2011-06-28

How Amendments to the Medical Device Directive Affect the Development of Medical Device Software

Martin McHugh
Technological University Dublin, martin.mchugh@tudublin.ie

Fergal McCaffery
Dundalk Institute of Technology, fergal.mccaffery@dkit.ie

Valentine Casey
Dundalk Institute of Technology, val.casey@dkit.ie

Recommended Citation

McHugh, M., McCaffery, F., & Casey, V. (2011). How Amendments to the Medical Device Directive Affect the Development of Medical Device Software. European Systems and Software Process Improvement and Innovation Conference. *EuroSPI* Roskilde, Denmark. doi:10.21427/xc06-tv96

Funder: Science Foundation Ireland

How amendments to the Medical Device Directive affect the Development of Medical Device Software

Martin Mc Hugh, Fergal Mc Caffery, Valentine Casey
Regulated Software Research Group
Dundalk Institute of Technology
martin.mchugh@dkit.ie, fergal.mccaffery@dkit.ie, valentine.casey@dkit.ie

Abstract

A recent revision to the European Medical Device Directive (MDD 2007/47/EC) made fourteen amendments to the original directive (93/42/EEC). A number of these changes directly affect the development of software for use in healthcare. The most significant change in relation to medical device software development is that standalone software is now seen as an active medical device. Prior to this amendment medical device software was developed in accordance with the IEC 62304 standard. However, IEC 62304 is not sufficiently comprehensive to provide guidance in the development of standalone software as an active medical device. Medi SPICE is currently being developed to fill the gaps left by IEC 62304 in developing standalone software as an active medical device and to provide medical device software developers a single point of reference for developing software for use in healthcare.

Keywords

IEC 62304, Medical Device Directive, MDD, Software Process Improvement, 2007/47/EC, AIMD, Medi SPICE

1 Introduction

Software is ubiquitous in daily life however, it is more critical in some areas than others. Failures in medical device software can have severe or fatal consequences. Between June 1985 and January 1987, four people died and two were left permanently disfigured by a software controlled radiation therapy machine known as Therac-25[1]. Therac-25 used software to control a beam spreader plate which reduced patient's exposure to radiation. However, due to software malfunctions the plate was not in place when required and patients received massive doses of radiation. This case highlighted the need for adequate safety measures to be put in place to protect patients and third parties i.e. clinicians, using medical devices controlled by software.

All medical devices used within the European Union (EU) must conform to the current MDD to achieve the CE conformance mark. The current directive i.e. 2007/47/EC[2], amends European directives MDD (93/42/EEC)[3], AIMD (90/385/EEC)[4] and the Biocides Directive (98/8/EC)[5] [6]. The revision to the MDD (2007/47/EC) covers all areas relevant to medical devices including risk and quality management. This latest amendment allows for standalone software to be used as an active medical device. With this amendment, incidents involving medical devices such as Therac-25 become more relevant, as now software can be the only element in a medical device subject to conformance requirements.

Consequently methods used to ensure that software is safe and fit for purpose must be reviewed. A number of benefits can be gained by manufacturers employing Software Process Improvement (SPI) techniques, one of which is a reduction in software faults that could potentially result in device recalls [7]. SPI is a continuous cycle of performing an assessment, implementing the recommendations of the assessment and restarting the cycle [8]. This process of continuous assessment and improvement can help reduce the amount of defective software being developed. Essentially the safety of medical device software is determined through the software processes followed during development [9].

IEC 62304[10] is the current medical software lifecycle process standard. This standard contains a number of processes for medical device software development which medical device software developers should follow in order to implement best practice medical device software processes. IEC 62304 is used in the development of software when software is part of a medical device system which consists of hardware plus software. It provides the minimum regulatory medical device requirements within a specified group of processes. However, it excludes all system level processes. As a result, IEC 62304 hands off the system processes to other standards such as ISO 13485[11], IEC 60601-1[12], IEC 61010-1[13], IEC/ISO 90003[14], IEC 61508-3 [15], ISO 14971 [16] and ISO 12207[17] [10].

Section 2 examines the revision to the MDD and highlights what this means with respect to medical device software development. This will include particular reference to the development of standalone software as an active medical device. In section 3, we discuss the existing standards that are appropriate to the development of medical devices with emphasis on satisfying the requirements of the MDD (2007/47/EC). In section 4, we discuss the importance of SPI techniques and recommend a specific SPI model to follow. Section 5 contains the conclusions from this research and our plans to progress this research further.

