Characteristics of a medical device software development framework

Paul Clarke[1], Marion Lepmets[1], Fergal McCaffery[1], Anita Finnegan[1], Alec Dorling[1], Sherman Eagles[2]
[1] Regulated Software Research Centre, Dundalk Institute of Technology, Dundalk, Ireland
[2] SoftwareCPR, USA
{paul.clarke; marion.lepmets; fergal.mccaffery; anita.finnegan; alec.dorling} @dkit.ie
and seagles@softwarecpr.com

Abstract
This paper aims to describe the software development settings of medical device domain focusing on the demands of the safety critical software processes. Medical device software developers have to adhere to a number of regulations and standards. This paper addresses the most important characteristics of a software development framework that could support medical device software developers in their efforts to comply with these regulations as well as to improve their software development processes.

Keywords

Process model, medical device software development, safety critical software process

1 Introduction

Software engineering has been around as an academic field and as an industry domain long enough to not be called novel anymore. It has been taught in universities for several decades. As with any domain that has moved from innovation to commonplace, the research questions have changed along with it. Software has become central to how we live and for many other domains whose knowledge is built on top of software engineering applications. The critical questions for software engineering have now shifted from the fundamental issues of how to develop software into an easier adoption and automation, adjustment and tailoring of these software development tasks. Software development in safety critical domains is one of these critical questions as an increasing amount of software is being embedded to medical devices, cars and airplanes.

A stable body of knowledge for software engineering exists in the world today which describes how to design and develop software. Two main philosophies of software development have emerged: *prescriptive development* and *agile development* [1]. Prescriptive development along with prescriptive process models are often associated with the development of detailed process definitions, followed by the application of process activities and tasks in accordance with the process definition. The intention of prescriptive process models is to improve the product quality by reducing the number of errors that are made and by supporting the achievement of delivery dates, budget constraints. In the case of medical device software development, the primary goal is to create safe and effective medical devices. Agile development, on the other hand, seeks to reduce the levels of bureaucracy associated with prescriptive process models and promote project agility. Increased agility allows development to respond rapidly to changing requirements and relies more on human-centric skills; thus empowering individuals to make decisions that best support emerging demands as opposed to strictly following an extensively defined and heavily prescribed process. As to which of these two philosophies is best (prescriptive or agile), Pressman points out that although an emotional debate has raged, it is really a question of trade-offs [1]. Or put another way, the selection of a software process philosophy (and indeed a software process itself) must first consider the benefits bestowed by different approaches and thereafter, identify the process characteristics that best service the demands of the given software development domain/environment [2].

This paper aims to describe how the generic software development approaches fit to the software development in highly regulated medical device domain. The next section describes prescriptive process models and their importance in medical device domain. The focus then shifts to the medical device software development - the medical device regulations and standards, and the current medical device software development practices. The authors then describe the demands of and appropriate approaches to medical device software development that lead to the characteristics that a medical device software development process model should have. Finally, the paper presents some concluding remarks and the future works.

2 Overview of Prescriptive Process Models

Quality Management System (QMS) standards provide various prescriptive process models that have long been established in software development (and earlier in other industrial sectors) as an approach to process management. A QMS is essentially concerned with the design and management of a suite of processes that support the achievement of consistent levels of quality in the delivery of products and services. Typically, a QMS standard will have an associated audit system whereby the performance of the QMS can be evaluated. The outcome of a QMS audit is either pass or failure, with instances of failure having an associated list of non-conformances with the standard. Audits can be conducted either internally by organisational personnel or by an external party. When the outcome of a third party audit is successful, it is generally possible to obtain official certification of compliance with the QMS. Perhaps the most widely adopted QMS is the ISO 9001 standard [3], which is applied in software development settings via the guidance provided in ISO 90003 [4]. Although ISO 9001 can theoretically be implemented in any software development setting, it has been suggested that the benefits are greater for large rather than small organisations [5]. It has also been reported that ISO 90003 may not be sufficiently rich in software development know-how [6]. Despite the guidance for performance improvement provided in ISO 9004, the primary aim of QMS standards like the ISO 9000

