

On the Impact of Medical Device Regulations on Software Architecture

Klaus Marius Hansen and Konstantinos Manikas

Department of Computer Science, University of Copenhagen, Njalsgade 128, Copenhagen, Denmark

Keywords: Medical Device, Regulation Compliance, Software Architecture.

Abstract: Compliance to regulations and regulatory approval are requirements for many medical device software systems. In this paper, we investigate the implications of medical device software regulations to the design of software systems. We do so by focusing on the American and European regulatory authorities and review the legal requirements for regulatory approval of medical devices. We define a simplified process for regulatory approval, consisting of five steps, and enhance this process by descriptions of how to decide whether a software system is a medical device and how to identify the class of the device. Moreover, we review software modularity in the implementation of software medical device and propose a set of preliminary principles for architectural design of software medical device based on a set of constraints identified from the reviewed regulations.

1 INTRODUCTION

Information and Communication Technology (ICT) and, more specifically, software systems have arguably become an integral part of healthcare services and support a wide variety of domains and functions within healthcare. This has become more apparent with trends like unified and interoperable Electronic Medical Records (EMRs) (Aanestad and Jensen, 2011), the increased use of telemedicine, e-health and m-health solutions (Christensen et al., 2014), or mobile apps with functionality that pose risks to human safety (Manikas and Hansen, 2013).

Many healthcare systems are *mission-critical* systems for which a failure may have severe consequences for human lives. To moderate issues of this kind, several authorities have issued regulatory requirements similar to those for traditional medical equipment and pharmaceutical substances. In this paper, we argue that the design of software systems under regulation compliance differs from other systems and identify the impact of regulation compliance on software design and software architecture. To do so, we focus on two regulatory authorities: the United States (US) Food and Drug Administration (FDA) and the European Union's (EU's) Medical Device Directive (MDD) and review their regulatory requirements. Our findings include: (i) a summary of the regulatory process in five simple steps, (ii) specifying how to identify if a system is a medical device in FDA and MDD, (iii) steps for defining the class of the medi-

cal device in each authority, (iv) identifying the constraints regulations put on system modularity, and (v) proposing three architectural principles for medical device software.

This work serves as a review of the legal requirements for medical device regulation in US and EU. We translate the formal requirements to a set of steps to be taken as part of regulatory compliance. Moreover, we identify a set of constraints on architectural design of software medical device and propose three principles to be included in the design of these systems.

2 REGULATION PROCESS

Examining the regulatory approval process for medical devices, we simplify the process for software systems into five steps (Intertek, 2015):

1. Determine if the software system is a medical device
2. Classify the software system as a medical device
3. Prepare technical documentation and develop and implement a Quality Management System (QMS)
4. Fulfill premarket requirements and apply for assessment
5. Maintain QMS and perform post-market surveillance

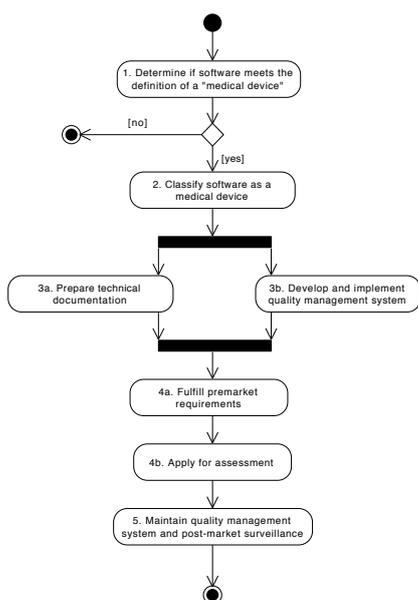


Figure 1: Overview of the regulatory approval process for medical devices (EU, US). Source: Intertek (2015).

Figure 1 shows the overview of the regulatory approval process¹. Our main sources are the laws and regulations that guide the approval and marketing of medical devices. In the US, the Food and Drug Administration (FDA) approves medical devices. The FDA is governed by the Federal Food, Drug, and Cosmetic Act (FD&C Act; (Federal Food, Drug, and Cosmetic Act of 1938, 2007, Chapter 9)) that is codified in the Code of Federal Regulations (CFR), Title 21 (FDA, 2014). In the EU, medical devices need to meet the requirements of the Medical Devices Directive (MDD; MDD (2007)) to be put on market. Below we examine steps 1 and 2 in more detail for FDA and MDD.

