International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:2, 2017

Supporting Embedded Medical Software Development with MDevSPICE® and Agile Practices

Surafel Demissie, Frank Keenan, Fergal McCaffery

***Abstract*—**Emerging medical devices are highly relying on embedded software that runs on the specific platform in real time. The development of embedded software is different from ordinary software development due to the hardware-software dependency. MDevSPICE® has been developed to provide guidance to support such development. To increase the flexibility of this framework agile practices have been introduced. This paper outlines the challenges for embedded medical device software development and the structure of MDevSPICE® and suggests a suitable combination of agile practices that will help to add flexibility and address corresponding challenges of embedded medical device software development.

***Keywords*—**Agile practices, challenges, embedded software, MDevSPICE®, medical device.

I. Introduction

THE medical device market world-wide is showing substantial impact with industry experts expecting this market to demonstrate robust growth over the next years with figures expected to expand from 133.6bn USD in 2014 to 173.3bn USD in 2019 [1]. A key characteristic of many medical devices is that of embedded software systems. Essentially, such systems are computerized systems that are unique as they are designed to perform specific task on specific platform. The complexity and growth rate of embedded software has been increasing over the past decades. From insulin pumps, pacemakers, cardiac monitors, to anesthesia machines, software is playing a major role in the functionalities of these devices. For example, infusion pumps today contain tens of thousands of lines of code with this number running into the millions for proton beam therapy devices [2]. However, the development of embedded software adds different challenges to the software engineer due to their complexity.

To attempt to control risk and overcome the challenges presented for such development, teams typically follow a plan-driven approach, such as the V-model, and need to provide evidence to show their software development process to get pre-market and post-market approval [3]. This is because such models have specific chick-ins and check-outs of each phase allowing regulations and audits to be performed at each checkpoint. As such, they are obliged to conform to regulations outlined by Medical Device Directive (MDD) in Europe or Food and Drug Administration (FDA) in the US. However, there have been calls for a better software development framework to address the trustworthiness of critical embedded software development. Indeed, most these regulations are high-level and do not dictate about low-level implementation [4]. These regulatory environments are complicated and changing due to the amendments that these regulations went through periodically [5].

MDevSPICE®, an integrated framework of medical device software development best practices, has been developed to assist software medical device developers reach regulatory compliance. MDevSPICE® integrates generic software development best practices with requirements from medical device standards enabling robust software process assessments to be performed when preparing for a regulatory audit. This framework has its origins in the ISO/IEC 15504 (SPICE) [34] series of standards for process assessment.

One approach that may offer assistance is the agile software development [6] which has been a hot issue in recent embedded software development projects. Generally, agile methods recommend a high degree of expert customer involvement, ability to incorporate changing requirements and short development cycles producing working software. Numerous Agile Methods are available including eXtreme Programming (XP) [7], Scrum [8] and Feature Driven Development (FDD) [9]. However, one challenge is to select and identify agile practices for this particular setting. The purpose of this paper is to identify these challenges, describe the structure of MDevSPICE® and appropriate agile practices to use in conjunction with MDevSPICE®, to address the challenges. The next section summarizes challenges for the Embedded Medical device software development process. This is followed by a description of the structure of MDevSPICE®. Next, we suggest agile practices that can satisfy the base practices of MDevSPICE®. Finally, we outline future work.

II. Medical Embedded Software Development and Particular Challenges

The development of medical embedded software development brings challenges from embedded software

This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie)

Surafel Demissie is with Regulated Software Research Centre and Lero, Department of Computing & Mathematics, Dundalk Institute of Technology, Co. Louth, Ireland (e-mail: surafel.demissie@dkit.ie).

Frank Keenan is with Regulated Software Research Centre and Lero, a senior lecturer at Department of Computing & Mathematics, Dundalk Institute of Technology, Co. Louth, Ireland (e-mail: frank.keenan@dkit.ie).