series is to evaluate the organizational conformance necessary for regulation – appropriate in the medical device sector because of the need to have consistent and equal application of the pertinent legislation. Continuous improvement of processes is the cornerstone of alternative prescriptive models that are referred to in this paper as the Capability Maturity Frameworks (CMFs). The underlying notion is that when the processes with higher capability are applied in the organization they will also advance the overall organizational maturity [7].
CMFs accept that process implementations vary greatly, ranging from complete disorganisation to extensive process implementation and management. CMFs provide a roadmap for process improvement that is based on the extensive experience of a large number of previous implementations over many years. It is also possible to undertake an assessment of process implementation using the infrastructure of a CMF. However, the outcome of a CMF-related assessment is different to the outcome of a QMS audit, in that it is concerned with the identification of the process capability when measured against established best practice for a process (as opposed to the pass/failure scenario associated with QMS audits).
Two of the best known and most widely adopted software development CMFs are ISO/IEC 15504 [8] and CMMI-DEV [9]. Software development CMFs address the need for process improvement in software development settings, and are noted to provide benefits, such as a reduction in the cost of quality [10], improved customer satisfaction and project performance [11]. However, software development CMFs are not entirely without criticism. As with QMSs, it has been noted that CMFs may be more challenging to implement in smaller software development companies, which are often confronted with customer pressures and a general lack of time and resources [12].
To date, a number of different software development standards and guidance documents have been developed for use in the medical device sector. These contemporary standards are presented in the following section, along with a brief outline of the medical device regulations. Thereafter, we examine the suitability of the two different software development philosophies (prescriptive process models and agile process models) to medical device software development.

3 Background to Medical Device Software Development

The medical device industry is focusing increasingly on software quality as more and more software is integrated into medical devices. According to data from the U.S. FDA [13], “software failures were behind 24% of all the medical device recalls in 2011” resulting in “gearing up the FDA labs to spend more time analysing the quality and security of software-based medical instruments and equipment.” Although the domain is heavily controlled by regulations, the regulation itself is satisfied in practice through the implementation of appropriate process and quality standards. Therefore, it is critically important that the medical device software process and quality standards, presented in Figure 1, adopt the expertise accumulated in the generic, best practice software process standards (that have proven to be the foundation of developing high quality software).

3.1 Medical Device Regulations

A medical device can consist entirely of software or have software as a component of the overall medical device system [14]. In order to be able to market a medical device within a particular region it is necessary to comply with the regulatory demands of that region (Figure 1). Two of the largest global bodies responsible for issuing and managing medical device regulation belong to the central governing functions of the US and EU.
In the case of the US, the Food and Drug Administration (FDA) issues the pertinent regulation through a series of official channels, including the Code of Federal Regulation (CFR) Title 21, Chapter I, Subchapter H, Part 820 [15]. Under US regulation, there are three medical device safety classifications: Class I, Class II and Class III. The medical device safety classification is based first and foremost on the clinical safety of the device. Class I devices are not intended to support or sustain human life, and may not present an unreasonable risk of harm. Examination gloves are an example of a Class I medical device. Class II medical devices are those devices for which Class I general controls alone cannot assure human safety and effectiveness. Class II devices could cause damage or harm to humans. An example of a Class II medical device is a powered wheelchair. Class III medical devices

are usually those that support or sustain human life, and are of significant importance in the prevention of human health impairment. An example of a Class III device is an implantable pacemaker. All implantable devices are Class III medical devices as the surgery required carries additional high risks from anaesthesia and possible infections that go beyond the technical and engineering safety risks of the correct performance of the device.

