A method to the specification of safety requirements in agile contexts

Ana Isabella Muniz Leite

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

Ana Isabella Muniz Leite

A method to the specification of safety requirements in agile contexts

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP,in accordance with the requirements of the Doctorate of Science – Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Elisa Yumi Nakagawa

USP – São Carlos
December 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

```
        Muniz Leite, Ana Isabella
M966m      A method to the specification of safety
        requirements in agile contexts / Ana Isabella Muniz
        Leite; orientador Elisa Yumi Nakagawa. -- São
        Carlos, 2023.
           144 p.


           Tese (Doutorado - Programa de Pós-Graduação em
        Ciências de Computação e Matemática Computacional) --
         Instituto de Ciências Matemáticas e de Computação,
        Universidade de São Paulo, 2023.


           1. Safety-Critical Systems. 2. Safety
        Requirements. 3. Agile Development. I. Nakagawa,
        Elisa Yumi, orient. II. Título.
```

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Ana Isabella Muniz Leite

Um método para especificação de requisitos de safety em contexto ágil

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Elisa Yumi Nakagawa

USP – São Carlos
Dezembro de 2023

I dedicate this work to my husband and my children, who inspire me every day to be a better human being.

Acknowledgements

In this long journey, I was never alone. When a woman, mother, teacher, homemaker, and wife decides to step out, a support network is created. It's wonderful to have people to express my gratitude to.

First and foremost, I want to thank God for granting me this opportunity to evolve, as well as my spiritual mentors for all their assistance and enlightenment. It was never just about the Ph.D.

I would like to express my sincere gratitude to my family, my husband Fábio, and my (incarnated) children Luis Filipe and Lucas, for all the support and love they have given me. To my late father Manoel and my mother Bezinha, my brothers and systems, Alexandre, Patricia, Tiago, and Rafael, for all the support, and patience, for being my foundation and refuge even from a distance. To my mother-in-law, Socorro, without her support, friendship, and care for my family, I wouldn't have made it.

My advisor Elisa Yumi Nakagawa deserves special recognition for all the support and guidance provided throughout this journey. Her example as a human being and a professional inspires and transforms me. The way she handled my entire process of transferring from Germany to Brazil, I am truly grateful for the opportunity to be her disciple. I also extend my gratitude to my research group colleagues, especially Anderson, who helped me at several steps, giving me the needed support to move on.

I would like to express my gratitude to Pablo Antonino for his guidance and support to make this Ph.D. possible.

To the "Lulus" (Adeline, Elisa, Debora, Laryssa, Claudia, and Karina) for the Brazilian warmth in German lands, as well as their families in the names of Rodrigo Falcão and Thiago Burgos. You have become my family.

To the State University of Paraíba for their financial and intellectual support, which made my full dedication to this research project possible. My department colleagues, especially Paulo Eduardo, for his support and friendship.

Lastly, I extend my thanks to the professors who participated in my Ph.D., including those who took part in the qualification of this project and all those who directly contributed to the production of this work.

"All you can do regarding your own life is choose the best path you believe in."

(Ichiro Kishimi)

Abstract

LEITE,A.I.M. **A method to the specification of safety requirements in agile contexts**. 2023. 144 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Context: Safety-critical software systems are increasingly being used in new application areas, such as the medical domain, in which health professionals are now relying more on software-based medical devices for diagnosing and treating patients more accurately and in a shorter time frame. These devices' software is becoming more and more complex due to disruptive technological improvements. Implementing larger parts of safety systems in software has led to a growing interest in adopting agile methods and practices to improve performance with respect to development efficiency, system quality, safety integrity, and effective assessment and certification. At the same time, recent accidents and recalls have shown that several failures have been caused by errors or faults introduced during development and resulted from the misunderstandings of safety requirements by agile development teams. Moreover, there is still a lack of techniques to ensure that safety requirements are properly addressed by both software architecture and implementation. **Objective:** This PhD thesis proposes a method to specify software safety requirements and support architectural design decisions that address them in agile contexts. **Method:** We systematically developed and evaluated the SCA3DA method proposed through a design science methodology. Two controlled experiments were conducted (with students and practitioners) to demonstrate the suitability and effectiveness of the SCA3DA method in safety-critical system development in agile contexts. **Results:** Our work provides an overview of safety-critical systems development in agile contexts. Although agile methods have been applied in all phases of the safety lifecycle, safety system requirements, and safety validation have received the most attention. This is due to the critical nature of this system. We also provide evidence that agile teams are more likely to fail to account for the real intention of safety requirements due to misunderstanding them in the safety-critical system development. Furthermore, the findings have shown that the application of the SAC3DA method is promising in terms of providing positive support to better understand the software safety requirements specification and that safety-centered architectural solutions derived led to a reduction in the time taken for their analysis, with no loss of requirements understandability. **Conclusion:** Our work represents a starting point toward developing effective communication in agile contexts. The solutions derived from the SCA3DA method serve as a guide for communicating safety-related needs to the agile team, thereby promoting cooperation in conflict resolution and decision-making. A major challenge encountered in defining the method is to make the real need (intention) of the safety requirement explicit in the agile context. While existing approaches have focused on "what should be done," this work seeks to introduce the concept of "how and why it should be done".

By doing so, understanding becomes clearer, and incorrect assumptions are avoided. We believe that this work provides valuable insights into the importance of improving the understanding of safety requirements specification. Therefore, agile teams can realize more accurate software safety specifications, use these solutions to improve team communication, and ensure a unique understanding of system criticality and a more accurate interpretation of safety requirements.

Keywords: software architecture, agile development, safety, safety-critical system.

Resumo

LEITE,A.I.M. **Um método para especificação de requisitos de safety em contexto ágil**. 2023. 144 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Contexto: Sistemas estão sendo cada vez mais utilizados em novas áreas de aplicação, como no domínio médico, em que profissionais de saúde estão confiando mais em dispositivos médicos baseados em software para diagnosticar e tratar pacientes de maneira mais precisa e em um prazo mais curto. O software desses dispositivos está se tornando cada vez mais complexo devido às melhorias tecnológicas disruptivas. A implementação de partes maiores de software em sistemas críticos em safety tem levado a um interesse crescente na adoção de métodos e práticas ágeis para melhorar o desempenho em termos de eficiência no desenvolvimento, qualidade do sistema, integridade de safety, e avaliação e certificação eficazes. Ao mesmo tempo, acidentes e recalls recentes mostraram que várias falhas foram causadas por erros ou defeitos introduzidos durante o desenvolvimento e resultaram requisitos de safety mal-compreendidos pela equipe de desenvolvimento ágil. Além disso, ainda falta técnicas para garantir que os requisitos de safety sejam adequadamente tratados tanto pela arquitetura de software quanto pela implementação. **Objetivo:** Este estudo propõe um método para especificar requisitos de safety em nível de software e apoiar decisões de design arquitetural que os considerem em contextos ágeis. **Método:** Desenvolvemos e avaliamos sistematicamente o método SCA3DA proposto por meio de uma metodologia de design science. Dois experimentos controlados foram realizados (com estudantes e profissionais) para demonstrar a adequação e eficácia do método SCA3DA no desenvolvimento de sistemas críticos em safety em contextos ágeis. **Resultados:** Nosso estudo oferece uma visão geral do desenvolvimento de sistemas críticos em safety em contextos ágeis. Embora os métodos ágeis tenham sido aplicados em todas as fases do ciclo de vida de safety, os requisitos de safety do sistema e a validação de safety têm recebido mais atenção, devido à natureza crítica desses sistemas. Também apresentamos evidências de que equipes ágeis são mais propensas a não levar em conta a verdadeira intenção dos requisitos de safety devido a mal-entendidos no desenvolvimento de sistemas críticos em safety. Além disso, os resultados mostraram que a aplicação do método SAC3DA é promissora em termos de fornecer suporte positivo para uma melhor compreensão da especificação de requisitos de safety de software e que soluções arquiteturais centradas em safety levaram a uma redução no tempo necessário para sua análise, sem perda de compreensão dos requisitos. **Conclusão:** Nosso trabalho representa um ponto de partida para desenvolver uma comunicação eficaz em contextos ágeis. As soluções derivadas do método SCA3DA servem como guia para comunicar necessidades relacionadas à safety à equipe ágil, promovendo cooperação na resolução de conflitos e tomada de decisões. Um dos principais desafios encontrados ao definir o método é tornar explícita a verdadeira necessidade (intenção) do requisito de safety no contexto ágil. Enquanto as abordagens existentes se concentraram em "o que deve ser feito", este trabalho buscou introduzir o conceito de "como e por que deve ser feito". Ao fazer isso, a compreensão fica mais clara e evita-se suposições incorretas. Acreditamos que este trabalho fornece insights valiosos sobre a importância de melhorar a compreensão da especificação de requisitos de safety. Portanto, as equipes ágeis podem realizar especificações de safety em nível de software mais precisas e também utilizar essas soluções para melhorar a comunicação da equipe e garantir uma compreensão única da criticidade do sistema e uma interpretação mais precisa dos requisitos de safety.

