Are the Current Architectural Practices Suitable for Safety Aspects of Medical Devices? An Exploratory Investigation.

Fabio Leite *, Pablo Oliveira Antonino †, Paulo Barbosa *, Soeren Kemmann † and Raphael Mendonca *
* State University of Paraiba, Campina Grande, Brazil
Email: {fabio, paulo}@cct.uepb.edu.br, raphael.mendonca@nutes.uepb.edu.br,
† Fraunhofer IESE, Kaiserslautern, Germany
Email: {pablo.antonino and soeren.kemmann}@iese.fraunhofer.de

Abstract. *We have investigated approaches in the literature to assess the quality of the software architectures of medical devices, and have found evidence that there is a lack of methodologies for evaluating the software architecture design aspects of medical devices that might affect system safety. Such evidences were identified when evaluating the software architecture of the FDA Generic Infusion Pump searching for architectural evaluation approaches available in the literature. In order to fill this gap, we propose a set of quality questions that focus on analyzing software architecture design aspects of medical devices aiming safety. We show arguments on why reference projects such as the FDA Generic Infusion Pump system must satisfy our new quality questions. The quality questions were integrated into a quality model commissioned by the Brazilian Health Ministery for the certification of medical devices.*

I. Introduction

Nowadays medical devices have an increasing number of software-intensive features. The incorporation of embedded processors into these devices allows benefiting from advantages such as size reduction, more control, less physical interaction, and tolerance to hostile environments, among others [1]. Naturally, the more software is embedded into these devices, the more design control is necessary across different software engineering disciplines. Quality attributes (QA) of software system architectures can be defined in many ways according to [2], and a set of non-functional requirements has been proposed to assess product quality. In this sense, one important issue to mention is that the design in the medical devices industry is very strongly affected by regulations [3]. The regulation states that the most relevant quality attribute to be considered is safety, due to the nature of laws, accident prevention, and so on. Therefore, due to our experience with regulations [4], the main QA to be considered when assessing architectures of medical devices is safety.

Recently, the Brazilian Health Ministry has started some initiatives and is investing resources into improvements of the quality of medical devices. One of these initiatives concerns the definition of a new quality model for assessing software-intensive systems [5] by evaluating the safety arguments for all the control features. This quality model is being developed by the Fraunhofer IESE in collaboration with the NUTES project, which is one of the main actions by the Brazilian government to generate know-how on how to assess and develop medical devices with the requested quality by surveying its usage in Brazil. NUTES stands for Nucleus for Strategic Health Technologies, and it is physically located at the State University of Paraba.

The quality model is being developed focused on the safety aspects of embedded software in electromedical devices, once the very few models already established in standards and in the literature are not focused in safety as the main quality attribute, what, consequently, imposes difficulties in achieving the actual interests in quality assurance. Our quality model includes questions for analyzing the following engineering disciplines: requirements, architecture, testing, code, and usability. Among these, in this paper we focus on the portion of the quality model that deals with evaluation of software architecture aiming safety.

We have observed that the set of relevant questions defined in the quality model for safety assessment of software architectures of medical devices provides a good basis for arguing with software architects and other professionals involved in the development of medical devices about the importance of architecting the software considering safety as a key quality attribute.

This paper is structured as follows: In Section II a review of the literature is provided to support our argument that we still need to care about safety when assessing systems and software architectures. Section III analyzes and criticizes a set of reusable questions identifying specific gaps if such questions were employed for regulatory purposes. Section IV introduces and justify new questions for safety assessment. Section V discusses other works that have safety as a concern for evaluating systems and software architectures. Finally, Section VI presents our final impressions about the work and how it continues being improved.

II. Investigating Safety Concerns in Architectural Assessment Approaches

ISO 42010 [6] defines architecture as fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution. In this regard, it is possible claim that architecture is a key artifact, considering medical device as software-intensive system.

Actually, the architectural decisions are directly related to the quality attributes exposed by the system [7]. Therefore, it is mandatory to evaluate the architectural decisions to check their impact (positive or negative) the safety of a medical device. However, there is a lack of evaluation methodologies concerning to safety, although several previous works related to architecture evaluation can be found in the literature [8].