2 European Medical Device Directive Amendment 2007/47/EC

The recent MDD [2] revision has made a number of amendments to previous directives i.e. [3-5].The MDD (2007/47/EC) came into force on March 21st 2010. In total there are fourteen changes introduced within the MDD (2007/47/EC)[18]. There are four areas within the amendment of the MDD (2007/47/EC) with important significance to medical device software development:

- Standalone Software as an active medical device;
- Validation of software as an active medical device;
- Software localisation;
- Safety Classification.

2.1 Standalone Software as an Active Medical Device

Prior to the release of the MDD (2007/47/EC) provision had been made within the MDD (93/42/EEC) for software to be used as a medical device. However MDD (2007/47/EC) Article 1 Section 2 makes explicit reference to software being a medical device.

*"any instrument, apparatus, appliance, **software**, material or other article, whether used alone or in combination, including the software intended by its manufacturer to be used specifically for diagnostic and/or therapeutic purposes and necessary for its proper application"*

To accompany this change provision has also been made for standalone software to be used as an active medical device. Within the MDD (2007/47/EC) Annex IX Section 1.4 amendment M5 states:

"Stand-alone software is considered to be an active medical device"

This can be difficult to understand particularly in relation to when software is or is not a medical device. An example of software as an active medical device is a software package which is used to calculate treatment doses for oncology treatment devices. A caveat has also been included into MDD (2007/47/EC) to avoid ambiguity in determining if a software package is a medical device.

"software for general purpose when used in a healthcare setting is not a medical device"

This caveat provides some clarity surrounding particular software used in healthcare. In Ireland we await a formal document from the Irish Medicines Board to define exactly what types of healthcare software applications will now be defined as a medical device. This document is expected in the spring of 2011 but in its absence there is still some confusion regarding what standalone software will be classified as a medical device.

2.2 Software Validation

As standalone software is now classified as an active medical device under the MDD (2007/47/EC) guidelines must be put in place to ensure that such software is safe and fit for purpose. To ensure this the MDD (2007/47/EC) Annex I Section 12.1a (amendment M5) states;

"For devices which incorporate software or which are medical software in themselves, the software must be validated according to the state of the art taking into account the principles of development lifecycle, risk management, validation and verification."

"State of the Art" is used to here to mean what is generally accepted as good practice. Since this requirement was introduced, developers must now validate the software integrated or standalone, regardless of device class. IEC 62304 and its aligned, standards, ISO 13485, IEC 62366 [19], IEC\TR 80002-1 [20], 80001-1 [21], EN 60601 and ISO 14971 are harmonised under the MDD (2007/47/EC) and are seen as a good place to start when validating software.

Whilst these standards are generally accepted and are harmonised under MDD (2007/47/EC) they do not necessarily cover the requirements introduced within the MDD (2007/47/EC).

2.3 Software Localisation

Another change which affects the development of medical device software is software localisation. Under the MDD (2007/47/EC), software sold or used within the EU must be localised into the language of each of the EU countries that the product will be marketed i.e. MDD 2007/47/EC, Article 4.4. Essentially, if an Irish medical device manufacturer wishes to market a medical device into France, the graphical user interface (GUI) must be available in French. A number of difficulties can arise when attempting to perform a software translation [22]:

- Differing file formats;
- Different character encoding;
- Character size constraints;
- Parsing may be required;
- Errors caused in code caused by repossessing.

There are currently a number of organisations that provide translation services to medical device manufacturers such as Global Translations [23].

3 IEC 62304 – Software Lifecycle Processes

Medical device manufactures wishing to achieve regulatory conformance are advised to follow the relevant applicable standards. Evidence of following the applicable standards can improve the process of achieving regulatory conformance. Medical device software developers typically follow IEC 62304 and its aligned standards.

3.1 Processes

IEC 62304 is a medical device software development standard. It was created to provide assistance in terms of the safe design and maintenance of medical device software. Software developed using the IEC 62304 standard is founded upon the assumption that the software is developed in accordance with a quality management standard (ISO 13485), a risk management standard (ISO/IEC 14971) and a product level standard (EN 60601-1). This standard provides a framework of processes divided into activities which are further divided into tasks.