Palavras-chave: Arquitetura de software, Desenvolvimento ágil, Safety e Sistemas safety-críticos.

List of Figures

List of Tables

Contents

CHAPTER 1 Introduction

This thesis investigates how to support software safety requirements specification of software-intensive safety-critical systems in agile contexts. This chapter introduces first the context of this thesis, then discusses the problem statement as well as presents the objectives and the research methodology of this thesis and, finally, presents a description of the overall structure of this thesis.

1.1 Context

This section introduces the context of this thesis, including safety-critical systems, agile development, and safety-critical systems development in agile contexts.

A. Safety-Critical Systems

Safety-critical system (safety-related system) is a designated system that both implements the required safety functions necessary to achieve and maintain a safe state for the equipment under control; and it is intended to achieve, on its own or with other safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions (IEC 61508, 2010).

Software-intensive safety-critical systems are the actual systems that are the subjects of this research. The software in such systems is called safety-critical software, which is stated by Leveson (2012). Safety-critical software is any software that can directly or indirectly contribute to the occurrence of a hazardous system state. Safety is related to hazards. Without analyzing the system hazards, it is impossible to ensure that a system is safe by avoiding accidents caused by these hazards. The issue is therefore to identify what constitutes a hazard. A hazard is a state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event) (AVIŽIENIS *et al.*, 2004). Hazards can be caused by the system itself, but they can also be caused by their environment.

There are basically two reasons why a hazard occurs: (i) from systematic failure, which are failures caused during system design by the developer, e.g., a calibration error, a specification error, or a software error; and (ii) from accidental failure, which are failures caused during the system's runtime, e.g., by accidental environmental changes, by accidental malfunction of a hardware component, etc.

It is interesting to observe that software only suffers from systematic failure because software failures are caused by errors made or faults introduced during development. Safety requirements for handling such systematic software failures that are predictable shall be identified and handled during the development(ISO26262, 2018). Safety requirements have a fundamental role when it comes to ensuring traceability in the development of safety-critical systems, since they often result from safety analysis of the architecture, and, ultimately, must be addressed by elements of the architecture (ANTONINO, 2016). In this regard, the literature has provided evidence that the following aspects should be addressed when creating safety requirements specifications: (i) It is recommended to break down the specification into smaller and sub-coordinated entities; (ii) The decomposition break-down structure should be traceable to the failure mode(s), which describe failures that motivate the existence of the safety requirement and the architecture elements from multiple abstraction levels (from the high-level functional specification to the software and hardware elements that realize them); and (iii) The traces to architecture have to be explicit (RTCA/DO-178, 2012) (MäDER; GOTEL, 2012).

In this perspective, techniques and recommendations used to specify software safety requirements shall be understandable by all the persons involved in a software life cycle (KASAULI *et al.*, 2021). This implies that people from different teams and with different experiences of requirements have to be able to use them in their daily work. It is therefore imperative that the software safety requirements are defined clearly and precisely so that difficulties and ambiguities in interpreting them are avoided (ZELLER, 2021). Indeed, repairing errors introduced during the specification of safety requirements and architecture design is more difficult and more expensive than errors introduced later in the life cycle (HATCLIFF *et al.*, 2014) (BECKERS *et al.*, 2014). Actually, it has been shown that errors within and across these artifacts are more likely to lead to safety-critical failures than implementation errors (HATCLIFF *et al.*, 2014). In this regard, it is important to provide means to address issues at this abstraction level.

B. Agile Development

Agile software development is defined in the Agile Manifesto (BECK *et al.*, 2001) on the level of core values and principles. The implementation of the principles are both the agile practices and the combination of agile practices called agile methods (MEHTA; SOOD, 2023). The goal of agile methods is to allow an organization to be able to "deliver quickly"; "change quickly and often" (SANKHE *et al.*, 2022).

According to Mehta and Sood (2023), agile development methods have been designed to solve the problem of delivering high-quality software on time under constantly and rapidly changing requirements and business environments. Agile methods have a proven track record in the software and IT industries. The main benefit of agile development software, according to Al-Saqqa *et al.* (2020), is allowing for an adaptive process—in which the team and development react to and handle changes in requirements and specifications, even late in the development process. Through the use of multiple working iterations, the implementation of agile methods allows the creation of quality, functional software with small teams and limited resources.

The number of methods grew over the years so there are now around 20 different agile or lean methods. According to The Annual Survey about Agile Software Development (VERSIONONE, 2022), Scrum continues to be the most commonly used method for agile development, increasing from 58% (in the 14th survey) to 87% in the current survey. While Scrumban (Scrum + Kanban) has 26%, and Scrum/XP hybrid 13%.

Scrum is an approach based on flexibility, adaptability, and productivity (SCRUM.ORG, 2023). Scrum allows developers to choose the specific software development techniques, methods, and practices for the implementation process. Scrum provides a project management framework that focuses on the development into 30-day sprint cycles in which a specified set of backlog features are delivered. The core practice in Scrum is the use of daily 15-minute team meetings for coordination and integration. Scrum has been in use for nearly ten years and has been used to successfully deliver a wide range of products (LAYTON *et al.*, 2022).

C. Agile Development of Safety-Critical Systems

In order to establish the state of the art on agile development of safety-critical systems (SCS), we have conducted a systematic mapping, which is presented in Chapter 2, we have observed that to face the challenges imposed by large-scale software development, different safety-critical application domains have already experimented with agile methods to reduce project risk and increase flexibility and even quality. Most companies adapt or tailor agile methods due to various problems such as complexity encountered during the introduction or change to agile development. These issues lead to companies applying agile practices, which are a small and very specific part of a method that addresses different aspects. Some known examples are pair programming or daily meetings. Thus, agile context refers to either the company adopting an agile or hybrid method, or a set of agile practices for some phase or the entire development of safety-critical systems.

In the automotive domain, the Automotive Software Development Report (2021) (SCHWEIZER *et al.*, 2021), a survey on agility in the automotive industry, states that the implementation of agile approaches stems from the need for adaptability to a changing environment and increased efficiency (as depicted in Figure 1.(A)). In addition, agile method is applied in all domains, not only in multimedia series development but in safety-critical functions, such as Braking Systems, Driving Assistance, and Powertrain (in Figure 1.(B)). Besides that, the use of agile methods occurs in all development disciplines, including hardware and mechanical development (in Figure 1.(C)). Among the agile methods/practices mentioned the Scrum framework is the method most used, however, continuous techniques, such as continuous integration and deployment, show promise (in Figure 1.(D)).