Even well known methods such as the ones from the Software Engineering Institute: Software Architecture Analysis Method (SAAM) [9], Architecture Tradeoff Analysis Method (ATAN) [10] and Active Reviews for Intermediate Design

(ARID) [11]. SAAM consists of evaluating the architecture according to scenarios that were prioritized, weighed and scrutinized with the stakeholders. The main tangible outputs of the SAAM are: scenario coverage in the architecture, a concrete idea of the critical portions in the architecture that have potential complexity, and potential points of possible changes.

ATAN is an early evaluation method that intends to evaluate how well an architecture satisfies particular quality goals and how they impact each other. The main idea is to identify problems at early stages of the development process in order to minimize the efforts to solve them. Furthermore, ATAM aims at identifying and documenting all possible architectural approaches in order to select the best one according to a quality attribute utility tree generated by the scenarios elicitation and prioritization.

Finally, ARID is an approach that is suitable for evaluating an incomplete architecture or a partial design. This is a lightweight evaluation approach that concentrates on the suitability of the new architecture according to scenarios and some questions that can help the reviewer to conduct the evaluation. This technique can be combined with ATAN to improve some aspects in the overall evaluation; for example, ATAN can put together more concrete and necessary artifacts to be analyzed in early phase of the ARID evaluation.

Despite a significant number of works related to architecture evaluation, there is a lack of works on specific evaluations concerning safety. We can find works on using the ATAN method in safety-critical systems (e.g. avionics domain) [12] but they are way much tight to constraints of the domain, and hardly extend to the context of medical devices, which, in a sense, demands special considerations that particular of the medical domain. Moreover, Rozanski and Woods [13] published in their book a set of questions concerning to high quality software architecture for each architecture view. However, the questions are too narrow for evaluating safety aspects (this experiment will be detailed in the next section).

Indeed, there are several design aspects related to safety that have to be considered when performing an evaluation. However, most of architecture evaluation methods available in the literature do not properly consider aspects related to safety engineering in the development of medical devices.

III. Evaluating Existing Literature Questions

In this section, we take a set of qualified questions from the literature and evaluate its feasibility for assessing systems and software architectures in a regulatory context. From our understanding, we consider the work of Rozanski and Woods [13] as a mature model for addressing such assessments, as it has been has been widely accepted and adopted as basis for documenting architectures of software based systems.

In order to analyze whether their approach was appropriate to the context of medical devices, we tried to apply their quality questions to the Generic Infusion Pump project [14] and obtained results that might be considered acceptable by the software engineering community. Other might claim that minor adjustments in the questions would be enough to provide a good support to meet our goal. In this section we provide evidences to claim that this assumption is not an appropriate choice.

We analyzed 94 questions and only 22 questions (less than 20%) were considered suitable for our purpose. One important justification is that most questions do not address quality attributes, especially safety.

Some examples of the questions that are not appropriate for evaluating architectures focusing on safety are shown below:

1) Can you simplify your concurrency design?
2) Have you considered how the architecture is likely to cope with possible change scenarios in the future?
3) Have you defined a clear strategy for organizing the source code modules in your system?

The concurrency aspect addressed by Question 1 is an important item to be considered when evaluating the safety criticality of a medical device, since, if not properly managed, errors like race conditions and permanent interruption of a process can be root causes of safety-critical failures. However, what matters for safety is the existence of mechanisms for detecting and handling such situations. The optimization of the concurrency model is important as a further design improvement step in order to improve other quality attributes like maintainability and performance. For safety, the main argument would be information about mechanisms to cope with concurrency failures, independently of how optimized the concurrency model is.

The core aspect evaluated by Question 2 is if the system is designed to evolve or be maintainable. Maintainability is a core aspect that architects and designers should take into account. However, the level of modularity will not determine whether the system is safe or not. The system can be absolutely hard to maintain, but still cover every safety requirement specified. In a nutshell, the level of modularity does not determine how safe the system is. Therefore, this item is also not appropriate to evaluate safety of medical devices.

