Understanding the Benefits of Agile Software Development in Regulated Environments

Jens Karrenbauer, Manuel Wiesche, Helmut Krcmar

Technische Universität München, Chair for Information Systems, Munich, Germany
jens.karrenbauer@tum.de, {wiesche, krcmar}@in.tum.de

Abstract. Agile software development has become increasingly popular in recent years. Applying agile methods, companies expect flexible planning, early delivery of the software product, and a continuous improvement of the development process itself. However, in regulated environments the use of agile development is not yet common practice. In such environments, various regulatory requirements apply which affect the software development process. This paper examines the use of agile software development in the regulated medical device industry and explores reasons for using agile methods although their use is limited. We interviewed agile software development teams in three different companies using semi-structured interviews. Using grounded theory methodology, we identify reasons why companies are using agile methods, even though problems and barriers exist. Our main achievement is the development of four categories, which describe the benefit of agile software development in regulated environments. These categories are master complexity, reduce effort, improve usability, and promote collaboration.

Keywords: agile software development, regulated environments, benefit of agile software development, medical device industry

1 Introduction

For some time, companies adopted agile methods, such as Scrum, within their software development process. Reasons for agile adoption are benefits such as flexible planning, early delivery of the software production, or a continuous improvement of the software development, which results in better quality software [1]. However, companies often find it difficult to implement agile methods due to challenges in the organizational, cultural, and human context [2]. In regulated environments, this is intensified by various regulatory requirements that must be met, while at the same time reacting flexibly to market or customer requirements is necessary [2].

One characteristic of regulated environments is the multitude of compliance procedures, regulations and standards that have to be considered in the software development process. There are various organizations and associations, such as the International Organization for Standardization (ISO) or the Food and Drug Administration (FDA),
which formulate specific requirements [3]. Furthermore, self-imposed requirements apply which are not mandatory for an industry, but which are generally recognized standards required by customers [4].

One example is the medical device industry, which is highly regulated. Companies that develop software for the US market must guarantee that their software complies with all FDA requirements.¹ For all stakeholders of these organizations, it is important to see that the system developed was designed safely and effectively for the intended purpose [6]. Breaching these requirements can quickly cause economic damage to a company [7]. For instance, a violation of regulatory requirements can lead to sanctions such as product recalls, product seizures or export restrictions [8].

A company in the medical device industry must, therefore, meet the challenge of reconciling the both existing and changing, strict regulations with the agile approach and at the same time constantly reducing development cost [2]. Thus, understanding why agile methods are used in regulated environments despite these barriers is important. This paper attempts to answer the following research question: What are typical benefits of the usage of agile software development in a regulated environment? To explore this question, we conducted 20 interviews with members of agile software development teams in three different companies from the medical device sector. Using the Grounded Theory Methodology (GTM) for data analysis, our main achievement is the identification of four categories, which describe the benefit of agile software development in regulated environments.

The remainder of the paper is structured as follows. In the next section, we provide a brief introduction of the background literature. This is followed by an introduction of our research approach as well as a presentation of the relevant cases. We then present our results and discuss the findings in detail. The paper concludes by highlighting the contributions that our study makes to research and practice.

2 Background

2.1 Agile Software Development in Regulated Environments

Agile software development methods became more and more popular in recent years. Different methods have been classified as agile, e.g. eXtreme Programming (XP), Crystal, Feature Driven Design, or Scrum. Scrum is one of the most widely used agile development methods in practice [3]. This method has been adopted in many sectors of the software industry. Companies expect flexible planning, early delivery, and continuous improvement of the development process [1]. Agile methods promise to reduce development time, increase product quality, and reduce development costs [9].

