Adopting Agile in the Sports Domain: A Phased Approach

Jennifer Callan-Crilly[1], Alan Moynagh[1], Özden Özcan-Top[1], Fergal McCaffery[1,2]

[1]Dundalk Institute of Technology, RSRC and Lero, Dublin Road, Dundalk, Ireland
[2]STATSports Group, Newry, Ireland

jennifer.callancrilly@dkit.ie, alan.moynagh@dkit.ie, ozden.ozcantop@dkit.ie, fergal.mccaffery@dkit.ie

Abstract. Sports Science is a new and evolving industry. There is a great potential in this domain which will be realised by capturing and analysing the performance data of the elite athletes and displaying all relevant information to them for better decision making and performance improvement. Establishing reliable systems to achieve performance monitoring in the sports science domain require hardware sensors, firmware and software algorithms work coherently. Such complex systems having also cyber-physical characteristics would bring their own challenges. In this paper, first we present the challenges related with the domain and the development environment based on our experiences in the STATSports Company. Then, we discuss how we adopted agile software development practices to overcome these challenges in a phased approach.

Keywords: Sports Science, Agile Software Development, Agile Adoption, Scrum, Distributed Teams, Feature Driven Development, GPS Tracking.

1 Introduction

Software engineering is different than the other engineering disciplines due to its essential and accidental difficulties [1]. The Agile Manifesto [2] has been established with a set of values and principles in 2001 by a group of software engineers in search of a better approach to software development. Agile software development (ASD) methods focus on customer satisfaction by offering continuous delivery of quality software at time-boxed intervals. They promote communication and collaboration between the development team and all stakeholders throughout the life of the project. ASD welcomes change during the development with implementation of specific practices such as product backlog grooming, sprint planning, face-to-face communication, automated testing, continuous integration etc. Reviews of the process at the end of each short cycle allow the team and stakeholders to continuously refine and improve their product and process [2].

Due to these benefits observed in the field [3][4][5], we aimed to adopt agile software development practices in the STATSports company, performing at the elite athletes domain to monitor performance data of the athletes and presenting analysed performance information. Sport Scientists use this information to monitor fatigue levels, appraise performance and assess injury risk. The detailed level of the data collected means that it can be classed as medical data and additional processes are needed to offer adequate data protection and ensure safety. The major reasons for agile adoption in the company are to improve communication between distributed development teams, produce quality software, promote customer involvement, offer stakeholders transparency throughout the development process, improve delivery of new software, increase data security and ensure traceability from design to release. The purpose of this paper is to present the domain and development related challenges and describe how we improved the development environment by adopting the agile software development practices.

The remainder of the paper is structured as follows: In Section 2, we present the background which includes a brief literature review on agile software development and the characteristics of the sports science domain. In Section 3, we present the challenges specific to the STATSports company which were identified with MDevSPICE® based process assessment. In Section 4, we provide the solutions developed for the organization which were adapted using a phased approach. Finally, In Section 5, we conclude and present the future work.

2 Background

In this section we provide a brief background on agile software development and the characteristics of the organization that we have performed the agile adoption.

Agile Software Development.

Agile software development methods are designed around four core values; *Individuals and Interactions over processes and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation and Responding to change over following a plan* [2]. While all of the above is valued within the agile process, it is the factors listed first which are considered most important [2]. These core values have been incorporated in a number of agile approaches which are widely used in software development. Crystal Methodologies [6], Dynamic Systems Development Method (DSDM) [7][8], Feature Driven Development (FDD)[9], Extreme Programming (XP) [10] and Scrum [11][12] are among the most common approaches and are implemented across a number of different industries and domains [4][5]. We focused on these five ASD Methods when assessing a best fit model for STATSports.

Agile Software Development promotes regular intense communication throughout the Software Development Life Cycle (SDLC), this offers transparency to all stakeholders and allows for more accurate risk assessment as the project progresses. The small timeboxed iterations or sprints used in the agile process mean that after a short time working software is produced and validation and verification are carried out frequently. This process improves the quality of the code produced by reducing bugs and ensures the system is delivered as expected by the customer. Agile practices also reduce the maintenance effort after product release[1][2].The Agile approach chosen greatly depends on the environment in which the software is to be developed, a detailed look at the characteristics of the project is needed to determine a suitable approach [14][15].