Question 3 is about verifying the existence of a clear hierarchical decomposition and organization of modules and source code. This aspect is fundamental when quality attributes such as understandability, modularity, and maintainability are being analyzed. With respect to safety, they are not relevant, since the system being analyzed can be extremely modularized, and hierarchically decomposed, and might even not address the necessary safety measures.

Some examples of the questions that are appropriate for evaluating architectures focusing on safety are shown below:

1) Will a clear set of standard third-party software elements be used across all element implementations? Have you defined the way they should be used?
2) Are the assessors satisfied that the deployment environment meets their requirements in terms of standards, risks, and costs?
3) Do all element interactions take place via well-defined interfaces and connectors that link the interfaces?

Question 1 deals with the very common practice of reusing parts of one system for building a new one. This reuse may be from legacy systems of the same company, or from systems built by other companies (so called third party systems). In any case, when any of these elements are reused, there should be evidence that they are appropriate for the new context. Such evidence generally includes failure mode and effect analysis, test plan, and any other mechanism that shows that this system is appropriate for the new context.

Question 2 is about deployment strategies considered in the implementation of medical devices. This aspect is of fundamental importance for safety since the computational nodes should be in conformance with the integrity levels specified in the standards. When this aspect is not observed, the safety of the medical device is strongly compromised.

The aspect discussed in Question 3 is also important for safety because it provides the basis for analyzing error propagation. Having a clear understanding of the interface and the connectors allows safety engineers to precisely identify which architectural elements must be considered while conducting failure identification techniques such as Fault Tree Analysis, and Failure Mode and Effect Analysis. Therefore, when properly described, beyond improving safety analysis efficacy, interfaces and connectors descriptions also improve the efficiency of safety analysis.

Finally, there are some questions that we were not able to classify as either useful or useless for safety. In order to show our point, let us take the following question and show our arguments.

1) Have you identified any standard approaches to design that you need all element designers and implementers to follow? If so, do your software developers accept and understand these approaches?

At first, there is evidences that this question would have some impact on system safety due to its appeal to be disciplined in the standardization of good practices in order to achieve some quality goal. Moreover, only with the acceptance and understanding of the development team it would be feasible to get a reliable implementation in terms of safety quality. For example, only with the right understanding of the importance of redundancy channel, a compromised implementation of such safety feature could be expected. Thus, our claim is that to be in conformance with design and implementation guidelines is a strongly recommended practice in development environments. That is why it is also important to ensure that members of the development team are committed to these rules. However, unfortunately, this subject does not offer a basis for determining or inferring anything about safety integrity levels of a medical device. It would be useful for improvements in structure, understandability, and, consequently, maintainability of the system. However, it is not a determinant factor whose absence would compromise the safety of the medical device.

IV. Architecture Quality Model for Medical Device Evaluation Addressing Safety

One of the goals of considering software systems at the architecture level is to ensure that the concerns of the different stakeholders involved in the development process are addressed and documented [15], and also that different aspects of the system (e.g. system structural decomposition and behavior) were considered [13]. To ensure that all the relevant safety aspects are considered, a requirements model for safety-related product artifacts was previously created [5]. The concrete quality demands for general engineering artifacts proposed in this work is used as basis for the architecture quality model proposed in this paper. This general quality model aims at evaluating medical device on four abstraction levels: (i) context level, (ii) system level, (iii) software architecture design level, and (iv) software unit design and implementation level. We understand that such abstraction levels should classify artifacts generated in the overall software development process. Thus, we considered four types of engineering information: ”Physical” Structure, Static Interaction, Dynamic Interaction and Integration . It is possible to observe all the abstraction levels and the interaction with the types of considered information in Figure 1.