¹ The EU directives for medical devices are in a period of change. On 25 May 2017, the new European Medical Devices Ordinance (MDR) came into force with a transition period of three years. This imposes extensive new requirements on the clinical evaluation of devices [5] and assimilates the requirements of the FDA and EU.
Agile was originally intended to be used by small teams in non-safety-critical projects and in co-located environments [10] [11]. However, agile methods need to be tailored to suit the needs of different circumstances [12]. Agile has been adopted to other contexts, including virtual environments [13], global software engineering [14], complex systems or Capability Maturity Model (CMM) environments [15] [16], as well as regulated environments [3].

Especially in regulated environments scholarly research finds puzzling results regarding the use of agile methods. The main conclusion of [3] is that agile practices and regulated environments are not necessarily incompatible. Others say that agile practices are unsuitable and can only be used successfully in combination with plan-based methods [17]. Most of the literature recommends an adaptation of agile approaches (e.g. [9], [18], [19]), while some consider it is too early altogether to apply agile methods in such environments (e.g. [15], [20]). One possible explanation for these inconsistent empirical findings are different regulated environments with different focuses.

Table 1: Relevant laws and regulations in the field of medical devices with their impact on the process.

<table>
<thead>
<tr>
<th>Law/Regulation</th>
<th>Comment/Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Focus: Control instrument for demonstrating the safety and medical-technical performance of medical devices.</strong></td>
<td></td>
</tr>
</tbody>
</table>
| EU Medical Device Directive (MDD), EU Medical Device Regulation (MDR) [23] [24] | - Define software as a medical device.  
- Divide medical devices into four classes which are used as the basis for the requirements to be implemented and the scope of an inspection by the authorities.  
- Require state of the art validation of the software, considering the principles of the software life cycle, risk management, validation and verification. |
| **Focus: Software Life Cycle Processes for Medical Device Software** |
| AANIS/AAMI/DIN EN/IEC 62304 [25] [22] | - Requires activities and tasks within the software life cycle required for the safe development and maintenance of medical device software.  
- Requests a prior determination of what is expected of the software and subsequent proof that the use of the software fulfils these intentions without causing unacceptable risks. |
| **Focus: Quality management systems for medical device manufacturers.** |
| ISO 13485, 21 CFR 820 [26], [27] [21] | - Requires the development of internal policies and procedures describing the QMS.  
- Requires the creation of Standard Operation Procedures (SOP).  
- Requires the creation and documentation of a comprehensible software development process.  
- Low-level processes must be consistent with those at a higher level and provide a coherent and consistent approach across the organisation that meets regulatory requirements. |
| **Focus: Application of risk management to medical devices** |
| ISO 14971 [28] [22] | - Calls for the introduction of systematic management of the risks associated with the development and operation of a medical device.  
- Requires consideration of product risks for patients, operators, others, and the environment.  
- Requires continuous monitoring of risks, even after delivery of the product. |
2.1 Requirements for Agile Software Development in Regulated Environments

For the medical device industry, different laws and regulations apply with regards to the software development process. Various organizations and associations such as the ISO and the FDA, formulate and publish regulatory requirements for product development [3]. For the development of software this implies a need for high quality, security and reliability. Table 1 illustrates the most relevant laws and regulations for the EU and the US and provides more information about potential impacts of these requirements on the software development process.

The table shows that the specific regulatory requirements differ between the countries. These international standards and guidelines make the medical device software development process unique [21]. Especially the IEC 62304, which places demands on life cycle processes for medical device software, is recognized by many regulatory bodies around the world as the gold standard [22].

There are strict requirements for the reliability and traceability of the products and their development process in the medical device industry. From these specific requirements, general rules can be derived which apply in a regulated environment in general. Table 2 highlights these common requirements. For instance, quality is one of the most important characteristics that a process and its software must show. Additionally, it is imperative to plan the entire project, as well as document all the activities done. Companies adopting agile methods face barriers when complying to the agile principles, for example, a lack of documentation or up-front planning. [29] Also, there are different agile barriers for the development teams which have an impact on the performance. Examples are the occurrence of subgroups or the relationship of personality models [30] [31] [32].