IEC 62304 provides guidance on the development of software as part of a medical device. However IEC 62304 does not provide guidance on all of the necessary processes required to develop standalone software as an active medical device. IEC 62304 states;

"This standard does not cover validation and final release of the medical device, even when the medical device consists entirely of software"

With the MDD revision i.e. (2007/47/EC) a medical device can consist only of software. As validation is required to ensure reliability and safety another method of validating standalone software as a medical device is required. One suggestion is to modify the scope of EN 60601-1 to include medical software systems. Another suggestion is to convert IEC 62304 to a product oriented standard in order to make it applicable to standalone software as an active medical device. All of these suggestions have advantages and disadvantages. A favourable outcome would be for IEC 62304 to remain as a software lifecycle standard and that a new standard be developed to specifically facilitate standalone software as an active medical device.

3.2 Safety Classification

IEC 62304 classifies all software based on the risk posed to the patient or user. The devices are classified as follows:

- Class A: No injury or damage to health is possible
- Class B: Non-serious injury is possible
- Class C: Death or Serious Injury is possible

This classification is similar to that of ISO 14971 Clause 4.4, 5 and 6.1. Safety critical software systems can be divided into items running a different software element each with its own safety classification. These items can be further sub divided into additional software elements. The overall software system assumes the highest classification contained within all of the software elements. For example if a software system contains five software elements, four of which may be classified as Class A, but one may be classified as Class C and therefore the overall device receives a classification of Class C. This can be seen in figure 1.0

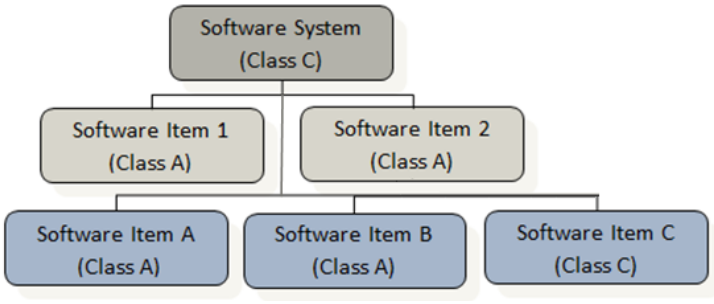

Figure 1.0 Classification of software items within complete software system

4 SPI and Medi SPICE

4.1 Importance of SPI

SPI is an important element within any software development lifecycle. Many organisations have difficulty in consistently developing high quality software. There are many benefits to be gained by using SPI including [24]:

- Improvements to overall quality
- Increased on-time delivery
- Budget consistency
- Reduced development costs

SPI places the emphasis on defining processes that are appropriate to the project and ensures that these processes are consistently followed. SPI maturity models focus on what has to be done, rather than how it should be done. The benefits of utilising SPI can be seen in many companies e.g. Siemens[25], Alcatel [26], NASA[27] and Motorola[28].

In order for SPI to be successful within an organisation, it relies on a number of critical factors. In 2005 a survey of one hundred and twenty software organisations identified six organisational factors as being crucial to ensure the success of SPI [29]:

- Business orientation;
- Involved leadership;
- Employee participation;
- Concern for measurement;
- Exploitation of existing knowledge;
- Exploration of new knowledge.

Research carried out by Embedded Market Forecasters in 2010[30] provided a comparison between software developed by the embedded industry and software developed by medical device producers. This research showed that 12.9% of medical device projects were cancelled, whilst 11.2% of embedded industry projects were cancelled. This research also revealed that on average 19.4% of the overall budget is wasted due to months lost during project development. The primary reason cited for these problems occurring is incomplete or vague requirements.

An empirical study in 2007 revealed how much importance medical device software developers place upon SPI [31]. The study surveyed organisations developing software for medical devices and medical information systems with the majority of respondents coming from Germany, USA and Sweden. 71% of respondents came from small and medium companies with between ten and two hundred and fifty staff. The remainder of the respondents came from organisations with over two hundred and fifty software developers. Of the respondents, 98% rate software as very important or an important part of their products. However, only 14% of the respondents have either a CMMI (Capability Maturity Model Integration) or ISO 15504 (SPICE) rating. Also only 50% of the respondents follow a defined process to perform requirements engineering, software architecture and design activities and implementation, over 50% of the time. This survey also asked participant's which process or activities cause the most issues for a software development project.