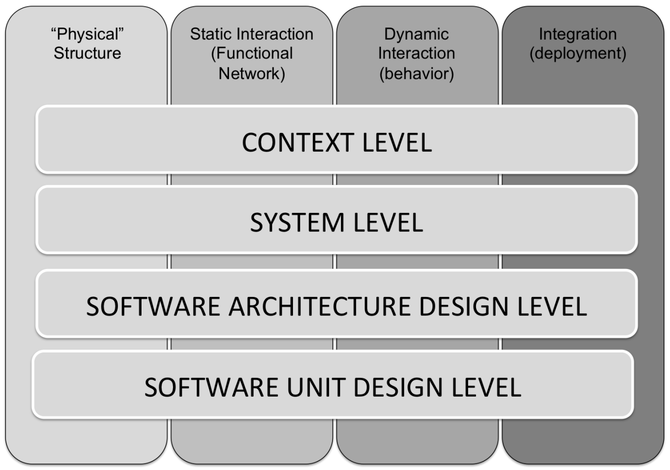

"Physical" Structure
Static Interaction (Functional Network)
Dynamic Interaction (behavior)
Integration (deployment)
CONTEXT LEVEL
SYSTEM LEVEL
SOFTWARE ARCHITECTURE DESIGN LEVEL
SOFTWARE UNIT DESIGN LEVEL

Fig. 1. Quality Model Criteria based on Abstraction Levels

The nest items are an overview of the architectural aspects that should be considered at each one of these abstraction levels.

1) Medical Device at Context Level (CONT) - it describes how the medical device is used and how it works in its intended environment [5]. The medical device is considered as a black box system and the architectural quality questions in this model are related to how the medical device interacts with other subsystems, stakeholders, and other devices.
2) System Level (SYSL) - it describes how the medical device provides the expected functionality [5]. On this level, the system is assumed as a white-box and the architecture description should describe the elements regardless of weather each functionality is implemented by software or by hardware.
3) Software Architecture Design Level (SADL) - it describes the realization of software while abstracting from details [5]. On this level, the described software architecture refines all logical components documented at SYSL considering modules and software components, the relationship between then and the interface exposed by each module.
4) Software Unit Design and Implementation Level (SUDIL) - it describes the implementation of the software that is specified at software architectural design level, the implementation of software units that realize all modules and software components that is described in SADL.

Now, we will present which architectural elements are important to be considered along the intersections between the abstraction levels and types of engineering information shown in Figure 1.

In the **Physical** structure we organize the information engineering elements concerning to the tangible structure of the medical device. For example, stakeholders, communication elements, modules, components, units, and so on. Thus, the categorization of this type of engineering information is:

- The **Physical Structure at the Context Level** involves understanding the stakeholders, other devices interacting with the medical device, and other environmental aspects such as electromagnetic activities in the environment where the medical device will be located.
- The **Physical Structure at the SYSL** comprises execution platforms and the communication paths among them.
- The **Physical Structure at the SADL** comprise elements that provide the understanding of the software in terms of components and modules, and how they interact.
- The **Physical Structure at SUDIL** cover code fragments that realize the software described in the other levels as, for example, packages, classes, methods, etc.

The **Static Interaction** engineering information type comprises elements related to static relationship of the structures which make software and hardware of the medical device. For instance, information changed among stakeholders and the medical device, information exchanged among the logical, components, modules and units, etc. Therefore, the categorization of this type of engineering information is:

- The **Static Interaction at the Context Level** involves understanding the static functional structure including logical information exchanged among the conglomerate of medical device, stakeholders, and other systems. For example, sensor/actuator interface, user interface, actuation and information features.
- The **Static Interaction at the SYSL** is focused on the static functional structure of the system as a white-box, with functions exchanging information via ports. For instance, functions interface, functions specifications (required and provide interface) and static interactions of functions are pieces of information required in this level.
- The **Static Interaction at SADL** requires elements for each software subsystem in order to have the complete understanding of their functionalities. In this way, this level involves system-level functions which are implemented by software subsystems, modular decomposition into software components, modules and software components interface, software components specification, static interaction between software modules.
- The **Static Interaction at SUDIL** comprises elements of static interaction that involves specific static information about software units, for example, decomposition of software modules and components in software units, software units interface, specification and static interaction of software units.

