The Significance of Requirements in Medical Device Software Development

Martin McHugh[1], Abder-Rahman Ali[2] and Fergal McCaffery[1]

Regulated Software Research Centre, Department of Computing and Mathematics
Dundalk Institute of Technology, Co. Louth Ireland
[1]*(Martin.McHugh, Fergal.McCaffery)@dkit.ie*
[2] *abder.rahman.ali@gmail.com*

Abstract

Software to be used in or as a medical device is subject to user requirements. However, unlike unregulated software, medical device software must meet both the user's requirements and the requirements of the regulatory body of the region into which the software will be marketed. Regulatory requirements are fixed and can be planned for; unfortunately, the same is not true with user requirements. As many medical device software development organisations are following traditional sequential Software Development Life Cycles (SDLC), they are experiencing difficulties accommodating changes in requirements once development has begun. Agile methods and practices offer the ability to overcome the challenges associated with following a sequential SDLC. Whilst the regulatory requirements are fixed, this paper presents these requirements and shows how they appear to mandate the use of a sequential SDLC. This paper also explains how agile methods and practices can be successfully adopted in the development of medical device software without hindering the process of achieving regulatory approval.

Keywords

Agile, medical, safety critical, regulated, requirements, FDA, SDLC

1 Introduction

Software is becoming an increasingly important component of medical devices, as it enables often complex functional changes to be implemented without having to change hardware [1]. Studies in the medical industry point to the fact that software is one of the most critical factors for cutting edge products. It is expected that, by 2015, that the research and development investment in software in this area will increase from 25% of the overall budget in 2002, to 33%. As the role of software in the medical device domain increases, so do the number of failures which arise due to software defects [2].

The subject of software in and as a medical device has become an important topic for the Food and Drug Administration (FDA). This interest began in 1985 when software in a radiation treatment therapy device failed as a result of software defects resulting in the administrating of a lethal overdose radiation. The FDA then analysed recalls by fiscal year (FY) to determine how many were caused by software problems. In FY 1985, for example, 20% of all neurology device recalls were attributable to software problems, while 8% of cardiovascular problems had the same cause [3]. An analysis of medical device recalls by the FDA in 1996 found that software was increasingly responsible for product recalls. A German survey on medical device recalls indicated that software was the top cause for risks related to construction and design defects of medical device products. This analysis, from June 2006, showed that 21% of the medical device design failures were caused by software defects [2]. This was an increasing trend, as the figures from November 2005 showed software was responsible for 17% of con-

struction and design defects. This continues to be the case, and in the period: 1st January 2010 to 1st January 2011, the FDA recorded 80 medical device recalls and stated software as the cause [2]. This type of analysis, along with the results of various corporate inspections, led the FDA to conclude that some type of regulations was required, especially that the agency's review of medical device reporting (MDR) incidents and analysis of product recalls has convinced the agency that software is a factor contributing to practical problems within devices [3].

Since there are many types of software in use by the medical arena, the problem of the best way to regulate it has become an issue to the FDA. Discussions have centred on what type of software is a medical device, the type of regulations required for such software, and what could be inspected under current regulations [3].

Requirements are central to all software development projects. They are used to develop the software and to demonstrate that the software is performing as intended. However, in medical device software, requirements play an extended role. Non-regulated software must meet the requirements of the customer or end user. Regulated software must also meet the requirements of the regulatory bodies also. To accompany this, as part of the regulatory requirements, a medical device software development organisation must be able to trace all stages of development back to the requirements.

2 FDA Stance on Requirements

The FDA regulations impose stringent requirements on the process by which software systems used in medical devices are developed. These requirements translate into various software artefacts that must be made available for the software to be FDA compliant [4] and, for medical device software, the FDA is responsible for assuring that the device utilizing the software is safe and effective [3].

FDA requires medical device manufacturers to submit their device requirements before beginning development. System and software requirements are taken from the FDA medical device quality system regulation [5]. FDA regulations cover all aspects of the medical device product lifecycle, and the FDA requires medical device manufacturers to submit evidence of product safety and efficacy for FDA review and clearance before the manufacturer can market, sell, or distribute the product [6]. Thus, it is critical to obtain information from the FDA on the requirements applicable to the proposed device [7].

