The Safety Requirements Decomposition Pattern

Pablo Oliveira Antonino[1(✉)], Mario Trapp[1], Paulo Barbosa[2], Edmar C. Gurjão[3], and Jeferson Rosário[4]

[1] Fraunhofer Institute for Experimental Software Engineering - IESE, Kaiserslautern, Germany
{pablo.antonino,mario.trapp}@iese.fraunhofer.de
[2] Nucleus for Strategic Health Technologies - NUTES, State University of Paraíba - UEPB, Campina Grande, Paraíba, Brazil
paulo.barbosa@nutes.uepb.edu.br
[3] Electrical Engineering Department, Federal University of Campina Grande - UFCG, Campina Grande, Paraíba, Brazil
ecandeia@dee.ufcg.edu.br
[4] Lifemed, Porto Alegre, Rio Grande do Sul, Brazil
jeferson.rosario@lifemed.com.br

Abstract. Safety requirement specifications usually have heterogeneous structures, most likely based on the experience of the engineers involved in the specification process. Consequently, it gets difficult to ensure that recommendations given in standards are considered, e.g., evidence that the requirements are complete and consistent with other development artifacts. To address this challenge, we present in this paper the Safety Requirements Decomposition Pattern, which aims at supporting the decomposition of safety requirements that are traceable to architecture and failure propagation models. The effectiveness of the approach has been observed in its application in different domains, such as automotive, avionics, and medical devices. In this paper, we present its usage in the context of an industrial Automated External Defibrillator system.

Keywords: Safety requirement · Architecture · Failure propagation model · Traceability · Completeness · Consistency

1 Introduction

Safety requirements are fundamental artifacts in the specification of safety-critical systems, since they often result from safety analysis of the architecture, and must ultimately be addressed by elements of the architecture [1, 2].

Because of this key role of safety requirements in safety engineering activities, it is important to assure that they meet certain quality attributes [3]. In particular, completeness and consistency of safety requirements have been widely discussed, as it has been demonstrated that the lack of guidance on how to properly specify traceable safety requirements is one of the main reasons for their incompleteness and inconsistency and, consequently, a root cause of safety incidents [4].

F. Koornneef and C. van Gulijk (Eds.): SAFECOMP 2015, LNCS 9337, pp. 269–282, 2015.
DOI: 10.1007/978-3-319-24255-2_20

In this regard, the work described in this paper addresses this challenge by providing a hierarchical decomposition structure to support the systematic decomposition of safety requirements specifications that are explicitly traceable to failure propagation models (e.g., Markov Chains and Fault Trees Analysis) and to architecture specifications–the Safety Requirements Decomposition Pattern (SRDP).

The SRDP has been used in several projects in the automotive, avionics, and medical device industries in Europe and Brazil, and has proven to be effective in improving completeness and consistency of safety requirements. In this paper, we present the use of the approach in the safety requirements specification of an Automated External Defibrillator (AED) system, which is being specified in a technology transfer collaboration process between the Nucleus for Strategic Health Technologies (NUTES[1]), which is an initiative of the Brazilian Health Ministry for promoting the technological development of medical devices, and the medical device producer Lifemed[2], which has more than thirty years of activity in the area of production of equipment and consumption material destined to the medical-hospital sector.

The remainder of this paper is structured as follows: In Sect. 2, we discuss the state of the practice in safety requirements specifications. In Sect. 3, we present the main drivers for establishing the SRDP, and in Sect. 4 we present it in detail. In Sect. 5, we describe its usage in the context of the NUTES/Lifemed Automated External Defibrillator system and discuss the results. In Sect. 6 we present the related works, and, in Sect. 7 we present conclusions and discuss future work.

2 State of the Practice in Specifying Safety Requirements

Standards and regulations such as ISO 26262 [2], DO-178C [5], and ANSI/AAMI/IEC 62304 [6] describe strict recommendations to be considered during the specification of safety requirements. ISO 29148 [7], e.g., recommends providing evidences showing that the requirements are consistent, complete, validated, verified, and fully traceable to all system artifacts. Nevertheless, none of these norms specifies a defined structure for safety requirements specifications, nor guidelines to be followed during the specification process. In practice, each safety requirements specification has a different structure, most likely based on the understanding and experience of the engineers involved in the specification process. Furthermore, because of this lack of a structured definition, following the argumentation of safety requirements specifications is not a trivial task, especially if the reader is not the author [8]. As a consequence, it becomes difficult to ensure that the aspects recommended by standards and regulations are considered, such as completeness and consistency of safety requirements with respect to the results of the failure analysis and to the architecture specification [4, 9]. For instance, Maeder et al. [4] analyzed the traceability documents of a medical device submitted to the US Food and Drug Administration - FDA and found that the majority

[1] http://nutes.uepb.edu.br/.