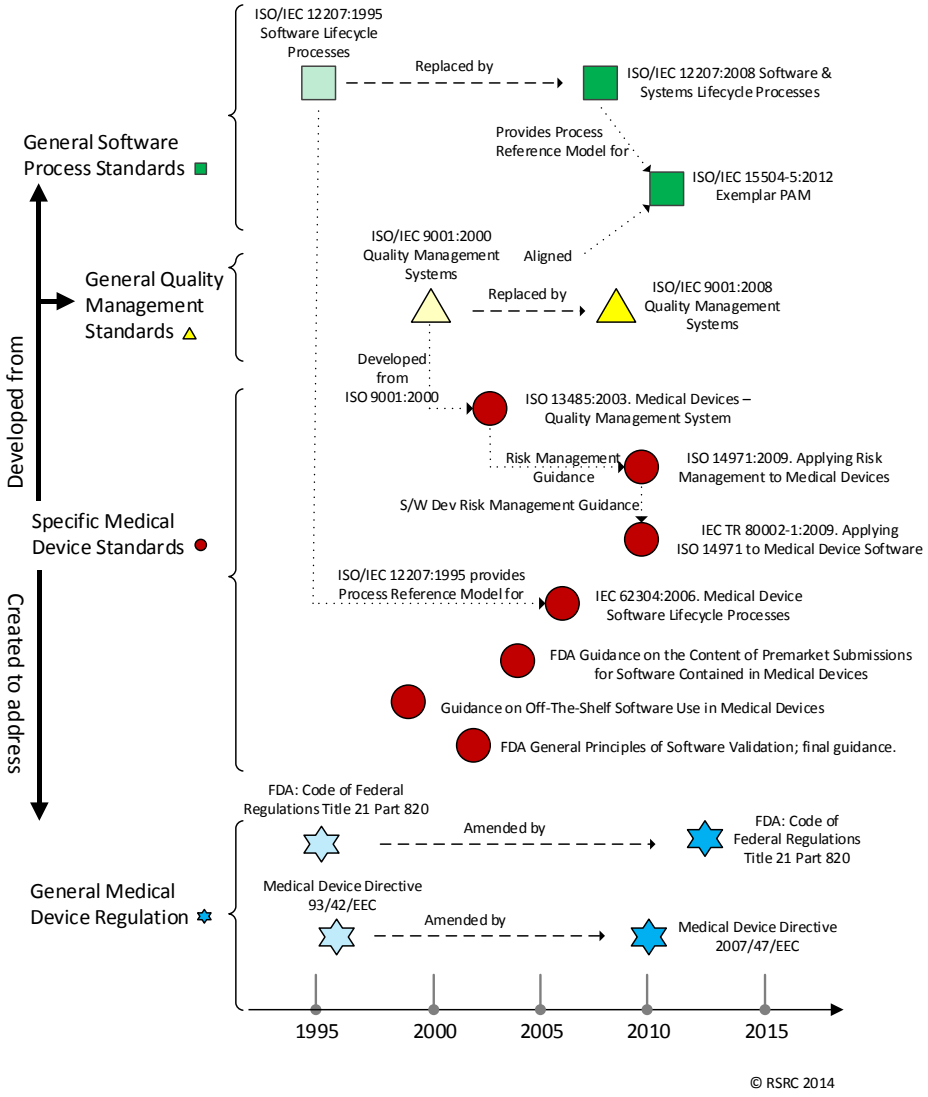

Figure 1. Medical Device Standards and Regulation

In the EU, the corresponding regulation is outlined in the general Medical Device Directive (MDD) 93/42/EEC [14], the Active Implantable Medical Device Directive (AIMDD) 90/385/EEC [16], and the *In-vitro* Diagnostic (IVD) Medical Device Directive 98/79/EC [17] - all three of which have been amended by 2007/47/EC [18]. Although slightly different to the US safety classifications that are based on clinical safety of the device, the EU classifications essentially embody similar classifications and limitations, where Class I corresponds to Class I, Class IIa and IIb to Class II, and Class III to Class III. A further safety classification applies to the software in the medical device as outlined in IEC 62304, wherein the safety classification is concerned with the worst possible consequence in the case of a software failure (as compared with general medical device safety classification which is based on the difficulty of a regulator to determine if the device will be safe). Hence, some Class II medical devices can cause serious injury or even death, but they are Class II because they are similar (in clinical use and safety) to well understood devices that have been used before. Since IEC 62304 safety