Fergal McCaffery is a senior lecturer and the director of Regulated Software Research Centre, Department of Computing & Mathematics, Dundalk Institute of Technology, Co. Louth, Ireland (e-mail: fergal.mccaffery@dkit.ie).

development which is related to technological factors including platform, hardware-software dependency and real-time nature. Also, progress typically requires input from multiple diverse stakeholder groups including, for example, software developers, hardware engineers, and possibly mechanical engineers in addition to the expected medical domain experts. Such diversity requires much interaction and multi domain communication.

Medical Embedded software are class of embedded software but that are only developed for medical devises. Software running inside Pacemakers, Magnetic Resonance Imaging (MRI), Infusion Pumps, Glucose Test cases and the likes are some examples of embedded medical software.

In addition to complexity, the Medical device software development process is under regulation of various international standards. Based on their geographical location, medical device companies have to follow the required international standards and guidance documents before marketing their products. Although beneficial, demonstrating conformance to such standards brings additional challenges on the software development process [10].

The development of medical embedded software is also characterized by the need to develop hardware and software concurrently. This is known as co-design. According to [11], co-design can be summarized into four tasks such as: Partitioning, allocating, scheduling and mapping. As described by the author, *Partitioning* divides the functions to be implemented into lower level interacting units. Such partitions will be *allocated* to the microprocessors or other hardware units to implement the functions in hardware or software. The next task, *scheduling*, allocates execution times for the functions and finally the *mapping* phase will transform generic functional description into an implementation on a particular set of components, either as software or logic. As we can observe from the summarized co-design activities, the development of both hardware and software is performed concurrently and testing of one unit will require stubs of the other and this can be challenging.

The challenge of co-design has also been discussed by [12] addressing the past, the present and future prospects of embedded systems. This report, when addressing the challenge of complex hardware software co-design, states that *imagine in a single vehicle, more than 100 million lines of code coexist and coexecute today. Imagine also the complexity of testing and verifying properties such as safety in such a complex system.*

Recently our research center has conducted a half-day workshop with a number of medical companies based in Ireland. The range of companies included mobile medical app developers, embedded device companies, small startups and wearable medical device development company. Some of the challenges stated by the companies were:

- ***Regulation Adaption*** – Most companies mentioned difficulty of regulation adaption as a main challenge.
- ***Communication Problems*** – Due to different stakeholders involved in a project, a medical device software development team can consist of Software Engineers, Hardware Engineers, Quality Assurance team, Regulatory auditors and Management teams. Creating effective and efficient communication and knowledge transfer between such diversified stakeholders were also reported challenges.
- ***Hardware and Software Platform Changing*** – This challenge was particularly addressed by some of the medical companies with different hardware and software platform options to choose from. A medical embedded software that is being built recently has to cope with platform change and provide upgrade features.
- ***Market Pressure*** – The medical device market is dynamic and changing frequently. The software has to be developed to the market in a limited time-to-market frame.

III. MDevSPICE®

The MDevSPICE® process reference model (PRM) consists of 23 processes and uses IEC TR 80002-3 [35] (which one of the authors developed in association with the IEC medical device standards community) as its foundation as this is the PRM for IEC 62304, which is the most significant medical device software standard. Ten of these processes are system lifecycle processes, eight are software lifecycle processes and the remaining five provide support for both the system and lifecycle processes as can be seen in Fig. 1.

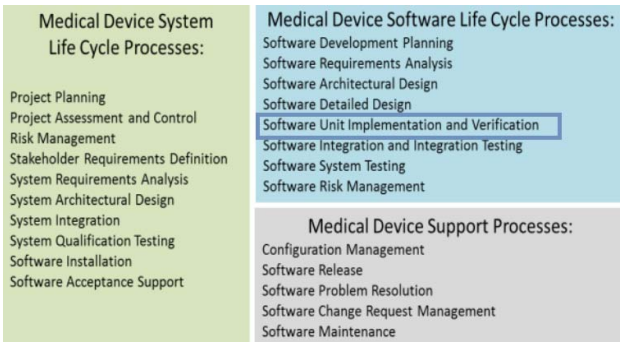

Fig. 1 MDevSPICE® PRM

A process assessment model (PAM) has been defined for MDevSPICE®. This PAM provides a comprehensive model for assessing the software and systems development processes against medical device regulations, standards and guidelines that a medical device software development organization has to adhere to. Similar to ISO/IEC 15504-5 (SPICE) [26], the PAM has two dimensions – a process dimension and a capability dimension. Each process is described in terms of a Process Name, Process Purpose, Process Outcomes, Base Practices, Work Products and Work Product Characteristics.