[2] http://www.lifemed.com.br/.

of the safety requirements were incomplete and that many of their traces to safety analysis results and to the architecture were inconsistent.

In a sense, there is a lack of means for intelligently and efficiently reasoning about existing and missing traceability between artifacts, which includes, among other things, the types of artifacts and relationships that are permissible and the granularity level to which the different artifacts should be decomposed [1, 10].

3 Designing the Safety Requirements Decomposition Pattern

The architecture specification is one core artifact to be considered when specifying safety requirements, as these requirements often result from a safety analysis of the architecture [1, 2]. In this regard, to design the SRDP, we analyzed several mature model-based architecture specification methodologies, such as (i) Kruchten's "4 + 1" View Model [11], (ii) the Software Engineering Institute's Viewtypes [12], (iii) the Siemens Four View Models [13], and (iv) the SPES 2020 Methodology for Model-Based Engineering of Embedded Systems [14]. Additionally, we had fruitful discussions with practitioners from the automotive, avionics, and medical devices domains, and, compiling the results from both sources, we observed that at least two aspects are usually addressed when specifying the architecture of a safety-critical system: (i) *the functions of the system*; i.e., implementation-independent specification of what the system should provide in terms of services to its users [11]; and (ii) *description of the architectural elements that realize the functions*, either software resources, e.g., a scheduling slot, or hardware resources, e.g., a processor [14]. In a sense, both the state of the art and the state of the practice indicate that it is important to document at the architectural level at least the functions of the systems and how they are realized by software and hardware artifacts (cf. Architecture Design slot of Fig. 1).

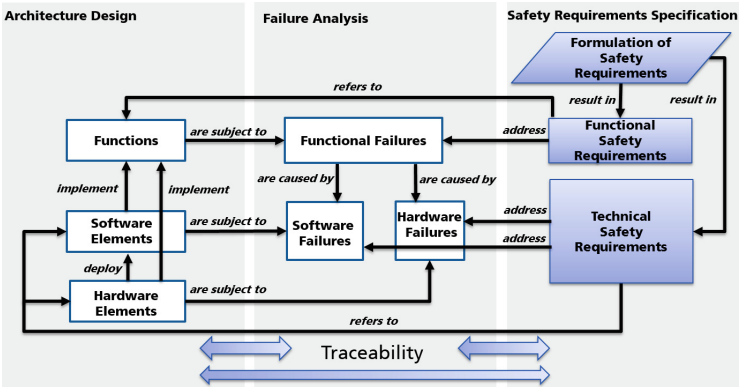

Fig. 1. Dependencies between development artifacts in safety-critical systems considering the functions and technical resources (software and hardware) that realize them.

In the development of safety-critical systems, portions of the architecture are subject to failure analysis, which, in turn, is performed using appropriate techniques, such as Markov Chains and Fault Trees Analysis, which aim at identifying and representing the logical relationships between software and hardware failures and how they cause functional failures (cf. Failure Analysis slot of Fig. 1) [1, 2].

Based on this context, we understand that it is also necessary to have a hierarchical decomposition of the safety requirements to ensure that failures associated with architecture elements of the different levels of abstraction (function, software, and hardware) are addressed with an adequate level of detail (cf. Safety Requirements Specification slot of Fig. 1). This is where the SRDP plays its role in guiding the construction of safety requirements specifications, enriching them semantically by considering hazard-contributing factors described in failure propagation models and risk control measures described in the architectural specification.

4 The Safety Requirements Decomposition Pattern

We split the description of the SRDP into three parts: first, in Subsect. 4.1, we present its decomposition structure for specifying functional safety requirements; then, in Subsect. 4.2, its decomposition structure for specifying technical safety requirements; and in Subsect. 4.3, the elements that are common to the functional and technical levels.