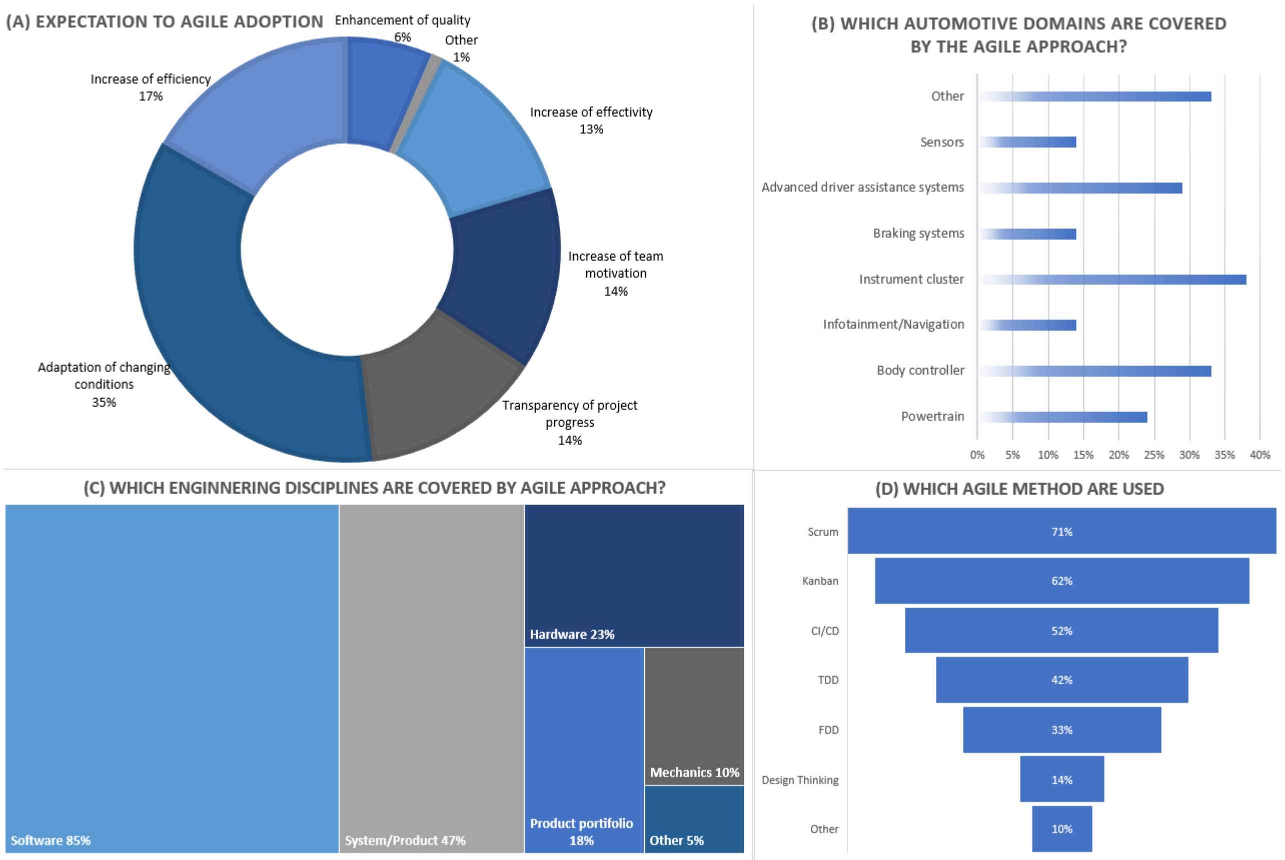


Figure 1 – Agile development in the automotive ecosystem (adapted from Schweizer *et al.* (2021)).

Regarding medical device development, in accordance with Medical Device Development Report (PERFORCE, 2019), a survey was conducted with global companies to find out about what method(s) they used previously in medical device development (as shown by blue bars in Figure 2(a)), and also what method(s) are they using today (as shown by orange bars). Therefore, we can observe an important growth in the use of agile methods/practices in medical systems development. However, medical device developers have more regulatory constraints. Teams that changed development methodologies migrated from Scrum to a hybrid agile development process, which combines agile with more traditional methods. It helps them get the documentation they need while accelerating release cycles (SCHIDEK; TIMINGER, 2022).

1.2 Problem Statement

Recent notable accidents, such as Boeing 737 crash (2019) (BBCNEWS, 2019), Nissan airbag's sensory detectors (2014) (JENSEN, 2014), crash of an Airbus A400M (2015) (HEPHER; MORRIS, 2015), Train crash in Singapore (2017) (PARK, 2017), Uber's autonomous cars struck and killed a woman (2018) (ISAAC *et al.*, 2018), have showed the vulnerabilities of

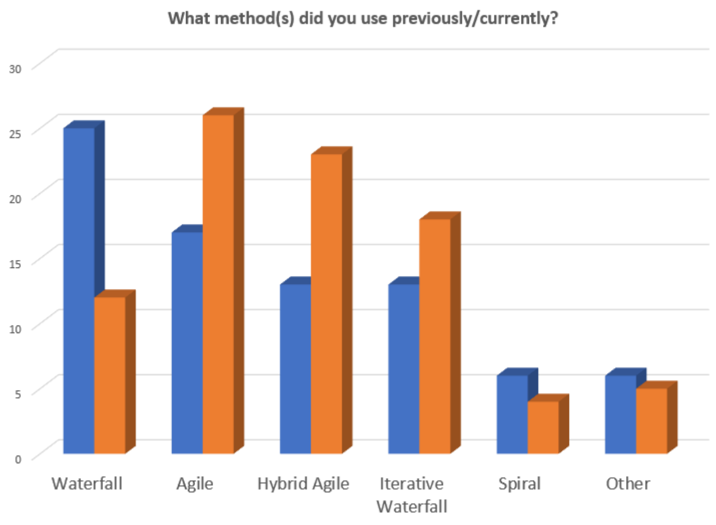


Figure 2 – Agile in medical device development.

software-intensive safety-critical systems. According to data from the U.S. Food and Drug Administration (FDA), software has been the top cause for medical device failures: a total of 22.7% of recalls in 2018 were due to software issues (TGA Health Safety Regulation, 2020). It is alarming to know that, from the top ten biggest medical device recalls list, nine out of ten of the items represent Class I (i.e., the highest safety-critical integrity level) recalls with more than half involving a significant software element (Critical Software, 2020). Likewise, in modern cars, the brakes, airbags, engines, and other important features generally need these software functions to work. A malfunction in the software can lead to an accident due to failures in the vehicle's electronic stability, braking, or forward collision avoidance systems. As the amount and complexity of software in safety-critical systems grow, even when the software does not directly control safety-critical hardware, software can provide operators and users with safety-critical data with which they must make safety-critical decisions (e.g., air traffic control or medical information such as blood bank records) (FIRESMITH, 2004). Hence, we observe the need to improve the understanding of safety requirements throughout the software development life cycle.

Moreover, these systems are heavily regulated and require certification against industry standards. These standards, for instance, ISO 26262 for automotive, IEC 62304 for medical, and DO-178B for avionics, describe strict recommendations to be included in safety requirements specification but do not present a strict methodology, method, or process to specify them. Safety requirements play an important role in the life cycle of safety-critical systems. Once, they are defined as requirements that describe means for avoiding, detecting, or handling potential failures in these systems (ISO26262, 2018). Further, those safety standards require that software and hardware safety requirements are derived from the system safety requirements and they should be traceable, but, in practice, that is done in an *ad-hoc* way, relying on the experience and background of the engineers/architects.

In the context of agile development is even more challenging, since the communication between analysis, design, and development is performed through user stories (BECK; WEST, 2014). However, such high-level specification is insufficient for properly communicating the challenges to be solved by the system in terms of safety (VUORI, 2011). In addition, when face-to-face communication is not practical among agile development teams across multiple functional areas and physical locations, each team may have an inconsistent interpretation of the system architecture (LO; CHEN, 2017). This is critical, especially because according to IEC 61508 (2010), *"from a safety viewpoint, the software architecture is where basic safety strategy is developed in the software"*. Thus, the greater communication challenges inherent in their environment require them to go beyond word-of-mouth architecture.

Based on the literature and the empirical research presented in Chapter 3 regarding the current means for providing precise software safety requirements specification in agile contexts, the following two practical problems (PP) and one scientific problem (SP) were identified:

PP1 - Ad-hoc specification of software and hardware safety requirements: When we investigated currently existing problems in state-of-the-practice (LEITE *et al.*, 2017), we observed that, although the safety standards require that software and hardware safety requirements are derived from the system safety requirements and they would be traceable, in practice, that is done in an ad-hoc way, relying on the experience of the team.

PP2 - The increased development of safety-critical systems based on agile methods/practices: Due to the increasing amount and complexity of software in safety-critical systems, the development of these systems based on agile methods has been raised as mentioned in Section 1.1. Thus, agile methods have to be adapted to address not only functional requirements but safety requirements as well.

SP1 - Lack of support to specify software safety requirements and the architectural decisions to address them in agile context: In safety-critical systems, considering the project-specific context is essential, especially because developers are more likely to fail to account for all of the variables in the software safety requirement intention, which increases the risk of bugs and errors slipping through the net. Thus, is important to know what safety-related information is needed to provide precise software safety requirements specifications and architecture solutions that address them.