Validation compares the final product to the original specifications [8], and is closely related to the requirements specification. You can validate the user's requirements; this is where ambiguity reigns most of the time and where formal methods, through the use of specification languages, have the biggest strides. There is still a wide gap between what the user wants and what the developer understands that the user wants. Very often this is where one of the causes of initial system failures can be found [9]. Software validation is the confirmation that all software requirements have been met and that all software requirements are traceable to the system requirements, provided that it is not possible to validate software without predetermined and documented software requirements [10]. There are two major types of validation that come into play with medical devices - design validation and process validation. Design validation means establishing, by objective evidence, that device specifications conform to the user's needs and the device's intended uses. Process validation, on the other hand, means establishing, by objective evidence, that a process consistently produces the desired result or a product meeting the predetermined specifications [11]. The FDA requires medical device manufacturers to submit their device specifications before beginning development [12]. Thus, validation could come at early stages of development if the user's requirements could be precisely defined, and which from them the rest of the development derived [12]. Ideally, validation work would be accomplished while the requirements are being written [9]. Any safety and regulatory requirements for medical devices necessarily call for rigorous software development methods to ensure reliability and to protect public health. In addition to that, requirements and specifications based on medical practice are needed to help ensure that devices will perform appropriately [6].

The regulatory bodies request that medical device software development organizations clearly demonstrate how they follow a software development life cycle without mandating a particular life cycle [13]. In order to comply with the regulatory requirements of the medical device industry, it is necessary to have clear linkages to traceability from requirements through the different stages of the soft-

ware development and maintenance life cycles. Traceability is central to medical device software development and essential for regulatory approval. Software traceability refers to the ability to describe and follow the life of a requirement in both forward and backward direction [13]. FDA for instance states that traceability analysis must be used to verify that a software design implements all of its specified requirements [14]. Thus, traceability is particularly important for medical device companies, as they have to demonstrate this in order to achieve FDA compliance [15].

2.1 Regulations and Software Development Lifecycles

As discussed, if a medical device software manufacturer wishes to develop software, this manufacturer must adhere to the regulations of the region into which the device is being marketed. These regulations do not mandate a Software Development Life Cycle which must be followed in order to achieve regulatory approval. Initial reading of these regulations and medical device software development standards would appear to imply that software developed for use in medical devices should be developed using a sequential plan driven development lifecycle such as the Waterfall or V-Model.

The FDA Quality System Regulations (QSR) [16] Subpart C – Design Controls provide information as to the processes which must be adhered to when developing regulatory compliant software. These include:

- Design & Development Planning; (Specifications);
- Design Output; (Coding)
- Design Review;
- Design Verification; (Was the Product Built Right);
- Design Validation. (Was the Right Product Built).

As mentioned, initial reading of the QSR would suggest completing these stages sequentially for example in accordance with the Waterfall Model. However, the FDA Design Control Guidance for Medical Device Manufacturers [17] states:

"*Although the waterfall model is a useful tool for introducing design controls, its usefulness in practice is limited… for more complex devices, a concurrent engineering model is more representative of the design processes in use in the industry*"

The FDA General Principles of Software Validation (GPSV) [18] continues to further clarify that it does not mandate the use of a specific SDLC when developing regulatory compliant software:

"*this guidance does not recommend any specific life cycle model or any specific technique or method*"

Furthermore the GPSV acknowledges that activities such as Requirements Specification are likely to be performed iteratively and provides guidance on how these iterative development models can be managed.

"*Most software development models will be iterative. This is likely to result in several versions of both the software requirement specification and the software design specification. All approved versions should be archived and controlled in accordance with established configuration management procedures*"

IEC 62304:2006 [19] is harmonised with the European Medical Device Directive (MDD) [20] and is approved for use by the FDA. IEC 62304:2006 is a software lifecycle model specific to the development of medical device software. As with guidance documents, adherence to IEC 62304:2006 is not mandatory, however, if a manufacturer chooses not to follow it, they would need to provide a sufficient explanation behind not following it. IEC 62304:2006 does not address software development lifecycle models; instead, it defines processes, which consist of activities that should be conducted in each medical device software development project [21]. As with the QSR, initial reading of IEC 62304:2006 would appear to suggest it should be followed in accordance with a sequential lifecycle model such as

Waterfall Model. The publishers of IEC 62304:2006 observed that the standard appeared to mandate following the Waterfall Model and added the following to remove any ambiguity;

"it is easiest to describe the processes in this standard in a sequence, implying a "waterfall" or "once through" life cycle model. However, other life cycles can also be used"