Although there are large number of studies in the literature regarding agile development methods in various domains, we were unable to find any publications relating directly to the very new Sports Science domain. To get an understanding of the challenges and successful application of agile development methods in the Sport Science domain, we examined publications relating to safety critical domain and mobile medical device software as they would be subject to similarly restrictive regulations which maybe challenging to implement within an agile development lifecycle. Research suggests that it is entirely possibly to implement agile practices within safety critical domain as many of the iterative practices offer repeated opportunities for risk assessment, verification and validation [4][5] [16][17].

Characteristics of the organization and the projects.

STATSports was founded in 2007 and launched its first product, a performance tracking device for elite sports clubs. GPS Tracking along with other instruments provided by the device allow the tracking of movement for a given player. The product consists of three components, hardware, firmware and software. The first product of the company had been very successful but rapid advances in sports science meant that the company needed to evolve to offer the most up to date analysis information to its clients. STATSports' new product, is designed to produce more detailed data and will calculate additional metrics offering a better experience with its superior real time functionality.

The field of Sport Science involves the study of physiology, psychology, anatomy, biomechanics, biochemistry and bio kinetics. Physics theories are applied to the movement of the body which is captured by a number of small highly sensitive sensors imbedded in the device. The Software is developed to extract data from the sensors, it is then passed through a calculation engine where algorithms produce metrics and statistics. These can then be viewed through the software for individuals and on a team level.

The data produced by the device and software is considered highly sensitive performance information data or Electronic Personal Health Information (ePHI), strict data security requirements have been specified to ensure compliance with the General Data Protection Regulations (GDPR) which come into effect in May 2018 [18]. The company is also striving to obtain HIPAA compliance (The Health Insurance Portability and Accountability Act of 1996). On obtaining HIPAA compliance, the device will be counted among a small group of non-medical devices worldwide which adhere to these data security standards [19].

3 Challenges

The product consists of hardware, firmware and software components, all three of which are developed in different locations; hardware component is developed in the Republic of Ireland, the firmware team is located in Romania and the software team is based in Northern Ireland. Dependency between the three components is incredibly high and a lack of process across all departments caused many road blocks and delays. We assessed the challenges of the project under two categories; Development Challenges and Domain Challenges. The development challenges were specified based on an MDevSPICE® assessment which took five business days, performed by two assessors. MDevSPICE® is a process capability assessment model for medical device software process assessment [20].

Development Challenges.

- There was no single point of contact for gathering new requirements/requirement changes in the organization. The organization's Sport Scientists were directly contacting developers and demanding features to be developed. This resulted in developers' receiving different tasks from multiple sources and it was difficult to understand which new features should be prioritized
- Software, hardware and firmware departments demonstrated an Ad-Hoc approach to development, developing features as they were suggested by various stakeholders. A major challenge was that new requirements and features were constantly introduced throughout development.
- The organization operates in a hardware led software development environment yet no documentation relating to design decisions existed within the hardware department.
- The firmware team who develop embedded software for the organizations' products produced infrequent emails regarding changes made in new firmware versions. Small adaptations in firmware could completely change the way the software communicated with the device causing existing software to stop working and unnecessary hours of debugging by the software team to determine the cause.
- There was no shared code repository meant that each team was looking at their piece of the product in isolation.
- There was no formal traceability from system requirements to the testing and release phases
- Poor communication between the distributed departments meant they followed separate Ad-Hoc development plans which led to development being out of sync.
- The software development team adopted a "Code and Fix" approach to development to ensure software delivery and responded to all requests without evaluating hardware and firmware requirements.

- Manual builds, poor code repository management and a lack of versioning meant that a number of versions of software were in production at the same time.
- The team spent a large amount of time post release bug fixing and performing maintenance.

Domain Challenges.

- Complex computations and algorithms needed to transform scientific concepts into meaningful performance metrics from GPS and sensor data.
- The highly sensitive nature of the data recorded and captured meant that documentation of decisions and development lifecycle would need to be provided to satisfy strict regulations.
- Additional Data security measures needed to be implemented to comply with data processing and storage regulations.