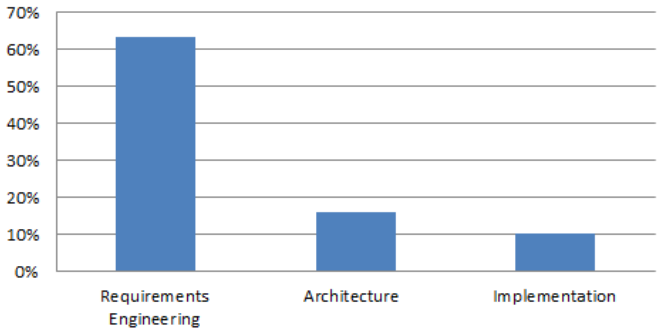

Figure 2.0 Activities & Processes most difficult to software development projects [31]

Figure 2.0 shows, the requirements engineering process is seen as causing most of the issues in the development of software for medical devices. Typically organisations following a defined set of processes or activities contained within an SPI model will have minimal problems with areas such as requirements engineering as they are provided guidance on all areas of development. This survey shows an inversely proportional relationship between importance placed on SPI rating/activities and difficulties caused by specific process areas.

Both of these pieces of research [30, 31] show medical device software developers have the most difficulties at the requirements elicitation/engineering stage of development..

4.2 Medi SPICE

Currently medical device manufactures developing standalone software as an active medical device have no single point of reference specific to the development of software for medical devices. To this end Medi SPICE[32] is currently being developed by the Regulated Software Research Group (RSRG) at Dundalk Institute of Technology (DkIT) in association with the SPICE User Group, to help in achieving regulatory conformance and improve overall device safety by employing SPI techniques through the entire development lifecycle.

Medi SPICE provides medical device software manufacturers with a Process Assessment Model (PAM), and a Process Reference Model (PRM), to help implement high quality medical device software processes that will enable seamless conformity to the medical regulatory standards. The PAM is being developed in accordance with the requirements of ISO/IEC 15504-2:2003, and is based on ISO/IEC 15504-5:2006. However ISO/IEC 15504 is not specific to the medical device industry. To overcome this, Medi SPICE incorporates the practices contained within IEC 62304 processes and the other relevant standards into appropriate the relevant ISO/IEC 15504 based processes. Due to the safety critical nature of medical device software, additional safety processes have also been incorporated from +Safe[33] and ISO 14971.

Medi SPICE aims to minimise the volume of software documentation and provide global harmonisation for all medical device software manufactures. Medi SPICE assessment results can be used to indicate the state of a medical device supplier's practice in relation to regulation. Medi SPICE results can also be used as a criterion for supplier selection. As discussed in section 4.1 many medical device software development companies cite the requirements engineering process as the most problematic of all of the processes. Medi SPICE will provide clear guidance on how to perform this process in addition to other processes of the medical device software development and maintenance lifecycle.

5 Conclusions

The recent revision to the MDD allows for standalone software to be an active medical device and states that software must be validated in accordance with state of the art practices. However IEC 62304 does not provide comprehensive guidance on the development of standalone software as an active medical device as it is purely a software development standard. An example of a process beyond the scope of IEC 62304 is the requirements elicitation stage of development. However research has shown that the requirements elicitations stage causes medical device software developers the most issues during development. This omission can be overcome by employing an SPI model specific

in the area of development of medical device software. To this end Medi SPICE is being developed to provide guidance in all stages of development of standalone software as an active medical device.

6 Acknowledgements

This research is supported by the Science Foundation Ireland (SFI) Stokes Lectureship Programme, grant number 07/SK/I1299, 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), and supported in part by Lero - the Irish Software Engineering Research Centre (http://www.lero.ie) grant 03/CE2/I303_1.