An example of one process, Software Unit Implementation and Verification, is included in Table I. Like all MDevSPICE® processes it includes a purpose, outcomes and in this instance 4 base practices (BP1 to BP4).

TABLE I
MDEVSPICE® PROCESS AREA, SOFTWARE UNIT IMPLEMENTATION AND VERIFICATION

MDevSPICE® process area	***Software Unit Implementation and Verification***
Process Purpose- *The purpose of the software unit implementation and verification process is to produce executable software units that properly reflect the software design.*	
Process Outcomes	
• *Software units are implemented*	
• *Software unit verification process is established.*	
• *Software unit acceptance criteria prior to their integration into larger software items are established and software units meeting the acceptance criteria is ensured.*	
• *Verification of the software units is performed and the verification results are documented.*	
Base Practices	
BP1 - *Implement each software unit*	
BP2 - *Establish unit verification procedures*	
BP3 - *Establish software unit acceptance criteria and ensure that software units meet the defined*	
BP4 - *Verify software units*	

As illustrated on Table I, the MDevSPICE® process lists the process purpose, process outcomes and base practices at a 'general level'. Each base practice (BP) in this example process area is not detailed in terms of describing HOW the practice should be implemented. For example, one of the BPs in this process area is "BP1-Implement each software unit". This practice at a general level requires developing and documenting the executable representations of each software unit, and updating test requirements and user documentation.

IV. AGILE METHODS

Over the past decade, the software development industry has experienced an explosion in the use of Agile Methods [13]. Surveys on the state of agile report that although Scrum and XP are the most popular agile methods; the current trend in adapting agile methods is also moving towards a hybrid mode of adaption [14]. The same proposition has been outlined by [15] showing how the two agile methodologies have complementary practices. This is because Scrum can be used for project management and XP, on the other hand, can be used for improving the quality of the software [16]. Another report by [17] proposes hybrid model containing both XP which compose engineering and Scrum for its management practices. According to this report, the reason for choosing the two methods is that Scrum is the most effective methodology for managing projects along with XP practices due to their widespread usage, simplicity, flexibility and adaptability in changing environments.

Generally, each agile method comprises a number of practices that achieve the main life cycle phases. Although initially the intention was that each method would be used to cover the complete life cycle it has become common that practices would be selected from each, and other plan-driven approaches if necessary, and combined to be used in a particular development situation. XP, for example started off with 12 practices [7] but an updated version, XP2, replaces these practices with 24 practices that are categorized as either primary or corollary. With the original version, XP1, the intention was that each practice was mandatory for each project. However, this has evolved and XP2 has been modified to follow a phased adoption of XP practices.

Previous studies on the adaption of agile for safety critical and regulated environments are case studies and expert opinions [18]-[21]. Case studies in safety critical domains such as aerospace industry, [22] and automotive [23] have reported that XP practices such as Pair Programming (PP), Test-Driven Development (TDD) and Acceptance Test-Driven Development (ATDD) are reported to bring benefits. On the other hand, Scrum practices such as Daily Scrum, Sprint Review and Sprint Retrospectives have been observed to be suitable for this process area.

Previous literature on the implementation of agile practices in medical domain is a mixture of case studies and expert opinions. An experience report on Abbott's adoption of agile software development practices in its molecular diagnostics division has reported the implementation of agile practices for FDA regulated environment [24]. The adapted agile practices were fixed and short duration iterations, continuous build, unit test, daily team meeting and retrospective. The authors compare two medical device projects; one before agile and one after. After introducing agile practices, the post-agile project has been reported to gain benefits such as:

- lower cost and shorter duration,
- better, less prescriptive test cases,
- frequent integration
- changing requirements development