4.1 Safety Requirements Decomposition Pattern at the Functional Level

As illustrated in Fig. 1, we understand that it is necessary to specify safety requirements associated with architecture elements of different levels of abstraction. At the functional level, the safety requirements should describe how the elements of the functional architecture should collaborate to fulfill the *Top-Level Safety Requirement* (cf. top-most part of Fig. 2), which, in turn, aims at indicating the high-level mitigation strategy for addressing the hazards identified in the hazard and risk analysis. The *Top-Level Safety Requirement* is called differently depending on the domain; for instance, in the road vehicles domain it is called Safety Goal [2].

In the remainder of this subsection, we describe the elements of the SRDP grouped inside the *Safety Requirements @ Functional Level* boundary depicted in Fig. 2. The first element described is the Functional Architecture Element, as safety requirements often result from a safety analysis of the architecture:

- **Functional Architecture Element (FAE):** Functional architecture describes the services that the system should provide [11], usually represented by networked functional hierarchies [14]. These elements are represented in the SRDP by the *FAE*.

Next, we describe the elements identified during the Failure Analysis (cf. central part of Fig. 1), which are depicted in the left-most part of Fig. 2:

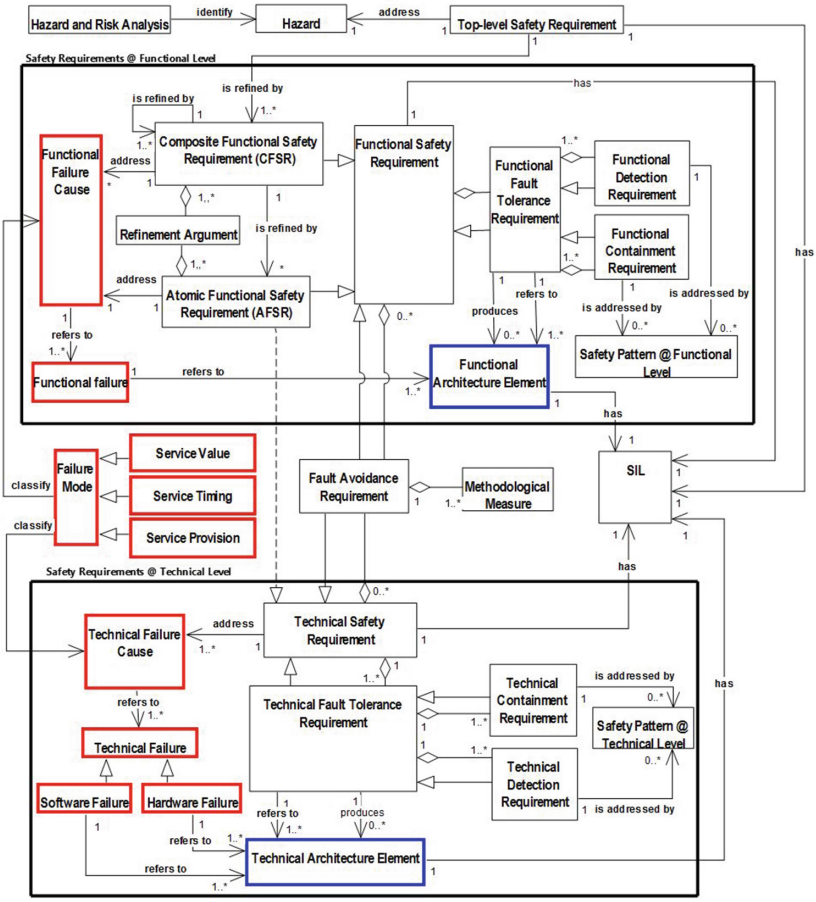

Fig. 2. The safety requirements decomposition pattern.

- **Functional Failure (FF):** ISO 26262 defines failure as the termination of the ability of an element to perform a function as required [2]. One of the reasons that might cause failures is the lack of compliance between the services delivered and the system specification at the functional and technical levels [1, 2]. At the functional level, we call this item Functional Failure (FF). It is about what might happen with elements of the functional architecture that can lead to the system failure. These failures are identified throughout the failure analysis using failure propagation models.
- **Functional Failure Cause (FFC):** Failure conditions associated with failures of elements of the functional architecture that lead to system malfunction. *FFCs* are classified according to Failure Modes, which, in turn, are described in Sect. 4.3.