The challenges listed above have led the company to search for alternative software development approaches. The decision was made to design and implement an agile software development life cycle to improve the continuous delivery of the new product and significantly reduce the number of bugs remaining in each release version. The method chosen must allow for change while offering opportunities to monitor and control. This project was a massive undertaking involving the development of several pieces of hardware which must integrate with firmware, and software (Apple/Windows) to produce accurate performance data. By introducing agile practices, we aimed to increase the transparency of the development process, reduce time to market, reduce bugs, reduce maintenance effort, increase data security and increase communication between all three development departments.

4 Adopting Agile

4.1 Choosing the Agile Methods and Practices to Implement

With a clear understanding of the development environment and the challenges involved, we needed to decide on an agile approach for the development of the new product. The complexity of the project with many custom algorithms and it's scientific application domain ruled out DSDM as research suggests it is best suited when applied in a business domain and is less effective when used in engineering or scientific applications [7][21][22]. The Crystal's stretch to fit model looked promising with four different levels of implementation to match the complexity level of any project yet problems had been noted when applying across a distributed team due to the core value of Osmotic Communication [6][21][23]. Having reviewed numerous publications and assessed the domain, FDD was chosen as the agile approach most suited to this project [9][24][21]. The new product needs to incorporate all the features offered by its predecessor along with new features identified by the Sports Science department. Visualising development by feature was already the preferred approach of the company so FDD was universally welcomed by the management. The FDD process model was followed, allowing a team made up of Business Stakeholders (IT, Management and Sports Scientists), the Senior Software Architect and Senior Developers to create an overall project model by defining the architecture and a high-level plan.

Creating a prioritised feature list allowed all three development departments to create a development roadmap and facilitated synchronized development planning. FDD improved the overall project planning and offered a clearer overview for all stakeholders. In order to offer full transparency to stakeholders throughout the development lifecycle it was decided that Scrum framework [25][16][12] should be applied in addition to a feature centric approach.