The following statement summarizes the aforementioned problems into **the main research problem addressed in this Ph.D. thesis: The misunderstood software safety requirements is one key reason for their inaccurate architecture solutions and implementation in agile development.**

1.3 Objectives

Considering the relevance of the research topic (i.e., the software architectures for developing safety-critical systems), as well as motivated by the problem statement listed above, the general goal of this thesis is to contribute to the development of large, complex software-intensive safety-critical systems, through the support for the specification of software safety requirements and architectural decisions that address them in an agile context. More specifically, **the main goal of this Ph.D. project is to provide a method to specify software safety requirements in agile contexts**. To achieve the goal, the four research questions were derived and presented below:

RQ1 - How safety is currently considered and specified in the agile context?

RQ2 - How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?

RQ3 - How to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well?

RQ4 - How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?

Figure 3 depicts the connection between the problems and the research questions. The answer to these research questions, we followed the research method detailed in the following section.

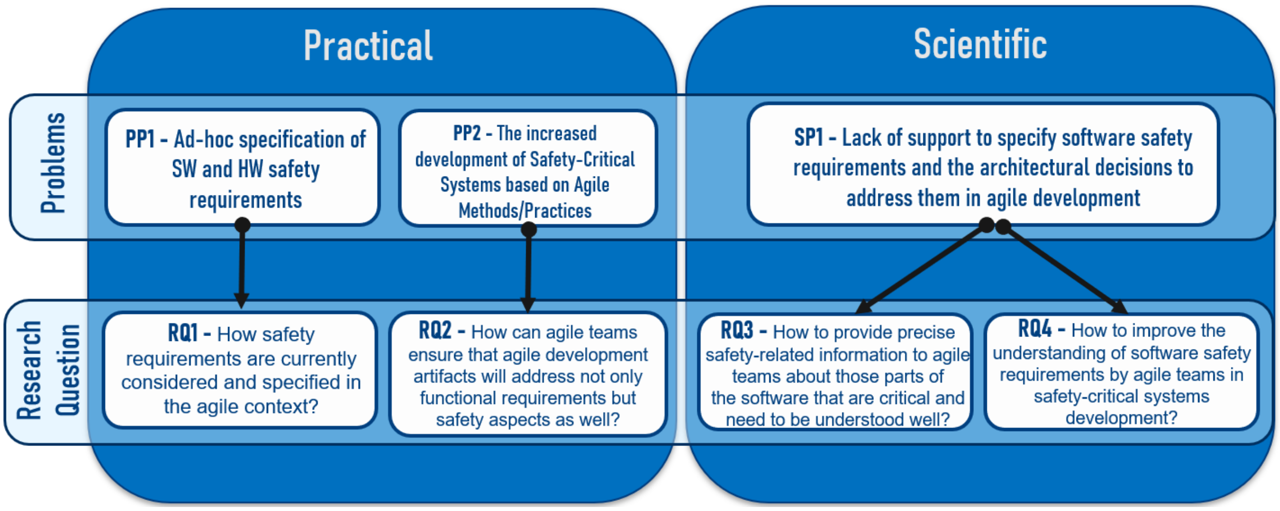

Practical
Scientific
Problems
PP1 - Ad-hoc specification of SW and HW safety requirements
PP2 - The increased development of Safety-Critical Systems based on Agile Methods/Practices
SP1 - Lack of support to specify software safety requirements and the architectural decisions to address them in agile development
Research Question
RQ1 - How safety requirements are currently considered and specified in the agile context?
RQ2 - How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?
RQ3 - How to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well?
RQ4 - How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?

Figure 3 – Mapping of problems to research questions

1.4 Research Methodology

The research project, documented in this thesis, adopted design science (WIERINGA, 2014), as a problem-solving methodology. The design science framework starts with a problem statement of a top-level practical problem. During the progress of the research, we got a deeper understanding of the problem and developed an initial solution to that. That solution was iteratively developed and evaluated with the stakeholders in the problem domain.

The process by Wieringa (2014) and Peffers *et al.* (2007) from our design science methodology consisted of six activities, which are depicted in Figure 4. Moreover, we present where the results derived from each activity are documented (from Chapters 2 to 6). Each chapter outlines a set of sub-questions that provide further specificity and structure to the research process. Table 1 presents a mapping of the research questions (depicted in Figure 3) and sub-questions, associated with their research methods that were employed to answer the RQs and the corresponding chapter where each question is addressed.

The first activity conforms to Peffers *et al.* (2007) was identifying and defining a specific problem or need that requires a solution. In order to characterize the problem we had to understand the software-intensive safety-critical systems development in agile contexts in depth. Hence, we formulated our first specific research question *RQ1 - How safety is currently considered and specified in the agile context?*. To achieve this, we performed a systematic mapping study (whose protocol is described in Appendix A) according to Petersen *et al.* (2015). This study involved a comprehensive review of existing research and literature to understand the current state of knowledge on this topic. In Chapter 2, we answered two specific questions: *RQ1.1 - Which safety activities of SCS development are being carried out in agile contexts?, RQ1.2 What safety-related information has been considered throughout agile development?*. These questions aimed to better understand how safety (measures, requirements, or strategies) has been addressed throughout agile development, focused on: (i) the context in which they have been applied to; (ii) what safety information has been considered; and (iii) challenges faced by safety-critical domains in agile safety-critical software development.

After carefully analyzing the state of the art in agile development of safety-critical systems, the second activity was defining the objectives, i.e., setting goals as based on the problem. In this regard, we needed to explore and understand the problem environment with stakeholders. For this, we defined our second research question: *RQ2 - How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?*. To answer this question, we conducted an observational case study, which involved the systematic and in-depth examination of a single or a few specific cases or instances within their real-world context or environment. This approach is often used in qualitative research to gain a deep understanding of complex phenomena, behaviors, or situations (WOHLIN, 2014). In Chapter 3, we presented the research problem, and clear objectives and goals were established for the design project. These objectives helped us to propose a solution idea and guided the