3 Agile Methods to aid with Requirements Management

The rapidly changing business environment in which most organizations operate is challenging traditional Requirements-Engineering (RE) approaches. Software development organizations often must deal with requirements that tend to evolve quickly and become obsolete even before project completion [22]. Agile methods and practices have advantages in accommodating change due to volatile requirements, and are most applicable to projects where requirements are ill-defined and fluid, since they seek to accommodate changes easily [23]. There are different agile practices and methods that can be used in the area of requirements management:

Face-to-face communication over written specifications; effectively transferring ideas from the customer to the development team, rather than creating extensive documentation, where simple techniques (i.e. user stories) are used to define high-level requirements. Here, developers discuss requirements in detail with customers before and/or during development. Thus, customers can steer the project in unanticipated directions, especially when their requirements change owing to changes in the environment or their own understanding of the software solution [22]. All agile approaches emphasize that talking to the customer is the best way to get information needed for development and to avoid misunderstandings. The CHAOS [24] report showed the critical importance of this customer involvement, as it was found to be the number one reason for project success, while the lack of user involvement was the main reason given for projects that ran into difficulties. A key point in all agile approaches is to have the customer 'accessible' or 'on-site'. Agile methods often assume an "ideal" customer representative: the representative can answer all developer questions correctly, and is empowered to make binding decisions and able to make the right decisions [25]. This informal communication with customers obviates the need for time-consuming documentation and approval processes which are perceived unnecessary especially with evolving requirements [22].

Iterative requirements engineering; requirements here aren't predefined, instead, they emerge during development. At each development cycle's start, the customer meets with the development team to provide detailed information on a set of features that must be implemented. And, during this process, requirements are discussed at a greater level of detail. Thus, requirements are clearer and more understandable because of the immediate access to customers and their involvement in the project when needed [22].

Requirements prioritization; agile development implements the highest priority features early so that customers can realize the most business value. The feature lists are prioritized repeatedly during development as the customer's and the developer's understanding of the project evolves, particularly as requirements are added or modified [22]. And, to keep priorities up-to-date, prioritization is repeated frequently during the whole development process [25].

Review meetings; at the end of each development cycle, a meeting with developers, customers, quality assurance personnel, management, and other stakeholders is held for requirements validation. During the meeting, the developers demonstrate the delivered features, provide progress reports to the customers and other stakeholders in the organization, and the customers and QA people ask questions and provide feedback, even though the meetings' original purpose is to review the developed features and get feedback [22].

4 Integrating Agile and Regulatory Requirements

As discussed previously, cursory reading of medical device software standards and regulations appears to advocate utilising a plan driven SDLC that should be followed when developing regulatory compliant software; however, research has shown this not to be the case. Following a plan driven SDLC can prove successful when developing medical device software once the requirements are fully established up-front and there is no risk of change to them. Unfortunately, this is rarely the case and plan driven SDLCs have difficulties accommodating changes. Research has also shown that through the use of iterative development techniques, changes in requirements can be accommodated easier.

While agile methods appear to solve the problems associated with following a plan driven SDLC, how well do agile methods align with the objectives of regulatory bodies? Agile methods appear undisciplined and to advocate producing none of the necessary deliverables; however, this is not the case. The agile manifesto states:

*Individuals and Interactions **over** processes and tools*
*Working software **over** comprehensive documentation*
*Customer collaboration **over** contract negotiation*
*Responding to change **over** following a plan*

It can be seen that statements one and two appear to be contradictory to regulatory requirements, as firstly, the safety of medical device software is determined through the processes followed during the development of the software [26], and secondly, comprehensive documentation is a necessity when seeking regulatory approval. However, as highlighted in the four key principles, agile methods do not dictate that working software instead of comprehensive documentation, nor does it states individuals and interaction instead of processes and tools. The key here is the use of the term **"over"**. For example, Robert Martin, a renowned agilest, clarifies this point further with regards to documentation by stating:

"Produce no documentation unless it is of immediate business value"

In essence, with the development of medical device software, documentation is of business value; therefore it would still be produced while developing software in accordance with agile development methods. Even below the four principles on the agile manifesto [27] website, this is clarified:

"While there is value in the items on the right,
We value items on the left more"

To accompany this, an additional reason cited for not being able to adopt agile methods when developing software, is that prior to development beginning, a medical device software organisation must register the requirements of the device. This being the case, the key benefit of adopting agile i.e. handle changing requirements, become void as there can be no changes allowed. While it is the case that a device's requirements must be registered with regulatory bodies prior to development, regulatory bodies do not require the organisation to register "nuts and bolts" requirements, rather they are concerned with high level requirements.

