The Integration of the Risk Management Process with the Lifecycle of Medical Device Software

F. Pecoraro; D. Luzi
National Research Council, Institute for Research on Population and Social Policies, Rome, Italy

Keywords
Medical device software, risk management, regulatory marketing procedures, software lifecycle, software safety

Summary
Objectives: The application of software in the Medical Device (MD) domain has become central to the improvement of diagnoses and treatments. The new European regulations that specifically address software as an important component of MD, require complex procedures to make software compliant with safety requirements, introducing thereby new challenges in the qualification and classification of MD software as well as in the performance of risk management activities. Under this perspective, the aim of this paper is to propose an integrated framework that combines the activities to be carried out by the manufacturer to develop safe software within the development lifecycle based on the regulatory requirements reported in US and European regulations as well as in the relevant standards and guidelines.

Methods: A comparative analysis was carried out to identify the main issues related to the application of the current new regulations. In addition, standards and guidelines recently released to harmonise procedures for the validation of MD software have been used to define the risk management activities to be carried out by the manufacturer during the software development process.

Results: This paper highlights the main issues related to the qualification and classification of MD software, providing an analysis of the different regulations applied in Europe and the US. A model that integrates the risk management process within the software development lifecycle has been proposed too. It is based on regulatory requirements and considers software risk analysis as a central input to be managed by the manufacturer already at the initial stages of the software design, in order to prevent MD failures.

Conclusions: Relevant changes in the process of MD development have been introduced with the recognition of software being an important component of MDs as stated in regulations and standards. This implies the performance of highly iterative processes that have to integrate the risk management in the framework of software development. It also makes it necessary to involve both medical and software engineering competences to safeguard patient and user safety.

1. Introduction

The growing use of medical device (MD) software has provided a great improvement in preventive, diagnostic and therapeutic activities, but it has also increased the complexity of the evaluation of patient safety considering both software to be integrated in MD and standalone applications used in healthcare settings. This has raised the number of MDs to be classified and thus subject to regulatory requirements, in particular considering its safety and estimating its risk classification. The manufacturer also has to put in place different procedures taking into account both quality management systems proper to any software development project and risk management that more accurately consider safety issues of MD software. These procedures have to be documented and have to cover the entire lifecycle of the product, from design to post-production monitoring. Several standards and guidelines have been drawn up to help manufactures fulfil these tasks. These are based on previous standards and tend to combine and/or consider the complex interaction of mechanical, electronic, software and hardware components from the perspective of MD safety requirements [1]. Of course, they also introduce changes to the whole MD development process that require the integration of different competences in order to assess risks from both clinical and software engineering perspectives.

Risk management in the MD domain has been analysed considering in particular the applications of the available techniques to detect risks and possible hazards. Some authors have addressed this issue proposing a Risk Management Capability Model based on SPI (Software Process Improve-
Risk management is a critical process that has been considered in different domains and, of course, from different perspectives. For instance, regulations and guidelines on software lifecycle consider risk management as part of the development of a project embedded in the organizational framework of a company that is developing new software. In particular, software engineering standards (such as IEC 12207:2008 [16]) emphasise the activities that reduce risks related to technical requirements such as functionality, performance, reliability and usability. When applied to MD software, risk management is considered “an integral part of a quality management system” as defined in ISO 13485:2003 [17]. The manufacturer has to put in place and document procedures of software lifecycle that demonstrate its safe design and maintenance (according to IEC 62304:2006 [18]), and systematically identify hazards of MDs (as required in ISO 14971:2012 [19]) as well as MD software (i.e. IEC 80002–1:2009 [20]). The accomplishment of these standards aims to reduce risks for patients, operators and the environment and balance them against the anticipated benefits of the newly introduced MD.

This process starts with the qualification of MD software and in particular the identification of standalone applications that represent a fundamental step, as they determine the risk class and consequently the procedures that have to be applied for risk management. Table 1 provides an overview of the main standards and guidelines to be applied in the US and Europe considering the different phases of software development.