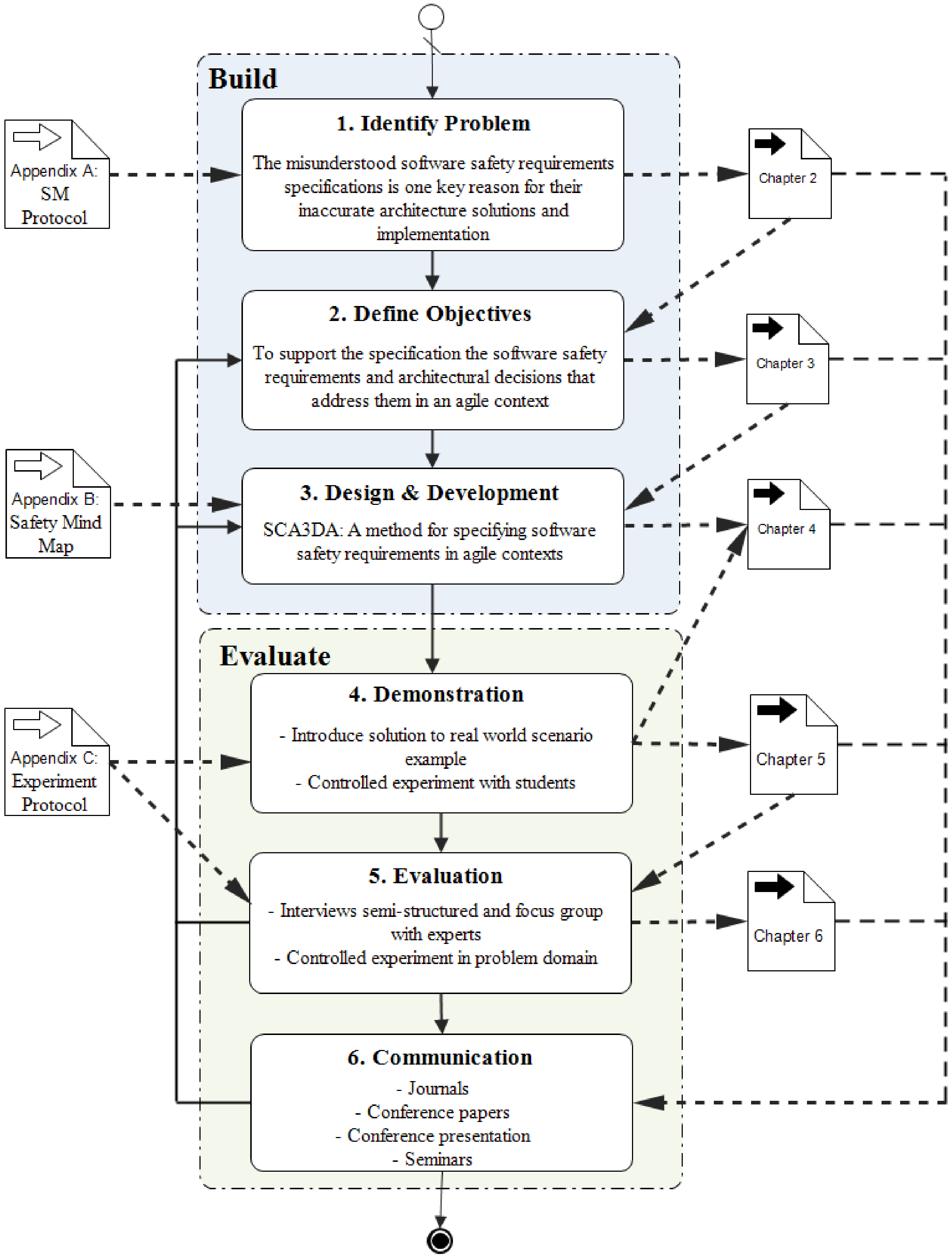

Build
1. Identify Problem
The misunderstood software safety requirements specifications is one key reason for their inaccurate architecture solutions and implementation
2. Define Objectives
To support the specification the software safety requirements and architectural decisions that address them in an agile context
3. Design & Development
SCA3DA: A method for specifying software safety requirements in agile contexts
Evaluate
4. Demonstration
- Introduce solution to real world scenario example
- Controlled experiment with students
5. Evaluation
- Interviews semi-structured and focus group with experts
- Controlled experiment in problem domain
6. Communication
- Journals
- Conference papers
- Conference presentation
- Seminars
Appendix A: SM Protocol
Appendix B: Safety Mind Map
Appendix C: Experiment Protocol
Chapter 2
Chapter 3
Chapter 4
Chapter 5
Chapter 6

Figure 4 – Our design science methodology (adapted from Wieringa (2014) and Peffers *et al.* (2007))

subsequent design and development efforts.

The third activity was design and development. At this stage, the artifact, SCA3DA (**S**afety-**C**entered **A**rchitectural **D**rivers and **D**ecisions **D**erivation) method, itself is created. The design & development, and evaluation of the method proposed were conducted through multiple design cycles with our industry collaborators. Initially, we addressed the following research question: *RQ3 - How to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well?*. To investigate this, we conducted a systematic analysis of safety standards (e.g., IEC 61508 (2010), ISO26262 (2018), IEC 62304 (2006), and RTCA/DO-178 (2012)), as a result, we defined a mind map with that information are recommended to specify safety strategies for fault detection and containment and take the system to safe state (as depicted in Appendix B). We also considered the related studies (gathered from the systematic mapping) and case study findings to define the first version of the solution.

The four activity consisted of demonstrating the artifact's functionality by implementing it in a suitable test environment and by using it to address the problem. This can also include experimental use of the artifact, simulations, case studies, and other applicable activities. Regarding this, we formulated our final specific research question: *RQ4 - How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?*. RQ4 was further decomposed into two sub-questions. To answer these questions, in Chapter 4, we presented the SCA3DA method composed of safety standard-independent metamodel and guidelines for the software safety requirements specification. More specifically, we focused on the metamodel and demonstrated its suitability in a real-world scenario example. Moreover, we performed a pilot controlled experiment with students (whose protocol and support material are depicted in Appendix C) according to the guidelines established by Wohlin *et al.* (2012), Jedlitschka *et al.* (2008) addressing the following research question: *RQ4.1 - How to measure the understanding of students on safety-critical systems development?*. In Chapter 5, we demonstrated the suitability of the method and primarily validated and refined the experiment protocol. Our objective was to comprehend how to properly measure understandability on safety-critical systems development addressing the following topics: *RQ4.1.1 - To what extent can the subject understand the safety requirements in order to derive specifications from them?*, *RQ4.1.2 - How accurate are the specifications of the safety requirements provided by the subjects?*, and *RQ4.1.3 - How much effort do subjects need to expend to specify safety requirements?*.

The fifth activity consisted of analyzing and measuring how well the method proposed solves the previously identified problem. The objectives of the solution are compared to the results observed during the demonstration. In light of this, once the SCA3DA method was proposed, it must undergo a rigorous evaluation to assess its effectiveness in addressing the identified problem in *RQ4.2 - Is the method proposed effective and suitable in agile contexts?*. To address this question, in the first evaluation activity (in the build phase), we evaluated the solution using semi-structured open-ended interviews (according to Robson and McCartan (2015)) and two focus groups to address the question of whether the SCA3DA method is suitable in agile development and the completeness of the concepts described in the metamodel. Based on the feedback, insights, and observations gathered from this activity, the method underwent further redesign and refinement cycles to continuously improve it. The second evaluation activity was to conduct a controlled experiment, considering the protocol already optimized, with practitioners from industry collaborators (in the evaluation phase), which aimed to analyze the effectiveness (with respect to understandability, accuracy, and effort) of the SCA3DA method to support the specification of software safety requirements. As a result of the activity, in each refinement cycle, we determined whether we must go back to the third activity in order to improve the effectiveness of the method, or if we should continue into the final activity and leave further development for future projects. In Chapter 6, we provided a detailed overview of SCA3DA as well as the whole evaluation process.

Finally, the sixth activity was communication. This involved sharing information about the artifact. The publications within the Ph.D. thesis, conference presentations derived from them, and the Ph.D. thesis itself serve as realizations of this final activity, representing communication with the scientific community and other stakeholders.

Table 1 – Relationships between RQs, empirical methods, and chapters

RQ ID	**Research Questions**	**Empirical Method**	**Presented in**
RQ1	How safety requirements are currently considered and specified in the agile context?	Systematic mapping study	Chapter 2
RQ1.1	Which safety activities of SCS development are being carried out in agile contexts?	Systematic mapping study	Chapter 2 (Section 2.4.3)
RQ1.2	What safety-related information has been considered throughout agile development?	Systematic mapping study	Chapter 2 (Section 2.4.4)
RQ2	How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?	Observational Case Study	Chapter 3
RQ3	How to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well?	Industrial Example	Chapter 4
RQ4	How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?		
RQ4.1	How to measure the understanding of students on safety-critical systems development?	Controlled Experiment (with students)	Chapter 5
RQ4.2	Is the method proposed effective and suitable in agile contexts?	Controlled Experiment (with practitioners)	Chapter 6

1.5 Overall Structure of this Thesis

The remainder of this thesis is comprised of six chapters. The research questions are addressed in Chapters 2 to 6. Chapter 7 contains the conclusions drawn and future research directions. Chapters 2 to 6 are based on scientific journals and conference papers, where this Ph.D. candidate served as the lead author and collaborated closely with the research group in designing and conducting the studies.

Chapter 2 is based on a paper that will be submitted to a high-impact journal soon. This chapter reports a systematic mapping study on the current evidence and gaps in how safety requirements are addressed by agile teams in safety-critical development.

Chapter 3 is based on a peer-reviewed conference paper published in the proceedings of the IEEE 25th International Requirements Engineering Conference (RE'2017) (LEITE, 2017). This chapter reports an empirical study that investigates the core problem addressed by this thesis, regarding software safety requirements specification by agile teams in safety-critical development.

Chapter 4 is based on a peer-reviewed conference paper published in the proceedings of the 34th Brazilian Symposium on Software Engineering (SBES'2020) (LEITE *et al.*, 2020). This chapter proposes the SCA3DA metamodel, which leverages the understanding of safety requirements by agile teams so that these requirements can be included in just enough safety-centered software architecture of safety-critical systems.

Chapter 5 is based on a peer-reviewed journal paper published in the IEEE Transaction Education (LEITE *et al.*, 2022). This chapter presents the results of a controlled experiment that reported the misunderstood of safety requirements by undergraduate students. Furthermore, that experiment was crucial to validate both the protocol and all materials utilized.

Chapter 6 is based on a peer-reviewed journal paper submitted in the Journal of Systems and Software. This chapter presents the SCA3DA method, which is a method for specifying software safety-critical requirements, in particular, requirements associated with the minimal set of architecturally relevant information required to specify failure detection and containment strategies to take systems to a safe state. It also presents the results of the controlled experiment that evaluated the effectiveness of the method.

Chapter 7 summarizes the results and contributions. Furthermore, it discusses limitations, and open questions that arose from the thesis, as well as potential future work regarding the different parts of this thesis: methodology, empirical validation, and tool support.