7 Literature

1 N. Leveson, Safeware: System Safety and Computers: Addison Wesley, 1995.
2 European Council, 2007 "Council Directive 2007/47/EC", September 2007
3 European Council, "Council Directive 93/42/EEC Concerning Medical Devices", 14-June-1993
4 European Council, 1990 "Council Directive 90/385/EEC", 5 September 2007
5 European Council, 1998 "Council Directive 1998/8/EEC", February 1998
6 European Council, 2009 "Implementation of Directive 2007/47/EC amending Directives 90/385/EEC, 93/42/EC and 98/8/EC ", 5 June 2009
7 P. Patra and S. Bartaki, "Productivity Improvement Using Ten Process Commandments," PMI Virtual Library, 2009.
8 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.
9 P. L. Jones, et al., "Risk Management in the Design of Medical Device Software Systems," Biomedical Instrumentation & Technology: July 2002, vol. 36, pp. 237-266, 2002.
10 ANSI/AAMI/IEC 62304, Medical device Software - Software life cycle processes, Association for the Advancement of Medical Instrumentation 19-July-2006
11 EN ISO 13485:2003 Medical Device: Quality Management Systems. Requirements for the Regulatory Process, 24-July-2003
12 EN 60601-1 Medical Electrical Equipment. General requirements for basic safety and essential performance. Collateral standard. Usability 31-May2010
13 IEC 61010-1:2010 Safety requirements for electrical equipment for measurement, control and laboratory use, June 2010
14 ISO/IEC 90003:2003 Software engineering -- Guidelines for the application of ISO 9001:2000 to computer software, February 2004
15 IEC 61508-3:2010 Functional Safety of electrical\electronic\programmable electronic safety related systems, May 2010
16 EN ISO 14971:2009 Medical Devices. Application of Risk management to medical devices 31-July-2009
17 ISO/IEC 12207:1995 Information Technology, Software Lifecycle Processes, 28-February-1995
18 Emergo. http://www.emergogroup.com/newsletters/europe-2007-47-ec-oct2007 2007.
19 IEC 62366 Medical Devices - Application of usability engineering to medical devices, 13-November-2007
20 AAMI/IEC TIR 80002-1:2009 Medical Device Software 1: Guidance on the application of ISO 14971 to Medical Device Software 31-May-2010
21 IEC TIR 80001-1 Application of risk management for IT networks - networks incorporating medical devices
22 Global Vision International Inc. http://www.globalvis.com/mdd-200745ec-and-software-localization/ 2010.
23 Global Translations http://www.gts-translation.com/medical_device_translation.asp. 2011.
24 G. O'Regan, Introduction to Software Process Improvement Springer, 2010.

25 T. Messer, et al., "Siemens Process Assessment and Improvement Approaches: Experiences and Benefits," in 22nd International Computer Software and Applications Conference, Vienna, Austria, 1998.
26 C. Debou, et al., "Alcatel's Experience with Process Improvement," in Better Software Practice for Business Benefits: Principles and Experience ed: CS Press, 1999, pp. 281-301.
27 V. R. Basili, et al., "Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory," in 24th International Conference on Software Engineering, Orlando Florida, 2002, pp. 69-79.
28 M. K. Daskalantonakis, "A Practical View of Software Measurement and Implementation Experiences Within Motorola," IEEE Trans. Softw. Eng., vol. 18, pp. 998-1010, 1992.
29 T. Dyba, "An Empirical Investigation of the Key Factors for Success in Software Process Improvement," IEEE Trans. Softw. Eng., vol. 31, pp. 410-424, 2005.
30 Embedded Forecasters - Embedded Market Forecasters Survey (2010)
31 C. Denger, et al., "A Snapshot of the State of Practice in Software Development for Medical Devices," in First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007, Madrid, 2007.
32 F. McCaffery and A. Dorling, "Medi SPICE: An Overview," Journal of Software Maintenance and Evolution: Research and Practice, vol. 22, pp. 255-267, 2010.
33 +SAFE, V1.2: A Safety Extension to CMMI-DEV, V1.2, Defence Materiel Organisation, Australian Department of Defence, March 2007,Software Engineering Institute, TECHNICAL NOTE CMU/SEI-2007-TN-006http://www.sei.cmu.edu/pub/documents/07.reports/07tn006.pdf

8 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 as part of the Regulated Software Research Group in Dundalk Institute of Technology.

Fergal Mc Caffery

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

Valentine Casey

Dr. Val Casey is a Senior Researcher with the Regulated Software Research Group in Dundalk Institute of Technology. His previous roles include Senior Lecturer and Research Area Leader at Bournemouth University, Researcher with Lero - the Irish Software Engineering Research Centre at the University of Limerick where he also lectured. He has over 20 years' experience in the software industry. He has also provided consultancy services focusing on software process improvement and software testing in the financial and telecom sectors.