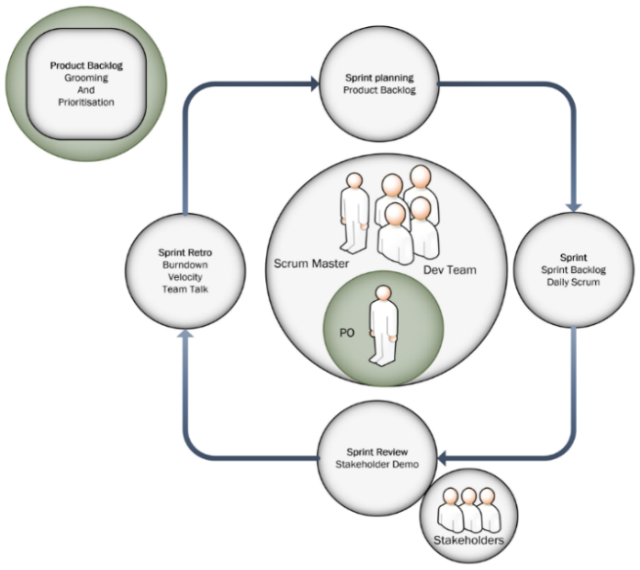


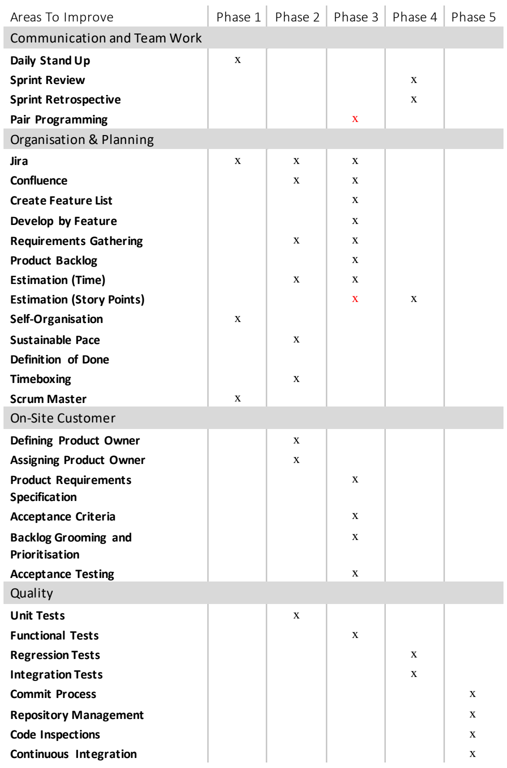
Fig. 1. Scrum Iterative Cycle

Scrum supports iterative development and promotes communication within the development team [3]. It encourages development cycles of between 1 and 4 weeks and facilitates frequent backlog grooming and requirements refining within each cycle (See Figure 1). The defined roles within the Scrum process allow for close collaboration between departments and increase transparency throughout development. The role of Product Owner (PO) is crucial in Scrum, the PO decides what will be selected for development and when, they are also responsible for specifying requirements for all features. The PO needs to have a very good understanding of what is required by the system and must be available to the Development team throughout the process. The Scrum Master (SM) is a servant leader role within the Development team, the SM is there to ensure that scrum values and practices are followed within each sprint. The SM facilitates communication between the PO, Dev Team and Stakeholders during the development Lifecycle, removing impediments for the Dev Team is also an important part of this role [21][11]. We decided to apply Scrum to FDD as a development management framework.

4.2 Implementing Agile

Changing to an Agile software development life cycle was a big undertaking.
It was decided to introduce agile methods' (FDD and Scrum) practices in phases to allow a gradual adoption and limit disruption to ongoing development activities (please see Table 1 for the practices adopted in each phase). The practices at Phase 5 are planned at this stage.

Table 1. Phases of Agile Implementation and Adopted Practices for each Phase



Areas To Improve	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Communication and Team Work					
Daily Stand Up	x				
Sprint Review				x	
Sprint Retrospective				x	
Pair Programming			x		
Organisation & Planning					
Jira	x	x	x		
Confluence		x	x		
Create Feature List			x		
Develop by Feature			x		
Requirements Gathering		x	x		
Product Backlog			x		
Estimation (Time)		x	x		
Estimation (Story Points)			x	x	
Self-Organisation	x				
Sustainable Pace		x			
Definition of Done					
Timeboxing		x			
Scrum Master	x				
On-Site Customer					
Defining Product Owner		x			
Assigning Product Owner		x			
Product Requirements Specification			x		
Acceptance Criteria			x		
Backlog Grooming and Prioritisation			x		
Acceptance Testing			x		
Quality					
Unit Tests		x			
Functional Tests			x		
Regression Tests				x	
Integration Tests				x	
Commit Process					x
Repository Management					x
Code Inspections					x
Continuous Integration					x

PHASE 1 (3 WEEK DURATION).

Communication within the development team was essential for the process to succeed, so it was identified as a priority for the first phase, the role of Scrum Master was assigned. the SM was to set up daily Stand-Up meetings for the development team. The team used these meetings to flag issues or concerns they had regarding impediments to development and team members were encouraged to offer advice and help. The first week of Stand-Up meetings was a little uncomfortable for the developers but by the end of week 2 they had become more relaxed and accustomed to speaking in front of their colleagues. This open discussion and collective ownership of problems helped the team build trust and began to change the mind-set of the developers from individual to team.

Face to Face communication was improving in the team, but, there was little to no record of development tasks, a software development tool needed to be introduced to help the team plan and track their progress. Jira was chosen after researching available options, it offered a well-designed UI, allowed for a significant level of customisation and integrated with a number of development tools which we planned to introduce. Initially a Scrum board was created with an active empty Sprint, the developers created tasks and added them to the active Sprint as they started development. The initial workflow consisted of 3 swimlanes (To Do, In Progress and Done), User Stories and Tasks did not require estimation. In this gentle introduction to Jira, the developers became familiar with the application and gradually creating and logging their tasks became part of the development itself.

PHASE 2 (2.5 MONTH DURATION)

For the second phase there were three main goals: The first one was to bring a structure and create transparency in the development process. The second one was to allow the team time to familiarise themselves with management tools, Jira and Confluence, by introducing additional functionality gradually. Writing quality user stories, recording requirements and gathering efforts were the learning goals for this phase. Quality was the third consideration in this implementation with the introduction of unit tests as a mandatory task for developers.

