates risk mitigation plans for each risk identified in step 1.

Step 5. Continually improve the development capabilities, value-oriented capabilities, communication capabilities, and expectations management capabilities and track progress and apply corrective action whenever the opportunity arises.

Boehm and Turner (Boehm & Turner, 2005) categorized the risks into three categories. Categories of environmental risks include technology uncertainties, diverse stakeholders, the complexity of systems. Agile risks are scalability and criticality, extensive refactoring due to the simple design, loss of tacit knowledge due to personnel turnover, insufficient skills in agile methods. Categories of plan-driven risks hold emergent requirements, fast-paced changes in modern technology, conditions of the market, the desire of rapid output, insufficient people skilled in plan-driven methods. Agile or Plan-driven is applied to the project based on the risk analysis, as mentioned in Step 2. The project is segmented into plan-driven and agile if it is not appropriate for pure agile or pure plan-driven, referred to as hybrid in step 3. And then, other steps are followed.

Boehm and Turner (Boehm & Turner, 2005) analyzed this risk-based method for three sample projects: SupplyChain.com (medium-sized application and intermediate complexity), Event Planning (small application and relatively non-critical), National Information System for Crisis Management -NISCM (extensive application and highly critical). They concluded that future trends are toward application developments that need agility during the development and discipline, where discipline refers to well-organized processes. Therefore, balanced and tailored hybrid methods can combine the benefits of both agility and discipline of plan-driven methods.

Other researchers also suggested that it is possible to develop safety-critical software by implementing agile practices (Abdelaziz et al., 2015; Schooenderwoert &

Shoemaker, 2018). Nonetheless, to fulfil the regulatory requirements, the development processes need to be tailored or modified (Stålhane et al., 2012) since agile processes do not comply with regulatory requirements in their original form.

Rasmussen and Rottier (2018) identified in their research that when designing medical device applications, no agile methods (including scrum) could be strictly followed because they do not cover all of the areas needed to achieve regulatory compliance. Besides this fact, they also encountered that using agile techniques during the creation of regulatory frameworks can be beneficial if they are carefully selected and integrated with a plan-driven lifecycle.

Henceforth, agile methods can be tailored to comply with regulatory requirements of the safety domain, such as a medical device to comply with the regulations. Hybrid methods can be introduced by implementing agile and plan-driven approaches within one project. The upfront planning for the design scheme can be developed to satisfy regulatory requirements. Safety concerns need to be communicated within the team to understand the safety requirements to balance the flexibility to the required changes to the necessary boundaries. Changes during the development must be traced using the proper toolset or by formal documentation to ensure safety. A detailed test plan needs to be introduced. In the development lifecycle, risk-management activities must be integrated for regulatory compliance so that the quality is maintained according to the regulatory perspective. The involvement of authorized notified bodies needs to be allowed to approve the release of the project. Agile methods can be more efficient for the medical device domain if the developers tailor them reasonably.

4.2 Scrum Tailored to Mitigate Complications

Scrum is the most popular agile technique, having been implemented by a large number of companies. Scrum is a software development method that provides a framework for the development processes. It describes how to structure the Sprints. However, it does not describe or define every individual Sprint and does not define the tasks for each of the iterations.

An assessment of how to adapt Scrum is performed by three experts in software development, certification, and agile development, respectively (Stålhane et al., 2012). They have proposed “Safe Scrum” to make Scrum a practically functional approach for developing safety-critical systems. They have considered the following issues: structure the development, plan for validating safety, create, review, select, design to ensure safety, write requirements for module testing, and test and evaluate the outputs from the safety lifecycle.

In Safe Scrum, safety-critical constraints and other design-related requirements are inserted into product backlogs by the development team, as shown below in Figure 7. By engaging an iterative and incremental approach, the development team can learn and adapt continuously, re-plan the project for progression and improvement on the basis of their experiences through the development. The product backlogs can be re-prioritized which makes the process flexible. The RAMS (Reliability, Availability, Maintainability, and Safety) validation process will be applied to each increment after each Sprint to reduce risk. When all the Sprints are complete, a final RAMS validation is performed, which is less extensive which will ultimately minimize the time and cost required for certification. Such a composition of a safety-oriented and agile software development process allows continuous feedback to the customer, the development team, and the independent test team, test-driven development, map functional requirements to safety requirements and enhance traceability (Stålhane et al., 2012). Summarizing these benefits, Stålhane et al. mentioned that this combination aids in making the production process more transparent and thereby allowing for better control.

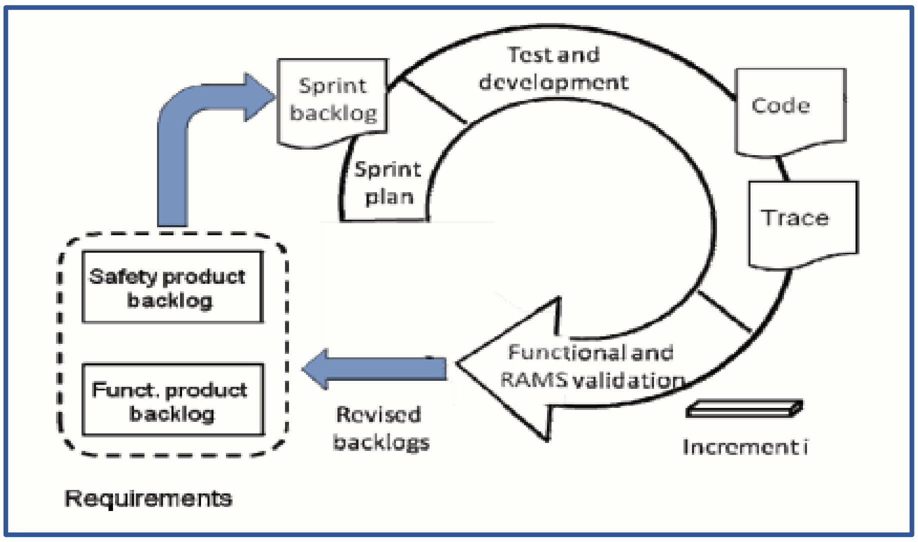

Figure 7. Safe Scrum (Stålhane et al., 2012).

Hanssen et al. (2016) worked with Autronica Fire & Security from 2014 to 2016 to perform a case study on the Safe Scrum process. They concluded that the project needed some clarification on “quality assurance” and the Scrum Master had a significant workload to maintain compliance with regulatory requirements. Consequently, they proposed a specialized Quality Audit position in the line organization for additional functioning. They decided to add this QA function to the Scrum team to conduct quality assurance keeping close connection to the development team.

Some other examples of Scrum implementation in the safety-critical domain are discussed briefly in the following paragraphs in this Section.

Wolff (2012a) presented an approach for implementing Scrum combined with a formal specification language in a fighter aircraft project. The project employed formal executable specifications to verify functionality, best explain the system's requirements, and more specifically implement the product (Abrahamsson et al., 2002). Along with implementing Scrum in the software implementation, formal specification models were implemented simultaneously. Tasks within a Sprint were incorporated with formal specification investigation tasks predefined by formalists who work within the team of the software engineers. Hence, it adjusted agile Scrum processes with formal methods commonly used in industry to model and validate high-risk system properties (Wolff, 2012a).

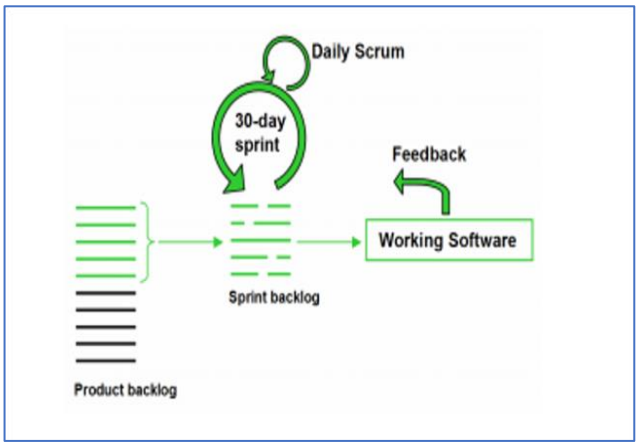

Figure 8a. Scrum Overview and (Wolff, 2012a).

Figures 8a (above) and 8b (below) show how formal specification methods are integrated into Scrum Sprints, where Figure 8a represents the overview of Scrum with a 30-day sprint and 8b represents the integration of the formal method into the Scrum process.