2.1 Determine if Software is a Medical Device

Step 1 in Figure 1 focuses on whether the software under consideration is a medical device or not. If the software is not a medical device, neither FDA nor MDD regulations apply.

Figure 2 gives an overview for the US (based on Section 201(h) of the FD&C act). Whether software is a medical device is based on the “intended use” of the software. Similarly, Figure 3 gives an overview for the EU (based on the MDD; (MDD, 2007, Article 1)). An important distinction is that in the US, an accessory is also a medical device whereas in the

¹Although these steps can also be applied to Canada and Japan, we focus on the regulatory processes of EU and US.

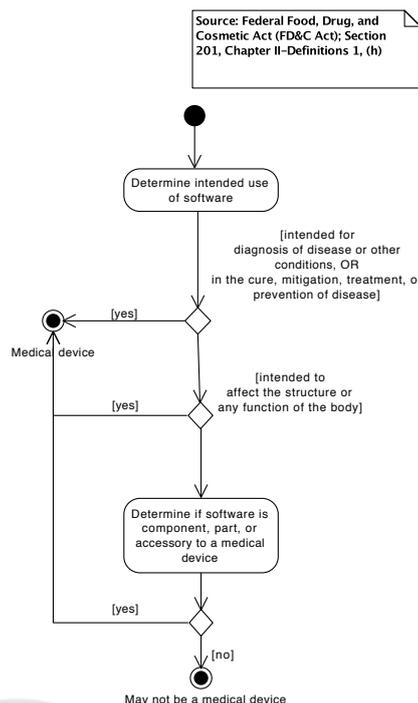


Figure 2: Determining if software is a medical device, US.

EU, accessories are governed by the MDD but are not themselves medical devices.

2.2 Classification of Software as Medical Devices

Assuming that the software is a medical device, the next step is to determine the classification category the software falls under. FDA distinguishes between three medical device classes: Class I, II, or III, while MDD distinguishes four: Class I, IIa, IIb, or III. These classifications imply requirements for the software development process (including steps 3, 4, and 5 in Figure 1).

The classification process of the US is shown in Figure 4. It is important to note that the classification found is only a suggestion and that it is the FDA that determines the class of the software. In the US process, the first step for classification is to find a device that is “substantially equivalent” to the software. If such a device can be found in the CFR, and an equivalence argument can be made, the software gets the same classification as the substantially equivalent device. Substantially equivalent devices may be sought through FDA’s product classification database². If no

²<http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfPCD/PCDSimpleSearch.cfm>. The database currently (2015-05-27) contains 103 matches for the search string “software”.

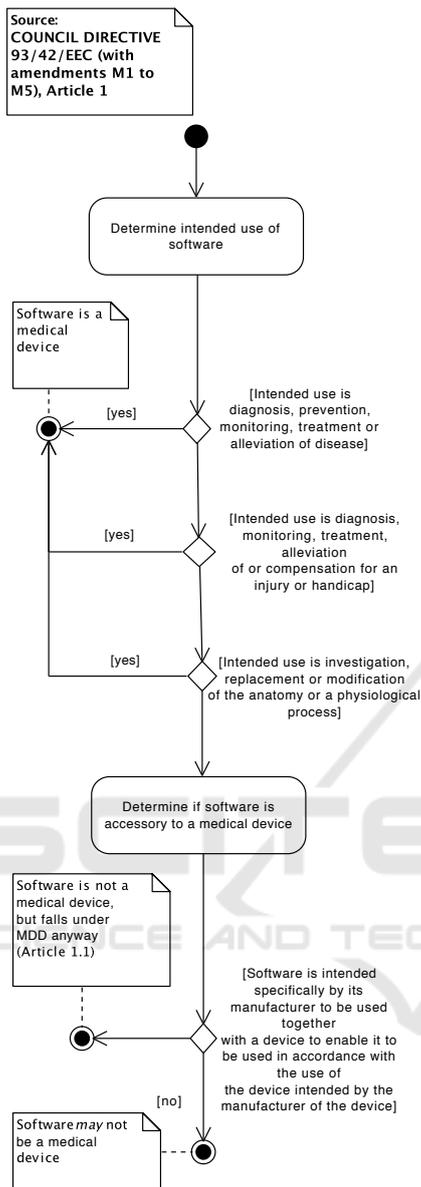


Figure 3: Determining if software is a medical device, EU.

match is found, the rest of Figure 4 applies.