Next, we describe how to hierarchically decompose the notion of Functional Safety Requirement and how the resulting elements are related to the results of the failure analysis and to the elements of the functional architecture.

- **Functional Safety Requirement (FSR):** Specification of implementation-independent safety measure, including its safety-related attributes [2].
- **Composite Functional Safety Requirement (CFSR):** *FSRs* that have more than one *FFC* motivating their existence. *CFSRs* can be refined by other *CFSRs* or by Atomic Functional Safety Requirements (AFSR).
- **Atomic Functional Safety Requirement (AFSR):** *FSRs* that have only one failure cause motivating its existence. *AFSRs* refine *CFSRs* that cannot be decomposed into other *CFSRs* anymore, and are realized by Technical Safety Requirements, which are described in detail in Sect. 4.2.
- **Refinement Argument (RA):** Reason given in support of refining *CFSRs* into other *CFSRs* or into *AFSRs*. For each refinement, there should be at least one refinement argument.
- **Functional Fault Tolerance Requirement (FFTR):** Avizienis et al. [15] introduced the notion of Fault Tolerance Techniques as a means that allows living with systems that are susceptible to faults. Avizienis' fault tolerance techniques basically consist of error detection and system recovery techniques. *FFTRs* comprise measures for detecting and containing failures with regard to elements of the functional architecture.
- **Functional Detection Requirement (FDR):** Refine *FFTRs* by describing means to detect errors at the functional level.
- **Functional Containment Requirement (FCR):** Refine *FFTRs* by describing means to handle errors at the functional level.
- **Safety Pattern @ Functional Level (SPFL):** Decisions made to detect and contain failures become more concrete when realized by one or more established safety patterns, e.g., Homogeneous and Heterogeneous Redundancies, Monitor-Actuator, and Watchdog [16]. In this regard, *SPFLs* indicate how to combine functional architecture elements to detect and contain functional failures.

4.2 Safety Requirements Decomposition Pattern at the Technical Level

At the technical level, the safety requirements should describe how the strategies specified at the functional level are realized using software and hardware resources. The artifacts used as inputs at this level are (i) the *AFSR*, as they are the most refined descriptions of the safety strategy at the functional level, and (ii) the technical architecture specification (i.e., software and hardware entities). In the remainder of this subsection, we describe the elements of the SRDP grouped inside the *Safety Requirements @ Technical Level* boundary depicted in Fig. 2.

- **Technical Architecture Element (TAE):** Software and hardware entities.

The following elements are identified during the Failure Analysis (cf. central part of Fig. 1):

- **Technical Failure (TF):** This is about what happened with technical architecture elements that led to functional failures. We adapted the convention proposed by Wu and Kelly [18] and categorize a technical failure according to one of the following two abstract types: (i) *Software Failure (SF)* - Incomplete or inaccurate specification, or incorrect design and implementation can cause unexpected behavior of the software; and (ii) *Hardware Failure (HF)* - Correct software can still misbehave due to unexpected behavior of the underlying hardware.
- **Technical Failure Cause (TFC):** Failure conditions associated with failures of elements of the technical architecture (hardware and software resources) that lead to functional misbehavior and finally result in a malfunction of the system. Technical failures are classified according to Failure Modes, which are described in Sect. 4.3.

The items listed below describe how to hierarchically decompose the notion of Technical Safety Requirement per se, and how the resulting elements are related to the results of the failure analysis and to the elements of the technical architecture.

- **Technical Safety Requirement (TSR):** ISO 26262 defines TSRs as *"requirements derived for implementation of associated functional safety requirement"* [2]. To follow this principle, in the scope of this paper *TSRs* comprise strategies for realizing an AFSR.
- **Technical Fault Tolerance Requirement (TFTR):** Specification for detecting and containing failures with regard to elements of the technical architecture. The measures described at this level should be consistent with those specified at the functional level with the FFTR.
- **Technical Detection Requirement (TDR):** Refine *FFTRs* by describing means to detect errors at the technical level.
- **Technical Containment Requirement (TCR):** Refine *FFTRs* by describing means to handle errors at the technical level.