Time-box sprints were a very big change for both the development team and management. The sprints were set at one-week duration as they were operating in the improvement phase of an old product. A working week was agreed to be 32.5 hours. The developers were instructed to only commit tasks to a sprint which they could complete in a working week. The short sprint duration offered the developers frequent opportunities to refine the process and with each new sprint they began to close the gap between committed and completed tasks. Time-boxed sprints allowed developers and Stakeholders gain a better understanding of the length of time required to complete a task or story. Starting with one-week iterations meant that developers had to place greater emphasis on requirements analysis to ensure any tasks chosen were concise and clear allowing for completion within the sprint. This left little room for ambiguity while gathering specifications and within three weeks began to show improvements in the requirements analysis process while also increasing the quality of the tasks and stories created.

Confluence [26] a document repository and project management tool by Atlassian was introduced to improve traceability and quality throughout the SDLC. All requirements being passed to developers needed to be recorded in a requirements document. Confluence was solely used by the developers at first, allowing them to complete requirements templates for upcoming features. The developers had to analyse the features they had committed to deliver from a different perspective, this allowed them to break down each feature into more manageable / deliverable pieces. During the requirements analysis and gathering process questions arose about implementation and requirements had to be refined. Features could evolve and change during this process. A greater transparency prior to development helped us to identify risks, for example, additional technologies needed to develop a feature, a lack of domain knowledge for the developer meant that research was needed before committing the tasks or stories to the sprint.

As mentioned in the first implementation phase above, the developers started to record their development tasks as user stories on a simple Scrum board. The purpose of this basic introduction was to get the developers using Jira to record their tasks as they developed. The tasks recorded contained minimal information and related to the technical implementation steps rather than describing the functionality. It was difficult to associate a task to a functional requirement. To improve this process and aid transparency for all stakeholders, we introduced a definition of User Story based on Bill Wake's INVEST approach [27] . Developers had to create a user story for small pieces of functionality, giving it a meaningful title, which was easy to read by non-team members. By adding readable stories to the backlog stakeholders could easily relate work items to a specific piece of functionality.

To improve the requirements analysis and gathering process, a Product Owner (PO) role was defined. The PO is the sole source of requirements specification for features. Only the specified product owner can request features and prioritise their delivery. An experienced senior Sports Scientist was picked to become the PO and all future feature requests had to be presented to the PO for approval.

PHASE 3 (3 MONTH DURATION).

The introduction of the Product Owner role had an immediate impact on the development process, fulfilling practices of both Scrum and FDD by creating a focused prioritised list of features for the new software. The PO is an experienced Sports Scientist who had worked closely with the clients and has a very detailed understanding of the clients' needs. The established relationship with the sports clubs meant that clarification on any suggested feature was easily obtained and this removed any delays in development.

Through the coaching of the Scrum Master, the Product Owner has developed a template on the Confluence tool for a detailed prioritised backlog. For the prioritization of the items, we specified the following process: One member of the development team sits down with the PO to detail a new feature through the product requirements template. The template records the goals, assumptions, business value, and user stories necessary to complete the requested feature. The product owner also specifies the Acceptance Criteria at this stage allowing the developer to understand what is necessary for delivery. Once all Users Stories for a feature have been completed, the template is then used to auto generate User Stories to the Development Backlog in Jira. The PO then reviews the backlog and prioritises the User

Stories to be included in the next sprint. Only the stories with a priority of High or Highest are considered for the sprint. Development by feature is now possible as Senior Developers, Stakeholders and the Product Owner has a detailed feature and user story list. At this phase, the sprint duration was changed to a two-week cycle, marking the end of a legacy product maintenance phase and the beginning of development for the company's new product.