Reference [25], which is an experience report of a software development group working in Cochlear™ introducing Agile practices, discusses the implementation of Scrum and XP practices. Practices such as automated nightly builds, Daily Scrum and Review Meetings and XP practices such as TDD using the Framework for Integrated Tests (FIT) have been used on Cochlears' product development process which was based on V-Model. The authors report the benefit of using TDD along with the Iterative development of features to produce high quality code. From these case studies we can observe that agile practices can bring benefit to the medical device software development but they need tailoring in the organizational context. Due to complex regulation process in the medical domain we can benefit from tailoring specific agile practices on traditional Software Development Life Cycles (SDLC) such as the V-Model

In this research, we are focusing on one of the processes from the MDevSPICE® framework, Software Unit Implementation and Verification. Initially, we are investigating agile practices that are reported to bring benefit for medical Embedded Software Development specifically for Implementation and verification part of SDLC. A summary of these candidate practices is presented in Table II. The first column shows the practice name while the second column has the practice description.

V. EXTENDING MDEVSPICE® BPS

MDevSPICE® framework, being a collection of international standards, does not recommend the use of a

specific SDLC. On the other hand, this framework has a sequential flow of development. This might suggest the adaption of traditional SDLC models might be compatible for this framework. For the previously mentioned process from the MDevSPICE® framework, of Software Unit Implementation and Verification, we have performed a mapping as shown in Table III.

TABLE II
PREVIOUSLY PROPOSED AGILE PRACTICES

PP	All code is written with two programmers at one machine. For each pair two interchanging roles are recommended. One is in control of the keyboard and is thinking about the best way to solve the problem. The other thinks strategically questioning the whole approach, looking for test cases and performing code inspections [7].
TDD	In TDD you write one single test that fails, write just enough code that makes this failing test pass (and all the other passing tests still pass), and then refactor your code to prepare it for the next tiny step.
ATDD	This process makes use of automated acceptance tests with the additional constraint that these tests be written in advance of implementing the corresponding functionality [26].
Daily stand-up meeting/ Daily Scrum	This is a daily 15-minute stand-up meeting to answer three questions: *What did you do yesterday? what do you plan to do today?* and *are there any impediments to your work?* It's a quick status update that identifies iteration progress, immediate plans and risks.
Sprint Review	Essentially, each iteration of Scrum is conducted in a short *sprint* that delivers a small number of requirements. At the end of each sprint a Sprint Review meeting is held. During this meeting the Scrum Team shows which Scrum Product Backlog items they completed (according to the Definition of Done) during the sprint. This might take place in the form of a demo of the new features
Sprint retrospective	The team inspects itself and creates a plan for improvements.

TABLE III
MAPPING BETWEEN MDEVSPICE® PROCESS AREA AND AGILE PRACTICES

BPs	***Suitable Agile Practices***	
	XP	***SCRUM***
BP1 - Implement each software unit	PP	Daily Scrum
BP2 - Establish unit verification procedures	PP, TDD	Sprint Review, Sprint retrospective
BP3 - Establish software unit acceptance criteria and ensure that software units meet the defined	ATDD	Sprint Review, Sprint retrospective
BP4 - Verify software units	PP	Sprint Review, Sprint retrospective

Table III shows the mapping between BPs of the selected MDevSPICE® process area and candidate agile practices. In the following subsection, we will discuss some of the suggested agile practice and demonstrate how we can benefit in addressing the challenges such as multiple domain communication and co-design.

A. Pair Programming

This practice is reported to increase the quality of the work product and increase the knowledge of each engineer such that the total time to implement a project is lower with PP than without [27].

On our literature and investigation on the web we have identified flavors of PP such as:

- Cross-functional pair programming (CFPP), Cross-functional pairing [28], [29]- pair development composing software engineer and hardware engineer working together.
- Distributed Pair Programming (dPP) [30]- two members of the team synchronously collaborate on the same design or code from different locations.

One of the challenges that have been reported on our assessment was dealing with multiple stakeholder input. Through pairing multiple stakeholders, they can share knowledge, for example through pairing a hardware engineer and a software engineer. Extended practices such as Cross-functional pair programming can also address other challenges such as co-design. Reference [31] stated that this challenge has to be understood and taken into consideration when we apply agile methods. The concurrent activities of hardware and software development practices can be managed through Cross-functional pair programming where different stakeholders work together for the same objective.