classifications are based on worse case failure of the software, it is possible that Class II medical devices can have Class III software.
In the medical device domain, ISO 13485:2003 (ISO 13485 from hereon) [19] outlines the requirements for regulatory purposes from a QMS perspective. ISO 13485, which is based on ISO 9001 [3], can be used to assess an organisation's ability to meet both customer and regulatory requirements. However, ISO 13485 does not offer specific guidance on software development.
IEC 62304:2006 (IEC 62304 from hereon) [20], which can be used in conjunction with ISO 13485, does offer a framework for the lifecycle processes necessary for the safe design and maintenance of medical device software. As a basic foundation, IEC 62304 assumes that medical device software is developed and maintained within a QMS such as ISO 13485, but does not require an organisation to be certified in ISO 13485. Therefore, IEC 62304 can be considered to be a software development specific supplement to ISO 13485. IEC 62304 is based on ISO/IEC 12207:1995 [21] which although a comprehensive standard for software development lifecycle processes has effectively been decommissioned following the publication of the more extensive ISO/IEC 12207:2008 [22]. Furthermore, other developments in the ISO and IEC communities for software development, such as ISO/IEC 15504, have provided significant additional levels of software process detail to support ISO/IEC 12207:2008. IEC 62304 is currently being revised to better align with ISO/IEC 12207:2008 (Figure 1). IEC 62304 is a critical standard for medical device software developers as it is the only standard that provides recommendations for medical device software implementations based on the worse consequences in the case where the software failures lead to hazards.
Furthermore, for general medical device risk management, IEC 62304 is used in conjunction with ISO 14971 [23], with IEC 80002-1 [24] providing guidance on the application of ISO 14971 for software development. Additionally, as IEC 62304 considers a medical device system to consist of software as part of an overall medical device system, the system level requirements are not included within IEC 62304 but instead within the medical device product standard IEC 60601-1 [25]. Also it should be noted that due to the increasing importance of usability within the medical device industry organisations should also adhere to the medical device usability requirements outlined in IEC 62366 [26].

3.2 Alignment between general software development and medical device standards

One of the more obvious examples of the gap that has emerged between the general software development standards and IEC 62304 (incl. ISO 14971 and IEC 80002-1) is the inconsistency in the use of language and terminology. For example, the Risk Management process that is present in ISO/IEC 12207:2008 (as opposed to ISO/IEC 12207:1995 upon which IEC 62304 is based) is concerned with project-level risks. In effect it is a project level process aimed at identifying and controlling general project risks of budget and schedule. However, the Risk Management process in IEC 62304 is concerned largely with product safety issues (i.e. addressing only negative outcomes) and how these might be reduced through robust process implementation throughout the entire software development lifecycle. Given that the medical device sector, and its many related standards, tends to term safety engineering as risk management, it is appropriate that IEC 62304 should adopt this language. In contrast, process safety issues are dealt with separately in ISO/IEC 15504 (in the Part 10 extension for Safety Critical software development [27]). This has resulted in different process related concepts for medical device software development as compared with generic software development – the Risk Management process. Many additional gaps also exist, and these extend beyond language and terminology, permeating the very architecture and design of the standards themselves. The major difference between these two standards is based both on their different design and purpose.
Figure 1 above presents numerous different medical device standards and regulations that exist today, some of which are interlinked and others which are inconsistent. Although the border between these two domains is potentially difficult to navigate (medical device development is focused on product safety management while general software development has a broader software development mandate), there are some shortcomings in the presently available approaches.
The dominant medical device software standards such as IEC 62304 are not yet aligned with the approach adopted in the general software development standards community since the 1995 publication of ISO/IEC 12207. One significant change in this respect has been the introduction of a

harmonised approach to process description (as defined in ISO/IEC 24774 [28]) which involves the identification of core process outcomes that can later be harnessed to develop a process assessment method. A further significant change relates to the movement in the general software development standards community (and in other safety-related domains) to include a software process improvement dimension that can be instrumental in guiding software development organisations towards the required process targets. In effect, the medical device standards have not kept up with the changes that have been made to the general software development standards. There are several reasons why the medical device standards lag the updates to the general software development standards, (perhaps) most importantly the IEC stability period during which adopted harmonised standards are not to be changed unless the proposed changes are necessary in terms of safety. With the increasing use of software in medical devices, there is a case to be made for introducing the accumulated up-to-date wisdom in the general software development standards into the medical device software development specific standards in a uniform fashion – and work in this direction should not wait for the IEC stability period to come to an end, but rather proceed in the interim period (such as the work reported upon in this paper).
In order to identify an appropriate architecture for introducing the significant body of general software process knowledge into the medical device process domain, an initial important step involves an evaluation of the general software development methodology to fit with the specific demands of medical device software development. The next section outlines the results of an evaluation of the general software development methodology with the specific demands of medical device software development.