4.1 Aligning on Goals

Agile software development methods are concerned with developing software using efficient techniques while meeting the needs of the customer. In the case of medical device software, two customers exist, the end user and the regulatory bodies. As a result, agile methods can support regulatory requirements; therefore, agile methods can be supportive of regulatory requirements rather than being contradictory

To accompany this, a key focus of agile development methods is the development of high quality software. Agile methods achieve this by increasing product development productivity and predictability. While regulatory bodies are also concerned about the development of high quality software, they

are not concerned with efficiencies used during the development; however, regulatory bodies do require medical device software organisations to produce objective evidence that the software they have developed performs exactly as described each and every time it is used. This can be achieved through the predictability delivered by agile methods.

4.2 Integration

Previous research has shown that it is not possible to wholly follow a single agile methodology when developing medical device software; however, the same research revealed that combining specific agile practices taken from multiple agile methods and combing them with a plan driven SDLC can be the most advantageous to medical device software organisation. Abbott Diagnostics integrated agile practices with a plan driven SDLC and reported cost savings of between 35% and 50% when compared to a project following a plan driven SDLC. There are a number of instances such as those that report the benefits of integrating agile practices; however, in each of the instances the organisations tailored their own SDLC with agile practices creating a proprietary SDLC. No research exists to date to supply a SDLC which combines agile practices with a plan driven SDLC which can be used by all medical device software organisation. This research will contribute to the development of such a model.

4.2.1 Tailored Software Development Lifecycle

When considering tailoring a SDLC, a foundation plan driven SDLC is required. For this purpose, we chose the V-Model for the following reasons:

- Medical device software organizations typically follow the V-Model to develop medical device software [28]. As a result, they are already familiar with the structure and phases of the V-Model and would be more willing to adopt a hybrid model based upon a SDLC with which they are familiar.
- Medical device software organizations may have received regulatory approval to follow the V-Model when developing medical device software. If these organizations move to a completely different SDLC, they may need to re-apply for regulatory approval for the new SDLC. This may be a barrier as organizations could be reluctant to undergo regulatory approval again.
- Whilst none of the regulatory requirements or development standards mandate the use of the V-Model, it appears to be the best fit with regulatory requirements, as it guides organizations through the process of producing the necessary deliverables required to achieve regulatory conformance.

Once the foundation model was chosen, an analysis of this model was performed to determine where agile practices could be integrated to overcome the problems associated with following a plan driven SDLC. A number of stages remain single pass stages in the tailored model. These stages include *"Requirements Specification*", *"System Testing"* and *"Acceptance Tests".* These stages must remain single pass stages to remain in line with regulatory requirements. The remaining stages "*System Analysis", "Software Architecture Design", "Software Detailed Design", "Software Unit Implementa-*

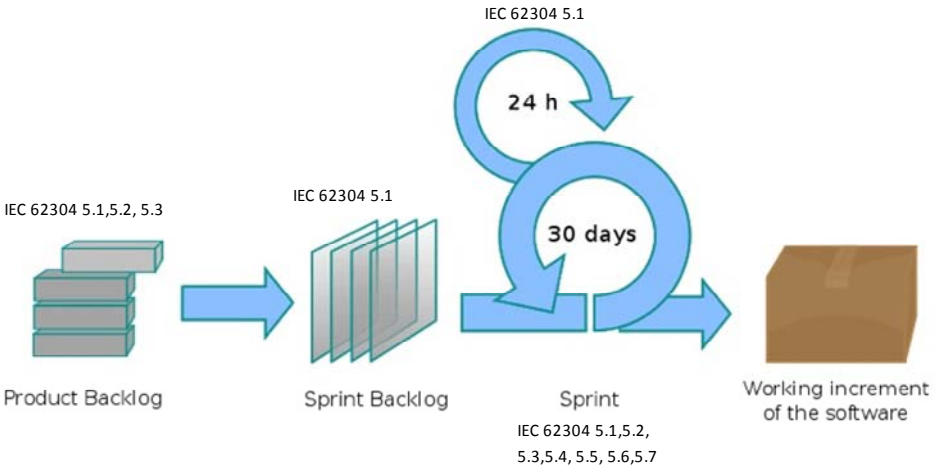

Figure 1 IEC 62304 Mapping to Scrum Lifecycle

tion", "Software Unit Test" and *"Software Integration and Integration Testing"* are integrated into a Scrum SDLC [29] (see figure 1).