B. Test Driven Development (TDD)

TDD is a design approach, and it helps users write better code, because testable code is written by default [32], [33].

In [34], the authors describe the advantage of TDD for embedded software development, where there is hardware development running in parallel. Bugs can be due to hardware, software, or a combination of the two. This report witnessed the benefit of TDD to deal with software bugs of both hardware and software. This practice is also reported to bring benefit in one the case studies discussed before [25]. With the implementation of TDD this case study was able to reduce the load of hardware testing.

By extending the BP, BP2 - Establish unit verification procedures, we can provide a detailed implementation and address additional challenges of co-design in testing and bug tracking. From these two examples and the mapping, we can observe that:

1) The MDevSPICE® framework can solve one of the main challenges medical companies are facing, complex process adherence. But this framework does not provide or dictate a specific SDLC and the BPs are only provided at a general level.
2) By extending the MDevSPICE® process, with suitable and extended agile practices, we can provide detailed implementation at a low level.
3) We can also address additional challenges that medical device companies are facing. Challenges such as multiple domain communication, knowledge sharing and co-design can be addressed with suitable agile practices.

VI. CONCLUSION

In this paper, we have discussed the challenges of medical embedded software development. Medical device companies are reporting the difficulty of dealing with complex process adherence. On other hand, embedded systems design brings its

own additional challenges such as Co-design to the development of medical embedded software. MDevSPICE® framework has been developed to give medical device companies a guidance and address the process adherence challenge. This framework does not provide low level implementation details and BPs are provided at a general level. By extending BPs of MDevSPICE® and adding flexibility through tailored agile practices, we can address challenges of medical embedded software development. In the future we are going to continue mapping the rest of process areas of MDevSPICE® framework with agile practices. We will also perform a structured interview with medical embedded companies. From the interview we will analyze the challenges and apply the extended MDevSPICE framework with agile practices for embedded medical software development through action research.