The **Dynamic Interaction** engineering information type comprises artifacts which are related to general behavior of the medical device. Information such as runtime behavior, actions, or relationship among modules, components and units modeling an system action are examples of elements. Thus, the categorization of this type of engineering information is:

- The **Dynamic Interaction at the Context Layer** involves understanding: (i) the runtime behavior of the systems that interact with the medical device; (ii) actions and their consequences for stakeholders; and (iii) how the system affect on the environment where the medical device is located.
- The **Dynamic Interaction at the SYSL** comprises all the runtime behavior of the functions.
- The **Dynamic Interaction at SYSL** includes the dynamic behavior of the logical components.
- The **Dynamic Interaction at SADL** includes the dynamic behavior of each software component.
- The **Dynamic Interaction at SUDIL** includes the dynamic behavior of each software modules (which refines the SADL components)

The **Integration** engineering information type comprises artifacts that supports mappings between elements, allocation of components and assignments of behavioral elements. Therefore, the information categorization of this type is:

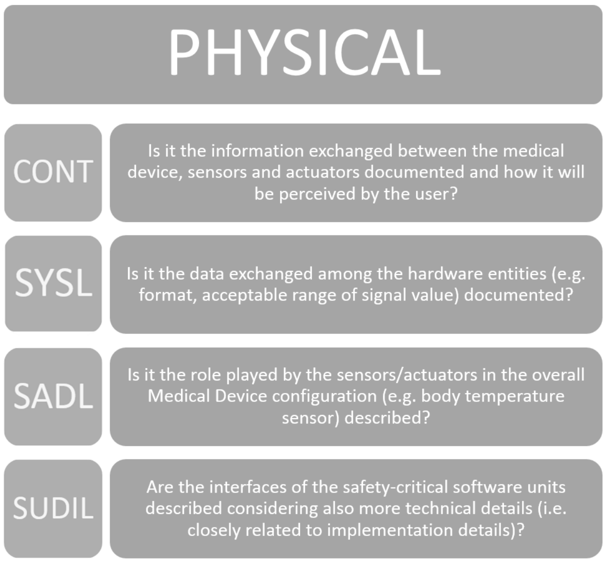
- The **Integration at the Context Layer** involves identifying the mappings and assignments of logical/behavioral elements (from static and dynamic interaction) to the physical structure. We assume that this mapping is in most cases straightforward.
- The **Integration at SYSL** comprises information that supports the knowledge of how the functions are allocated in the physical structure of the medical device.
- The **Integration at SADL** involves elements to improve our understanding of the assignment of functions to software modules and components.
- The **Integration at SUDIL** comprises elements that indicate where the software units are deployed.

A. *New Quality Questions for Safety Assessment*

In this section, we will discuss some questions which we use to evaluate safety in the software architecture of the medical devices. Figures (2, 3, 4, and 5) provide illustrative questions for each level of the previously described model in Figure 1 in order to clarify the model's purpose. The reader should have in mind that the actual model has a lot more questions and will be published at a late date by the Brazilian Health Ministry. Due to space restrictions we selected closely related questions for each column.

First, for the physical layer we selected questions asking about safety arguments regarding user perception of the data exchanged by the medical device and external systems at the context level. For example, besides other issues, we need to have clear information about such exchanged data at the system level. This information can be enriched by asking for the description of the role of the stakeholders who manipulate such exchanged data at the architectural level and can be refined by asking about the description of the interfaces at the software unit level. Some evaluation questions can be seen in Figure 2.

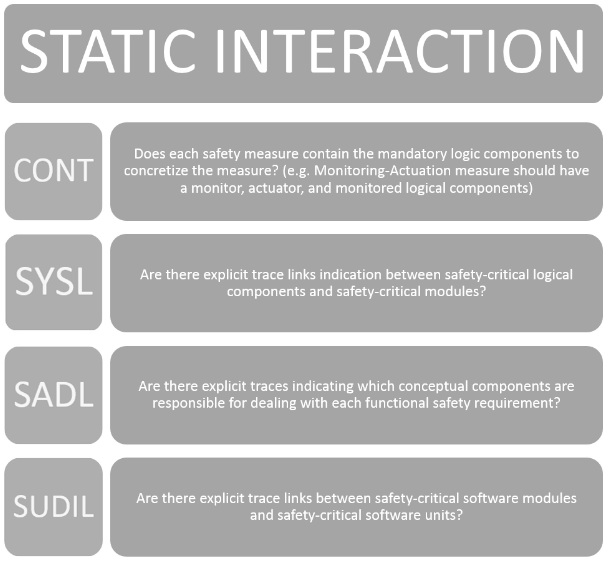
For the static interaction layer, we selected questions asking about a clear description of which logical components will concretize each safety measure at the context level. For example, besides other issues, we need to have information about explicit traces between the safety-critical components and the safety-critical modules at the system level. This information can be enriched by checking the explicit traces between the conceptual components and the safety requirements at the architectural level, further it can be refined by asking about