A special case of interest for software in the FDA classification are the “Medical Device Data Systems (MDDS)”. MDDS transfer, store, convert, or display medical device data without controlling or altering the functions of connected medical devices or being used for active patient monitoring. MDDS fall under Class I. This is shown in Figure 5. Food and Drug Administration (2011) contains detailed questions and answers related to MDDS.

The EU classification process is shown in Figure 6 (MDD, 2007, Annex IX). In the EU, software that “drives or influences the use of” a device is clas-

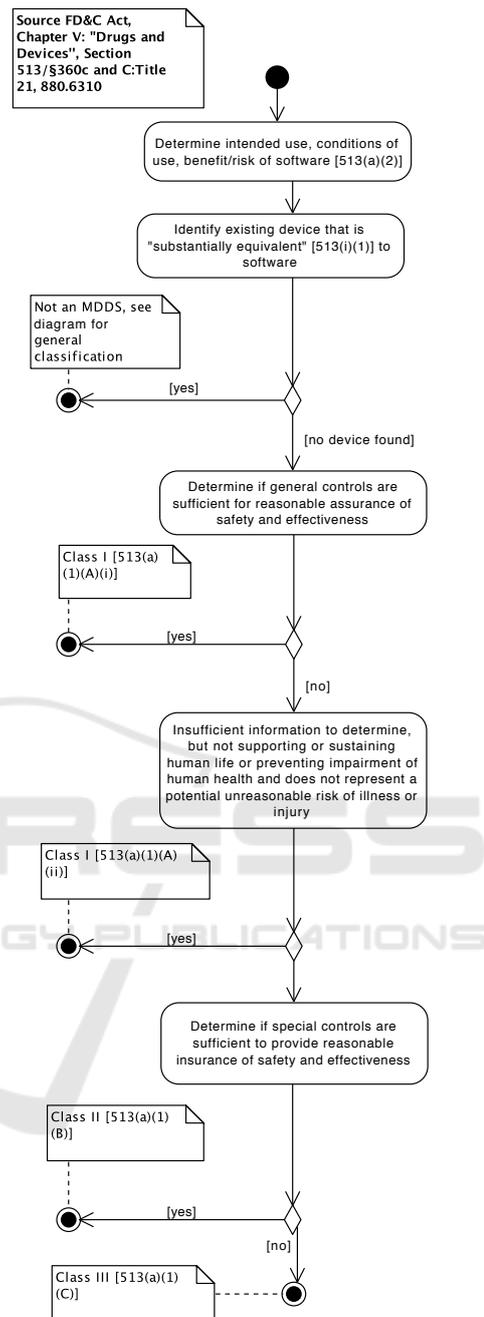


Figure 4: Classifying medical device software, US.

sified as the device is. Otherwise, software is considered a “standalone, active device” and can be classified in Class I, IIa, or IIb. In particular, according to Figure 6, software that is a standalone, active device, and an MDDS is in Class I.

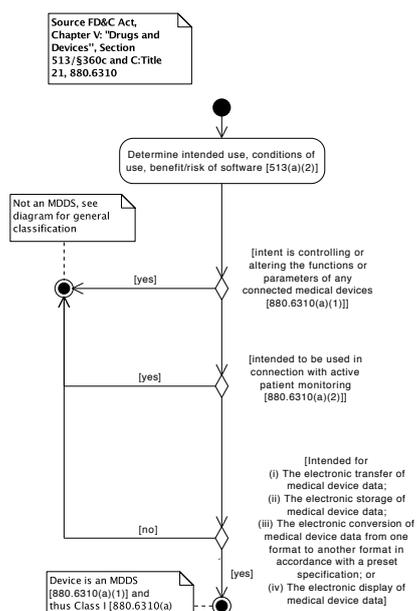


Figure 5: Classifying medical device software, US. Special case: Medical Device Data System (MDDS).

3 MODULARITY OF SOFTWARE AS MEDICAL DEVICES

In this section we examine what constitutes “software”, i.e., which parts of a (software) system is a medical device. As noted in Sections 2.1 and 2.2 this is important since restrictions apply for how software medical devices may be implemented. The high-level division of a software system is its “software architecture”, i.e.,

“the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both” (Bass et al., 2013)

As such, the module, component-and-connector (C&C), and allocation structures are all relevant to how a software medical device should and could be divided into parts.