Although the stories and tasks were being created, they had no assigned unit of measurement. Estimation using story points following the Fibonacci sequence (1,2,3,5,8,13,21) was chosen. One in the Fibonacci sequence represented a story which required little effort and low risk while 21 represented a very difficult and possibly risky story. All stories were pointed by individual developers at first and they were instructed how to use the outliers to determine a story's estimation. Story point estimation was a difficult concept for the developers to grasp. After four weeks of confusion regarding story points estimation and most developers swinging wildly from over estimation to under estimation, it was decided to revert to effort estimation. Developers felt that they had a better understanding of how long something would take them to complete rather than using the more abstract scale. The developers looked at the tasks and committed to what they felt could completed within 32.5 hours (per week). The Tasks / Stories were time tracked and within two weeks estimations were almost in line with completion rate.

Pair programming was also tried in this phase but, it was unsuccessful due to the small size of the development team. In a development team of five, it was unrealistic to have two developers working on the same piece of functionality. Developers also felt that they lost interest in the feature when they were not coding it themselves. This practice was stopped after two weeks.

The introduction of a Quality Assurance (QA) role allowed functional and acceptance test cases to be written once a feature was detailed in the Confluence tool. The TestRail [28] tool was introduced to record and manage test cases allowing a quality assurance person to create Test Runs for each sprint. The results of the tests can be linked directly to each Jira user story allowing full transparency of bug creation and fixes.

PHASE 4 (4 MONTH DURATION).

By Phase 4, the developers had become comfortable using the project management tools and had embraced all the adoptions made to date. It was time to fully implement Scrum with the introduction of regular structured Sprint Review and Retrospectives meetings. All the Stakeholders, the Product Owner and the Development Team had started attending the bi-weekly Sprint Review meetings where the development progress was demonstrated to all parties. The Sprint Review meeting was also used to clarify upcoming development functions (features and stories) and to adjust backlog prioritisation when necessary. The Retrospective meetings were a chance for the development team to examine their performance during the recent sprint and to identify the positive and negative factors influencing delivery. It also gave the developers an opportunity to refine the process by highlighting practices which they felt did not work in the development environment, pair programming was universally disliked by the team. The Retrospective meetings were very new to the team at this phase, the first couple of meetings produced little feedback as developers struggled to vocalise their experiences within the sprint. The Scrum Master facilitated these sessions and changed the format for each meeting to establish the optimum set up for the team. Novel approaches were employed using games and Lego to get the development team thinking and talking about ways to improve the overall process. "The Retrospective Game"[29] is an engaging fun way to gather feedback from your team, based on the popular game "Cards against Humanity"[30] but tailored for a development environment, it offers Object, Context and Feedback cards. The Object card relates to a random character (the president, an alien), the Context card relates to the development lifecycle (Sprint, Release, Development Team) and the Feedback cards which contain positive and negative statements prompt the players to fill in the blanks. The game creates a relaxed environment in which the team can create hilarious yet relevant feedback on things done well during the sprint and areas in need of improvement. Within the "Legospective" normal retrospective questions where prosed to the developers but instead of vocalising their responses they were asked to visualise and build a Lego construct to represent their views. 15 minutes was allotted for consideration and construction and once completed each developer introduced their creation explaining its significance to the recent sprint. After five consecutive retrospective meetings, the team began to become familiar with the format and started to produce valuable actionable feedback to improve the overall software development approach. The PO role also intensified during this stage with the SM and PO working closely to ensure that weekly backlog grooming was carried out. Regularly refining and reprioritising the backlog allowed the development team to easily plan for up-coming sprints by allowing the creation of sprint backlogs and estimating the delivery of features more accurately.

The development team was now accurately estimating user stories through time tracking as Story Points estimation had proven difficult for them to visualise in the earlier stages of agile adoption. The decision was made to reintroduce estimation by Story Points as it offers a better reflection of the overall effort and value of each story [31]. After two sprints with the introduction of the Planning Poker Game [32], the developers started to feel confident in estimating the user stories based on the Fibonacci sequence.

With comprehensive user stories defined, detailed functional and behavioural requirements for the software and explicit acceptance criteria, the QA tester started to develop functional integration and acceptance tests which run during the sprint. During each test run, a QA can define a bug relating to a particular user story directly to the development board for the developer to fix. The introduction of the TestRail tool added another level of traceability to the development process.