The *TDRs* and *TCRs* as well as *FDRs* and *FCRs* (cf. Sect. 4.1) should refer to technical and functional architecture elements that concretize the safety measures. If these elements already exist in the architecture, they just need to be referenced in the requirement; if not, they have to be included according to the demands described in the requirement, as indicated by the *refers to* and *produces* relationships, respectively, between Functional and Technical Fault Tolerance Requirements and Functional and Technical Architecture Element shown in Fig. 2.

- **Safety Pattern @ Technical Level (SPTL):** At this abstraction level, elements of the technical architecture should be considered in order to realize the *SPFLs*.

4.3 Safety Requirements Decomposition Pattern Elements that are Common to the Functional and Technical Levels

In this section, we discuss the SRDP elements that are common to both functional and technical levels: *Failure Mode*, *Safety Integrity Level*, and the *Fault Avoidance Requirement*.

Failure Mode (FM) is the manner in which an element or item fails [15]. At the functional level, the Failure Mode is associated with the manner in which elements of the functional architecture fail. At the technical level, it is associated with the manner in which elements of the technical architecture fail. We adopted the failure mode classification proposed by Fenelon et al. [17, 18]: (i) *Service provision* - Omission, Commission; (ii) *Service timing* - Early, Late; and (iii) *Service value* - Coarse incorrect, Subtle incorrect.

Safety Integrity Levels (SIL) are discrete levels, corresponding to a range of safety integrity values, where SIL 4 has the highest level of safety integrity and SIL 1 has the lowest level [19]. In this regard, the SIL needs to be indicated explicitly throughout the specification of functional and technical safety requirements to ensure safety integrity compatibility between *Top-Level Safety Requirement*, *Functional Safety Requirement*s, *Functional Architecture Elements, Technical Safety Requirement*s, and *Technical Architecture Elements.*

Fault Avoidance Requirements (FAR) Descriptions of methodological measures to modify the development process (e.g. testing strategies and code conventions) in order to reduce the number of faults introduced during the development time of safety-critical systems [15]. As FARs are related to methodological measures, their demands affect system development as a whole, independent of how the system specification is abstracted.

5 Specifying the Safety Requirements of an Automated External Defibrillator with the Safety Requirements Decomposition Pattern

AED is a medical device used for ventricular fibrillation/tachycardia treatment, which are cardiac arrhythmias with the highest incidences of fatal cases. The AED is operated by a rescuer to deliver controlled energy to the patient's chest. The amount of energy delivered is based on an automated analysis of the Electrocardiography (ECG), which is also performed by the AED. Delays in the application of the defibrillator shock are safety-critical because every minute without the application of the shock implies a 7 % to 10 % decrease of the chance of survival [20]. In this regard, it is important to ensure optimized recharging to minimize discharge intervals. We explored the AED, which is being specified in a technology transfer collaboration process between the medical device producer Lifemed and the Brazilian Health Ministry initiative NUTES, to demonstrate how the SRDP contributed positively to improving the specification of traceable safety requirements.

5.1 Preliminary Specification of the Automated External Defibrillator

Figure 3 depicts the context diagram for the AED system. This diagram aims at showing each external entity, the main functional units, and their interactions. In simplified terms, the main input variable to be received is the patient's cardiac pulse. The Signal Analyzer employs algorithms for detecting the signal complexity and

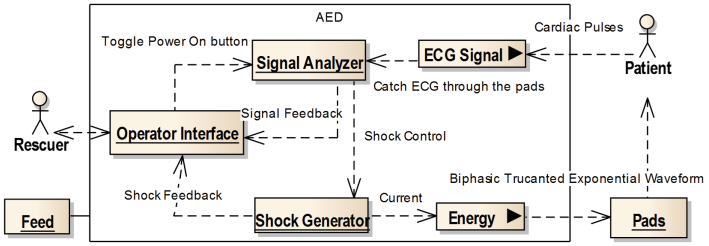

Fig. 3. Context diagram of the AED system.

determines whether a cardiac arrest has occurred. If this is the case, the Shock Generator provides controlled energy to the patient's chest through the cardiac pads.