For the EU, guidance documents state that “the determination of the class of a particular device may be made with respect to the simplest configuration that can still be considered, in view of its proper functional features, as a device in its own right” (MEDDEV 2.4/1 Rev. 9, 2010). MEDDEV 2.1/6 (2012) discusses modularization of software medical devices into separate applications including that non-medical modules of a software medical device system is not to be treated as a medical device³. Rather, such

³This guidance relates to that accessories are “treated as medical devices in their own right”(MDD, 2007).

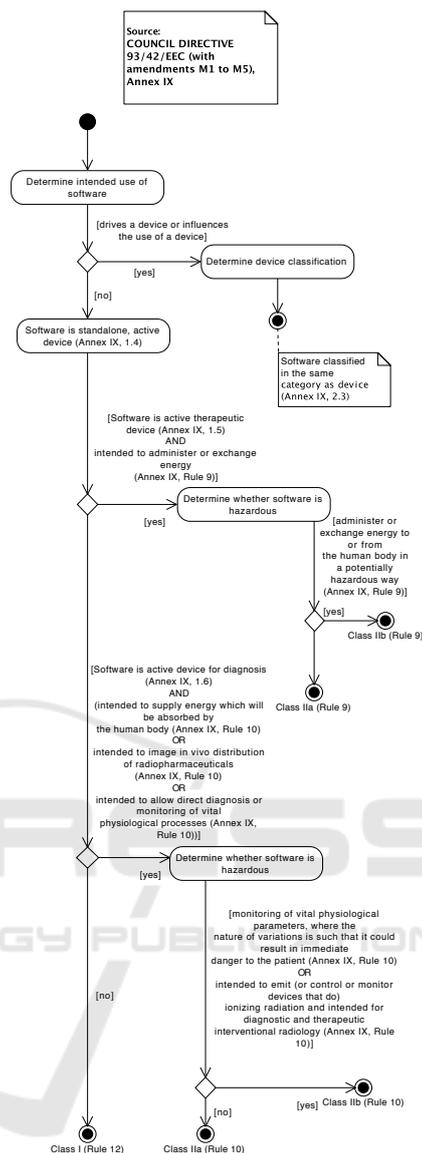


Figure 6: Classifying medical device software, EU.

non-medical modules are to be treated as “Software Of Unknown Provenance” (SOUP; IEC 62304:2006 (2006)). Still “the whole combination [...] must be safe and must not impair the specified performances of the devices” (MDD, 2007).

For the US, guidance on modularity is less clear as is the status of “accessories” (Food and Drug Administration, 2015) including their definition. For some modularizations, a module may become an accessory to a parent device/module. The guidance clarifies how accessories are defined and how they may be classified (and suggests the use of the “De Novo” process for classifying accessories in a lower class than their parent device; Food and Drug Administration (2015)).

The IEC 62304:2006 (2006) standard is harmonized and adopted by both the EU and the US (Bundtz, 2010) and thus applies to both cases. The standard specifies that a software medical device system should first be classified as a whole based on a hazard analysis (Bundtz, 2010). Subsequently, modules (“software items”) may be classified separately if they are “segregated”. The standard only mentions one example of segregation: “to have software items execute on different processors. The effectiveness of the segregation can be ensured by having no shared resources between the processors”.

4 ARCHITECTURAL PRINCIPLES FOR SOFTWARE AS MEDICAL DEVICES

To decide on a proper modularization for software medical devices, multiple constraints have to be taken into account:

- The cost of putting a device to market increases with increased classification. For PMAs, the FDA, e.g., charges more than \$250,000⁴. Moreover, for Class II or above devices in the EU, an external, notified body has to assess conformity to regulations.
- Time-to-market increases with increased classification. The higher the classification, the higher the requirements to plans, requirement specification, software architecture design etc. (IEC 62304:2006, 2006).
- Evolvability decreases with increased classification. For the US, e.g., changes to a Class III device may require a “PMA supplement” that may take up to 180 days to process⁵.

On the other hand, if a device is marketed according to a lower class than it really is, the consequences may be that a device is taken off the market⁶.