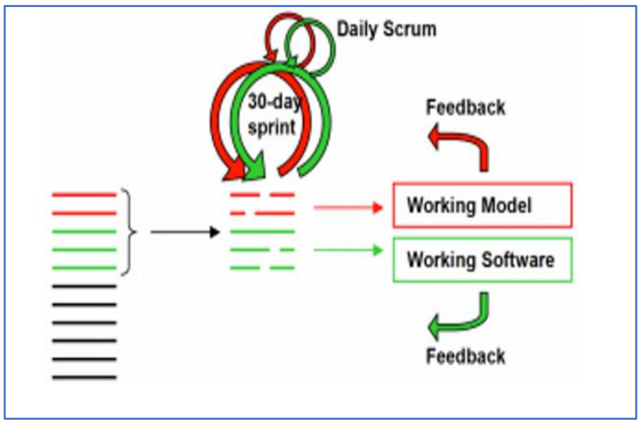

Figure 8b. Integration of formal method into Scrum (Wolff, 2012a).

Wolff (2012b) presented an industrial case study where development engineers developed an executable model using a formal specification language VDM (Vienna Development Method) to implement on a fighter aircraft project. Wolff also mentioned that using a formal model and lightweight formal method principles such as the scenario-based tests and manual inspection of the generated proof obligations proved to be very valuable (Wolff, 2012b). The project outcome was quite positive, and the customer was satisfied too.

Regulated Scrum (Fitzgerald, 2013) is an example of an adapted approach that has been implemented and validated in a highly regulated organization. An example of implementing Scrum with Test Driven Development, Continuous Integration, and Pair Programming in a regulated environment is a European space industry company named QUMAS (Ahmad et al., 2010).

QUMAS had employed the Waterfall model since the company was founded in 1994. This methodology resulted in a long time-to-market and a significant release overhead, which were considered drawbacks in the quickly changing market in which QUMAS operates. Eventually, they have adopted the Scrum methodology over approximately two years, as mentioned in an industry case study (Stålhane et al., 2012) of implementing agile in a regulated environment. This company solved the core issues that conflict with Scrum, such as quality assurance, safety and security, effectiveness, traceability, and verification and validation by adding enhancements to the generic Scrum methodology to meet the compliance requirements of a regulated environment. The regulated Scrum by QUMAS is presented as R-Scrum in the case study.

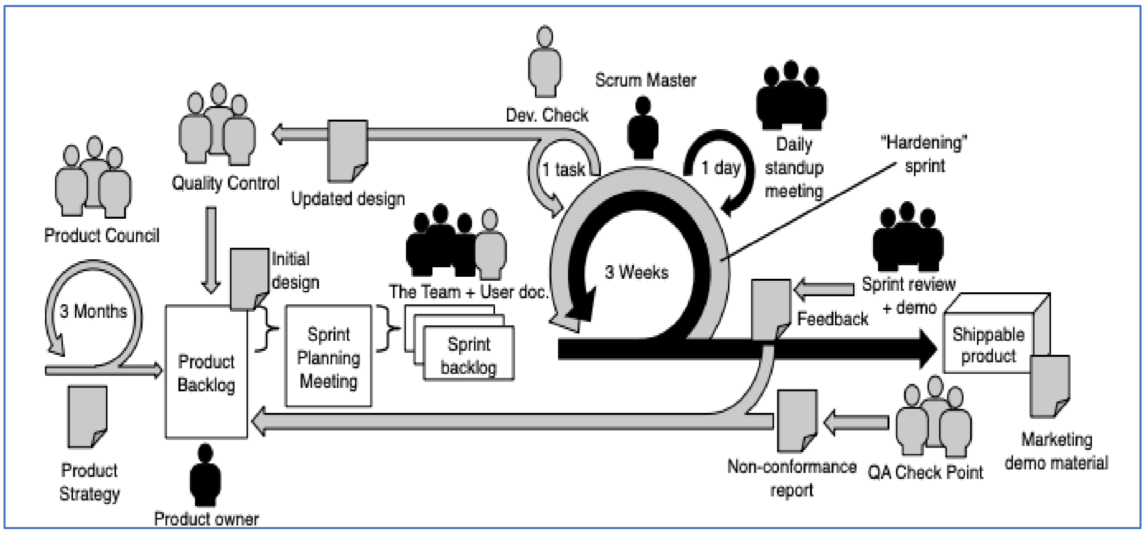

Figure 9. R-Scrum (Regulated Scrum) in QUMAS (Ahmad et al., 2010).

As shown in Figure 9, QUMAS implements QA (quality assurance) audits at the end of each Sprint, allowing improved visibility, traceability, and measurement. Traceability facilitates "Continuous compliance" with the regulatory environments. They undertake QA audits to verify that the output from each Sprint adheres to the required procedures and standards. Eventually, risk mitigation is facilitated significantly by the transparency of ascertaining project status at a glance by the outputs of the audits, which allows solving safety issues. QUMAS also operates a four-stage prioritization scheme for tasks and bugs to prioritize better risk factors, ranging from P1 to P4. Regular customer audit and verification by the auditor of functionality implemented in the product via the agile processes enhanced the effectiveness of their development. Full end-to-end traceability is established by using toolset such as JIRA, Confluence, and others, which can trace all initial requirements, tasks and sub-tasks, design documentation, source code, builds unit tests, and bugfixes. At the start of and production of Sprint, requirements are explicitly checked with the Product Owner. Unit tests are done within JIRA, and functional tests are the responsibility of the test team using a specific quality center testing suite. Any failures are recorded, and emails are sent to the developers and Scrum Master. QA does not sanction a release with any open issues, 'definition of done' must also include regulatory compliance. QUMAS ensures the verification and validation of their product. Therefore, agile methodology such as Scrum is highly suitable when tailored to meet the needs of regulated environments and supported with appropriate tools represented in this case study (Fitzgerald, 2013).

Kircher and Hofman (2012) documented the experiences of Siemens Healthcare, a leading provider of biomedical technology worldwide, in resolving challenges when transitioning a large-scale dispersed platform development organization to Scrum. According to them, Siemens Healthcare integrates both approaches: PLE (Product Line Engineering) and Scrum to leverage the long-term benefits; PLE for strategic reuse and agile practices for achieving steady progress. Scrum is merged into their software development process and, in addition to that, implements "feature orientation" practice to resolve the challenge of managing the flow of requirements coming from several product lines. "Feature Orientation" refers to the sum of understanding about the structuring and development of features described clearly in the report (Kircher & Hofman, 2012). This process is a pure example of implementing Scrum in the medical device domain.

Furthermore, Abrahamsson et al. (2002) compared different agile software development methods. Based on the comparison, he stated that Scrum processes could cover project management, requirements specification, integration test, and system test phases as required (McCaffery, 2017). However, it's complicated because there are inconsistencies between agile and plan-driven processes (Heegar & Neilson, 2020).

Despite the barriers or complexities, organizations developing safety-critical software increasingly seek to create better practices by combining agile and plan-driven development processes (Heegar & Neilson, 2020). However, in the above research papers, it is commonly mentioned that Scrum has not been used in the safety-critical domain with its original versions. Instead, it is tailored for this domain and combined with supplementary practices to ensure safety and regulatory compliance. Therefore, Scrum cannot be implemented directly to the safety-critical MDS development without any modification or tailoring.

4.3 Mitigate complications of Implementing Scrum in MDSD

After studying different methods of integrating Scrum in regulatory software development for the safety domain, it is comprehensible that there are some issues manufacturers or developers must consider during the development process of MDSD. The concerns listed below for these issues are supposed to successfully mitigate all complications, including risk factors, while implementing Scrum in the Medical device domain.

The required actions to mitigate the complications of adopting Scrum in MDSD are listed as follows:

1. In consideration of the safety life cycle, safety requirements should be inserted into product backlogs along with other user requirements or design requirements.
2. The project should be designed to handle the flow of requirements from both the customer and regulatory bodies.
3. DoD must consider regulatory compliance.
4. Integrate risk-management activities into Scrum process to mitigate risks.
5. Traceability must be enhanced for both changes and effects of changes to maintain transparency and accountability to the regulatory authorities.
6. Each increment must go through a clinical verification process to minimize end-user risk.
7. Develop a complete test plan to apply on each of the features of the product to meet the safety domain's requirements.
8. The Scrum team should possess both flexibility and accountability to the regulatory bodies.
9. Assign an individual in charge of the team (other than the Scrum Master) to handle the conformity and quality with respect to the regulations. Additionally, assign a team outside the development team to ensure regulatory compliance. Involve notified bodies in the development process.
10. Continuous compliance must be considered to a higher degree.