REFERENCES

[1] Espicom, "United States Medical Devices Report," 2015. (Online). Available: http://www.espicom.com/usa-medical-device-market.html. (Accessed: 25-Feb-2016).
[2] Z. Jiang and R. Mangharam, "High-Confidence Medical Device Software Development," *Found. Trends® Electron. Des. Autom.*, vol. 9, no. 4, pp. 309–391, 2015.
[3] R. F. Munzner and D. Ph, "Entering the U. S. Medical Device Market," pp. 3548–3550, 2003.
[4] AAMI, "AAMI TIR45: Guidance on the use of AGILE practices in the development of medical device software," 2012.
[5] M. McHugh, F. McCaffery, and V. Casey, "Changes to the International Regulatory Environment," *J. Med. Device.*, vol. 6, no. 2, p. 21004, 2012.
[6] D. Greer and Y. Hamon, "Agile Software Development," Aug. 2011.
[7] K. Beck, "Extreme Programming Explained: Embrace Change," *XP Ser.*, no. c, p. 224, 1999.
[8] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, vol. 18. 2001.
[9] S. R. Palmer and M. Felsing, "A Practical Guide to Feature-Driven Development," Nov. 2001.
[10] N. Hrgarek, "Certification and regulatory challenges in medical device software development," *2012 4th Int. Work. Softw. Eng. Heal. Care, SEHC 2012 - Proc.*, pp. 40–43, 2012.
[11] W. H. W. H. Wolf, "Hardware-software co-design of embedded systems," *Proc. IEEE*, vol. 82, no. 7, pp. 967–989, 1994.
[12] J. Teich, "Hardware/software codesign: The past, the present, and predicting the future," *Proc. IEEE*, vol. 100, no. SPL CONTENT, pp. 1411–1430, 2012.
[13] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1213–1221, 2012.
[14] VersionOne Inc, "VersionOne 10th Annual State of Agile Report," 2016.
[15] K. Mar and K. Schwaber, "Scrum with XP," *Informit. com*, 2002.
[16] H. Kniberg, Scrum and XP from the Trenches. Lulu. com, 2015.
[17] Z. Mushtaq and M. R. J. Qureshi, "Novel Hybrid Model: Integrating Scrum and XP," *Int. J. Inf. Technol. Comput. Sci.*, vol. 4, no. 6, pp. 39–44, 2012.
[18] H. Jonsson, S. Larsson, and S. Punnekkat, "Agile Practices in Regulated Railway Software Development," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, 2012, pp. 355–360.
[19] B. Fitzgerald, K. J. Stol, R. O'Sullivan, and D. O'Brien, "Scaling agile methods to regulated environments: An industry case study," *Proc. - Int. Conf. Softw. Eng.*, pp. 863–872, 2013.
[20] F. McCaffery, M. Pikkarainen, and I. Richardson, "Ahaa --agile, hybrid assessment method for automotive, safety critical smes," *Int. Conf. Softw. Eng.*, p. 9, 2008.
[21] T. Myklebust, T. Stalhane, and N. Lyngby, "An agile development process for petrochemical safety conformant software," in *2016 Annual Reliability and Maintainability Symposium (RAMS)*, 2016, no. Fitzgerald, pp. 1–6.
[22] S. H. VanderLeest and A. Buter, "Escape the waterfall: Agile for aerospace," in *2009 IEEE/AIAA 28th Digital Avionics Systems Conference*, 2009, p. 6.D.3-1-6.D.3-16.
[23] R. Y. Takahira, L. R. Laraia, F. A. Dias, A. S. Yu, P. T. S. Nascimento, and A. S. Camargo, "Scrum and Embedded Software development for the automotive industry," *Proc. PICMET '14 Conf. Portl. Int. Cent. Manag. Eng. Technol. Infrastruct. Serv. Integr.*, pp. 2664–2672, 2014.
[24] R. Rasmussen, T. Hughes, J. R. Jenks, and J. Skach, "Adopting agile in an FDA regulated environment," *Proc. - 2009 Agil. Conf. Agil. 2009*, pp. 151–155, 2009.
[25] P. A. Rottier and V. Rodrigues, "Agile Development in a Medical Device Company," *Agil. 2008 Conf.*, pp. 218–223, 2008.
[26] B. Haugset and G. K. Hanssen, "The Home Ground of Automated Acceptance Testing: Mature Use of FitNesse," in *2011 AGILE Conference*, 2011, pp. 97–106.
[27] L. W. Alistair Cockburn, "The Costs and Benefits of Pair Programming," *Extrem. Program. examined*, pp. 223–247, 2000.
[28] Hoby Van Hoose, "Experience Design and Cross-Functional Pairing - SolutionsIQ," 2013. (Online). (Accessed: 11-Oct-2016). Available: http://www.solutionsiq.com/experience-design-and-cross-functional-pairing/.
[29] James E. Hewson, "Cross-functional pair programming | Embedded," 2003. (Online). (Accessed: 11-Oct-2016). Available: http://www.embedded.com/design/prototyping-and development/4024901/Cross-functional-pair-programming.
[30] B. F. Hanks, "Distributed Pair Programming: An Empirical Study," Springer Berlin Heidelberg, 2004, pp. 81–91.
[31] J. Ronkainen and P. Abrahamsson, "Software development under stringent hardware constraints: do agile methods have a chance?," *Extrem. Program. Agil. Process. Softw. Eng.*, pp. 1012–1012, 2003.
[32] K. Beck, "Test-Driven Development By Example," *Rivers*, vol. 2, no. c, p. 176, 2003.
[33] R. Jeffries and G. Melnik, "TDD--The Art of Fearless Programming," *Software, IEEE*, vol. 24, no. 3, pp. 24–30, 2007.
[34] M. Karlesky, W. Bereza, and C. Erickson, "Effective Test Driven Development for Embedded Software," in *2006 IEEE International Conference on Electro/Information Technology*, 2006, no. 616, pp.382–387.
[35] ISO/IEC 15504-5. Information technology – process assessment - Part 5: an exemplar process assessment model. 2012. p. 211
[36] IEC TR 80002-3: Medical device software -- Part 3: Process reference model of medical device software life cycle processes (IEC 62304). 2014. IEC: Geneva, Switzerland. p.23.