In order to identify potentially safety-critical failures of the AED, we conducted quantitative and qualitative safety analysis considering heterogeneous artifacts of the system, such as those depicted in Fig. 3. An excerpt of the analysis is depicted in Fig. 4, where the failure paths leading to the *Overshocking* hazard can be seen. The information presented in this analysis will later be traced to elements of the safety requirements using the SRDP. As shown in Fig. 4, the *Overshocking* hazard occurs as a result of the combination of intermediate and basic events inside the *Shock Generator* (cf. Fig. 3), which is composed of *Charge* and *Discharge* hardware and software entities inside their controllers. In the *Charge module*, the main causes stem from the loss of ability of the *Charge Controller* to maintain the stability of the energy transfer process. In the *Discharge module*, the main causes stem from the wear of hardware components and the failure in detecting and mitigating problems due to the high level

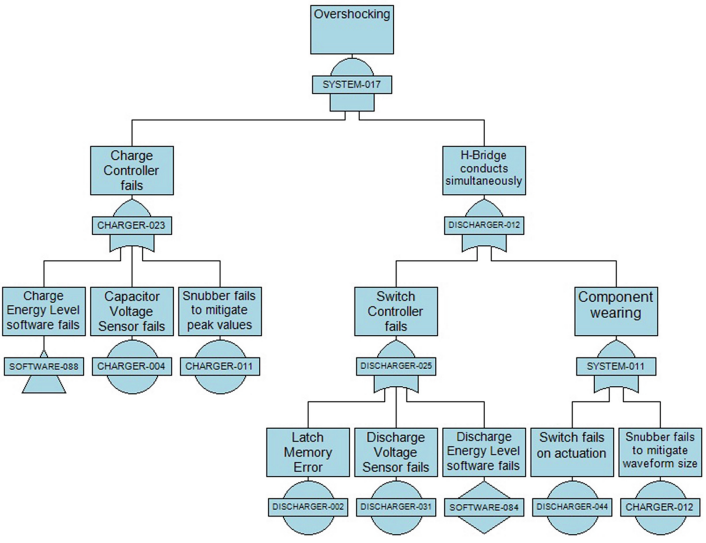

Fig. 4. Failure propagation analysis as a fault tree.

of energy to be delivered originated in the *Charge module*. Finally, all these controllers have embedded software, where logical problems such as data error or wrong control flow might generate events that could somehow contribute somehow to raising the undesirable top event.

5.2 Safety Requirements Specification for Addressing the *Overshocking* Hazard

We specified safety requirements for mitigating the hazard of *Overshocking* using the SRDP presented in this paper, and the result is summarized in Fig. 5.

One of the measures for mitigating AED overshocking is to ensure that the switching voltage of the shock generator (cf. Fig. 4) will not deliver energy above the