CHAPTER

2 Background

[LEITE, A., ANTONINO, P. O., NAKAGAWA, E. Y., Agile Methods in Safety-Critical Systems Development: A Mapping Study, 2023.]

Status of publication: This paper soon will be submitted to a high-ranked journal in the SE area.

Abstract

Context: Regarding the strong demand for increasing the efficiency of software development processes (in terms of effort, user satisfaction, and time) while still respecting the safety requirements imposed by relevant standards and regulations, many companies have incorporated agile practices into safety-critical projects. However, for engineering safety-critical systems using agile methods, most researchers prefer the combination of agile methods and traditional development processes that rely on safety standards. Besides that, little emphasis has been given to agile methods to address safety aspects throughout the development process. **Objective:** The goal of this paper is to better understand how safety has been addressed by agile methods, focused on: (i) the context in which they have been applied; (ii) which has safety information been considered; and (iii) challenges faced by safety-critical domains in agile safety-critical software development. **Method:** We conducted a Systematic Mapping (SM). The search strategy identified 1,598 studies, of which 52 studies were selected as the relevant ones for our SM. **Results:** The results of this SM should encourage further research into the design of studies to improve the integration between agile practices with safety practices, particularly to enable the communication of the safety aspects among the project team actors. **Conclusion:** We observe that there is still a range of open research topics to be explored, including the need for more industry-oriented studies with the participation of practitioners using and validating new approaches, that could result in high-quality safety-critical systems.

2.1 Introduction

Safety-critical systems refer to the computer software and hardware system in the safety field (DU *et al.*, 2014), whose failure could result in accidents that cause damage to the environment, financial losses, injury to people, and even the loss of lives (LEVESON, 2012), such as in the automobiles, aircraft, medical domains. The development of these systems is traditionally associated with waterfall and V-model that involve huge effort in planning, specification, and documentation. These development processes are typically rigorous and enable software certification (RUSCIO *et al.*, 2014). However, due to the ever-increasing usage and complexity of software in safety-critical systems, the adoption of these processes makes it difficult to manage requirements changes and the introduction of new technologies (COMAR *et al.*, 2009). Furthermore, the cost and time to produce safety-critical systems are perceived to be too high, especially in domains where the combination of regulations and implementation of safety-related software are relatively new (JONSSON *et al.*, 2012).

There is no specific process model required in the safety standards such as IEC 61508 (2010). These only require certain activities and documentation with some quality according to the corresponding Safety Integrity Level (SIL) (VUORI, 2011). This makes it possible for many companies to adopt agile methods to manage the increasing complexity during both development and operation and increase flexibility, transparency, and, even quality (PAGÈS *et al.*, 2012b). Although these methods do not seem at first glance to be suitable for the development of safety-critical systems, works have demonstrated the use the agile methods in the development of these systems (GARY *et al.*, 2011) (GE *et al.*, 2010) (LIAN, 2014) (WEBER, 2015). They have also discussed how to manage the challenges of introducing agile methods and practices without compromising safety.

In spite of this scenario already observed in the software industry, there is a lack of a detailed overview that could support a good understanding of how and which agile methods have been currently adopted for the development of safety-critical systems. This overview could also be the basis from which new, essential research lines on this topic could be identified, which could contribute to improving the way these systems are developed.

Aiming at characterizing the state-of-the-art current usage of agile methods in safety-critical systems development, we performed a Systematic Mapping (SM) (PETERSEN *et al.*, 2015), which is a technique for finding and summarizing available pieces of evidence on a particular research topic. In this SM, specifically, we analyzed the agile methods applied in safety-critical systems development, as well as the application domains, safety information considered, and challenges faced by these domains in agile safety-critical software development.

The remainder of this work is organized as follows. Section 2.2 presents concepts of safety-critical systems and agile methods, and also an overview of related works. Section 2.3 describes the protocol used in our SM as well as the conduction phase. Section 2.4 presents in detail the results. Section 2.5 provides a discussion and analysis on further directions. Section 2.6 describes the threats to validity. Finally, in Section 2.7, we summarize the paper and draw conclusions.

2.2 Background

In this section, we present the main concepts and terminology on safety-critical systems and agile methods, relevant to the understanding of this paper.

2.2.1 Safety-Critical Systems

Safety-critical systems are those systems whose failure could lead to consequences that are unacceptable for the user(s) and/or the environment (AVIŽIENIS *et al.*, 2004). There are many examples of such systems in diverse application areas, such as medical devices, aircraft flight control, weapons, and nuclear systems. These systems must be certified by regulatory agencies (such as the FDA[1] and TuV[2]) in order to ensure that they are safe for use. Actually, each standard or regulation, such as (IEC 62304, 2006) for medical devices and ISO26262 (2018) for the automotive domain, describes strict recommendations to be included in the safety requirements specification. These requirements describe means for avoiding, detecting, or handling potential failures in a safety-critical system (ISO26262, 2018). They include all assumptions necessary to address the respective safety goal obtained from the hazard analysis and risk assessment (BECKERS *et al.*, 2014). In this context, a safety engineer (a safety specialist) is a member of the software development team responsible by ensuring that an acceptable level of safety is achieved and maintained throughout the life cycle of the system being developed (SMITH; SIMPSON, 2011).

2.2.2 Agile Methods

Agile is a mindset supported by principles, methods, and practices; i.e., time boxing, communication, iterations, empowerment of the team, flexibility, and short feedback loops. The implementation of these principles is both the agile practices and the combination of agile practices called agile methods (DIEBOLD; DAHLEM, 2014). Agile methods, such as Extreme Programming (XP) and Scrum, are designed to get feedback early and often during the course of product design and use this feedback to continuously improve the product (ABRAHAMSSON *et al.*, 2002). These methods emphasize continuous integration and delivery rather than assuming the definition of all detailed requirements and architecture upfront (ANTONINO *et al.*, 2014). In the agile development processes, things are done in small increments with minimal long-term planning. Each iteration is worked on by a team through a full software development cycle,

[1] https://www.fda.gov/
[2] https://www.tuv.com

including planning, requirements analysis, design, coding, unit testing, and acceptance testing. This helps to minimize the overall risk and also allows the project to be more quickly adapted to changes (GARG, 2009).

2.2.3 Related work

Few systematic literature reviews (SLR) and systematic mappings discussing widely and deeply the combination of agile development and safety-critical systems are found in the literature as for example the works of Cawley *et al.* (2010), Kasauli *et al.* (2018), and Heeager and Nielsen (2018).

The work of Cawley *et al.* (2010) focuses on lean/agile software development methodologies within the medical device industry. Their work investigated how Lean/Agile methodologies are adopted in safety-critical embedded software development due to its high regulation and what its benefits were. As the main results, the authors point out that agile practices can be used in the development of medical device software and that a combination of agile and plan-driven methods appears to be suitable for the current industry context. However, their work restricted their string search to capture results related to the medical domain only.

A systematic mapping study was conducted by Kasauli *et al.* (2018) aiming to provide an overview of existing research, potential benefits, challenges, and solutions candidates of agile development of safety-critical systems. They also analyzed the current state of practice based on a cross-company workshop, in which experts were invited to discuss through the mapping findings focusing on benefits, challenges, and potential solutions.

Heeager and Nielsen (2018), synthesized the existing knowledge in the academic literature about agile development in safety-critical software through a systematic mapping concentrating on challenges in three facets: requirements, documentation, and traceability.

The above works are complementary since they explore different aspects of agile development of safety-critical systems. Nevertheless, they do not perform an extensive identification and mapping of the state-of-the-art on how safety is addressed in the agile development of safety-critical systems aiming to enhance the safety requirements understanding by agile contexts.

2.3 Research Method

We adopted the empirical method through a systematic mapping study to conduct our research. The methodology to conduct the SM, as depicted in Figure 5, was based on the guidelines proposed by Petersen *et al.* (2015). The SM was motivated by the increase in the adoption of agile methods/practices in the safety-critical system development, as reported in many studies (Ge *et al.* (2010), Paige *et al.* (2011), McHugh *et al.* (2013), Lukasiewicz and Górski (2016), Wang *et al.* (2017), Hanssen *et al.* (2018), and Barbareschi *et al.* (2022)). The gaps between the traditional development of these systems, the safety standards recommendations, and agile methods and practices also encourage the need for this SM.