### 2.1 Qualification of MD Software

According to IEC 62304 [18], an MD software is “a software system that has been developed for the purpose of being incorporated into the medical device being developed or that is intended for use as a medical device in its own right”. This general definition divides MD software into two main categories: standalone and incorporated in an MD. The latter is easy to qualify: it includes software that drives a device or influences the use of a device and falls automatically in the MD domain within systems that have to be assessed as a whole. Dissimilarly, when seeking to determine if a standalone application falls into the MD definition, a critical role is played by the software functionalities as well as by the intended use as defined by the manufacturer. Regulations in Europe and the US agree upon the identification of MD software when they are intended for: a) Analysis of patient data generated by MDs with a view to diagnosis and monitoring; b) Use for by patients to diagnose or treat physical or medical ailments. These categories include software that manipulate and analyse clinical data as well as applications that generate images, determine measurements and identify imaging regions of interest. On the contrary, they exclude systems used for healthcare administrative settings, or for epidemiological studies and statistics. However, Europe and the US adopt different rules to distinguish between standalone and software that is a part or accessory/ component of MDs. The FDA provides a comprehensive list of products along with their descriptions and intended use, while the new European Directive defines standalone software as an active MD, being dependent “on a source of electrical energy or any source of power other than that directly generated by the human body or gravity and which acts by converting this energy”. The further specification that “software for general purposes when used in a healthcare setting is not a medical device” has introduced a certain level of ambiguity to the interpretation of standalone software to be classified as MD [3, 32]. To help the manufacturer to solve this ambiguity, the European Commission has delivered a guideline (MEDDEV 2.1/6, 2012) [21] that provides a workflow to identify standalone MD software, as well as some examples of computer programs that manipulate and analyse clinical data.

These approaches also lead to a different interpretation in the qualification of border-line software such as the Electronic Healthcare Record (EHR) systems. European Directives exclude these systems, given that actions performed by the software are limited to the “representation of data for medical purposes” [22]. FDA instead considers EHR one of the main sources for practitioners to make diagnoses and establish patient treatment, according it requires enforcement discretion to determine the applicability of MD regulation on each specific EHR system.

### 2.2 Classification of Risk

In order to bring a new MD to the market, it is necessary to demonstrate its safety and efficacy performing conformity assessment procedures. In Europe this is regulated by the EU Directives that provide the description of the essential requirements that a manufacturer has to demonstrate in order to get a CE mark. In the US, the manufac-

---

**Table 1**

<table>
<thead>
<tr>
<th>Phase</th>
<th>US</th>
<th>EU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The manufacturer has to provide objective evidence of the MD’s conformity, either by submitting the 510(k) forms or following a Pre-marked Approval (PMA) procedure. These procedures take into account the MD risk classification associated with its intended use. Once the device’s intended use is defined, the manufacturer has to establish the MD risk class according to classification rules provided by the relevant authorities. Also, software is evaluated and classified under three risk classes: Class A: no injury; Class B: no serious injury, Class C: death or serious injury. These are equivalent in Europe and the US and determine the performance of specific activities as well as the production of documents that demonstrate software safety.

Europe and the US adopt different methods to determine the software security level: in the US, the FDA has published a specific guide, updated in May 2005 [24], where the manufacturer can determine the “level of concerns” for its software by answering to a set of questions. Conversely, in Europe no specific method is provided to establish the risk level of an application and the manufacturer determines the security class of the software considering the possible severity associated with each identified malfunction, as described by the standard IEC 62304:2006 [18].

Software that is a part or accessory of MDs has the same risk class as the related MD, while standalone software follows the rules applied to active MDs in which each module has to be evaluated considering the risk posed to the patient and user. According to IEC 62304:2006 [18], that guides manufacturers in establishing the safety and effectiveness of an MD software, the overall software system assumes the highest classification contained of all software elements.

### 2.3 Risk Analysis

Risk identification is a probabilistic analysis based on statistical and experimental data as well as on theoretical considerations that take two essential aspects into account: the probability that an injury can occur and the severity of its consequences. At the end of this phase the manufacturer develops the risk analysis document that describes the expected risks. For each of them it reports how to reduce the probability of their occurrence and then provides the assessment of acceptability.

The same procedure is also implemented by the manufacturer on the software, whether it is standalone or embedded in a MD. Distinctly from the MD risk