5 Interview

5.1 Interview Design

The thesis has illustrated the perceived complications of implementing Scrum in MDSD reported in Scrum and Safety-Critical Regulatory Software studies. In order to gain insights into the complications and practices of implementing Scrum in MDSD, we further interviewed practitioners who have been working with MDSD for more than five years, being firmly involved in regulatory compliance for the medical device domain. This interview process intends to answer the second research question:

RQ2: Can Scrum mitigate these complications to develop medical device software?

This interview is a semi-structured interview with a combination of directive and non-directive questions (Hove & Anda, 2005). The interview form is open-ended to encourage the participants to express their opinions and feelings freely and extensively. Four experts are interviewed who are strongly involved in developing software for the medical device domain.

Upon the suggestions of the interviewees, the interviews have been arranged in Microsoft Teams or Zoom video conferencing online. The question set was shared with the interviewee before the interview to familiarize them with the research topic and interview goals. The interviews have initially started with a discussion about the background of the interviewees, their work experience, and current positions. Later the questions and answering proceeded to an explanatory and spontaneous conversation. Three of the interviews have been recorded; one was conducted by taking notes during the interview and asking necessary tails later via email.

The interview results provide broad coverage of issues implementing Scrum in the medical device domain and help connect the outcomes of the studies to current MDS development practices.

5.2 Interview Participants

Candidates of the interview are IT personnel working in different IT companies at Tampere, Finland, offering strategic consulting, service design, software development, analytics, cloud services, application and integration services, reliable technological solutions for health care use and other operational services in Finland, Sweden, Denmark, Estonia, Belgium, and Germany. These companies facilitate modern approaches for their product and service development processes. The participant interviewees have been playing different roles in these companies.

Roles of the participants:

- Product Owner
- Scrum Master
- Developer
- Regulation Affairs Specialist

Table 2 lists the job title and the company, the year of experience on working in the MDSD field. The length of the interviews varies, ranging from 45 minutes to one hour.

Table 2: Overview of the Interviewees.

Job Title	Company Name	Year of experience with SD/MDSD	Duration of the Interview	Interview Date
Regulatory Affairs Specialist	A	12 / 5	60 minutes	22.04.2021
Product Owner/ Scrum Master	B	9 / 6	50 minutes	03.05.2021
Junior Fullstack Software Developer	C	11/ 5	45 minutes	04.05.2021
Software Specialist	D	9 / 7	60 minutes	05.05.2021

The first interviewee is a regulation specialist currently working in a renowned company at Tampere. He has been involved in regulatory MDSD for the last five years. He is highly acquainted with the agile process implementation, conformity assessment, QA audits, technical file assessment, medical product development, international standards, regulatory perspectives, and other development practices in the medical device industry. Currently, he is a Regulation Affairs Specialist in a medical device manufacturing company and a PhD researcher at Tampere University. His extensive knowledge of different software development methodologies, compliance, regulatory frameworks, and standardization contribute to cover a broad range of practical information in the interview session.

The second interviewee has also been working with MDS since 2013, closely experiencing regulatory compliance and implementation of Scrum in this domain.

Currently, he is a Software Specialist, working for the leading Finnish supplier of health care laboratory and diagnostics information systems. He is also researching "Continuous Development of Medical Device Software under European Union Regulatory Framework."

Another interview participant is working on a thesis project, "AHMED (Agile and Holistic Medical Software Development)," related to the pain points of regulated development. He is also serving as a full-stack software developer at the University of Helsinki. The other interviewee has been working in multiple companies for more than five years as Scrum Master, Product Owner, Software Developer proving experience with agile coaching, servant-leadership, Scrum and MDSD.

The author has recruited them for the interview because of their profound knowledge of agile, Scrum and multidisciplinary experiences in implementing agile methods, standardization, and compliance in MDSD in the industry.

5.3 Interview Questions

Findings from the literature review extract the essence that Scrum can be implemented in MDS with necessary alteration adapted within the basic framework. Since some medical device manufacturers antecedently apply Scrum for developing such software systems, this thesis endeavours to formulate feasible solutions, based on the current practices, to mitigate complications considered to be encountered while implementing Scrum.

The questions of the interview are composed prospectively to obtain in-depth information possible to perceive the argument on how organizations negotiate with the trade-offs while applying Scrum during their development process. All the interview questions are the gateway to the discussions about the eight issues confined from the literature review. Anticipating each complication to require some modifications, the author set the interview questions to pinpoint the current industry's related practices to find the answer for the last research question. Thereupon, the questions are prepared accordingly to focus on the subject matter to view how organizations mitigate the complications of implementing Scrum in MDSD alongside maintaining the regulations.

All the interview questions are listed in Appendix 1. Some of the most relevant interview questions are listed below.

General Questions

1. What are specified as regulatory requirements? When are they specified for a project? When are they analyzed? How are they adjusted into the project requirements?

Related Questions to RQ2

1. To consider a project (or piece of project) to be Done/Complete, what sort of validation or verification does it require to comply with regulations? Who is responsible for specifying the product definition?
2. Have you ever faced problems to handle continuous changes in requirements for the development of a regulatory medical device software? If yes, how did you handle it?
3. How do you introduce risk management in the development lifecycle for complying with regulations?
4. What kind of documents are required in medical software development in compliance with regulations?
5. Do you think it is necessary to introduce roles such as Regulation Specialist/Regulation Officer within or outside the development team to comply with the regulatory requirements?
6. How do you think "Unpredictability/Uncertainty" of SCRUM conflicts with the rules of regulatory requirements?

In the last question, the author emphasized that Scrum leverages uncertainty and flexibility, whereas regulatory systems such as MDS enforce predictability for complying with regulation. This question is utilized to outstretch an all-embracing, pervasive conversation to collect a wide-ranging view from the interviewees to figure out the conflicts regarding the unpredictability of Scrum, the practices imposed to mitigate the conflicts. All the other questions are orientated in harmony to derive the other practices to mitigate all the eight complications listed in Section 3.5.

5.4 Interview Results

The interview sessions have covered practical implementation techniques and issues such as tailoring agile methods such as Scrum, flexibility in requirement changes in the medical device software domain, documentation and traceability issues concerning the regulatory requirements, verification and validation processes required for regulatory compliance, mandatory risk management activities to ensure safety, issues related to self-controlling of the Scrum development team, the number of roles to handle the regulatory conformity during the development of MDS projects. During the interview, the interviewees were cooperative enough to discuss all the practicalities relevant to this research.

5.4.1 Tailored Scrum Complying with Regulations

The interview participants commonly described that all agile approaches currently used in MDSD are hybrid methods since they require a plan-driven part and agility and incremental aspects during the development. Some companies still follow plan-driven sequential models as the development base, but they also integrate agile principles into their development process. Companies implementing agile methods also include plan-driven quality management processes in their development procedure, which must comply with the regulation. In the case of implementing Scrum, other processes require to be used on top of the regular framework of Scrum. One of the interviewees mentioned, "Processes outside the core software development are always plan-driven. On top of Scrum, we need to use them, also."

Interviewees reported that commonly manufacturers have a compliance department, and it is expected that a person is responsible for an R&D project's compliance activities. In handling regulatory compliance, there are variations in different companies. Some companies may also handle it by employing a separate compliance team of several members or hiring outsourced specialists. In addition, International and Harmonized standards require a Quality Management System (QMS) to confirm the effectiveness of a product. It ascertains the consistency in design, development, and delivery of MDS that is safe for the end-users. Henceforth, the development process is designed in association with the QMS.

5.4.2 Product Definition and quality Assurance

For developing MDS, the most crucial concern is compliance with regulations. The regulation requires a QMS to be followed, as mentioned above. After verifying the product by the internal regulatory team, an external body should also verify the product to be released. A development team cannot release a piece of product whenever that is ready.

“We cannot release any product or a piece of the product when we want. We need to go through external bodies for assessment. It could take even several months, but we need to wait,” a participant expressed the idea of QA audits in these words. Once a product is ready and the documentation must be adequately prepared, the notified body responsible for the regulatory compliance performs an external quality assessment to check the reliability and safety issues. There are two companies to perform in Finland as the notified body: SGS Finland (Finnish) and Eurofins (French). Once the assessment is done, the notified body approves the release of the product.

5.4.3 Flexibility in Requirement Changes

In MDSD, all the requirements are documented, verified by the regulatory bodies. Requirements in MDS development include two types of requirements: product requirements (including product definition, DoD) and regulatory requirements from the harmonized standards. While implementing agile Scrum as a development methodology, there may be some changes in the requirements during developing a product that satisfies certain conditions.