<table>
<thead>
<tr>
<th>Topic area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensuring quality</td>
<td>planning, implementation and proof of quality assurance measures [6]</td>
</tr>
<tr>
<td>Ex-ante planning</td>
<td>additional process steps (e. g., risk analysis or additional tests) [15]</td>
</tr>
<tr>
<td>Roles &amp; responsibilities</td>
<td>mandatory roles (e. g., security expert or a quality manager) [33]</td>
</tr>
<tr>
<td>Documentation</td>
<td>extensive, development-related documentation (e. g. requirements documentation or test specifications) [8]</td>
</tr>
</tbody>
</table>

The impact of these requirements leads to different problems, for instance during the overall process implementation, during the transition to the testing phase, or within the documentation. For example, all tasks and activities must be completed before a release takes place (including a complete documentation). This also means, software cannot be delivered without a complete documentation [18]. Additionally, it is not worth to deliver small releases in short cycles as the additional work to be done is extensive. Also, the costs of refactoring will become quickly very high as soon as a change involves a re-verification and re-evaluation of the artifacts concerned [20].
Literature showed several examples for such problems, which redundant occur in this context. That is the reason why we want to clarify and understand the motivation and benefits for companies in a regulated environment to use agile methods.

3 Research Approach

To understand why agile is used in regulated environments requires a rich understanding of the field and yields several potential explanations. We, therefore, find an inductive, qualitative research design appropriate. We conducted 20 interviews in three different cases and used the GTM for data analysis. Based on the case studies and the results of the GTM, four categories which describe the benefits of agile software development in a regulated environment have been elaborated.

Table 3: Background information of the interviewees

<table>
<thead>
<tr>
<th>ID</th>
<th>C</th>
<th>Role</th>
<th>Background</th>
<th>Exp.</th>
<th>Exp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Agile Coach</td>
<td>Computer Science</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Scrum Master</td>
<td>Medical Technology</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Product Owner</td>
<td>Developer</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Architect</td>
<td>Telecomm.</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Program Mgr.</td>
<td>Medical Devices</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Verification</td>
<td>Telecomm.</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Risk manager</td>
<td>Biology</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Req. Eng.</td>
<td>IT Specialist</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>Ergonomics Mgr.</td>
<td>Medical Informatics</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Product Owner</td>
<td>Industrial Eng.</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Scrum Master</td>
<td>Industrial Eng.</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Architect</td>
<td>Medical Informatics</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Risk Manager</td>
<td>Mechatronics Eng.</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Tester</td>
<td>Computer Science</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Developer</td>
<td>Electrical Eng.</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Program Mgr.</td>
<td>Electrical Eng.</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Program Mgr.</td>
<td>Electrical Eng.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Scrum Master</td>
<td>Medical Informatics</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Developer</td>
<td>Computer Science</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>20</td>
<td>C</td>
<td>Product Owner</td>
<td>Computer Science</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>

Legend: C = case; CO = company; IN = industry; MD = medical devices; S = Scrum; K = Kanban; N = Nexus; XP = eXtreme Programming; Exp. = Experience with regulated environments; AV = aviation; AT = Automotive; RW = Railway

3.1 Research Method

For sampling our data, we chose a holistic multiple-case design, where we identified several distinct case organizations to be included in our sample [34]. We identified three companies, each of which is regarded as a separate case. A description of these cases is provided in the following.
In all cases semi-standardised interviews were conducted to collect data [35]. An interview guideline was developed which served as a basis for the interviews. This guideline included questions on known regulations, agile procedures used in the company, characteristics of the implemented active development process, and one's own opinion on existing conflicts. Additionally, we asked questions about the benefits and existing barriers of the usage of agile software development in the regulated environment. The aim was to interview each role of the agile software development team in order to gain a comprehensive picture of the benefits using agile software development methods in regulated environments [35].

We conducted 20 interviews. Each interview lasted between 40 - 90 minutes. Table 3 provides background information of all interviewees. After each interview, a theoretical memo was prepared to record our impressions and thoughts about the interviewee [36]. The interviews were transcribed and analyzed with Atlas.ti.

