A Survey of Software Engineering Techniques in Medical Device Development

Raimund L. Feldmann¹⁾, Forrest Shull¹⁾, Christian Denger²⁾, Martin Höst³⁾, Christin Lindholm³⁾

¹⁾ <i>Fraunhofer Center for Experimental Software Engineering (CESE) 4321 Hartwick Rd - Suite 500 College Park, MD 20742 USA</i>	²⁾ <i>Fraunhofer Institute for Experimental Software Engineering (IESE) Fraunhofer-Platz 1 D-67663 Kaiserslautern Germany</i>	³⁾ <i>Department of Communication Systems Lund University Box 118 221 00 Lund Sweden</i>
<i>{rfeldmann, fshull}@fc-md.umd.edu, Christian.Denger@iese.fraunhofer.de, {Martin.Höst, Christin.Lindholm}@telecom.lth.se</i>		

Abstract

A wide variety of the functions provided by today's medical devices relies heavily on software. Most of these capabilities could not be offered without the underlying integrated software solutions. As a result, the medical device industry has become highly interdisciplinary. Medical device manufacturers are finding an increasing need to incorporate the research ideas and results from traditionally disconnected research areas such as medicine, software and system engineering, and mechanical engineering. In 2006, we conducted a survey with more than 100 companies from Europe and the USA to shine some light on the current status of the integration of software engineering technologies into the medical device domain. The initial results of this survey are presented in this paper. Both software engineers and the medical device industry can use these findings to better understand current challenges and future directions, to achieve a better integration of the fields.

1. Introduction

Today, many medical devices could not fulfill their intended use without the software embedded within them, which implements a variety of functions and features. Surveys of trends in the medical device industry (e.g., [1], [2], [3]) indicate that software is one of the most decisive factors for producing innovative

products with new capabilities, and predict that the importance of software will only further increase in the future [4]. Studies also predict that the research and development (R&D) investment in software in this market will increase to 33% of the overall budget by 2015 [2].

As the role of software in the medical device domain increases in importance, so do the failures due to software defects. An analysis of medical device recalls by the FDA in 1996 [5] found that software was increasingly responsible for product recalls: In 1996, 10% of product recalls were caused by software-related issues. This was up from 6% in the years 1983–1991. A German survey on medical device recalls in the medical sector indicates that software is the top cause for risks related to construction and design defects of medical device products. This analysis, from June 2006, shows that 21% of the medical device design failures are caused by software defects [6]. This is an increasing trend, since the same figures from November 2005 show software responsible for 17% of construction and design defects.

To address such issues, the development of medical device software is regulated by various standards, laws and recommendations (e.g., [7], [8], [9]). In general, these standards describe software life-cycle models that should be implemented by manufacturers. The overall objective is the definition of general process steps and intermediate work-products. Adhering to the regulations and following the specified processes increases an organization's ability to produce safe, high

quality medical device software. However, in many cases the standards are quite vague regarding the concrete software engineering techniques that should be used in different development steps. Thus, in practice there is a high degree of freedom in instantiating the processes. This may be an indicator that currently software engineering standards are only loosely integrated into this domain.

Given this context, and the general lack of empirical knowledge about the state of the practice regarding medical device software development, we designed an international survey. Our objective was to understand the current goals and concerns of companies in this market and how they choose software practices to address the issues that they see. We were interested in understanding the extent to which the standards that have been developed have been recognized and instantiated by industry. Furthermore, we were interested in eliciting the most important challenges with respect to developing embedded software for medical purposes. These results can be used by researchers and practitioners to get an overview of the level of usage of various processes and tools in the domain as well as to understand useful targets for developing new techniques and methods aimed at further improving software quality in the medical sector.

This paper presents the basic statistical results of that survey, which was run in 2006. The questions targeted in the survey can be summarized as:

- 1) How can the medical device area in general be characterized regarding software engineering?
- 2) What are the most frequently applied software engineering techniques, methods and tools during medical device production?
- 3) What are the most recent challenges with respect to software engineering in medical device production?

More than 100 companies from Europe and the USA participated in the survey, ranging in size from small organizations with less than 20 employees to global players with several thousand developers. The products of the companies range from devices implanted during surgery to diagnosis systems to non-invasive devices and information systems. Most of these products are classified as products of class II (Special Controls) according to the FDA standards and class IIa/IIb according to European standards (i.e., devices with medium risk such as electro-medical devices). On this basis, the survey covers a wide cross-section of the industry.