Another interviewee responded this way regarding the flexibility in requirement changes, “In medical device development, we need to comply with certain process requirements of the standards, in addition to product requirements. Within these boundaries, teams can make decisions - as long as regulatory requirements are met.” It means a requirement cannot be added or deleted, or altered if it contradicts the regulatory standards, if the requirement change seems unsafe or unreliable for the user, or if the requirement change hinders the clinical efficiency of the software or the associated medical device. A verification process is conducted by the regulatory teams from the compliance department that manufacturers commonly have to check these issues.

One participant discussed an example of this issue during the interview. While developing an MDS, once a development team improved a better version of the algorithm used in the development after starting the development process. However, since the notified body did not approve it, they could not implement it even though the better efficiency of the algorithm was already tested and verified by the team.

Therefore, with some limitations, it is possible to change requirements in the development process. However, since Scrum iterations and Sprints possess functional improvements during the development, there should not be any problem learning and adapting continually, keeping a reasonable boundary.

5.4.4 Documentation and Traceability

One of them stated this for explaining the traceability issue in the medical device domain, “When we think of traceability in medical device software development, we think of the requirement management and traceability between artifacts.” For ensuring traceability, all the phases are documented in the medical device domain, starting from requirement elicitation to testing. Technical documentation (such as Usability Specification Document or User guide) can be done using tools such as Jira, Confluence or other technical management tools; or manually by utilizing verified and licensed documentation templates for different development phases (Michaud, 2012). Regarding the quality assurance issue, the interviewees described that there should be enough tool support for documentation and requirement management to ensure the highest quality of a product.

Another of the interviewees shared additional information regarding the documentation required. “We still have some documentation in Word and Excel -format. Requirements are written as Gherkin scenarios and stored in Github. All documents are stored as PDF as well. We plan to get rid of Word and Excel and start using Confluence and Jira instead, as we can pull data to Github directly from those, without need to do PDF conversions.” It means there is various documentation required in a different format. Requirements are documented as Gherkin scenarios and stored in Github for easy access online. Gherkin scenarios are formalized syntax to exemplify the behaviour of a software system under development. These scenarios may be sub-parts of the requirement specification, or the test suite used to gather information among stakeholders, testers, and developers (Gutiérrez et al., 2017). Other required documentation is done by using MS Word, MS Excel as per the need. All documents are to be accessed easily by the team or delivered to other parties if needed. Hence, documents are also stored in PDF format for future reference. Jira and Confluence are two agile project management tools: both support documentation by allowing direct file access from Github. Using such tools save time and help to maintain organized documentation. Most of the companies now use Jira and Confluence to maintain the required documentation, confirmed two interviewees.

5.4.5 Risk Management

According to the interviewees, while implementing Scrum, risk management starts with risk identifying during the Sprint planning. Across the execution of the Sprints, the risk management file is updated with necessary risk mitigation.

“When I was working on a medical project, we identified risks and documented them (in association with one backlog) in Sprint planning. The compliance team verified them. After that, we mitigated the risks; simultaneously, we updated the risk file. The compliance team always reviews updated risk files”, one interviewee addressed the risk management activities in the project he was involved in.

Risks are verified and appropriately documented during the Sprint. The operational risks mentioned in the first paragraph are verified by testing and peer review among the developers and manual checks. However, the end-user risk is reviewed and checked by the clinical specialist or by the compliance team. The compliance team also reviews the updated risk management file.

5.4.6 Unpredictability and Variability

Through the interview session, the issue of unpredictability and variability is also discussed. As discussed earlier, the medical device domain requires enough predictability regarding safety issues. The interviewees spontaneously responded that Scrum is not contradictory to this issue. One participant detailed, "We need to have a high-level understanding of the product: clinical benefits, clinical performance, and efficiency. Those need to be scientifically proven, so there might be, and usually is, clinical trials with real persons. So implicitly, the environment where software development is happening is not chaotic, where Scrum works best. Inside the software development, we can be more agile as long as we implement product level requirements.

Predictability is required for the clinical performance and efficiency perspective, even if it is not required for the development perspective. Predictability or uncertainty may be possessed depending on the situation and the product's requirements during the development process.

5.4.7 Verification and Validation

Verification and validation are a must for regulatory compliance. Different verification methods are employed in different stages of development. Before the development begins, the requirements are also verified. There is a different person responsible for the different verification processes.

One of the interview participants exclaimed this for answering questions related to the verification and validation process in the MDS domain, “The team is following pre-defined workflow, which ensures that different verification activities are carried by the right persons at the right time. When we think about high-level product requirements, the Product Owner would be the right person for verifying product requirements. Then again, if we think about software requirements, it could be a lead architect or the person who is responsible for the tech leading.”

5.4.8 Self-organized and Self-controlled Team

In MDSD, teams can be self-organized, but they are not always self-controlled. Team members cannot make their own decisions without approval from the compliance department. Indeed, they can make some decisions that are not contradictory to the compliance or do not disaffirm the standards, but not in every aspect. In some cases, the team needs an audit trail for approving decisions.

While discussing the team's decisions, one of the interviewees who has been working with MDS development for the last eight years set an example.

"We cannot make our own decisions regarding a project without approval from the regulatory department. One example is that we wanted to start a pilot project with an existing product and a new client located in Spain. Local regulations require that in Spain, the medical device must be available in the Spanish language. So, in this specific case, we could not decide to enter Spain with our current product version."

5.4.9 Number of Roles

All the interviewees mentioned that there must be a team or a person who must work on the compliance issues. Most companies have a regulatory team; some may have one person involved directly in the development who can handle the regulatory compliance. It is possible that the compliance checking is done by an entirely different team outside the development team.

Generally, companies have roles such as compliance officers and regulation specialists. The number of staff in that role may vary depending on the size of the company. Large-scale companies may have extra members such as clinical specialists to handle the risk-control activity for satisfying the regulatory standards.

Though the interview is conducted particularly in a particular region, it portrays the perspectives and standpoints of the regulatory medical device development worldwide regarding the EU regulatory framework.

6 Discussion

Scrum software development methodology advocates an adaptive approach, whereas MDSD administers a regulated procedure to comply with the regulatory standards and guidance. Scrum adaptation in MDS development arises complications in this circumstance. However, Scrum does not specify each step of the development, leverages flexibility, provides freedom to the team. These confinements leave significant scope for the Scrum team to accommodate the regulatory requirements within the development procedure, which allows the medical device manufacturers to modernize their development approaches and maintain compliance in regulatory environments.

The literature review suggests that Scrum must impose some tailoring to meet regulatory requirements. Tailoring is an inevitable necessity to employ specific improvers to balance the essence of Scrum and the regularity. The activities described in IEC 62304 standard requires a linear sequence. However, this thesis represents that development teams can practice agility during the development process while developing MDS by adopting practices to balance the agility of Scrum to incorporate the regulatory standardization.

The components of a Scrum framework are Scrum events and Scrum roles, and Scrum artifacts. Each one necessitates some additional effort. The predicaments in adapting Scrum in MDS development include product definition, flexibility in requirements, unpredictability, documentation, verification, risk management, self-controlled team, and the number of roles in the Scrum team. Resolving these issues, manufacturers can administer an efficient development for their product using Scrum proven that Scrum fits the regulatory framework for the medical device domain.

Only a few manufacturers have already made the transformation from the plan-driven traditional approach to Scrum. Different companies may encounter different complications because of their internal organizational structure and work principles. Since Scrum has some complications considered as trade-offs, it is possible to adopt a Scrum model for the organizational needs and for fulfilling all the demands of the regulatory standards. This provides the development team with the freedom to choose modernized approaches to ensure the highest productivity and efficiency. This thesis intends to summarize all the measures to prevent Scrum from becoming complicated for the manufacturers.

6.1 Scrum Events incorporating Compliance

For implementing Scrum in MDS, the MDSD can be divided into three stages, and they are Requirement Elicitation, Implementation and Release, where all the three stages must incorporate risk management activities and documentation.

6.1.1 Requirement Elicitation

Usually, Scrum starts with requirement gathering from the Product Owner. The Product Owner defines the requirements and the definition of done. Then the development team inserts the requirements into the product backlog. For complying with regulations, the product definition, including all requirements, must be verified by the regulatory bodies. Requirements must append the regulatory requirements into the product backlog. The processes required for the Quality Management System (QMS) begin in the same phase. In the requirement elicitation phase, the Product Owner defines the customer's need and the project's goal, which entail a plan-driven approach to follow. At the same time, the Quality Manager or the Regulation Specialist specifies how to achieve the goals and the boundaries from a regulatory perspective. The team must document each requirement. Hence, the documentation and verification must start from the requirement elicitation phase while developing MDS to satisfy standards such as IEC 62304, even though Scrum does not prioritize activities such as documentation and verification.