For data evaluation and analysis we followed Grounded Theory Methodology (GTM) [37]. In the first step of the analysis, initial concepts were identified in the data. These were divided into categories (open coding). Concise and self-explanatory codes were used to create the categories. Next, the open codes were divided into groups by selective coding. Due to the large number of codes, we also worked with subcategories of the respective categories.

3.2 Cases

We identified three companies from the medical device industry which are already using agile methods within their software development process. The first case, INFUSE, is a software design house with experiences in the automotive, safety & security, aerospace, and medical device sectors. The project focuses on the development of software for an automatic infusion pump. The implementation of Scrum is close to the framework. Concurrent to the agile process, there is a risk management, a verification, and a configuration management process. Issues or requirements of the concurrent processes are given as requirements in the Scrum process. In our view, the company does not distinguish between regulatory requirements and product requirements as they want to use the opportunity of developing a high-quality software product.

The second case, XRAY, is a company in the dental industry. The interviews were based on the development of a software, which is used for the imaging of X-ray machines (image display, image enhancement). A special feature is that the software can be used as a variant with other imaging devices (e.g. industrial area). Currently, there are two agile Scrum teams working on this project. There are also many other roles involved, e.g. an Ergonomics Manager or a Requirements Engineer. At the start of a new project (or a new release) all requirements have already been defined. The whole software development process is very extensive and rigid. The interpretation of the existing regulations is always enforced strictly.

The third case, DATA, is a provider of products and solutions for ophthalmology, neurosurgery, dentistry and oncology. The software can be used to collect, process and archive data from their system as well as third-party devices. In addition, the data is prepared for the physician and the information is made available for diagnostics. The
implementation of the Scrum framework at DATA is a pragmatic approach. Nevertheless, it takes them an average of one year until a new release takes place.

4 Results

4.1 Software Development Process in the Cases Applied

The three projects are similar at a high level. The same regulations apply to all three cases, since all software products are grouped in the same classification of medical devices (IIb). In addition, the same agile practices are predominantly used. In all cases the embedding of the software development process in a higher-level product development/medical product process took place and run as a V-model. For the software development they move into the agile process. After several iterations within Scrum, the agile process ends, and the V-model is completed with verification/system tests and other necessary activities. Table 4 provides an overview of the three cases.

<table>
<thead>
<tr>
<th>Table 4: Comparison of the three cases</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case A</strong> (INFUSE) &amp; <strong>Case B</strong> (XRAY) &amp; <strong>Case C</strong> (DATA)</td>
</tr>
<tr>
<td><strong>Employees</strong></td>
</tr>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td><strong>Form of Software</strong></td>
</tr>
<tr>
<td><strong>MD classification</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Relevant regulations</strong></td>
</tr>
<tr>
<td><strong>Team Size</strong></td>
</tr>
<tr>
<td><strong>Used agile practices</strong></td>
</tr>
<tr>
<td><strong>Agile strategies</strong></td>
</tr>
</tbody>
</table>

In order to compare these cases, we need to take a deeper look into the processes and have, therefore, identified three main characteristics by using GTM which are handled differently. These characteristics are implementation of requirements, documentation and final integration/system test. Implementation of requirements means time and procedures for the implementation of requirements. Documentation describes the time and
procedure of the documentation. Final integration/system test characterises the handling of these final tests. Table 5 compares the three cases.