2.3.1 Research Question

The purpose of this systematic mapping study is to investigate how safety has been addressed by safety-critical system development in an agile context. This leads to the research questions (RQ) described in Table 2.

Table 2 – Research questions and their motivations.

Research Questions	**Description and Motivation**
RQ1: Mapping of sources by contribution and research facets	The purpose of this question is to get information regarding the addressed topics and major contributions (e.g., process, model, tool, and technique), and the different research type facets addressed by the studies.
RQ1.1: How many studies present processes, models, tools, or techniques for addressing safety in the agile development of SCS? RQ1.2: What types of research have been used in these studies?	
RQ2: According to the safety lifecycle, which phases and activities of these systems development are being carried out in an agile context?	This question maps the activities (phases) of safety-critical system development that have been conducted in an agile context.
RQ3: What safety-related information is considered throughout the agile development of SCS?	This question aims to identify which safety-related information is considered by agile methods and practices or even to specify agile artifacts that address safety.

In the following the search protocol is presented and the systematic mapping protocol is depicted in Appendix A.

2.3.2 Search Strategy

To establish the search strategy for answering the research questions, we developed the search string by specifying the main terms of the phenomena under investigation. A number of pilot searches were performed to refine the keywords in the search string using a control group for our SM. We used five previously known studies (McDermid *et al.* (2010) and Paige *et al.* (2011), Gary *et al.* (2011), VanderLeest and Buter (2009), and Stålhane *et al.* (2012)) that were suggested by a knowledge expert. They were our baseline to check whether our search string was properly defined, i.e., if our string was able to find these studies in the publication databases. We removed

Table 3 – Searches in databases.

Database	Search
IEEE	((agile OR agility OR "extreme programming" OR xp OR "feature driven" OR scrum OR crystal OR lean OR iterative OR evo OR adaptative OR evolucionary OR "continuous development") AND ((critical OR complex) AND (safety)))
ACM	((agile OR agility OR "extreme programming" OR xp OR "feature driven" OR scrum OR crystal OR lean OR iterative OR evo OR adaptative OR evolucionary OR "continuous development") AND ((critical OR complex) AND (safety)))
Scopus	TITLE-ABS-KEY (agile OR agility OR "extreme programming" OR xp OR "feature driven" OR scrum OR crystal OR lean OR iterative OR evo OR adaptative OR evolucionary OR "continuous development") AND TITLE-ABS-KEY (safety) AND TITLE-ABS-KEY (critical OR complex)
ScienceDirect	TITLE-ABS-KEY ((agile OR agility OR "extreme programming" OR xp OR "feature driven" OR scrum OR crystal OR lean OR iterative OR evo OR adaptative OR evolucionary OR "continuous development") AND ((critical OR complex) AND (safety)))
Web of Science	((agile OR agility OR "extreme programming" OR xp OR "feature driven" OR scrum OR crystal OR lean OR iterative OR evo OR adaptative OR evolucionary OR "continuous development") AND ((critical OR complex) AND (safety)))

terms whose inclusion did not yield additional papers in the automatic searches. After several iterations, we settled on the following search string. This search string, which is expressed as a conjunction of three parts, was used to search within keywords, title and abstract.

The first part of the search string relates to agile methods/practices. The synonymous were extracted from systematic literature reviews, such as: Elghariani and Kama (2016), Albuquerque *et al.* (2012), and Dybå and Dingsøyr (2008). The second part of the search string captures keywords related to the criticality of applications (i.e. critical software, critical system, critical equipment, critical application). The third and final part of the search string concerns safety (i.e. safety certification, safety requirements, safety case, safety assurance).

We performed an automatic search on a set of five recognized scientific database libraries, namely ACM Digital Library, IEEE Xplore, ScienceDirect, Scopus, and Web of Science to find publications relevant to the scope of the review. These databases have been selected based on the experience reported by Kitchenham *et al.* (2015) and Dybå and Dingsøyr (2008).

The search strings used for each database can be found in Table 3. Parsifal (PARSIFAL, 2020), a tool to support researchers in performing systematic literature reviews, was used to remove duplicates and to manage a large number of references, as well as to extract the data. Table 4 shows the number of search results per database.

Table 4 – Number of studies per database

Database	Search Results
IEEE	199
ACM	248
Scopus	1475
ScienceDirect	23
Web of Science	297

2.3.3 Study Selection and Data Extraction

The selected primary studies needed to be assessed for their relevance, enabling the inclusion of studies that provide evidence for the research questions as well as the exclusion of studies that do not. This is achieved by the definition of Inclusion Criteria (IC) and Exclusion Criteria (EC). Thus, our criteria were:

IC_1: The study addresses agile methods/practices in safety-critical systems development;
EC_1: The study is about safety-critical systems but not in the agile context;
EC_2: The study does not present any abstract or it is not available for further reading;
EC_3: The primary study is a table of contents, short course description, tutorial, keynotes, copyright form or summary of an event (e.g., a conference or a workshop);
EC_4: The study is a previous version of a more complete one the same research, of the same authors;
EC_5: The study is written in a language other than English;

Our protocol was validated by professionals with know-how in agile development and safety engineering areas. Figure 5 depicts the results during the conducting phase, in which the primary studies were identified, selected, and evaluated using the aforementioned criteria.

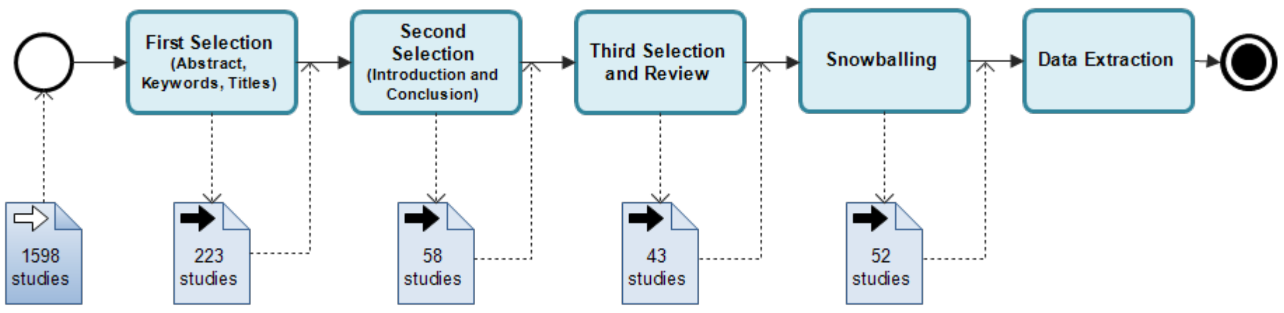


Figure 5 – Search conduction results.

First selection: The search string was customized and applied to the selected publication databases. As a result, we obtained 2242 primary studies and removed duplicate ones (i.e., 644 studies), remaining 1598 studies for analysis. Further, we excluded all editorials, prefaces, discussions, workshops, panels, and article summaries as well as studies that were clearly not about the agile development of safety-critical systems. As an example, because our search strategy included the term "lean", we got several "hits" on articles related to *Lean Manufacturing*. Articles with title and abstract that indicated clearly that the articles were outside the scope of this systematic mapping were excluded. As result of this first selection activity, a total of 223 studies were included.

Second selection: The introduction and conclusion of the 223 primary studies were read and the selection criteria were again applied. As a result, 48 primary studies were included and 175 studies were excluded. The application of inclusion and exclusion criteria to titles and abstracts, introduction, and conclusion was conducted by the first author. Thus, each article was only reviewed by a single author, which poses a threat to the reliability of the mapping study (see also the discussion of validity threats Section 2.6). In order to reduce the threat, actions have been taken to evaluate the final set of articles included, which was done at the third selection and review.

Third selection and Review: Each of the 48 studies that remained after the second selection activity was full-text read. At the end of the selection activity, a safety expert performed the selection review. First, he inspected all studies excluded in the third selection (i.e., 21 of them). Afterward, he checked the studies included by reading the abstract, and any disagreements were discussed. As a result of this discussion, a total of 43 studies were finally included as relevant to our SM.

Reference list Review (Snowballing): We used the backward snowballing technique (WOHLIN, 2014) intending to cover the whole research area. This technique allowed us to identify and examine works referenced in the studies selected in the previous activity. Among all the works evaluated, we selected 9 relevant primary study, which had not been previously identified. Finally, a set of 52 studies was selected and considered to answer the research questions, as presented in Table 5.