To our knowledge, this is the first survey of this size that explicitly focuses on the topic of software

engineering in the medical device domain. We perceive the results as a basis for the identification of current and future challenges with respect to software engineering and quality assurance in the field. Researchers and practitioners can use the identified challenges as input for...

- ... a better integration of software engineering standards into the applied processes and standards.
- ... the development of new, innovative techniques and methods to further improve software quality, tailored to the medical device domain.

The remainder of this paper is structured as follows: In section 2 we describe the design of our survey. Key objectives are discussed, as well as our sample target group and the chosen approach for collecting the data. Next, we present an overview of the survey outcomes (section 3). Insights into the original data set as well as the methods used for analysis are given. Based on the results we provide an interpretation of the findings from a software engineering point of view. We summarize our findings and conclude in section 4.

2. Survey design

This survey was designed by software engineering researchers from three institutions in the United States and Europe (the Fraunhofer Institute for Experimental Software Engineering in Germany; the Fraunhofer Center in Maryland, USA; and the Lund University, Sweden). The main objective of the survey was to characterize the state of the practice of software development in the specific context of medical devices and medical information systems, in order to understand the extent to which software engineering practices are integrated and used in the medical device domain. In all cases, our objective was to characterize the state of the practice in this context. The survey collected no information that could be used to directly evaluate the effectiveness of the practices that are being applied, nor did it collect information that would compare respondents in this survey to those in another domain.

2.1. Sample and target group

The target population for the study consists of organizations developing software for medical devices and medical information systems. This is a large population and it is impossible to carry out a study with the complete population where every organization is

included. Therefore, a sample of the population has to be chosen.

The invited organizations were chosen based on the contacts that the authors had and were able to obtain in the initial steps of the survey. Email invitations were sent to all of these contacts as well as to contacts in the databases of large industry associations. These associations supported the authors in conducting the survey by mailing out the link to the questionnaire throughout their distribution lists. However, since we were not allowed to access these third party databases directly, we do not know how many invitations were sent out in total. Obviously, the chosen methodology of the survey does not necessarily result in a rigorous random sample of medical device companies. However, in the given circumstances it was the best possible design we could come up with for our research interests.

2.2. Conducting the survey

The survey was carried out through a web-based questionnaire. Potential participants were given a link to the URL and asked to fill out the questionnaire. In order to motivate people to participate they were given the possibility to register their email in order to obtain the result of the survey. As a further motivation for participants we donated \$1 to the International Red Cross for each completed on-line survey.

3. Collected data

As a response to the invitations, 349 individuals looked at the starting page of the questionnaire. From these, survey responses at some level of completeness were received from 113 participants.

Since it was possible to skip questions when answering the survey, some respondents did not provide answers to all survey questions. We filtered the dataset by deciding on a minimum threshold, below which too little information had been provided to be usefully included in the analysis. The survey ends with a set of characterization questions about the respondent's organization and the developers working there. Only the answers of those respondents who have answered at least one of these characterization questions are included in the survey. We believe that subjects, who reviewed all of the questions, even if they did not provide answers to all of them, are likely to provide the most valid data. Because of the filtering, the dataset was left with 57 valid responses, which were

included in the analysis. In the following, we call this set of companies the "core data set."

The respondents come from many different types of companies. 71% of the companies participating in the survey are small and medium sized companies with 10-250 employees. Within these companies, the size of the software development departments / teams varies. Almost 50% of the software development teams are smaller than 11 people. Only 18% of the companies have development teams with more than 50 people. Of the 57 companies from the core data set, most respondents came from Germany (38), the USA (8), and Sweden (5).

With these numbers, our survey may not be indicative of the state of the entire industry. However, a large percentage of our respondents were alike in having primary focuses on safety critical software. Thus, our result can shed some light on software engineering practices in this specialized area. In addition, the results seem to be consistent with the experience of some domain insiders from which we received feedback, and thereby seem to provide a fair characterization of the industry.

3.1. Analysis of data

One of the most important questions focused on how companies who produce medical devices obtain the software components (i.e., whether the software is developed in-house or obtained from outside). Of the 57 respondents, 20 companies use only software developed in-house in their products and seven companies use only third party software in their products. 29 companies do both, obtaining some software from third parties and integrating it with software developed in-house. One company stated that their products do not contain software.