PHYSICAL	
CONT	Is it the information exchanged between the medical device, sensors and actuators documented and how it will be perceived by the user?
SYSL	Is it the data exchanged among the hardware entities (e.g. format, acceptable range of signal value) documented?
SADL	Is it the role played by the sensors/actuators in the overall Medical Device configuration (e.g. body temperature sensor) described?
SUDIL	Are the interfaces of the safety-critical software units described considering also more technical details (i.e. closely related to implementation details)?

Fig. 2. Quality model questions for physical layer

the explicit traces between the safety-critical software modules and the safety-critical software units. In Figure 3 we stated some evaluation questions for each layer.

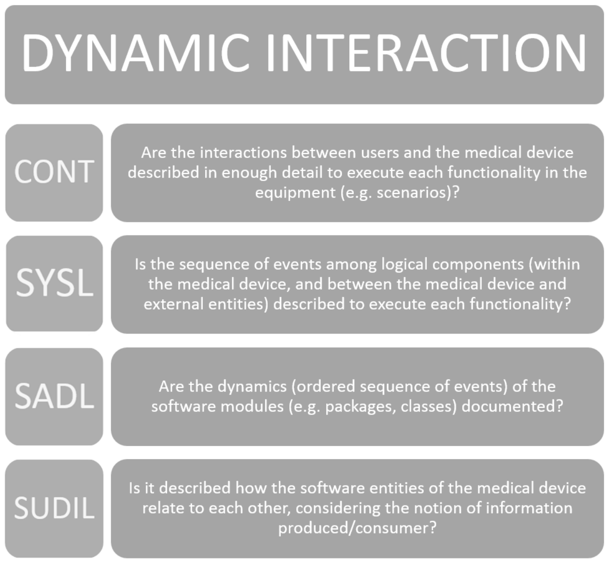


STATIC INTERACTION	
CONT	Does each safety measure contain the mandatory logic components to concretize the measure? (e.g. Monitoring-Actuation measure should have a monitor, actuator, and monitored logical components)
SYSL	Are there explicit trace links indication between safety-critical logical components and safety-critical modules?
SADL	Are there explicit traces indicating which conceptual components are responsible for dealing with each functional safety requirement?
SUDIL	Are there explicit trace links between safety-critical software modules and safety-critical software units?

Fig. 3. Quality model questions for static layer

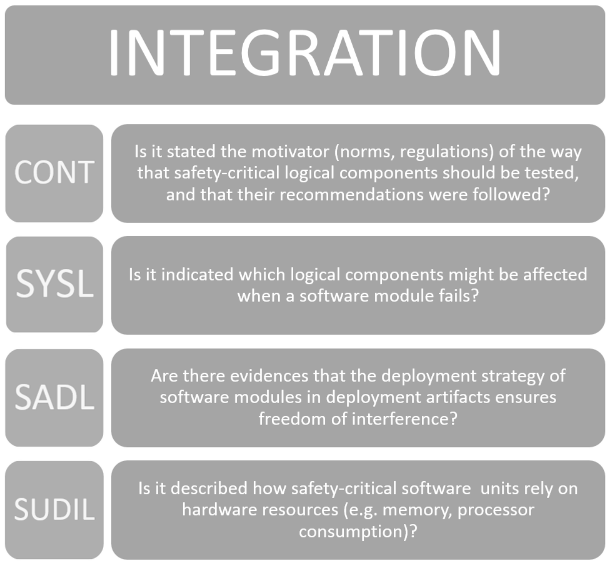
For the dynamic layer, we selected questions asking about details of usage scenarios of medical devices at the context level. For example, besides other issues, we need to have the description of the sequence of events among logical components to execute each functionality at the system level. This information can be enriched by checking if the ordered sequence of events between software modules is well documented at the architectural level and can be refined by asking how the software entities relate to each other considering the information produced and consumed in the software units. Figure 4 shows some evaluation questions for the static layer.