Table 5: The software development process in the three different cases

<table>
<thead>
<tr>
<th>Infuse</th>
<th>Xray</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Implementation of requirements</strong></td>
<td>- user stories are partly defined in advance, or they are created dynamically parallel to the development</td>
<td>- strict modular processing of the various steps, requirements are defined and approved in advance</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>- documentation is usually made in parallel or after development</td>
<td>- without finalized documentation, the software may not be implemented</td>
</tr>
<tr>
<td><strong>Final Integration / System test</strong></td>
<td>- a full system test is performed after the last sprint and before release</td>
<td>- a full system test is performed after the last sprint and before release</td>
</tr>
</tbody>
</table>

4.2 Benefits of Using Agile Methods in Regulated Environments

If one looks at the problems and barriers identified in the previous chapters, it is noticeable that these areas match the aspects that are considered as an incentive for the introduction of agile development methods. The coding of the data revealed four categories of motivation to introduce agile development methods. These four categories are master complexity, reduce effort, improve usability, and promote collaboration.

Master complexity

Software development and software itself have grown more and more complex in the previous decades. With new technological possibilities, increasingly complex systems found their way into medicine. In this context, agile methods help the software development team to deal with complexity-related issues such as compatibility with modules, integration with hardware, or handling all the regulations applying to the software development process.

In contrast to an agile approach, the classic V-model provides a complete definition of all requirements at the beginning. In the case of complex systems, this is sometimes difficult to implement, as the entire set of requirements must be described initially. If a V-model is used for software development today, it leads to additional efforts as existing developments have to be continuously changed. Regardless of the selected process model, an iterative procedure is required to master the complexity.

There are different possibilities how agile helps to master the complexity in a software development project. For instance, agile is suited very well if the definition of the final product is not quite clear at the beginning of the project: “Actually, as soon as it becomes more complex and at the beginning is still a certain blurriness in the whole,
then it is always helpful, even if it is only that I go through the first phases agile and iteratively.” (INFUSE). Additionally, iterative learning is nowadays almost indispensable for complex system development:

“Mastering the complexity is in an agile team, where I don’t have quite rigid requirements, and that has to be on the table in three years (...). But, I have a vision, and also the vision I adjust in my reflections constantly after a bit. Of course, I can’t throw it all over the place, that there will be something completely different. Because then I produced for the trash can. But that it is just a matter of bringing a vision into the world and with the knowledge I learn about the iterations, that it really grinds itself so finely that in the end what I need really comes out.” (XRAY).

Another point and what we already mentioned is, that complex software can no longer be defined completely in advance. The reason for this is the high uncertainty regarding the features to be developed, as well as the difficulty of understanding complex interdependencies in advance: “I don’t have the chance to initially describe all requirements, all interfaces in such a way that I can easily implement them.” (XRAY).

The following quote explains this in more detail:

“The whole world and technology and systems are becoming increasingly complex. This means that we are networked, all devices talk to each other, and the complexity is meanwhile on one level that it is not controllable in this classical process model topic, like also a V-model.” (INFUSE).

The degree of complexity in such an environment became so high, that the traditional process models are no longer suitable: “And that’s why a big key of agility is that none of the very complex systems, no single person or individual, is able to describe the system in such a way that it is then processed by a team, and in the end what I really want, and need comes out.” (INFUSE).

This illustrates how the use of agile methods helps software development teams to deal with complexity. In a regulated environment, an agile approach is relatively rigid, but still more dynamic than in a classical process model.

**Reduce effort**

An agile approach is far more efficient than a classic process model. It can help software development teams to reduce effort in a better way than classic process models do. Efficiency can be achieved by saving time (and thus increasing speed) or preventing waste (and thus reducing the number of tasks). Speed is an essential factor in software development. With agile methods and procedures, the development of the product begins earlier in the development process.

Generally, an agile approach is leaner than a classic approach and, thus, is better able to reduce effort in a software development process. In addition, a minimum viable product is usually available at an early stage: “Because the agile is more like saying with the whole toolset, you start working much earlier on what the Minimal Viable Product is. That means you try to understand the problem much better, so you don’t generate waste by specifying something that no one needs. And there is also such a bit of cultural change necessary.” (INFUSE).