IEC 62304 5.1 – Software Development Planning
IEC 62304 5.2 – Software Requirements Analysis
IEC 62304 5.3 – Software Architectural Design
IEC 62304 5.4 – Software Detailed Design
IEC 62304 5.5 – Software Unit & Implementation
IEC 62304 5.6 – Software Integration & Integration Testing
IEC 62304 5.7 – Software System Testing

A mapping study was performed in accordance with [30] which identified instances of where agile methods have been adopted in the development of medical device software. The mapping study identified 10 instances of where agile methods have been successfully used over the period of 2002 to 2012. Of these the majority of the organisations involved adopted a Scrum approach with their traditional plan driven SDLC. In figure 1 the relevant processes, in accordance with IEC 62304, are mapped to specific stages of the Scrum Lifecycle.

5 Conclusions

Regulatory bodies place a large emphasis on requirements when developing medical device software. These requirements are used to achieve traceability and to determine if the medical device software is performing as intended. In an ideal scenario, prior to the development of medical device software, all of the stakeholders in a development team could agree and sign off on the device requirements. Once these requirements are agreed, a medical device software development team could adopt a sequential plan driven SDLC to develop the software effectively. Unfortunately, the ideal scenario rarely exists, and at times, changes in requirements are unavoidable, and if a development team has begun to develope in accordance with a plan driven SDLC, they can experience great difficulties when introducting a change.

Agile methods appear to offer the silver bullet to the problem of changing requirements in development project. As a project is broken into iterations, a change can be introduced into the iteration cycle easier than compared to a plan driven SDLC. However, while agile methods appear to be the silver bullet, there remains reluctance amongst medical device software organisations to adopt them. Research has shown that where agile practices have been adopted, they have proved successful. Where they have been successfuly introduced, they have been intergrated with the existing plan driven SDLC resulting in a scenario where the organisation can rely on the stability of following a plan driven SDLC whilst reaping the benefits of agile methods such as accomodatng changes.

This paper maps stages of medical device software development to a Scrum development lifecycle. This mapping can be used by any organisation wishing to develop medical device software in accordance with agile methods whilst remaining compliant with regualtory controls.

6 References

[1] F. McCaffery, V. Casey, M. Sivakumar, G. Coleman, P. Donnelly, and J. Burton, "Medical Device Software Traceability," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., ed: Springer-Verlag, 2012.

[2] C. Denger, R. L. Feldman, M. Host, C. Lindholm, and F. Schull, "A Snapshot of the State of Practice in Software Development for Medical Devices," presented at the First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007, Madrid, 2007.

[3] R. C. Fries, *Reliable Design of Medical Devices* vol. 3rd ed. Boca Raton FL: CRC, 2012.

[4] H. Mehrfard, H. Pirzadeh, and A. Hamou-Lhadj, "Investigating the Capability of Agile Processes to Support Life-Science Regulations: The Case of XP and FDA Regulations with a Focus on Human Factor Requirements," presented at the In Proceedings of SERA (selected papers), 2010.