This has implications for data that can be used in further analysis. We use 49 respondents for questions dealing with software development and 36 respondents for questions with respect to software from third parties.

The analyses have been carried out with two data sets. First, the analysis was carried out with the core data set consisting of the answers from the 57 people who had answered the characterization questions. Where relevant, the results from the core dataset were further tested by analyzing the entire set of 113 respondents who answered the entire questionnaire. In no case were the results of this different from those for the core dataset, so we do not report these cases explicitly in the following sections.

Figure 1. Educational background of software developers

Figure 2. Importance and safety criticality of software

Figure 3. Development activities from which most issues originate

3.2. Characterizing software development in the organization

Participants were asked to characterize the educational background of the majority of the members of their software development team. Figure 1 shows the results: In 64% of the cases, the majority of the software developers have a background other than computer science. These developers come from disciplines such as electronics, medical sciences, or electrical systems engineering. Only in 36% of the cases is the medical device software created mainly by computer scientists.

Figure 2 indicates the relevance of software in medical device products. It is evident that software is an important part in many medical devices: 98% of the companies rate software as either a very important (84%) or important (14%) component of their products. The figure also indicates that software is a safety-critical element of the product in over 75% of the cases. In contrast, for only 16% of the companies does the software clearly realize non-safety-critical functionalities.

Asked about some of the perceived challenges, 64% of our participants noted that finding sufficient software developers (i.e., staff with a computer science background) is a challenge. At the same time, most issues regarding software quality stem from activities involved in planning the software, the functionality it should accomplish, and how it will accomplish it. That is, most challenges stem from requirements activities (63%) and architecture and design activities (16%) (see Figure 3). Actually implementing the software code is perceived as the most challenging activity by only 10% of the respondents. This indicates that investments in requirements engineering activities seem to be most promising to gain significant improvements in the software development process.

Looking in more detail at issues that cause problems for each type of activity, for requirements-related activities 86% of the companies perceive changing requirements as the main problem. Missing requirements (33%) and misinterpreting requirements (39%) are also perceived as important.

Related to architecture and design, the main challenge is missing information in the diagrams (53%) and inconsistencies between the planned architecture and the software (39%). Missing opportunities to reuse software code in a systematic way (33%) and difficulties in maintaining the software code (29%) are perceived as the main issues during implementation.

Figure 4 shows that around 50% of the companies follow a defined process to perform the activities mentioned above on a regular basis, that is, they said that they always or frequently follow such a process. (If the criteria are loosened to include companies that follow defined processes in about half of their projects, then 78% of the respondents had a defined process for implementation, 71% had one for architecture, and 69% had one for requirements).

3.3. Characterizing the challenges of using notations and tools

In order to document the results of the various activities involved in constructing software, different notations and languages can be applied. Most of our respondents were using relatively informal notations and techniques to do so. Formal languages (e.g., temporal logic, architecture description languages) describing software requirements or architectures were rarely utilized. For example, for describing software requirements as well as architecture and design, only 2% of the companies use formal languages in all of their projects. In 22% of the companies, formal languages are used frequently in the requirements phase and in 14% formal languages are used frequently for architecture and design.

Consequently, less formal notations and languages are most popular for use in all types of activities. Figure 5 shows the results for requirements engineering. There, natural language (e.g., English or German) is used in almost all companies in all projects. In 92% of the companies this kind of notation is used always or frequently.

A detailed analysis of the answers reveals that for 46% of the companies natural language is the one and only notation to specify requirements. Structured notations such as use cases are used by 40% of the companies on a regular basis (i.e., always or frequently used).

For architecture and design, structural diagrams are the most frequently applied notations for modeling the software. These diagram types (e.g., class diagrams, package diagrams, functional block diagrams) are used by 64% of the companies on a regular basis (always or frequently applied). Sequence and data flow diagrams seem to have a lower importance. These diagrams are frequently used by 40% and 36% of the companies, respectively. More formal notations such as state charts and/or Time Petri Nets seem to be of low importance in the medical device domain. State charts are used on a regular basis by 23% of the companies,

Figure 4. Defined processes for development activities

Figure 5. The usage of natural language in the requirements phase

Figure 6. Usage of tools for different activities

while Time Petri Nets are not applied regularly by any respondent. Only 6% of the companies occasionally (i.e., less than half the time) use this notation.