Scrum holds flexibility in requirement change. In any phase, the Scrum team can make changes in requirements listed as the product backlog, as required. Regulatory compliance demands quality management for ensuring safety, also restricts changes if the change affects the safety concerns. While developing MDS implementing Scrum methodology, the requirement change requires an inspection from the regulatory bodies. If the proposed change or additional requirement disaffirm the clinical safety of the end-user of the software or tend to be unreliable, then the change must not be accepted. Otherwise, it is possible to accept if the change does not affect the safety issues. The way the IEC 62304 requirements are converted into practical procedures varies from company to company, depending on the scope of the software and the safety risks it poses. The methods for managing the requirements in a low-risk device would vary from those used to handle software in a complex, safety-critical system.

6.1.2 Implementation

After the requirement analysis, the Scrum team plans time-boxed Sprints to develop product features one by one as prioritized in the product backlog. A Sprint starts with a Sprint planning meeting within the team. The development team defines the Sprint backlog in this meeting. Finally, items in the Sprint backlog are verified by peer review.

Risk management is incorporated with the whole implementation process of the software. Accordingly, the Scrum team must perform risk identification during the Sprint planning. All the identified risks must be documented in the risk management file. There may be a case that there are no such risks identified for the product's use since it is neither directly applied for sensitive medical machinery nor relate to a health hazard of the users. Regardless of this, the team must carry on the formal risk management procedure because regulatory compliance requires a risk management file.

For maintaining the traceability required to comply with the regulations, the documentation in each step is a must. It can be done for easier access of the documents by using tools such as Jira, Confluence, and other management tools as standards such as IEC 62304 do not dictate any particular type of documentation. Using such a tool also provides a better management experience for the team.

Afterwards, Sprint execution starts. Each Sprint follows the same workflow during the development. Functional risks are mitigated by testing and verification through manual checks and peer-reviews among the developers. The development team must mitigate the patient's risk during the Sprints. Each of the identified end-user risks is reviewed and checked by the Clinical Specialist or by the compliance team to ensure safe use of the product. At the end of each Sprint, the team tests the increment developed in that Sprint. The software development team have to set up configuration management and bug tracking systems at the beginning of the project to manage risks according to the standard. If there is any bug detected in the operation of the software, the team must solve it. Then they must check risks for that particular piece of product to identify any possibility of a hazardous situation. The risk management file is updated when risk is mitigated. However, the updated risk management file is required to be reviewed by the compliance department.

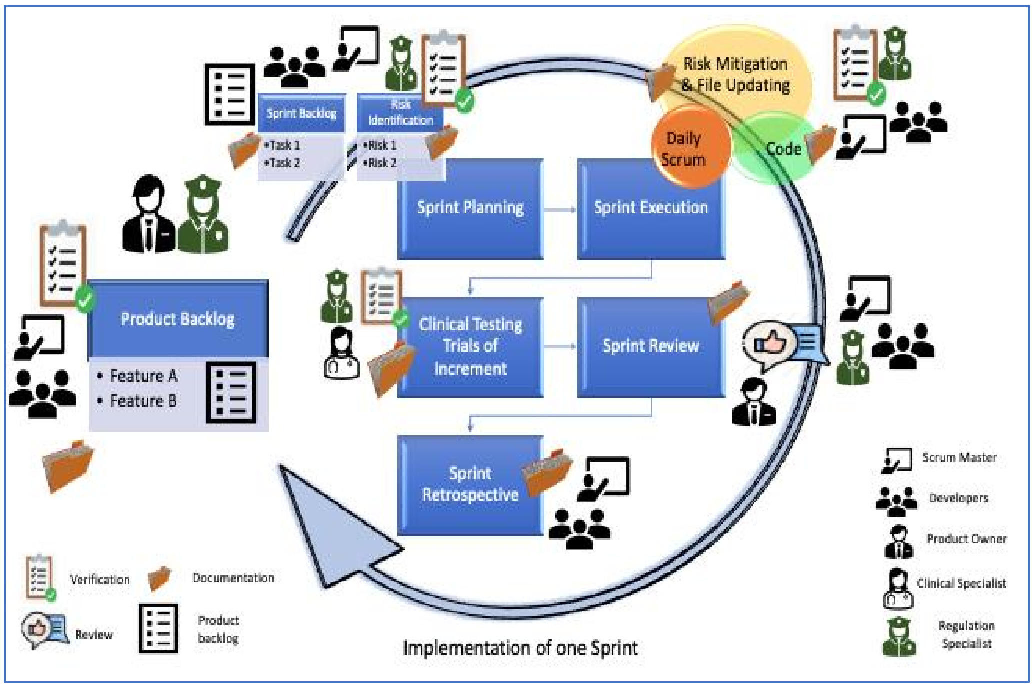

Figure 11. Sprint during Scrum implementation in MDSD.

Figure 11 is summarized from the discussion which represents how a Sprint in a Scrum process can be executed in order to mitigate the possible complications during MDS development in compliance with regulatory requirements. With the regular Scrum events mentioned in Section 2.2 (Figure 4), such as Sprint Planning and Sprint Execution, the regulatory confirmation requires some additional activities to be performed, as such clinical trials for increment produced during a Sprint and risk management. Accordingly, the figure includes "Clinical Trial" as a part of the sprint. It also indicates the verification, documentation and involvement of additional roles (such as regulation specialist, clinical specialist) during the Sprint required to comply with the regulations.

As we discussed in Sub-section 3.5.3, Scrum leverages variability. Subsequently, the development in Scrum is considered unpredictable. Since medical device regulatory software development requires enough predictability regarding the end-user's safety, the Scrum team needs to balance the variability during the development. From the development perspective, unpredictability or inconsistency may be present in the production phase depending on the product's requirements. Predictability must be employed from the regulatory perspective to prove the clinical performance and efficiency of the product by following the guidance from the Regulation Specialist, which contains a planned process to maintain compliance. Henceforth, the development can be more agile by leveraging variability if there is no conflict with regulations.

Similarly, the Scrum team must shrink the self-control by involving regulatory bodies during MDS manufacturing. The development process must be highly controlled from the beginning as the complexity of the software process is more than for hardware; the problems in software are not easily detectable later in the development process. The teams can decide the development tasks and other issues until the decision disregards the regulatory requirements. If a team's decision contradicts the development and the regulatory compliance, they cannot take it into action. Otherwise, within a specific reasonable boundary, the team can make their decisions. All the decisions need to be communicated within the development team, including Product Owner, Scrum Master and verified by the Compliance Officer or Regulation Specialists. These roles enact further verification within or outside the development team if necessary. In addition, the Clinical Specialist is involved in performing the clinical trials and verifying the safety of the user. The responsibilities of these roles are further discussed in Section 6.2.

6.1.3 Release

Scrum allows short releases of the product during the Sprint iterations. After each Sprint, the Scrum team delivers a deliverable increment to the Product Owner and the customer in some cases. Regulatory MDS development does not allow short releases without system testing and verifying the entire software system. On this account, the Scrum team has to hold the releases till the final product is ready. The product increment can be tested by employing a clinical specialist who can perform medical or clinical trials. The increment is accepted once it is verified clinically on the condition that the increment will be integrated into the final releasable product.

Figure 12 depicts the whole process of Scrum implementation for developing MDS in compliance with the regulations and harmonized standards from requirement elicitation to release.

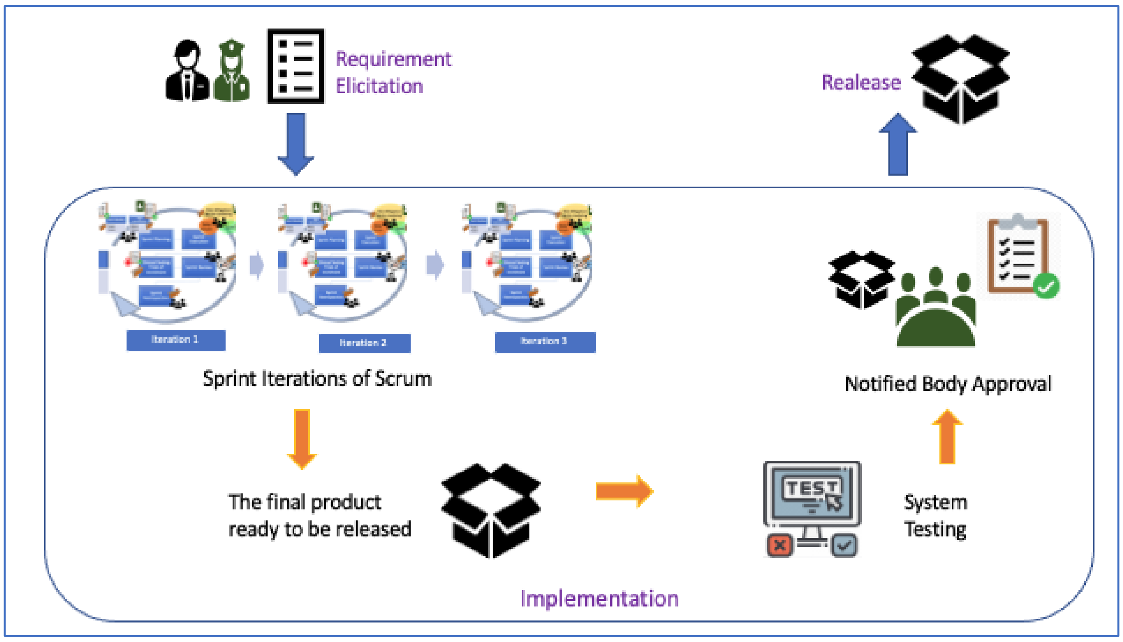

Figure 12. Scrum in compliance with regulation in MDSD.