Having created and executed a number of successful Functional and Acceptance Test Suites, the QA could now identify tests to be included in ongoing regression suites. The tests were identified and created which would offer a significant level of code coverage with the fewest test runs. Once full testing processes where in place, the team created their Definition of Done, a user story is Done when it has been coded and unit tested by a developer and has successfully passed all functional, integration, regression and acceptance tests by a QA. Only on successful completion of all tests, a user story is moved to the Done column on the sprint board.

5 Conclusion

There were many challenges involved in adopting an Agile Software Development approach within the Sports Science domain. The domain in which the company operates requires additional documentation and planning to satisfy strict data security regulations while also needing flexibility and agility in development to allow for changing requirements. A number of factors needed to be considered while assessing the development environment, a distributed development team comprising of Hardware, Firmware and Software departments needed a process which would work remotely and suit all three disciplines.

In this paper, we identified the challenges related with the domain and the development environment. Following this, we described how we adopted FDD and Scrum methods in four phases in the organization. A tailored FDD combined with Scrum approach was chosen as the SDLC that suited to the current strategic approach of the company. We had to find a balance that would suit the company and all three departments for this project and future development. We achieved full traceability within the development process which was an essential requirement for the domain.

The phased-based introduction of agile practices was a slow, yet a steady process, offering all stakeholders the opportunity to familiarise themselves with agile practices gradually. Selecting a small number of practices for introduction in each phase allowed us to monitor the process and assess the suitability of each practice within the development environment. In this way, we could identify the practices which were unsuited to the development team and adjust the time spent on each phase according to the complexity of implementation.

Adopting FDD, Scrum practices and the use of project management tools allowed the development team to plan, capture requirements, refine requirements, capture decisions, implement solutions and perform verification and validation in two-week cycles. Adopted approach has improved the communication between all three development teams and offered transparency to all stakeholders. We created an agile culture within the company which offers comprehensive yet lightweight documentation and allows for change while meeting strict development guidelines for safety critical software.

The main contributions of the paper are the presentation of how we tailored the agile practices in the company, the order that we introduced the practices to the teams, the practices that were resisted by the team and the solutions to overcome such challenges.

As future work, we plan to adopt a repository management strategy to ensure the quality and maintainability of the code base. A defined branching model for both local and remote repositories, the introduction of pull requests rather than a code and push policy and a commit process detailing the compulsory comment format will be established.

Repository management roles will be assigned to specific developers within the development team ensuring that each commit is reviewed and inspected before being merged to the development branch for integration and regression testing. Only on successful testing will the development branch be merged to the release ready master branch by a designated repository manager. Our experiences regarding these new adoptions are planned to be shared as well.

Acknowledgement. This research is supported by the Science Foundation Ireland Research Centres Programme, through Lero - the Irish Software Research Centre (http://www.lero.ie) grant 10/CE/I1855 & 13/RC/2094.

References .