Many tools exist that assist with various constructive techniques (e.g., requirements management tools, design modeling tools, test execution tools). Software engineers have long found that, in order to make effective use of a tool, it is important to have a good process definition in place. A tool cannot overcome deficiencies regarding the processes followed. Consequently, frequent tool usage may be an indication of higher process maturity in software development activities when paired with well-defined processes. However, Figure 6 shows that tools are not frequently used in the medical device domain. For example, even though testing can be easily supported by tools, less than 13% of the companies use tools on a regular basis for this purpose.

Tool support for other types of activities is also infrequent. Specifying and managing requirements is tool-supported in 20% of the companies. (For this purpose DOORS™ (16%) and RequisitePro™ (10%) are the most frequently used tools.) Architecture and design activities are supported by tools in 35% of the companies. Here UML modeling tools (25%) had the highest importance. Tools that are typically applied in many other domains of embedded software (e.g., in the automotive industry), such as Matlab Simulink or LabView, seem to have a lower importance in the medical device domain. Matlab is applied by 8% of the companies on a regular basis, LabView by 12%.

Risk management activities seem to be the activities that are most frequently supported by tools (63% of the companies use a risk management tool on a regular basis). However, these numbers must be carefully interpreted. A detailed analysis of the data shows that almost all of the companies state that they do not use commercial tools specialized to support this activity but common applications such as text editors (like MS Word™) or tabular calculation sheets (e.g., MS Excel™).

3.4. Characterizing quality assurance for software

Quality assurance activities are an integral part of software development processes. Especially in safety-critical domains such as the medical device domain, risk management activities that are used to analyze, control and monitor safety risks of the devices and the software included therein are of high importance.

We found that software quality assurance activities are important in the overall medical device development process. 60% of the companies frequently plan quality assurance activities, while another 14% establish such plans for about half of their software development projects. In addition, we found that the results of software quality assurance techniques are frequently documented. 62% of the companies always or frequently document them.

However, mature software quality assurance activities are supported by activities aimed at quantitatively assessing progress and quality, such as the collection of code or design metrics or performing benchmarks of the software. It seems that these supporting activities are not applied in a wide range of medical device companies. Less than 5% of the companies always collect code or design metrics of their software. Benchmarks are frequently used by fewer than 13% of the companies. The most frequently applied supporting activity is defect classification: 35% of the companies use a defect classification to understand better the nature of the software faults. However, it remains unclear how detailed these classifications are defined in practice. Consequently, judging their value for supporting quality assurance planning is not possible.

Among the quality assurance techniques, testing was the one most frequently used. System validation activities applied to the software (i.e., testing / validating that the software fulfills its requirements) are performed by 78% of the companies in all projects and in a further 16% of the companies frequently. Inspections and reviews (i.e., the verification of the quality of intermediate development products) are also applied quite frequently: 32% of the companies perform this activity in all of their software development projects, 24% apply the technique frequently, and another 16% perform inspections in half of their projects.

Risk analysis can be performed using different techniques such as Fault Tree Analysis (FTA) [10] or Failure Mode and Effect Analysis (FMEA) [11]. These techniques are recommended by various standards as suitable approaches for systematically identifying the most relevant risks during development of a medical device. Our findings indicate that FMEA is the most frequently applied technique for risk analysis. 42% of the companies always or frequently use the FMEA technique for this purpose. FTA seems to be of lower importance as only 18% of the companies always or frequently use this technique. However, only 18% of the respondents stated that they perform software

FMEA on a regular basis in their development processes. Since this is rather less than the number of respondents applying FMEA in general, companies seem to be applying the FMEA technique at the system level, rather than specifically to the software component. This finding is consistent with our practical experiences that even though many developers understand that risk analysis should be performed on software and software components, it remains unclear in practice exactly how this should be performed. Other risk analysis techniques (e.g., hazard and operability analysis (HAZOP) [12], reliability block diagrams, or event trees) seem not widely used in the medical device domain as they were not mentioned by the respondents as alternative techniques for risk analysis.

4. Conclusion

Our survey supports the assumption that software engineering methods, techniques, tools, and standards could be better integrated into the used development processes and existing standards for medical devices. One reason for this might be the fact that software in organizations developing medical devices is mainly created by developers with a background other than computer science. Even though we currently do not know of studies regarding the differences of computer sciences vs. non computer sciences in developing software in specific domains, the findings for the medical device industry are similar to those in the automotive or embedded system industry. In all of these disciplines, which traditionally have more of a system engineering legacy, the number of computer scientists who are developing the software is relatively low while at the same time the issues with the produced software are relatively high.