We thus propose the following preliminary principles for architectural design:

- *Form Equivalence Classes of Components Through Segregation.* Divide/segregate software medical devices into modules that are separate applications/devices or accessories with separate classes if possible.

⁴<http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/MDUFAIII/ucm313673.htm>

⁵<http://www.fda.gov/RegulatoryInformation/Guidances/ucm089274.htm>

⁶<http://www.fda.gov/ICECI/EnforcementActions/WarningLetters/2013/ucm376296.htm>

- *Segregate Transforming Components from Transmitting Component.* (For the US) MDDS devices are of particular interest since they are Class I.
- *Segregate Evolving Components from Stable.* Components that are expected to evolve frequently should present as little risk as possible, i.e., be in a low class. In this way, e.g., adaptive maintenance can be performed faster.

5 CONCLUSION

In this paper, we investigate the implications of medical device software regulations to the design of software systems. We do so by focusing on the US and EU authorities and review the legal requirements for regulatory approval of medical devices. We define a simplified process for regulatory approval, consisting of five steps, and enhance this process by steps that aid in deciding whether a software system is a medical device and how to identify the class of the device. Moreover, we review software modularity in the implementation of software medical devices and propose a set of preliminary principles for their architectural design based on a set of constraints identified from the reviewed literature.

Plans for future work include the improvement and empirical evaluation of the identified process and steps in different domains of software systems. Moreover, we plan to apply and evaluate the proposed principles for architectural design in practice.

ACKNOWLEDGEMENTS

This work has been conducted as part of the SCAUT Project⁷, co-funded by the Innovation Fund Denmark.

REFERENCES

- Aanestad, M. and Jensen, T. B. (2011). Building nationwide information infrastructures in healthcare through modular implementation strategies. *The Journal of Strategic Information Systems*, 20(2):161 – 176.
- Bass, L., Clements, P., and Kazman, R. (2013). *Software Architecture in Practice*. Addison-Wesley, 3rd edition.
- Bundtz, B. (2010). Developing medical device software to iec 62304. *European Medical Device Technology*, 1(6). <http://www.emdt.co.uk/article/developing-medical-device-software-iso-62304>.

⁷<http://www.scaut.dk/>

- Christensen, H. B., Hansen, K. M., Kyng, M., and Manikas, K. (2014). Analysis and design of software ecosystem architectures – towards the 4s telemedicine ecosystem. *Information and Software Technology*, 56(11):1476 – 1492.
- FDA (2014). Title 21 of the code of federal regulations. <http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cf/cfr/cfrsearch.cfm>.
- Federal Food, Drug, and Cosmetic Act of 1938 (2007). United States Code, Title 21.
- Food and Drug Administration (2011). Medical devices; medical device data systems. *Federal Register*, 76(31):8637–8649. <http://www.gpo.gov/fdsys/pkg/FR-2011-02-15/pdf/2011-3321.pdf>.
- Food and Drug Administration (2015). Medical device accessories: Defining accessories and classification pathway for new accessory types. draft guidance for industry and food and drug administration staff. <http://www.fda.gov/downloads/medicaldevices/device/regulationandguidance/guidancedocuments/ucm429672.pdf>.
- IEC 62304:2006 (2006). Medical device software – software life cycle processes. ISO/IEC.
- Intertek (2015). Quick guide: 5 steps to market approval for medical devices. <http://www.intertek.com/5-step-market-approval-medical-devices/>. Accessed May 2015.
- Manikas, K. and Hansen, K. M. (2013). Reviewing the health of software ecosystems - a conceptual framework proposal. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 33–44.
- MDD (2007). Council directive 93/42/eec of of 14 june 1993 concerning medical devices. <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:01993L0042-20071011&from=EN>. Amended up until and including Directive 2007/47/EC.
- MEDDEV 2.1/6 (2012). Guidelines on the qualification and classification of stand alone software used in health-care within the regulatory framework of medical devices. http://ec.europa.eu/growth/sectors/medical-devices_old/documents/guidelines/files/meddev/2_1_6_ol_en.pdf.
- MEDDEV 2.4/1 Rev. 9 (2010). Medical devices: Guidance document – classification of medical devices. http://ec.europa.eu/health/medical-devices/files/meddev/2_4_1_rev_9_classification_en.pdf.