4 Characteristics of a medical device software development framework

The primary observation in relation to the identification of the demands of medical device software development is that a large degree of variation is evident from a development process perspective. Although agile software development approaches are increasingly adopted in industry, the medical device software development still needs to comply with several standards indicating the need for a more disciplined software development approach. This variation in the demands presents a significant challenge to any general framework for medical device software development. Any such framework should be capable of supporting both an agile software development philosophy while also addressing the very high levels of process rigour associated with the more disciplined software development process philosophies [29]. Since these opposing software development philosophies are essentially discordant, it is not surprising that a general framework supporting both philosophies does not presently exist. In developing the medical device software framework, it is useful to first identify the characteristics that such a framework should ideally exhibit:

Characteristic 1: The framework should support the development of software for all medical device safety classifications.

Characteristic 2: The framework should offer greater levels of detail on the implementation of software development processes than presently exists in the dominant medical device software development standards.

Characteristic 3: The framework should identify a roadmap that companies can follow in order to implement the process improvements required in order to progress towards both regulatory compliance and best practice.

The three characteristics highlighted above can be summarised as follows: a general framework for medical device software development should be capable of supporting a spectrum of process implementation, it should offer a high level of detail regarding software process implementation, and it should facilitate software process improvement. A purely agile software development methodology would be unsuited to these characteristics since without adaptation; it can lack a significant investment in up-front requirements elicitation and formal documentation (such as is required for Class C medical device software). Equally, a wholly prescriptive process approach would also be unsuitable for the identified characteristics, since the levels of bureaucracy associated with such approaches would not be ideally aligned with the needs of Class A medical device software development. Therefore, an approach should be designed that supports agility for some types of development, while also providing a prescriptive process for other types of development. Considering that a spectrum of process

implementations is required in order to satisfy the characteristics identified above, it is proposed herein that of the contemporary general software development approaches, a CMF offers the most natural fit. However, some caution should be applied to this judgment as CMFs are more challenging to implement in smaller companies [12, 30-34]. Plus, rapid product innovation may be an important survival characteristic for Class A medical device software developers. Therefore, we must examine if a CMF exists that supports the needs of medical device software development – and if such a CMF does exist, can it be adapted for use in organisations that are engaged in the development of highly innovative, though regulated, Class A medical device software. One possibility would be to adapt the ISO/IEC 15504 CMF (and the underlying ISO/IEC 12207:2008 process activities and tasks) to harmonise with the explicit requirements of IEC 62304 – an approach which (owing to the large amount of descriptive text required) the authors intend to address in a separate publication. A further adaptation of this framework could include the regulatory requirements from FDA and QMS that the medical device products need to be compliant with prior to be placed on the market.

5 Conclusions and Future Works

Medical device domain is heavily controlled by regulations. Regulations are satisfied in practice through the implementation of appropriate process and quality standards. It is therefore critically important that the medical device software process and quality standards adopt the expertise accumulated in the generic, best practice software process standards (that have proven to be the foundation of developing high quality software). As outlined in this paper, this is not presently the case in the medical device software development sector – with IEC 62304 being based on the now withdrawn ISO/IEC 12207:1995. The result is that medical device software development standards are no longer up to date with the acknowledged best practice or process definitions set forth in the international standardisation community (particularly with respect to ISO/IEC 27447, ISO/IEC 12207:2008 and ISO/IEC 15504).
The goal of the work presented in this paper was to describe the landscape of software development in medical device domain focusing on the demands of safety critical software processes. There is a myriad of regulations and standards that need to be applied in medical device software development and the authors presented the most important characteristics of a capability maturity framework that could support medical device software developers in their efforts to adhere to these regulations as well as improving their software development processes.
The work is currently underway integrating the generic software development process requirements with the specific medical device regulatory requirements and guidelines into a comprehensive medical device software development capability maturity framework.

Acknowledgment. This research is supported by Enterprise Ireland and the European Regional Development Fund (ERDF) under the National Strategic Reference Framework (NSRF) 2007-2013, grant number CF/2012/2631, and in part by the Science Foundation Ireland (SFI) Stokes Lectureship Programme, grant number 07/SK/I1299, and the SFI Principal Investigator Programme, grant number 08/IN.1/I2030 (the funding of this project was awarded by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund).