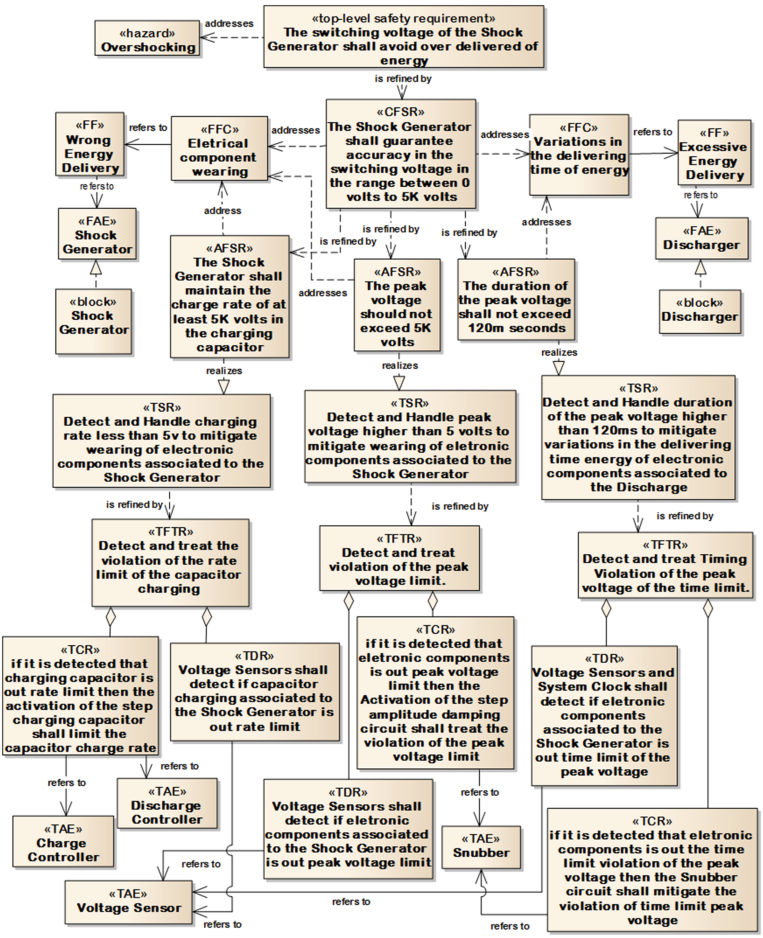

Fig. 5. AED safety requirements for mitigating the *Overshocking* hazard.

specified level. This aspect is represented by the *Top-level safety requirement* element in Fig. 5. The main causes that might lead to the occurrence of this hazard are (i) wear of electronic components or (ii) an unexpected loss of control by the software leading to unexpected values. This motivates new fine-grained specifications such as charging rates for capacitors, specific voltage peak values, or specific frequencies via semiconductor switching. Therefore, three technical safety requirements were derived and matched perfectly to those defined in the SRDP. Two cases (the leftmost ones) refer to the architectural element Shock Generator module, and the other one (the rightmost one) refers to the architectural element Discharge module. This parameterization makes it easier to clearly define the goal of each subtree specification.

Going towards fine-grained specifications, the next refinements will be fault tolerance requirements such as indication of components wear; alerts indicating the need for maintenance and repair; coupling of new amortization circuits in order to deal with unexpected peak values; or coupling of new circuits for the management of periods to avoid extrapolation of time limits for issuing peak tensions. This generates technical containment requirements and technical detection requirements, which are also defined according to the parameterized templates. Finally, the last layer of specification will require specific sensors and actuators for efficient detections and actions over all hazardous conditions.

5.3 Brief Discussion

The proposed methodology applied to medical devices allows mapping internal risks from a possible hazard and is critical in AED design due to its crucial utilization. In addition, the analysis permits both software and hardware risk specification in an integrated way, which is difficult to perform. As engineers are usually not completely aware of how a failure in a small piece of internal circuits affects the whole system safety, we consider the developed traceability to be an important achievement regarding the challenge of introducing safety engineering practices into system engineering processes. Finally, the clear decomposition proposed in this approach constitutes a concrete tool for meeting the very abstract recommendations provided by ANSI/AAMI/IEC 62304 [6].

6 Related Works

Habli et al. [21], investigated how to hierarchically decompose safety concepts [4] using the Goal Structuring Notation (GSN) in such a way that the safety argumentation is traced to a set of SysML models. Nevertheless, they present no systematic means for instantiating the approach. Actually, there is no guideline on how to decompose from high-level safety requirements to software-/hardware-related safety requirements and no indication of the types of artifacts and relationships that are permissible, nor of the granularity level to which the different artifacts should be decomposed. These aspects are extremely important for precisely and intelligently decomposing complete and consistent safety requirements [1, 10], and are provided by our approach. The approach

targets mainly inter-traceability (traceability among different artifacts) and neglects intra-traceability (traceability within the safety requirements decomposition). The SRDP improves both aspects: regarding intra-traceability, it provides a systematic hierarchical decomposition of the safety requirements that precisely indicate how top-level safety requirements as well as functional and technical safety requirements are related to each other; regarding inter-traceability, it requires matching the safety requirements decomposition to the hierarchical abstraction of the architecture elements and their respective failure propagations. Finally, our approach is not strictly tied to any particular modeling approach such as GSN or SysML because we understand that engineers should be free to use whichever approach and notation they are used to and should just have to consider the decomposition strategy described in the SRDP.