Additionally, with an agile approach, it is possible to react and develop faster. This is a reason why companies are choosing this method: “That was our main motivation
to actually introduce agility. Namely to be able to react faster to changing market requirements in our process world as well, and in development.” (DATA). Also, traditional activities already planned are often not pursued further, resulting in unnecessary planning effort. In addition, requirements are often implemented in the software that are not required in the end. Software developers refer to this as "waste":’

“[…] a motivation to originally introduce agility was that the predictability about what is the feature scope had nothing to do with what was then in the product requirement spec. So, there was something in it, but during the development period, it felt 50% completely different. And then it had made absolutely no sense to stay in this rigid process model now.” (DATA).

“[…] and because he knows that, he defines what he thinks what will be needed in five years. The next big thing. And most of it is just over-specified. This means that you simply create a solution to problems that no one has.” (DATA).

In summary, this means that in an agile environment only needed requirements are implemented: “And then I have all the requirements that I want to implement, which I would not find with a classical approach. Because then I add insane things and have as requirements that no one really needs, and I would still forget a lot.” (DATA). This, combined with regular feedback, leads to an efficient development process. Although mandatory activities for the provision of software must take place for a regulatory environment, these are, as expected, equally mandatory in a classical process model. The advantage in an agile environment is that the waste previously generated with conventional methods is no longer necessary, which reduces the effort and increases the efficiency in software development.

**Improve usability**

Even though some essential aspects of agile methods, such as early delivery or more flexible planning, are difficult to apply in a regulated environment, there are other facets that are beneficial and suitable for that environment. Software development teams can use these lean frameworks with a limited number of rules and specifications in various scenarios. Due to their simplicity, they are adaptable and can be combined with various agile and non-agile methods and procedures.

One example for the improvement of the usability is the fact that an agile approach leads to more innovation through employees. “I can let the employees go, those who are normally motivated, they do it. And others, who are highly motivated, bring in more innovation and so on.” (INFUSE). Another example is the continuous process improvement which is very well supported by the Scrum framework. “[…] the knowledge, and then contribute it back to the improvement process. So continuous process improvement. That's good.” (XRAY). This is accompanied by regular feedback and adjustments which are carried out within the iterations. “And what is definitely useful are the short iterations, where you quickly realize, ok, maybe we are on the wrong path after all, maybe we still have to readjust.” (XRAY).

Additionally, while using Scrum, it is possible to be nearer to the reality: “And it's just the same, as always, the problems are actually always human. And that doesn't matter which process it is, you will always have that. Only the agile process helps us to be much more in reality than the waterfall process would do.” (DATA). Last but not least, the documentation within a software development project is simpler than in a
Agile methods have several advantages which improve the usability within a software development process. Continuous process improvement is characteristic for an agile environment. This is particularly helpful in a regulated environment due to the existing rigidity and necessary additional activities. This includes providing regular feedback to stakeholders or the entire Scrum team, as well as promoting innovation in the development process by employees. A final point is the fact that an agile approach helps to develop realistically. With classic process models, development is a kind of black box for outsiders, as little feedback is given to the developers during development. Agile procedures provide transparency in the development process due to continuous feedback and a continuous target/actual comparison.

**Promote collaboration**

Agile methods promote collaboration between different roles and stakeholders within the software development process. Agile software development methods encourage an active collaboration between different stakeholders. This can lead to an increase in quality both in the process and the software product itself.

According to the interviews, the developers prefer to work in an agile environment rather than in a classical process model: “[...] typically, it does not apply to all developers, but the developers prefer to work in this agile model rather than in a classic model. So, for me, the classic model in software development has simply outlived itself in any industry.” (DATA). Developing agile entails, a greater appreciation for the developers is shown because they work on their own responsibility and the team is organized independently: “Many developers say they want to develop agile. That is the motivation. Then we can offer you whatever you need.” (INFUSE). Active involvement is desired and enabled. This can be transferred from the team level to each team member.