6 Literature

1. Pressman, R., *Software Engineering - A Practitioner's Approach*. 2005, Boston, MA: McGraw-Hill.
2. Dyba, T., *Contextualizing Empirical Evidence.* IEEE Software, 2013. **30**(1): p. 81-83.
3. ISO, *ISO 9001:2000 - Quality Management Systems - Requirements*. 2000: Geneva, Switzerland.
4. ISO, *ISO 90003:2004 - Software Engineering - Guidelines for the Application of ISO 9001:2000 to Computer Software*. 2004: Geneva, Switzerland.
5. Stelzer, D., W. Mellis, and G. Herzwurm, *A critical look at ISO 9000 for software quality management.* Software Quality Journal, 1997. **6**(2): p. 65-79.
6. Yang, Y.H., *Software Quality Management and ISO-9000 implementation.* Industrial Management & Data Systems, 2001. **101**(7): p. 329-338.
7. Humphrey, W.S., *A Discipline for Software Engineering*. 1994: Addison-Wesley.

8. ISO/IEC, *ISO/IEC 15504: Information Technology - Process Assessment*. 2005: Geneva, Switzerland.
9. SEI, *CMMI® for Development, Version 1.3*. 2010, Software Engineering Institute.
10.Gibson, D., D. Goldenson, and K. Kost, *Performance results of CMMI-based process improvement*. 2006, Software Engineering Institute, Carnegie Mellon University: Pittsburg, PA, USA.
11.Wegelius, H. and M. Johansson. *Practical experiences on using SPICE for SPI in an insurance company*. in *EuroSPI 2007 Industrial Proceedings*. 2007.
12.Laporte, C.Y., et al., *Initiating software process improvement in small enterprises: Experiments with micro-evaluation framework.* Proceedings of the International Conference on Software Development, 2005: p. 153-163.
13.FDA. *FDA News on Software Failures Responsible for 24% of all Medical Device Recalls*. 2012 [cited 2013 12.04]; Available from: http://www.fdanews.com/newsletter/article?articleId=147391&issueId=15890.
14.European Commission, *Directive 93/42/EEC of the European Parliament and of the Council concerning medical devices*, in *OJ o L 247 of 2007-09-21*. 1993: European Commission, Brussels, Belgium.
15.FDA. *Chapter I - Food and drug administration, department of health and human services subchapter H - Medical devices, Part 820 - Quality system regulation*. [cited 2013 15.05]; Available from: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820.
16.European Commission, *Council directive 90/385/EEC on active implantable medical devices (AIMDD)*. 1990: Brussels, Belgium.
17.European Commission, *Directive 98/79/EC of the european parliament and of the council of 27 october 1998 on in vitro diagnostic medical devices*. 1998: Brussels, Belgium.
18.European Commission, *Directive 2007/47/EC of the European Parliament and of the Council concerning medical devices*, in *OJ no L 247 of 2007-09-21*. 2007, EC: Brussels, Belgium.
19.ISO, *ISO 13485: Medical Devices - Quality Management Systems - Requirements for Regulatory Purposes*. 2003, ISO: Geneva, Switzerland.
20.IEC, *IEC 62304: Medical Device Software - Software Life-Cycle Processes*. 2006, IEC: Geneva, Switzerland.
21.ISO/IEC, *ISO/IEC 12207:1995 - Information Technology - Software Life-Cycle Processes*. 1995, ISO/IEC: Geneva, Switzerland.
22.ISO/IEC, *ISO/IEC 12207:2008 - Systems and Software Engineering - Software life cycle processes*. 2008, ISO/IEC: Geneva, Switzerland.
23.ISO, *ISO 14971 - Medical Devices - Application of Risk Management to Medical Devices* 2009, ISO: Geneva, Switzerland.
24.IEC, *IEC TR 80002-1 - Medical Device Software - Part 1: Guidance on the Application of ISO 14971 to Medical Device Software*. 2009, IEC: Geneva, Switzerland.
25.IEC, *IEC 60601-1 - Medical electrical equipment – Part 1: General requirements for basic safety and essential performance* 2005, IEC: Geneva, Switzerland.
26.IEC, *IEC 62366 - Medical devices - Application of usability engineering to medical devices*. 2007, IEC: Geneva, Switzerland.
27.ISO/IEC, *ISO/IEC 15504-10 - Information Technology - Process Assessment - Part 10: Safety Extension*. 2011, ISO/IEC: Geneva, Switzerland.
28.ISO/IEC, *ISO/IEC 24774 - Systems and Software Engineering - Life Cycle Management - Guidelines for Process Description*. 2010: Geneva, Switzerland.
29.M. Mc Hugh, et al., *Balancing Agility and Discipline in a Medical Device Software Organisation*, in *13th International SPICE Conference on Process Improvement and Capability dEtermination*. 2013, Springer: Bremen, Germany.
30.El Emam, K. and A. Birk, *Validating the ISO/IEC 15504 Measure of Software Requirements Analysis Process Capability.* IEEE Transactions on Software Engineering, 2000. **26**(6): p. 541-566.
31.Jung, H.-W., et al., *Findings from Phase 2 of the SPICE trials.* Software Process: Improvement and Practice, 2002. **Vol.6**(no 4): p. 205-242.
32.Staples, M., et al., *An exploratory study on why organizations do not adopt CMMI* Journal of Systems and Software, 2007. **80**(6): p. 883-895.
33.Saastamoinen, I. and M. Tukiainen. *Software process improvement in small and medium sized software enterprises in Eastern Finland: A state-of-the-practice study*. in *Proceedings of the 11th European Conference on Software Process Improvement*. 2004.
34.Khurshid, N., P. Bannerman, and M. Staples. *Overcoming the first hurdle: Why organizations do not adopt CMMI*. in *Proceedings of the International Conference on Software Process*.