Finally, for the integration layer we selected questions asking about motivations and recommendations for testing safety-critical logical components at the context level. For example, besides other issues, at the system level we need to have indications of what logical components are affected by any component failures. This information can be enriched by checking if the deployment strategy for software modules provides evidence of freedom of interference at the architectural level and can be refined by asking how the safety-critical software units rely on hardware resources. An example of evaluation questions at integration layer in Figure 5.



DYNAMIC INTERACTION	
CONT	Are the interactions between users and the medical device described in enough detail to execute each functionality in the equipment (e.g. scenarios)?
SYSL	Is the sequence of events among logical components (within the medical device, and between the medical device and external entities) described to execute each functionality?
SADL	Are the dynamics (ordered sequence of events) of the software modules (e.g. packages, classes) documented?
SUDIL	Is it described how the software entities of the medical device relate to each other, considering the notion of information produced/consumer?

Fig. 4. Quality model questions for dynamic layer



INTEGRATION	
CONT	Is it stated the motivator (norms, regulations) of the way that safety-critical logical components should be tested, and that their recommendations were followed?
SYSL	Is it indicated which logical components might be affected when a software module fails?
SADL	Are there evidences that the deployment strategy of software modules in deployment artifacts ensures freedom of interference?
SUDIL	Is it described how safety-critical software units rely on hardware resources (e.g. memory, processor consumption)?

Fig. 5. Quality model questions for integration layer

V. Related Works

This section briefly discusses some architecture-based safety evaluation methods. It is important to advise that the almost all analyzed approaches are based on model driven techniques, thus the architecture have to be formally described in a specific language (e.g. Simulink) in order to assess safety applying model-based evaluation methods. Model-based techniques for design embedded safety-critical systems have been effectively implemented for avionics and automotive systems, but in medical systems domain this is still at the beginning. Moreover, general regulatory bodies (e.g.: FDA) do not obligate the industry to use model-driven development, consequently the evaluation quality model for medical device software systems have to work with any approach for systems specification and designing, further, regulatory agencies do not have access to the concrete project (for example, development artifacts), they just have to assess safety (and another quality attributes) analyzing project documentation. To illustrate, we can observe the a set of works that have been focused on the traceability form safety requirements to the architecture such as in the Grunke et al. [16] focused on automatic process for trace safety requirements in the architecture and another approaches like FTA ([17]), FMEA ([18]), or strategies based on FPTN ([19]) and CFT ([20]).

In addition, the most of safety assessment methodologies for architecture evaluation have been implemented during the design and development. On the other hand, all regulatory bodies need to evaluate the architecture of a product that had already been built. For example, Rupanov et al. [21] presents an early model-based safety evaluation of design decisions based on metamodels that support the task for automotive domain. Although safety is assessed by the proposed methodology they are based on model-based techniques and they need all the development artifacts. Therefore, there is a lack of safety assessment methodologies for evaluate architecture of a medical device systems based on evidences exposed in the project documentation.

VI. Conclusions

This paper has discussed critically safety issues in quality models for software architectures of medical devices. From our point of view, the existing evaluation guides provided by the literature were still not sufficient for addressing the safety quality attribute bringing serious problems for regulation and even manufacturers. We provided contributions to this field by showing the architectural perspective of the safety quality model ordered by the Brazilian government and built by NUTES and Fraunhofer-IESE. This perspective is structured in abstraction layers and pieces of information, providing an useful support for higher level safety arguments.

Current and future works involve, besides the application of the quality model, the development of a toolset for Enterprise Architect and Eclipse Modeling Framework able to specify safety integrity levels and execute such architectural verifications in UML and SysML models.

Acknowledgments

This work was partially financed by the NÚCLEO DE TECNOLOGIAS ESTRATÉGICAS EM SAÚDE - NUTES - Agreement FINEP 01.08.0613.00MCT/FINEP/AT - PRO-INOVA - 01/2008.