1. Brooks, F. P. No silver bullet. *IEEE Computer*, *20*(4), 10-19, 1987.
2. B. Kent *et al.*, "Agile Manifesto," *Agile Manifesto*, 2002. [Online]. Available: http://agilemanifesto.org/principles.html.
3. K. Trektere, F. McCaffery, M. Lepmets, and G. Barry, "Tailoring MDevSPICE® for mobile medical apps," *Proc. Int. Work. Softw. Syst. Process - ICSSP '16*, no. Md, pp. 106–110, 2016.
4. K. Trektere, G. Regan, F. M. Caffery, D. Flood, M. Lepmets, and G. Barry, "Mobile medical app development with a focus on traceability," *J. Softw. Evol. Process*, vol. 29, no. 11, 2017.
5. M. Mchugh, F. McCaffery, and G. Coady, "An Agile Implementation within a Medical Device Software Organisation," *Ccis*, vol. 477, pp. 190–201, 2014.
6. Cockburn, A., Crystal clear: a human-powered methodology for small teams: Addison-Wesley Professional, 2004.
7. Stapleton, J. (1997). *DSDM,* dynamic systems development method: the method in practice. Cambridge University Press.
8. A. Sani and A. Firdaus, "A Review on Software Development Security Engineering using Dynamic System Method (DSDM)," *Int. J. Comput. Appl.*, vol. 69, no. 25, pp. 37–44, 2013.
9. Martin Bauer, "FDD and Project Management | Martin Bauer," *Cutter IT Journal*, 2004. [Online]. Available: http://www.martinbauer.com/Articles/FDD-and-Project-Management. [Accessed: 07-Jun-2018].
10. Beck, K. "Embracing change with extreme programming." Computer 32.10 (1999): 70-77..
11. J. Sutherland and K. Schwaber, "The Scrum Papers : Nuts , Bolts , and Origins of an Agile Process," *Origins*, no. December, pp. 1–202, 2007.
12. Y. Khmelevsky, X. Li, and S. Madnick, "Software development using agile and scrum in distributed teams," in *2017 Annual IEEE International Systems Conference (SysCon)*, 2017.
13. T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, no. 9–10, pp. 833–859, 2008.
14. H. Ayed, B. Vanderose, and N. Habra, "Supported approach for agile methods adaptation: an adoption study," *Proc. 1st Int. Work. Rapid Contin. Softw. Eng. - RCoSE 2014*, no. March 2015, pp. 36–41, 2014.
15. P. Rai and S. Dhir, "Impact of Different Methodologies in Software Development Process," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 2, pp. 1112–1116, 2014.
16. Özcan-Top, Ö., & McCaffery, F. "How Does Scrum Conform to the Regulatory Requirements Defined in MDevSPICE®?." In *International Conference on Software Process Improvement and Capability Determination* (pp. 257-268), 2017. Springer, Cham..
17. M. McHugh, F. McCaffery, and V. Casey, "Software process improvement to assist medical device software development organisations to comply with the amendments to the medical device directive," *IET Softw.*, vol. 6, no. 5, p. 431, 2012.
18. "General Data Protection Regulation (GDPR) – Final text neatly arranged." [Online]. Available: https://gdpr-info.eu/. [Accessed: 28-Mar-2018].
19. "About HIPAA.com – HIPAA.com." [Online]. Available: https://www.hipaa.com/about/. [Accessed: 28-Mar-2018].

20. Lepmets, M., McCaffery, F., and Clarke, P., "Development and Benefits of MDevSPICE, the Medical Device Software Process Assessment Framework". In: Journal of Software: Evolution and Process, Wiley. April 2016. 28: 800–816. doi: 10.1002/smr.1781. Volume 28, Issue 9 September 2016 Pages 800–816.
21. P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis,” *Espoo, Finl. Tech. Res. Cent. Finland, VTT Publ.*, p. 112, 2002.
22. M. E. Moreira, *Being agile: your roadmap to successful adoption of agile*. New York: Apress, 2013.
23. M. T. Dilamani, “A short review on Crystal Clear methodology and its advantages over scrum, the popular software process model,” 2014.
24. S. Goyal, “Major Seminar On Feature Driven Development,” p. 22, 2007.
25. M. Umbreen, J. Abbas, and S. M. Shaheed, “A Comparative Approach for SCRUM and FDD in Agile,” *Int. J. Comput. Sci. Innov.*, vol. 2015, no. 2, pp. 79–87, 2015.
26. Atlassian, “Confluence - Team Collaboration Software | Atlassian.” [Online]. Available: https://www.atlassian.com/software/confluence. [Accessed: 14-Apr-2016].
27. Wake, B. “INVEST in Good Stories, and SMART Tasks,” Posted on August 17, 2003, (http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/).
28. “Test Case Management and Test Management Software Tool - TestRail.” [Online]. Available: http://www.gurock.com/testrail/ [Accessed: 14-Jun-2018].
29. “The Retrospective Game - The Feedback Game That’s Fun and Does Good – theretrospectivegame.” [Online]. Available: https://theretrospectivegame.com/. [Accessed: 10-Feb-2018].
30. “Cards Against Humanity.” [Online]. Available: https://cardsagainsthumanity.com/. [Accessed: 15-Jun-2018].
31. D. Radigan, “Secrets to agile estimation and story points | Atlassian.” [Online]. Available: https://www.atlassian.com/agile/project-management/estimation. [Accessed: 14-Jun-2018].
32. Planning Poker, “PlanningPoker.com - Estimates Made Easy. Sprints Made Simple.” [Online]. Available: https://www.planningpoker.com/. [Accessed: 14-Jun-2018].