Author CVs

Paul Clarke

Dr. Paul Clarke is a Research Manager in the Regulated Software Research Centre based at DkIT and a Research Fellow with Lero – the Irish Software Engineering Research Centre. Primary among Paul's present research interests is the establishment of a best practice framework for medical device software development.

Marion Lepmets

Dr. Marion Lepmets is a Senior Research Fellow at the Regulated Software Research Centre in Dundalk Institute of Technology and a member of the Irish Software Engineering Research Centre (Lero). Her PhD and post-doc research focused on the impact of process model implementation and of process assessment on process improvement success, respectively. Her current research is about the development of the process reference and process assessment models for medical device software development. She is also involved in the development of software engineering standards in the International Standardization Organization (ISO/IEC JTC1 SC7).

Fergal McCaffery

Dr Fergal Mc Caffery is a Science Foundation Ireland (SFI) Principal Investigator. He is a Senior Lecturer in the Department of Computing and Mathematics, Dundalk Institute of Technology (DkIT). He is Director of the Regulated Software Research Centre in DkIT and the Medical Device Software Engineering competency area in Lero. He has been awarded SFI funding through the Stokes Lectureship, Principal Investigator and CSET Programmes to research the area of medical device software. He has published over 150 peer-reviewed conference and journal papers and is on the editorial board/programme committee for a number of leading software engineering conferences and journals.

Anita Finnegan

Anita Finnegan is a Senior Software QMS Researcher in the Regulated Software Research Centre based at DkIT and a member of Lero – the Irish Software Engineering Research Centre. Anita's primary focus is establishing the relationship between QMS requirements and software development requirements. Her PhD research addresses the development of a framework for the security assurance of medical devices using cybersecurity assurance cases.

Alec Dorling

Alec Dorling is a member of the research team at the Regulated Software Research Centre in Dundalk Institute of Technology and a member of the Irish Software Engineering Research Centre (Lero). He is the ISO convener of the ISO/IEC 15504 series and ISO/IEC 330xx family of standards on Process Assessment, and the international project leader of SPICE (Software Process Improvement and Capability dEtermination). He has held key positions in software engineering at both national and European levels at IVF's Centre for Software Engineering in Sweden, IT Research and Technology Transfer Centre at the University of Boras in Sweden, European Software Institute in Spain and the National Computing Centre in the UK. He is on the editorial board/programme committee for a number of leading software engineering conferences and journals.

Sherman Eagles

Sherman Eagles is a convener of IEC/ISO joint working group that developed IEC 62304 Medical device software life cycle processes and IEC 80002-1 Guidance on the application of ISO 14971 to software. He has 45 years of experience in the software field, including 18 years at Medtronic. He is also a Co-chair of the Association for the Advancement of Medical Instrumentation (AAMI) software working group and of the ANSI/AAMI SW87:2012 – Application of quality management system concepts to medical device data systems. He is the author of numerous journal papers on software development, software risk management, and software standards.