References

[1] J. Webster, *Medical Instrumentation: Application And Design, 3Rd Ed.* Wiley India Pvt. Limited, 2009. [Online]. Available: http://books.google.com.br/books?id=bxXcYL29SUMC

[2] ISO/IEC, "ISO/IEC 25000 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE," Tech. Rep., 2005.

[3] M. D. Mary Beth Privitera and J. Johnson, "Interconnections of basic science research and product development in medical device design," in *Proceedings of the 31st Annual International Conference of the IEEE EMBS*. IEEE, 2009.

[4] P. Barbosa, M. Morais, K. Galdino, M. Andrade, L. Gomes, F. Moutinho, and J. Figueiredo, "Towards medical device behavioural validation using petri nets," in *Proceeding of the IEEE CBMS'13*, 2013.

[5] R. Adler, S. Kemmann, D. Filho, and J. A. Neto, "Safety assessment of software-intensive medical devices: Introducing a safety quality model approach," in *Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2013.

[6] ISO/IEC/(IEEE), "ISO/IEC 42010 (IEEE Std) 1471-2000 : Systems and Software engineering - Recomended practice for architectural description of software-intensive systems," p. 23, 07 2007.

[7] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2001.

[8] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," *Software Engineering, IEEE Transactions on*, vol. 28, no. 7, pp. 638–653, Jul 2002.

[9] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-based analysis of software architecture," *Software, IEEE*, vol. 13, no. 6, pp. 47–55, Nov 1996.

[10] R. Kazman, M. H. Klein, M. Barbacci, T. A. Longstaff, H. F. Lipson, and S. J. Carrière, "The architecture tradeoff analysis method," in *ICECCS*. IEEE Computer Society, 1998, pp. 68–78.

[11] P. C. Clements, "Active reviews for intermediate designs," Carnegie Mellon, Technical Note CMU/SEI-2000-TN-009, aug 2000. [Online]. Available: http://www.sei.cmu.edu/library/abstracts/reports/00tn009.cfm

[12] B. Mario, C. Paul, L. Anthony, N. Linda, and W. William, "Using the architecture tradeoff analysis method (atam) to evaluate the software architecture for a product line of avionics systems: A case study," Software Engineering Institute, Carnegie Mellon University, Technical Note CMU/SEI-2003-TN-012, 2003. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6447

[13] N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Pearson Education, 2011. [Online]. Available: http://books.google.com.br/books?id=nXRF77-gxRkC

[14] GIP, "The generic infusion pump (gip)," 2013, http://rtg.cis.upenn.edu/gip.php3.

[15] D. Jackson and E. Kang, "Separation of concerns for dependable software design," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 173–176. [Online]. Available: http://doi.acm.org/10.1145/1882362.1882399

[16] L. Grunske and J. Han, "A comparative study into architecture-based safety evaluation methodologies using aadl's error annex and failure propagation models," in *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*, Dec 2008, pp. 283–292.

[17] C. A. Ericson, *Hazard Analysis Techniques for System Safety*, 1st ed. Wiley-Interscience, Aug. 2005. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471720194

[18] D. H. Stamatis, *Failure mode and effect analysis : FMEA from theory to execution*. ASQ Quality Press, 2003. [Online]. Available: http://www.worldcat.org/isbn/0873895983

[19] P. Fenelon, J. A. McDermid, M. Nicolson, and D. J. Pumfrey, "Towards Integrated Safety Analysis and Design," *SIGAPP Appl. Comput. Rev.*, vol. 2, no. 1, pp. 21–32, Mar. 1994. [Online]. Available: http://dx.doi.org/10.1145/381766.381770

[20] B. Kaiser, P. Liggesmeyer, and O. Mäckel, "A New Component Concept for Fault Trees," in *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33*, ser. SCS '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 37–46. [Online]. Available: http://portal.acm.org/citation.cfm?id=1082054

[21] V. Rupanov, C. Buckl, L. Fiege, M. Armbruster, A. Knoll, and G. Spiegelberg, "Employing early model-based safety evaluation to iteratively derive e/e architecture design," *Science of Computer Programming*, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642313002554