<table>
<thead>
<tr>
<th>Reference phase</th>
<th>Standard/Guidelines</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole MD software development process</td>
<td>IEC 62304:2006. MD software – Software life-cycle processes [18]</td>
<td>Definition of the requirements for the development and maintenance of both standalone and software embedded in a MD.</td>
</tr>
<tr>
<td></td>
<td>FDA Guidance for the Content of Premarket Submissions for Software Contained in MDs:2005 [24]</td>
<td>Guide on the documentation that manufacturers should include in premarket submissions for MD software, including standalone applications.</td>
</tr>
<tr>
<td>MD design and development</td>
<td>ISO 13485:2003. MDs – Quality management systems – Requirements for regulatory purposes [17]</td>
<td>Based on ISO 9001 [27], it specifies requirements for a quality management system that can be used by an organization for the design, development, production, installation and servicing of MDs. It includes the design, development, and provision of related services.</td>
</tr>
<tr>
<td></td>
<td>21 CFR part 820 – Quality System Regulation [28]</td>
<td>Good practice in the design, development, storage, installation and servicing of MD. Equivalent to ISO 13485 [17].</td>
</tr>
<tr>
<td>Validation</td>
<td>General Principles of Software Validation; Final Guidance for Industry and FDA-2002 [29]</td>
<td>Description of the MD quality system regulation that has to be applied to perform software validation.</td>
</tr>
<tr>
<td>Safety</td>
<td>IEC 60601–1-X [30] collateral standards</td>
<td>Definition of safety and performance requirements. IEC 60601–1–4 refers to MDs that incorporate programmable electronic systems (i.e. software).</td>
</tr>
<tr>
<td></td>
<td>IEC 61010–1:2010 [31]</td>
<td>Specifications of safety requirements for different types of electrical equipment and their accessories, such as Electrical laboratory equipment.</td>
</tr>
</tbody>
</table>
analysis, software has to be based on systematic models since it is not possible to determine the probability of a malfunction using traditional statistical methods. For this reason both Europe and the US recommend that the manufacturer performs the risk analysis considering the severity of an injury and assuming that the probability of its occurrence is equal to 1 (i.e. is certain to occur) (IEC 80002–1:2009 [20]).

3. Risk Management within the Software Lifecycle: a Proposal

In this paragraph we propose a framework that integrates the software development lifecycle and risk management process. Our hypothesis is that risk management applied to MD software represents a crucial complex macro-activity that provides the input to model the application starting from the preliminary activities of software planning, design and development. Figure 1 shows a high level description of the software development lifecycle based on IEC 62304:2006 [18] (upper part of the diagram) and the risk management process based on IEC 80002–1:2009 [20] and ISO 14971:2012 [19] (lower part of the model). The interaction between these processes is described highlighting the main activities to be carried out by the manufacturer in the development of safe software, whether it is embedded within an MD or a stand-alone MD software (middle part of the diagram). Further, some available techniques for risk analysis are reported and analyzed throughout the software development lifecycle [1, 33].

The software development lifecycle is represented as a sequence of the high level activities implying a pure “waterfall” process in which each phase starts when the previous one is completed. This methodology is typically used to develop safety critical software [34]. However, the proposed framework is not strictly dependent on a specific lifecycle model and can be adopted using other implementation strategies, such as V-model [2] and iterative model [35]. This is in line with the regulatory requirements and development standards such as ISO 62304:2006 [18] that do not recommend a specific lifecycle when developing MD software. As mentioned in paragraph 2.2, one of the key activities to be carried out by the manufacturer before the development of the software application concerns its safety evaluation and classification. This classification determines the type of documentation to be provided as well as the activities to be performed. For instance, during the design phase the identification of decoupling items is required only for class C software and the refinement of software architecture into units is required only for class B and C software. Since the safety classification might not be determined at the beginning of the software development process, the manufacturer has to apply the highest class until a safety class can be assigned.

3.2 Software Design

When software requirements have been defined the design phase can start. It can be divided into two activities: architectural and detailed design. In the former activity the requirements are transformed into an architectural framework describing the overall software structure and identifying the items that compose the software application as well as the interfaces between them and with other parts of the whole MD (i.e. hardware and/or software). The detailed design activity consists of an iterative process to refine the software architecture into further software items and/or units. Once the architecture has been determined, the manufacturer has to define the safety class of each item and re-classify the whole software applying the highest class (see paragraph 2.2). In the design phase the manufacturer has also to recognize which software objects can be related with potential safety concerns. On the one hand, he/she has to identify which item can cause a hazardous situation and thus perform the overall risk management process to identify potential risks and control measures to mitigate them. On the other, the manufacturer has to identify items that can contribute to the implementation of risk control measures and consequently update the risk management plan of the overall MD. Also in this phase of the development process, after performing risk management the manufacturer has to determine which items can lead to unacceptable residual risks and, if necessary, update the de-
sign of the software architecture identifying, for instance, items that can help mitigate these risks implementing specific risk control measures. In this phase risk analysis is further updated and two alternative methods can be adopted by the risk manager to identify and prioritize hazards and hazardous situations: Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA). The former is particularly used early in the design and development phases. It is based on top-down deductive analysis where an identified potential failure of the system (called top event that represents the root of the tree) is used to detect all contributing failures of single or multiple faults of a single component of the complex system. Conversely, the FMEA is a bottom up method in which effects or consequences of a single component are identified and evaluated. It follows that this method is often used in the later stages of the development process, when the software design is more mature and the system architecture is more detailed. Furthermore, unlike FTA, FMEA does not consider external events that can result in a hazardous situation, for instance a fire that can damage a server farm resulting in a loss of data.