Since the availability of computer scientists may not significantly increase in the near future, training programs tailored for software developers in this specific domain may be helpful. Such training programs need to be developed in a joint venture between medical experts, software engineering experts and (hardware) developers.

Activities related to the requirements phase are perceived as the major source of issues regarding software quality, which is thus a primary target for improvement activities. Focusing on this phase is also important given the fact that errors in this phase are often not detected until the very end of the development process. Hence, in order to fix these errors, it is necessary to go through some rework in all development phases, which is obviously more

expensive than fixing errors that only affect one or maybe two development phases. In 50% of all cases requirements are managed with commercial standard applications (e.g., MS Word™ or MS Excel™). The use of natural language (instead of formal languages and notations) seems to go along with these findings, since it is known that natural languages offer room for ambiguities (e.g., [13]). Better integration of software engineering knowledge (i.e., methods, techniques, tools, and standards) into the requirement phases seems to be a promising way to save effort and thereby development cost without reducing the necessary quality of the developed software.

Because only about 50% of the organizations represented in this survey follow defined processes, it is not surprising that tools in general are used to a lesser extent than in other domains. Similar results hold for quality assurance activities. This is likely an indicator that verification and validation techniques commonly used in other domains, or process standards like SPICE and CMMI®, cannot simply be applied for this domain but need customization to be successfully integrated. This calls for actions on both sides, the medical device industry as well as in the field of software engineering. Besides the development of new, tailored solutions for the domain, additional training for the development teams seems to be a promising way. Integrating better guidance on how and when to use which tools could help to improve the existing standards.

Studies like the survey described in this paper can help to identify focuses for improvements and allow for goal-oriented actions to further integrate software engineering methods, techniques, tools, and standards into the medical device domain. For researchers interested in starting this work, a more complete set of survey results can be obtained by request to the authors. We are currently in the process of conducting further analysis on the collected data, which will include correlations between different variables (e.g., the developer's background and tool usage). Outcomes of these results will be made public as they become available.

Acknowledgements

We would like to thank the numerous participants of the survey for providing their feedback. The help of Andreas Schlichting from the Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, in summarizing the raw data sets is much appreciated.

References

- [1] Future Trends in Medical Device Innovation. Advanced Medical Technologie Association (AdvaMed), 2004, <http://advamed.org>
- [2] ITEA 2 Blue Book. September 2005, www.itea-office.org
- [3] Wirtschaftsfaktor Gesundheit: Chancen und Potenziale für Deutschland. BDI-Initiative Vitale Gesellschaft, Bundesverband der Deutschen Industrie, 2005, <http://www.vitalegesellschaft.de>
- [4] Studie zur Situation der Medizintechnik in Deutschland im internationalen Vergleich. Bundesministerium für Bildung und Forschung, <http://www.bmbf.de>
- [5] Dolores R. Wallace and D. Richard Kuhn: Failure Modes in Medical Device Software: An analysis of 15 years of recall data. International Journal of Reliability, Quality and Safety.
- [6] Bundesinstituts für Arzneimittel und Medizinprodukte, 2006, <http://www.bfarm.de>
- [7] ISO 14971:2000(E): Medical Devices - Application of risk management to medical devices. Genf, 2000.
- [8] IEC 60601-1-4, Medical electrical equipment - Part 1-4: General requirements for safety - Collateral Standard: Programmable electrical medical systems. Ed. 1.1, 2000.
- [9] CDRH, General Principles of Software Validation; Final Guidance for Industry and FDA Staff. 2002, (see also <http://www.fda.gov/cdrh/comp/guidance/938.html>)
- [10] Fault Trees And Reliability Block Diagrams. <http://www.weibull.com/basics/fault-tree/index.htm>
- [11] Glenn D Krasker: Failure Modes And Effects Analysis: Building Safety into Everyday Practice. HC Pro, Inc., 2004.
- [12] The HAZOP (Hazard and Operability) Method. http://www.acusafe.com/Hazard_Analysis/HAZOP_Technique.pdf
- [13] Erik Kamsties: Surfacing Ambiguity in Natural Language Requirements. PhD Thesis in Experimental Software Engineering, University of Kaiserslautern, Kaiserslautern, Germany. Published in 2001 by Fraunhofer IRB Verlag, Stuttgart, Germany, ISBN 3-8167-5903-3.