When the final product is ready with all the proposed features from the product backlog and the regulatory requirements, the features are integrated into the final product. The product must be tested, engaging in system testing, and submitted to the notified body responsible for the final verification of the product. This external body performs not only the functional verification but also the conformity assessment. Once the external conformity assessment is done, the notified body approves the product's release to the stakeholders to place in the market.

In Section 4.2, “Safe Scrum” is discussed, which applies to tailor Scrum for implementation in any critical systems regardless of the field. Then a case study of “Regulated Scrum” is discussed, which applies Scrum in the regulated domain as such space industry. These tailored Scrum implementations proposed a Quality Audit position to check on the quality assurance and continuous compliance according to the regulatory requirements. On top of that, Safe Scrum employs RAMS validation. Regulated Scrum facilitates significant risk mitigation by prioritizing risk factors and maintaining traceability. Above-described tailored Scrum is highly focused on the medical device domain, which applies every measure to mitigate the complication of Scrum when developing MDS. It proposes quality and compliance checking by a Regulation Specialist instead of general quality audit roles. Similar to Regulated Scrum, traceability and risk-mitigation is highly considered in this process, focusing on regulatory standards such as “IEC 62304” to confirm compliance to the highest degree.

6.2 Scrum Roles and Artifacts incorporating Compliance

As discussed in Section 6.1, the tailored Scrum method employs verification and documentation in each step of the development. The team shares artefacts continuously. All decisions and development progress are documented and communicated, incorporating the standardization. The development phases are appropriately traced to comply with regulations. The Scrum team in this process require regular Scrum Roles such as Product Owner, Scrum Master, Development Team and customer or other stakeholders who play similar roles to a Scrum approach as described in Section 2.2. Developing a regulated and critical MDS system, the software development team must involve regulatory bodies in the development as mentioned in Sub-section 3.5.8 and introduce some new roles to the team such as a Regulation Specialist or Compliance Officer and a Clinical Specialist to enforce the compliance. The Regulation Specialist is responsible for guiding the development team on how to proceed with the development through setting up boundaries as a part of the QMS. Through the entire development process, he must check and verify the compliance being close to the development team. This position can also be titled as a Compliance Officer since it is the key role to manage compliance conformity. The clinical Specialist is responsible for the clinical trials of the software increments, which is entailed by the risk management process during the development of the MDS.

According to the latest EU legislations (MDR & IVDR), cybersecurity checking is also a part of regulatory compliance (Granlund, 2017) to ensure data and system security since the medical devices are now linked to IT networks and may be vulnerable to cybersecurity threats (such as viruses) as well as unanticipated problems caused by deteriorated network performance, software glitches, or the installation of operating system patches or malware protection updates among other things (Hrgarek, 2012). Henceforth, organizations may also need a cyber-security analyst to ensure data protection and the safe online use of the software. All or some of the roles can be responsible for continually checking the regulatory compliance throughout the development procedure.

The notified body is involved in the development though they are not a part of the team. Notified body approves the release of the software. The development team performs the integration test for the whole software system; then, they send it to the notified body for approval. Finally, the software is ready to be released after the efficiency of the software is validated by this external body.

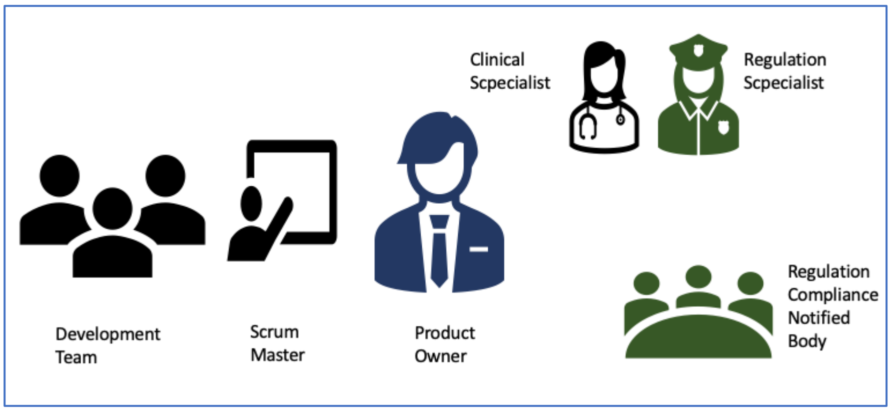

Figure 13. Scrum roles involving regulatory bodies.

Figure 13 shows the Scrum roles mentioned above required to be involved in the software development lifecycle for regulatory compliance in the medical device domain. Depending on the company's organizational scale, it can differ from company to company. A separate compliance team must be formed if the manufacturer works on a large scale in the MDSD industry.

6.3 Solutions Mapped to Complications

In the circumstances of regulatory MDS development, companies manufacturing such software engage Scrum in the development lifecycle because of some significant advantages. Scrum allows developers to maintain agility even though the standards and regulation impose control on them. In Scrum, the Project Managers or Quality Managers can track the workflow very easily. Scrum introduces the latest tool support for developing and managing the development processes, making it more convenient for the developers and manufacturers of MDS.

The tailored Scrum procedure prescribed above summarizes the argument that Scrum can successfully resolve the entanglements that originated from implementation in MDSD in conformance of regulatory proclamations. Furthermore, the complications of Scrum implementation in MDSD can be mitigated by adopting the discussed measures in the development process. Solutions to each complication are mapped in this section to provide a detailed view of how Scrum can be applied in MDSD, mitigating all conflicts with regulatory requirements.

Table 3: Complications mapped with corresponding solutions.

Complications	Solutions to mitigate the complications	Practices for complication-mitigation
Product Definition and Quality Assurance	Definition of done for the product features must be defined by the regulatory compliance team (Sub-section 3.5.1). All product backlogs and Sprint backlogs during the Scrum process must be verified. Before releasing, the product must be verified by the regulatory bodies (Sub-section 3.5.5 & 5.4.2).	Regulation Specialist handles the QMS by defining the regulatory boundaries and Product Owner defines the product backlogs. Development team verifies the Sprint backlogs. The external notified body verifies the product before releasing the software.
Flexibility in Requirement Changes	Feature of the software can be structured, and the project should be developed accordingly to manage the flow of requirements from both the customer and the regulatory compliance team. No change can be made without approval (Sub-section 3.5.2 & 5.4.3).	The software development team adopt functional changes if there is no contradiction with regulatory standards. The Regulation Specialist or Compliance Officer can negotiate the changes within the team
Validation and Verification	Validation process must be applied for each increment to reduce risk. Detailed test plan of the final product should be designed, clinical trials must be applied on each of the features to satisfy the requirements of the safety domain (Sub-section 3.5.5 & 5.4.7).	Unit testing for functional risks is performed by development team, peer review is applied for verification of code. Each increment is tested by a clinical specialist to ensure safety. The final product is validated by the notified body to ensure efficiency.
Documentation and Traceability	Traceability must be enhanced for both changes and effects of changes to maintain transparency and accountability to the regulatory bodies (Sub-section 3.5.4 & 5.4.4).	Documentation is produced for each of the development phases including requirement elicitation, implementation, release, risk management; all through the Sprints.
Risk Management	Extra attention should be given to risk mitigation, risk controlling measures can be considered along with safety requirements (Sub-section 3.5.6 & 5.4.5).	Risk management begins from the requirement elicitation phase. Risk identification is done during Sprint planning. Identified risks are mitigated during the Sprints and documented accordingly.
Unpredictability and Variability	Safety critical requirements can be inserted into product backlogs along with other functional requirements to avoid uncertainty related problems with regulatory requirements A complete design scheme can be introduced for the final product, scope for the changes in functional design can be allowed. (Sub-section 3.5.3 & 5.4.6).	Safety requirements are inserted into the product backlog by the Regulation Specialist in the beginning of the Scrum process. The final product can never be uncertain since the regulatory requirements are very clear in the beginning since the efficiency of the software is also required. Complete design and test plan are documented before the development begins. Uncertainty is leveraged only for the functional perspective which does not disaffirm the regulation.
Number of Roles	Regulatory conformity assessment & Quality Assurance can be handled by involving regulatory bodies, by assigning a person in charge within the team to focus on regulatory perspective (Sub-section 3.5.8 & 5.4.9).	Roles include (a)Regulation Specialist, (b)Clinical Specialist, (c)Cyber Security Specialist, depending on the scale of the manufacturing organization. Compliance department is often formed for conformity assessment. Notified body is involved at the end of the development for releasing the software.
Self-organizing & Self-controlled Team	Safety requirements must be considered during the decision making within the team. Decisions must be verified by the compliance team (Sub-section 3.5.7 & 5.4.8).	Though the team is self-organizing, they are not self-controlled. The development is always monitored and incorporated with compliance; hence the team must adhere to the internal or external compliance team regarding the development of MDS.