[5] F. Pavese and A. B. Forbes, *Data Modeling for Metrology and Testing in Measurement Science*: Birkhäuser, 2009.
[6] T. H. Faris, *Safe And Sound Software: Creating an Efficient And Effective Quality System for Software Medical Device Organizations*: Asq Press, 2006.
[7] J. K. Shapiro, "The Pathway to Market for your Medical Device: A Primer on Obtaining Information from FDA " *FDLI,* vol. Update May/June, 2008.
[8] M. S. Sivakumar, V. Casey, F. McCaffery, and G. Colema, "Verification & Validation in Medi SPICE," presented at the The 11th International SPICE Conference Process Improvement and Capability dEtermination, Dublin, 2011.
[9] J. O. Grady, *System Requirements Analysis* Amsterdam: Elsevier Academic, 2006.
[10] D. Farb and B. Gordon, *Pharmaceutical Computer Validation Introduction Guidebook*: UniversityOfHealthCare, 2005.
[11] C. T. DeMarco, *Medical Device Design and Regulation*: Asq Press, 2011.
[12] A. Dasso and A. Funes, *Verification, Validation and Testing in Software Engineering*: Idea Group Pub., 2007.
[13] V. Casey and F. McCaffery, "Med-Trace: Traceability Assessment Method for Medical Device Software Development.," presented at the European Systems & Software Process Improvement and Innovation Conference, (EuroSPI). Roskilde, Denmark, 2011.
[14] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," presented at the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, Cape Town, South Africa, 2010.
[15] F. McCaffery and G. Coleman, "The need for a software process improvement model for the Medical Device Industry," *International Review on Computers and Software,* vol. 2, pp. 10-15, 2007.
[16] FDA, "Title 21--Food and Drugs Chapter I --Food and Drug Administration Department of Health and Human Services subchapter h--Medical Devices part 820 Quality System Regulation," ed: *U.S. Department of Health and Human Services*, 2007.
[17] *FDA Design Control Guidance for Medical Device Manufacturers*, 1997.
[18] FDA, "General Principles of Software Validation: Final Guidance for Industry and FDA Staff," ed: *Centre for Devices and Radiological Health*, 2002.
[19] AAMI, "ANSI/AAMI/IEC 62304, Medical device Software - Software life cycle processes," ed. Association for the Advancement of Medical Instrumentation, 2006.
[20] *Directive 2007/47/EC of the European Parliament and of the Council of 5 September 2007*, 2007.
[21] M. McHugh, F. McCaffery, and V. Casey, "Standalone Software as an Active Medical Device " presented at the The 11th International SPICE Conference Process Improvement and Capability dEtermination, Dublin, 2011.
[22] C. Lan and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study," *Software, IEEE,* vol. 25, pp. 60-67, 2008.
[23] M. Coram and S. Bohner, "The impact of agile methods on software project management," in *Engineering of Computer-Based Systems, 2005. ECBS '05. 12th IEEE International Conference and Workshops on the*, 2005, pp. 363-370.
[24] Standish Group, "Chaos Report," ed, 1995.
[25] F. Paetsch, A. Eberlein, and F. Maurer, "Requirements engineering and agile software development," presented at the Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on, 2003.
[26] P. L. Jones, J. Jorgens, A. R. T. Jr, and M. Weber, "Risk Management in the Design of Medical Device Software Systems," *Biomedical Instrumentation & Technology: July 2002,* vol. 36, pp. 237-266, 2002.
[27] (2001). *Manifesto for Agile Software.* Available: http://agilemanifesto.org/
[28] M. McHugh, F. McCaffery, and V. Casey, "Barriers to using Agile Software Development Practices within the Medical Device Industry," in *European Systems and Software Process Improvement and Innovation Conference, EuroSPI* Vienna Austria, 2012.
[29] M. Cohn, *Succeeding with Agile - Software Development Using Scrum*. Upper Saddle River NJ: Addison Wesley, 2011.
[30] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), University of Bari, Italy, 2008.

7 Author CVs

Martin Mc Hugh

Martin received his B.Sc. (Hons.) in Information Technology Management in 2005 and M.Sc. in Computer Science in 2009, from Dundalk Institute of Technology. He is now undertaking research for his Ph.D. in the area of software process improvement for medical devices with emphasis on the usage of agile

practices when developing medical device software, as part of the Regulated Software Research Group in Dundalk Institute of Technology.

Abder-Rahman Ali

Abder-Rahman received his BSc in Computer Science in 2006 from the University of Jordan and MSc Software Engineering in 2009 from DePaul University. He is interested in applying technology to medicine, and in building computer aided diagnosis systems that aid in diagnosing disease. He is currently pursuing his Ph.D. degree at Université d'Auvergne in France, as part of the ISIT lab, in the area of fuzzy clustering and discrete geometry for image analysis of MRI and ultrasound imaging sequences for Hepatocellular Carcinoma (HCC).

Fergal Mc Caffery

Dr Fergal Mc Caffery is the leader of the Regulated Software Research Centre in Dundalk Institute of Technology and a member of Lero. He has been awarded Science Foundation Ireland funding through the Stokes Lectureship, Principal Investigator and CSET funding Programmes to research the area of software process improvement for the medical device domain. Additionally, he has received EU FP7 and Enterprise Ireland Commercialisation research funding to improve the effectiveness of embedded software development environments for the medical device industry.