Katta and Stalhane [22] proposed a conceptual model of traceability for safety systems, which comprises a variety of artifacts and their traceability links. Their model includes elements from (i) the System Development Process, which comprises elements such as functional description, system and software requirements, and architectural components, and (ii) the Safety Assessment Process, which comprises elements such as hazards, risks, and common cause failures. Some of the intra-artifacts traceability inks proposed were considered during the elaboration of the SRDP. In this regard, we adapted their model by clearly separating safety requirements, architecture elements, and their associated failures at the functional and technical levels, and we provide systematic decomposition guidelines at each level. Moreover, we provide a clear execution path from the top-level safety requirements to the software and hardware requirements. Katta and Stalhane, on the other hand, do not provide any decomposition flow to be followed, which causes confusion and uncertainties when using their model. As for Habli et al. [21], Katta and Stalhane do not specify the granularity level to which the different artifacts should be decomposed, all of which is provided by our approach.

Birch et al. [23] analyzed the implicit safety argument structure of ISO 26262 and propose using GSN for structuring and logically decomposing the argumentation in order to justify "why" a safety requirement is indeed a Safety Requirement. The SRDP also handles justification of the existence of safety requirements, as it requires indicating the failure modes/failure paths that motivate the existence of each safety requirement. However, the SRDP requires referencing the failure propagation model that describes the failures being addressed in the safety requirement and the related architecture elements; it does not specify the level of detail to justify why a safety requirement is, indeed, a safety requirement.

Beckers et al. [24] proposed an approach centered in UML and GSN to breakdown safety goals into functional safety concepts [2]. The approach consists of seven steps that covers important aspects like, e.g., checking the completeness of a functional safety concept and ASIL decomposition, functional safety concepts allocation. Our approach differs from Beckers' as we also consider technical requirements and their synergies with functional requirements, contributing then for multi-level consistent specifications. Moreover, beyond explicitly referencing architecture elements of both levels, the SRDP offers basis for referencing failure models associated to these elements that triggered the creation of the safety requirements. Additionally, the global approach of which the SRDP is part supports not only completeness, but also

consistency checks, as described in a previous work of ours [25]. Last, the SRDP improves Becker's approach in terms of context appropriateness as its current version results from intense use not only in the road vehicles, but also in the avionics and medical devices domains.

7 Conclusions and Future Works

In this paper, we presented the Safety Requirements Decomposition Pattern, which addresses the challenges regarding completeness and consistency of safety requirements by providing a hierarchical decomposition structure to support the systematic decomposition of safety requirements specifications that are explicitly traceable to failure propagation models and to architecture specifications.

We have seen that our approach offers a reasonable basis for ensuring that there is enough information at each abstraction level of a system specification describing how the appropriate safety measures should be considered in order to address the failure causes described in the failure propagation models.

As future work, we intend to define relationships between the Safety Requirements Decomposition Pattern and other safety analysis artifacts such as safety cases and customize controlled natural languages to support content elaboration of the elements that compose the Safety Requirements Decomposition Pattern.

Acknowledgements. This work is supported by the Fraunhofer Innovation Cluster Digitale Nutzfahrzeugtechnologie. We would also like to thank Sonnhild Namingha for proofreading, and to the head of NUTES Prof. Dr. Misael Morais.

References

1. Hatcliff, J., Wassyng, A., Kelly, T., Comar, C., Jones, P.: Certifiably safe software-dependent systems: challenges and directions. In: FOSE 2014, Hyderabad, India (2014)
2. International Organization for Standardization: ISO/DIS 26262 - Road Vehicles – Functional Safety. Technical Committee 22 (ISO/TC 22), Geneva, Switzerland (2011)
3. Adler, R.: Introducing quality attributes for a safety concept. SAE Technical Paper 2013-01-0194, Detroit, Michigan, USA (2013)
4. Maeder, P., Jones, P.L., Zhang, Y., Cleland-Huang, J.: Strategic traceability for safety-critical projects. IEEE Softw. **30**(3), 58–68 (2013)
5. DO-178C/ED-12C: Software Considerations in Airborne Systems and Equipment (2011)
6. ANSI/AAMI/IEC 62304:2006: Medical Device Software—Software Life Cycle (2006)
7. International Organization for Standardization: ISO/IEC/IEEE 29148:2011 Systems and software engineering - Life cycle processes - Requirements engineering. IEEE (2011)
8. Kaiser, B.: Approaches Towards reusable safety concepts. Presentation at the VDA Automotive SYS Conference, Berlin, Germany (2012)
9. Antonino, P.O., Trapp, M.: Improving consistency checks between safety concepts and view based. Architecture design. In: PSAM12, Honolulu, Hawaii, USA (2014)