Table 3 exhibits the complications mapped with corresponding solutions to mitigate the complications to comply with the regulatory requirements. The items in the first column are the complications listed in Section 3.5. The second column describes what is required for mitigating those complications based on the findings from the literature review. Finally, the last column encapsulates what practices MDS manufacturing companies may associate to incorporate the regulatory standards while developing their projects implementing agile Scrum as the software development model, according to the interview participants.

These solutions are derived from the interview process. Practitioners from different companies provided similar frameworks for their regulatory development in the case of MDS. They discussed different aspects of encountering the complications during the development. In addition, they discussed the corresponding alterations they adopt to mitigate those complications. The modifications conclude a general overview of mapping the complication to a particular solution that may solve the conflict between agility and regulatory requirements.

7 Conclusion

Scrum principles do not specifically address issues in relation to regulatory requirements or quality management processes. In the previous research in this domain, researchers have studied how agile can be implemented in MDSD and how efficiently agile can be integrated into such a domain. However, there is insufficient research on integrating Scrum in the MDS development regulatory framework.

This research has been initiated to identify the complication of adopting agile Scrum in MDSD. Agile, especially Scrum, has a structure where it is always flexible to changes. When implemented within a sequential plan-driven regulatory software development lifecycle, there is always a risk management process required. The thesis provides a general overview of the complications and proposed possible solutions for developing MDS implementing Scrum. It provides a clear idea of how Scrum can be implemented in the medical device domain by mitigating the complications related to safety and quality issues for regulatory compliance. This idea is validated by the feedback analysis for Scrum practices in the MDSD sector. Feedback from the domain experts has provided strong views of how organizations allow Scrum to mitigate the complications of adopting it under regulatory compliance. Furthermore, practitioners provided sufficient information regarding the current practices engaged in developing regulatory medical device software during the interview, which helped to portray the necessary enhancement to be adopted while implementing Scrum. Henceforth, the developers of today´s MDS may reconsider their methods and make a transition to Scrum if they are currently using pure plan-driven approaches only to comply with the regulatory standards.

The scope of this thesis calls for further research to specifically propose guidance to adopt Scrum in the medical device domain. For example, a risk management process in the Scrum framework satisfying the regulations, a complete validation and verification plan including 100% traceability for the final delivery according to safety requirements of the regulatory framework.

References

AAMI, ANSI/AAMI/IEC 62304 (2006), Medical Device Software-software life cycle processes. *Association for the Advancement of Medical Instrumentation,*94-101.

Abdelaziz, A. A., El-Tahir, Y., & Osman, R. (2015, September). Adaptive Software Development for developing safety critical software. In: *Proceedings of the 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)* (pp. 41-46). IEEE.

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. ESPOO 2002, VTT Publications 478. 107p.

Ahmad, E., Raza, B., Feldt, R., & Nordebäck, T. (2010). ECSS standard compliant agile software development: An industrial case study. In: *Proceedings of the 2010 National Software Engineering Conference,* 1-6.

Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*, *30*.

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-model vs. agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management, 2*(1), 26-30.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Jeffries, R. (2001). Manifesto for agile software development.

Bhavsar, K., Shah, V., & Gopalan, S. (2020). Scrum: An agile process reengineering in software engineering. *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, vol. 9, no. 3*, 840-848.

Black, S., Boca, P. P., Bowen, J. P., Gorman, J., & Hinchey, M. (2009). Formal versus agile: Survival of the fittest. *Computer*, *42*(9), 37-45.

Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed.* Addison-Wesley Professional.

Boehm, B., & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. *IEEE Software, 22*(5), 30-39.

Bouissou, M., Martin, F., & Ourghanlian, A. (1999). Assessment of a safety-critical system including software: A bayesian belief network for evidence sources. In: *Proceedings of the Annual Reliability and Maintainability. Symposium. 1999 (Cat. no. 99CH36283),* 142-150.

Bozzano, M., Cimatti, A., & Mattarei, C. (2019). Formal reliability analysis of redundancy architectures. *Formal Aspects of Computing, 31*(1), 59-94.

Bowen, J., & Stavridou, V. (1993, April). The industrial take-up of formal methods in safety-critical and other areas: A perspective. In: *International Symposium of Formal Methods Europe* (pp. 183-195). Springer, Berlin, Heidelberg.

Bulska, K., & Górski, J. (2011). Applying agiile practices to the development of safety-critical software. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej.Technologie Informacyjne, 1*, 65-68.

Commission communication in the framework of the implementation of the council directive 93/ 42/EEC concerning medical devices. (2017). *Official Journal of the European Union.*

Chittister, C., & Haimes, Y. Y. (1993). Risk associated with software development: a holistic framework for assessment and management. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(3), 710-723.

Derek Rayside, Aleksandar Milicevic, Kuat Yessenov, Greg Dennis, and 1764 Daniel Jackson (2009, October). Agile specifications. In: *Companion to the 24th Annual ACM SIGPLAN Conference 1766 on Object-Oriented Programming, Systems, Languages, and Applications, 1767 OOPSLA 2009,* Shail Arora and Gary T. Leav1765 ens, Orlando, Florida, USA, pages 999– 1768 1006. ACM.

Ericson, C. A. (2011). *Concise encyclopedia of system safety: Definition of terms and concepts* John Wiley & Sons.

FDA (1997). Design control guidance for medical device manufacturers. U.S. Food & Drug Administration.

Fitzgerald, B., Stol, K., O'Sullivan, R., & O'Brien, D. (2013). Scaling agile methods to regulated environments: An industry case study. In: *Proceedings of the 35th International Conference on Software Engineering (ICSE),* 863-872.

Fowler, K. (2004). Mission-critical and safety-critical development. *IEEE Instrumentation & Measurement Magazine, 7*(4), 52-59.

Ge, X., Paige, R. F., & McDermid, J. A. (2010). An iterative approach for development of safety-critical software and safety arguments. In: *Proceedings of the 2010 Agile Conference,* 35-43.

Gutiérrez, J., Ramos, I., Mejias, M., Arévalo, C., Sánchez-Begines, J. M. & Lizcano, D. (2017). Modelling Gherkin Scenarios Using UML. In: *Proceedings of the ISD2017* *on Information Systems Development: Advances in Methods, Tools and Management.* ISBN: 978-9963-2288-3-6.

Granlund, T. (2016). *Implementing a medical device software risk management process by ISO 14971 in compliance with agile principles,* Master´s Thesis, Tampere University

Granlund, T. Vedenpää, J., Stirbu, V., & Mikkonen, T. (2017). On Medical Device Cybersecurity Compliance in EU, *Regulation (IVDR), 746*, p.2.

FDA (2011). Guidance for industry and FDA staff. *Center for Devices and Radiological Health (CDRH),*

Hanssen, G. K., Haugset, B., Stålhane, T., Myklebust, T., & Kulbrandstad, I. (2016). Quality assurance in scrum applied to safety critical software. In: *Proceedings of the International Conference on Agile Software Development,* 92-103.

Harvey, J. (1959). TITLE 21—FOOD AND DRUGS CHAPTER I—FOOD AND DRUG ADMINISTRATION, DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE SUBCHAPTER B—FOOD AND FOOD PRODUCTS Part 121—FOOD ADDITIVES DEFINITIONS AND PROCEDURAL AND INTERPRETATIVE REGULATIONS. *Food, Drug, Cosmetic Law Journal, 14*(4), 269-290.