3.3 Software Implementation, Testing and Release

When the detailed architecture is defined the manufacturer implements and verifies the single unit, integrates them into the whole system and performs a system testing activity. In terms of risk management the manufacturer has to verify the risk control measures and if necessary repeat the risk management process based not only on the new possible risks identified, but also on the anomalies that can cause a hazard. If the risk management process determines unacceptable residual risks the implementation and verification activity is triggered again to identify possible updates to both risk control measure functionalities and items that are related with unacceptable risks. If the overall residual risk is judged unacceptable the manufacturer should determine if the medical benefits of the MD outweigh the overall residual risk. In this case the overall residual risk can be
judged acceptable and the manufacturer can release the product. During these stages of the development process the risk manager can update the risk analysis using the Hazard and Operability Study (HAZOP) method to verify and then optimize design concepts or changes. This method is based on the assumption that hazards can be caused by the deviation of the system from the software design intent.

### 3.4 Software Maintenance

This process often comprises two main activities: 1) establishment of the maintenance plan; 2) problem and modification analysis. In particular in the latter, the manufacturer has to establish monitoring procedures to collect feedback from users. Each problem notified by end users has to be evaluated to determine whether it affects the safety of the released product and whether it requires changes to the released version. Modifications in the software design, implementation and verification imply an iterative process in which the detection of new possible hazards triggers again both the software development and the risk management processes to identify new risks as well as the update and/or the implementation of new risk control measures. During the maintenance stage of the development process, the risk manager can update the risk analysis using the methods reported in the development phases. In addition, in order to collect and analyze data submitted by end users, the risk manager can use the Failure Reporting, Analysis and Corrective Action System (FRACAS) to report, classify and analyze failures as well as to plan corrective actions to eliminate or mitigate the risks related to such failures. FRACAS can also be integrated with different software applications (such as Weibull) facilitating risk analysts’ use of the failure data to perform statistical analysis.

### 4. Conclusions

The recent enforcement of the EU Directive as well as the release of standards and guidelines on MD software have introduced important changes in the process of MD development both for the MD industry and software development companies. They have to conform to regulatory requirements to demonstrate safety and reliability of the MD software, putting in place procedures to perform risk management within the framework of quality management. The development of software in the MD domain makes it necessary to involve a multidisciplinary team that includes risk management practitioners who have to assess risk from a clinical and patient/user perspective, as well as software engineers who need to specifically address safety requirements within the software development process. The interaction between different disciplines allows software developers to contribute to the overall safety of the MD already at the early stages of its design, also considering the role of software before the MD design is finalized.

In this framework the software development process implies a careful analysis of possible hazards resulting from the interaction between software and other MD components as well as the identification of potential failures of single and/or related software items. This makes it necessary to perform risk analysis at each phase of the software development process, carrying out refinement activities to assess possible hazards and implement risk control measures in a highly iterative process. In this perspective, the proposed framework emphasizes the role of the risk management as an essential activity to be carried out by the manufacturer wanting to design and develop safe and effective MD software. In our view, information resulting from the risk management activities, i.e. from the risk analysis to the assessment of the acceptability of any residual risk, should be included as input at the different phases of the software development rather than considered simply the result of an isolated activity. The integration of these processes should be performed already at the requirements stage since the quality of both user and regulatory requirements have a direct impact on the quality of risk management for the determination of risk sources and methods to mitigate them. In this paper, we also include a set of tools to facilitate the manufacturer in the identification and potential failure of the whole system, as well as of the role of software in hazardous situations. These tools use different approaches and are specifically adopted at different stages of the software development lifecycle. The results deriving from appropriate use of these techniques throughout the design and development process can be used as input for the overall software development process and lead to a reduction of the MD development cycle without compromising safety requirements.

## References

12. Issler L, Spreeckelsen C, Welzel C. Implementing Software Development Guidelines in a Medical
31. IEC 61010-1:2010, Safety requirements for electrical equipment for measurement, control, and laboratory use – Part 1: General requirements.