Furthermore, external stakeholders can be involved in the process in Scrum: “And then we continued to pursue this Scrum idea. Even if they're not on the team, just get them closer. So, they will be there on every Scrum-Day. And at the review meeting and see what's new.” (INFUSE). “So, if you don't want to join the team and the same office, just get as close as possible and do as much as possible in small steps in the sprint.” (XRAY). Different roles and stakeholders are working closer together. An agile approach also increases employee satisfaction: “Because the satisfaction is better after all, yes. Employee satisfaction is also important.” (XRAY). The team becomes more dynamic, more flexible and more efficient. The human component is a (success) factor and reason to consider agile developments in regulated environments.

To summarize, an agile approach supports working together with different roles and responsibilities. It is, therefore, suitable in a regulated environment as many stakeholders need to be involved and coordinated.
5 Discussion and Conclusion

The goal of this research was to shed light on the benefits and barriers of agile methods used for software development in regulated environments. The aim was not to find new agile practices, but to find the right fit for a regulated compliance context.

Our results confirm that software development in regulated environments emphasize quality assurance, ex-ante planning, additional roles, and documentation. Furthermore, our results show that organizations use agile methods in regulated environments to master complexity, reduce efforts, improve usability, and promote collaboration. Hereafter, we discuss our contributions, limitations and provide avenues for future research.

Our research contributes to the ongoing question why organizations use agile software development methods, given the strict regulations in compliance savvy environments [3]. Our results suggest that agile procedures, such as a project vision, do not only help mastering complexity of both software and hardware solutions that involve APIs and different stakeholders, but also support project planning by separating long-term and short-term planning in different mechanisms. Thereby, our results suggest that a good tailoring of the agile method leaves developers with a suitable methodology to work user-oriented and efficient, meet regulatory requirements [38] [39] as well as perform innovation projects [40].

Our results provide additional detail on how agile methods can be tailored to regulated environments [41]. We identified two different strategies, which we refer to as (1) embedding agile in existing V-model based development and (2) using agile as toolbox of procedures. When embedding agile in an existing plan-based environment, the agile part focuses on the development aspects and usually uses plan-driven approaches for testing and verification. When using agile as a toolbox to improve software development, we found that the teams developed what we refer to as mini-Vs within each sprint: A hybrid approach of combining agile and plan-based elements in each iteration.

Overall, we contribute to the literature on usability of agile methods by enhancing literature on scaling agile methods [42]. Our results provide evidence that agile methods can be successfully applied in regulated environments. We analysed three cases that tailored agile methods to their context by using agile procedures to organize daily work to foster collaboration and focus on user-centricity procedures to ensure usability and cope with complexity. The teams combined this with plan-driven elements for planning and interfaces, especially when complex hardware was involved.

Our study is subject to limitations. First, we selected teams that reported to actively use agile methods and, thus, we might have missed configurations where plan-driven approaches were enriched with agile elements. However, we examined three different cases and asked interviewees about the process, tailoring, and their experiences to ensure a broad perspective. Second, we focused on cases regulated by medical device compliance requirements. We argue that this is a typical regulatory environment but call for future research in different industries such as pharma, banking, and aerospace. Third, the focus of this paper is mostly on the benefits of agile software development in regulated environments. Drawbacks and problems were only touched briefly. We recommend for future research to also consider these aspects. Lastly, it is inherent to exploratory qualitative work that generalizing the results is challenging. For example,
we derived our results from agile teams that used Scrum as agile method. Thus, they cannot be taken for granted for methods such as XP or lean. Further research should extend our theoretical sampling and examine other organization size and company context, including distributed teams and other agile methods. Research and practice would benefit from design science research that develops new artifacts, e.g., tools for agile teams in regulated environments or apply method engineering to develop a new development method for applying agile in such environments.

Overall, this research empirically illustrates the tailoring of agile methods for regulated environments. Our findings illustrate that regulated environments emphasize quality assurance, ex-ante planning, additional roles, and documentation. We show that organizations make use of agile methods to master complexity, reduce efforts, improve usability, and promote collaboration.

References