Heeager, L. T., & Nielsen, P. A. (2018). A conceptual model of agile software development in a safety-critical context: A systematic literature review. *Information and Software Technology, 103*, 22-39.

Heeager, L. T., & Nielsen, P. A. (2020). Meshing agile and plan-driven development in safety-critical software: A case study. *Empirical Software Engineering, 25*(2), 1035-1062.

Hron, M., & Obwegeser, N. (2018). Scrum in practice: An overview of scrum adaptations. In: *Proceedings of the 51st Hawaii International Conference on System Sciences,* 5445-5454.

Hrgarek, N. (2012). Certification and regulatory challenges in medical device software development. In: *Proceedings of the 4th International Workshop on Software Engineering in Health Care (SEHC),* 40-43.

Kircher, M., & Hofman, P. (2012). Combining systematic reuse with agile development: Experience report. In: *Proceedings of the 16th International Software Product Line Conference-Volume 1,* 215-219.

Kornecki, A., & Zalewski, J. (2009). Certification of software for real-time safety-critical systems: State of the art. *Innovations in Systems and Software Engineering, 5*(2), 149-161.

Bozzano, M., (2010). *Design and safety assessment of critical systems* (1st ed.) Auerbach Publications.

Mc Caffery, F., Casey, V., Sivakumar, M. S., Coleman, G., Donnelly, P., & Burton, J. (2012). Medical Device Software Traceability. In: Cleland-Huang J., Gotel O., Zisman A. (eds) Software and Systems Traceability. pp. 321-339. Springer-Verlag.

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

Mc Hugh, M., McCaffery, F., & Casey, V. (2012). Barriers to using agile software development practices within the medical device industry.

Moe, N. B., Dingsøyr, T., & Dybå, T. (2008). Understanding self-organizing teams in agile software development. In: *Proceedings of the 19th Australian Conference on Software Engineering (Aswec 2008),* 76-85.

Medical Device Corporation Group Document (2021), MDCG 2021-05,

Guidance on standardisation for medical devices April 2021.

Myklebust, T., & Stålhane, T. (2018). *The agile safety case* Springer.

Michaud, C. (2012). Templates repository for software development process. Published on 18 January 2012. Retrieved on 10 May,2021 from *https://blog.cm-dm.com/pages/Software-Development-Process-templates.*

Nair, S., de la Vara, Jose Luis, Sabetzadeh, M., & Briand, L. (2013). Classification, structuring, and assessment of evidence for safety--a systematic literature review. In: *Proceedings of the 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation,* 94-103.

Özcan-Top, Ö, & McCaffery, F. (2017). How does scrum conform to the regulatory requirements defined in MDevSPICE. In: *Proceedings of the International Conference on Software Process Improvement and Capability Determination,* 257-268.

Pawar, R. P. (2015). A comparative study of agile software development methodology and traditional waterfall model. *IOSR Journal of Computer Engineering (IOSR-JCE),* Vol. 2, No. 2, 1-8.

Rasmussen, R., Hughes, T., Jenks, J. R., & Skach, J. (2009). Adopting agile in an FDA regulated environment. In: *Proceedings of the 2009 Agile Conference,* 151-155.

Rottier, P. A., & Rodrigues, V. (2008). Agile development in a medical device company. In: Proceedings of the *Agile 2008 Conference,* 218-223.

Rubin, K. S. (2012). *Essential scrum: A practical guide to the most popular agile process* Addison-Wesley.

Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices. *http://data.europa.eu/eli/reg/2017/745/oj*

Regulation (EU) 2017/746 of the European Parliament and of the Council of 5 April 2017 on in vitro diagnostic medical devices. *https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32017R0746*

Regulation (EU) 2020/561 of the European Parliament and of the Council of 23 April 2020 amending Regulation (EU) 2017/745 on medical devices. *https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32020R0561*

Salo, O., & Abrahamsson, P. (2005). Integrating agile software development and software process improvement: A longitudinal case study. In: *Proceedings of the 2005 International Symposium on Empirical Software Engineering, 2005.* 10 pp.

Scholtes, M., Buedenbender, S., Behrend, A., Sohrabi, K., & Gross, V. (2018). Integrating a usability engineering process into a consisting risk management. *Current Directions in Biomedical Engineering, 4*(1), 645-647.

Schooenderwoert, N. V., & Shoemaker, B. (2018). Agile Methods for Safety-Critical Systems: A Primer Using Medical Device Examples. CreateSpace Publishing.

Schwaber, K., & Sutherland, J. (2012). *Software in 30 days: How agile managers beat the odds, delight their customers, and leave competitors in the dust* John. Wiley & Sons.

Shore, J. (2007). *The art of agile development: Pragmatic guide to agile software development.* O'Reilly Media, Inc.

Stålhane, T., Myklebust, T., & Hanssen, G. K. (2012). The application of safe scrum to IEC 61508 certifiable software. In: *Proceedings of the 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference,* 6052-6061.

Stephenson, Z. R., McDermid, J. A., & Ward, A. G. (2006). Health modelling for agility in safety-critical systems development. In: *Proceedings of the 1st Institution of Engineering and Technology International Conference on System Safety Engineering*, 260-265

Sutherland, J. (2014). *Scrum: A revolutionary approach to building teams, beating deadlines, and boosting productivity.* Random House.

Sutherland, J., & Schwaber, K. (2020). The scrum guide. *The Definitive Guide to* Scrum: The Rules of the Game. Published on November 2020. Retrieved on 11 February, 2021 from *https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf*

Turk, D., France, R., & Rumpe, B. (2002). Limitations of agile software processes. In: *Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering,* 43-46.

Turk, D., Robert, F., & Rumpe, B. (2005). Assumptions underlying agile software-development processes. *Journal of Database Management (JDM), 16*(4), 62-87.

Vogelzang, J., Admiraal, W. F., & van Driel, J. H. (2019). Scrum Methodology as an Effective Scaffold to Promote Students' learning and motivation in context-based secondary chemistry education. *EURASIA Journal of Mathematics, Science and Technology Education, vol. 15*, no. 12, em1783.

Wolff, S. (2012a). Scrum goes formal: Agile methods for safety-critical systems. In: *2012 First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches (Formsera),* 23-29.

Wolff, S. (2012b). Using Executable VDM++ Models in an Industrial Application - Self-defence System for Fighter Aircraft. *Technical Report Electronics and Computer Engineering,* 1, 1-18.

Zema, M., Rosati, S., Gioia, V., Knaflitz, M., & Balestra, G. (2015). Developing medical device software in compliance with regulations. In: Proceedings of the *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC),* 1331-1334.

Appendix 1
Interview Questions

General Questions

1. How many years have you been working in Medical Device Software Development?
2. Have you worked in any project which adapted SCRUM for the medical software development? If yes, to what extend was the SCRUM integrated into the development process? and what type(s) of medical software was delivered by the project?
3. What are the most important issues with developing medical device software? (Software lifecycle, regulatory constraints or what else?)
4. What are the main concerns for developing medical device software using an `Agile´ software development lifecycle? (Safety, regulations, anything else)
5. What are specified as regulatory requirements? When are they specified for a project? When are they analyzed? How are they adjusted into the project requirements?

Related Questions to RQ2

1. How do you mitigate the "Lower Quality Assurance" of agile development model while working on a medical software project? What sort of verification and validation you impose?
2. To consider a project (or piece of project) to be Done/Complete, what sort of validation or verification does it require to comply with regulations? Who is responsible for specifying the product definition?
3. Have you ever faced problems to handle `continuous changes in requirements´ for the development of a regulatory medical device software? If yes, how did you handle it?
4. Do you consider it unacceptable/inappropriate for regulatory software to release a piece of project (an increment, according to SCRUM) without detailed testing of the whole project in the medical device domain?
5. How do you introduce "Risk management" in the development lifecycle for complying with regulations?

6. What kind of documents are required in medical software development in compliance with regulations? What sort of tools do you use for documentation and requirements? Name some if possible.
7. For maintaining the compliance with regulatory requirements, who requires to be involved in the development process? Do you think it is necessary to introduce roles such as Regulation Specialist/Regulation Officer" within/outside the development team to comply with the regulatory requirements?
8. How do you think "Unpredictability/Uncertainty" of SCRUM conflicts with the rules of regulatory requirements? Does medical device software development require predictability in every aspect? Being unpredictable, how Scrum can handle this issue to comply with regulations?

Appendix 2
Abbreviations

MDS	Medical Device Software
MDSD	Medical Device Software Development
MDR	Medical Device Regulation
IVDR	In-Vitro Diagnostic medical device Regulation
MDD	Medical Device Directives
IVDD	In-Vitro Diagnostic Device Directives