Data extraction: During this stage, all necessary data for our mapping study was extracted based on a predefined extraction form. This form allowed extracting all data (as described in Table 6) of 52 primary studies needed for the analysis of the research questions. As well as the selection process, data extraction was fully aided by the Parsifal tool. Besides, the extraction was performed by the first author and reviewed by the second author by tracing back the information in the extraction form to the statements in each paper and checking their accuracy.

Table 5 – Studies selected.

ID	Title	Author	Year
S1	Open-do: An open-source initiative for the development of safety-critical software	(COMAR *et al.*, 2009)	2009
S2	Adopting agile practices when developing software for use in the medical domain	(MCHUGH *et al.*, 2014)	2013

ID	Title	Author	Year
S3	Quality assurance in scrum applied to safety critical software	(HANSSEN *et al.*, 2016)	2016
S4	An iterative approach for development of safety-critical software and safety arguments	(GE *et al.*, 2010)	2010
S5	High-integrity agile processes for the development of safety critical software	(PAIGE *et al.*, 2011)	2011
S6	Agile practices in regulated railway software development	(JONSSON *et al.*, 2012)	2012
S7	Early safety analysis	(STÅLHANE; MYKLEBUST, 2016b)	2016
S8	The agile safety case	(STÅLHANE; MYKLEBUST, 2016a)	2016
S9	Quality assurance in agile safety-critical systems development	(MCBRIDE; LEPMETS, 2016)	2016
S10	An agile V-model for medical device software development to overcome the challenges with plan-driven software development lifecycles	(MCHUGH *et al.*, 2013)	2013
S11	Challenges in flexible safety-critical software development - An industrial qualitative survey	(NOTANDER *et al.*, 2013)	2013
S12	Towards agile development of critical software	(GÓRSKI; LUKASIEWICZ, 2013)	2013
S13	Escape the waterfall: Agile for aerospace	(VANDERLEEST; BUTER, 2009)	2009
S14	AgileSafe - a method of introducing agile practices into safety-critical software development processes	(LUKASIEWICZ; GÓRSKI, 2016)	2016
S15	Health modelling for agility in safety-critical systems development	(STEPHENSON *et al.*, 2006)	2006
S16	Agile methods for open source safety-critical software	(GARY *et al.*, 2011)	2011
S17	Scrum goes formal: Agile methods for safety-critical systems	(WOLFF, 2012)	2012
S18	Agile change impact analysis of safety critical software	(STÅLHANE *et al.*, 2014)	2014
S19	An assessment of avionics software development practice: Justifications for an agile development process	(HANSSEN *et al.*, 2017)	2017

ID	Title	Author	Year
S20	CARD-RM: A reference model for airborne software	(MARQUES *et al.*, 2013)	2013
S21	How can agile and documentation-driven methods be meshed in practice?	(HEEAGER, 2014)	2014
S22	Investigating the suitability of using agile for medical embedded software development	(DEMISSIE *et al.*, 2016)	2016
S23	Agile medical device software development: Introducing agile practices into MDevSPICE	(MCCAFFERY *et al.*, 2016)	2016
S24	Agile Development in Automotive Software Development: Challenges and Opportunities	(KATUMBA; KNAUSS, 2014)	2014
S25	Towards Agile Testing for Railway Safety-critical Software	(LI *et al.*, 2016)	2016
S26	An Integrated Process for Developing Safety-critical Systems using Agile Development Methods	(GUO; HIRSCHMANN, 2012)	2012
S27	Agile development in a medical device company	(ROTTIER; RODRIGUES, 2008)	2008
S28	An agile development process for petrochemical safety conformant software	(MYKLEBUST *et al.*, 2016b)	2016
S29	Agile Safety Plan	(MYKLEBUST *et al.*, 2016c)	2016
S30	Agility in the Avionics Software World	(WILS *et al.*, 2006)	2006
S31	An Exploratory Study on Applying a Scrum Development Process for Safety-Critical Systems	(WANG *et al.*, 2017)	2017
S32	A Study of Safety Documentation in a Scrum Development Process	(WANG *et al.*, 2017)	2017
S33	Adapting SafeScrum®	(HANSSEN *et al.*, 2018)	2018
S34	Assessment of Risks Introduce to Safety Critical software by Agile Practices	(GORSKI; LUKASIEWICZ, 2012)	2012
S35	The Agile Hazard Log Approach	(MYKLEBUST *et al.*, 2017)	2017
S36	A model-based agile process for DO-178C certification	(COE; KULICK, 2013)	2013
S37	Scaling agile methods to regulated environment: an industry case study	(FITZGERALD *et al.*, 2013)	2013

ID	Title	Author	Year
S38	Agile practices when developing safety systems	(MYKLEBUST *et al.*, 2016a)	2018
S39	Keeping continuous deliveries safe	(VÖST; WAGNER, 2016)	2016
S40	Discovering, Analyzing, and Managing Safety Stories in Agile Projects	(CLELAND-HUANG; VIERHAUSER, 2018)	2018
S41	ED-12C/DO-178C vs. Agile Manifesto–A Solution to Agile Development of Certifiable Avionics Systems	(MARSDEN *et al.*, 2018)	2018
S42	Compliance of agilized (software) development processes with safety standards: a vision	(GALLINA *et al.*, 2018)	2018
S43	Defining Autonomous Functions Using Iterative Hazard Analysis and Requirements Refinement	(WARG *et al.*, 2016)	2016
S44	Adaptive Software Development for developing safety critical software	(ABDELAZIZ *et al.*, 2015)	2015
S45	Test-Driven Approach for Safety-Critical Software	(ÖZçELIK; ALTILAR, 2015)	2015
S46	Hazard stories, HazId and safety stories in SafeScrum	(STaLHANE; MYKLEBUST, 2018)	2018
S47	Agile Safety Analysis	(STaLHANE; MYKLEBUST, 2016)	2016
S48	Speed up BDD for safety verification in agile development: a partially replicated controlled experiment	(WANG *et al.*, 2018)	2018
S49	The Agile FMEA Approach	(MYKLEBUST *et al.*, 2018)	2018
S50	A case study of agile software development for safety-critical systems projects	(ISLAM; STORER, 2020)	2020
S51	Automatic Test Generation to Improve Scrum for Safety Agile Methodology	(BARBARESCHI *et al.*, 2023)	2023
S52	Scrum for safety: an agile methodology for safety-critical software systems	(BARBARESCHI *et al.*, 2022)	2022

Table 6 – Extraction form

	Study Data	Description	RQ
1	ID	An unique identifier for the study	Overview
2	Title, Author, Year	bibliographic information	Overview
3	Publication Database	ACM, IEEE, Springer, Science Direct, Scopus	Overview
4	Venue	Journal, Conference, Symposium, workshop, book chapter	Overview
5	Application Context	Industrial, Academic, Both	Overview
6	Application Domain	Automotive, Medical, Avionics, Aerospace, Railway	Overview
7	Agile Method	Scrum, XP, Kaban	Overview
8	Contribution	What contributions (e.g. process, models, tools, techniques) are identified in the research literature that address safety in the agile development of SCS?	RQ1.1
9	Research facet	What type of research (e.g. validation research, evaluation research, solution proposal, philosophical papers, experience papers) have been used in approaches that address safety in agile development of SCS?	RQ1.2
10	Safety Activities	Which activities of safety-critical system development are being carried out in an agile context?	RQ2
11	Safety Information	What safety-related data/information artifacts are considered throughout agile development?	RQ3
12	Project/Team Actors	Who are stakeholders responsible for defining and verifying whether safety is addressed during the agile development of SCS?	RQ4

2.4 Results of mapping

In this section, we present an overview of the different studies by providing the general data that resulted from this study. Then the results needed to answer the research questions.

2.4.1 Overview of the Studies

This section presents an overview of the different studies by providing the general data that resulted from this study. Firstly, It is worth observing the intersection between agile and safety-critical systems, by analyzing the distribution of the published studies over the years, most of these publications were recently published, concentrating mainly on the last seven to eight years. Figure 6 presents the distribution of publications through the years. From a total of 52 publications, almost 90% were published from 2012 to 2023. That can indicate a growth in the interest in that research area in the last few years. If we consider it a trend, as well as considering the events that will occur in this and next years, we can foresee a considerable number of contributions to the development of safety-critical Systems. Moreover, it is important to highlight that these systems are quite complex and require teams of developers with high skills and differentiated competence and abilities, this justifies the need for research and the formation of high-quality human resources.

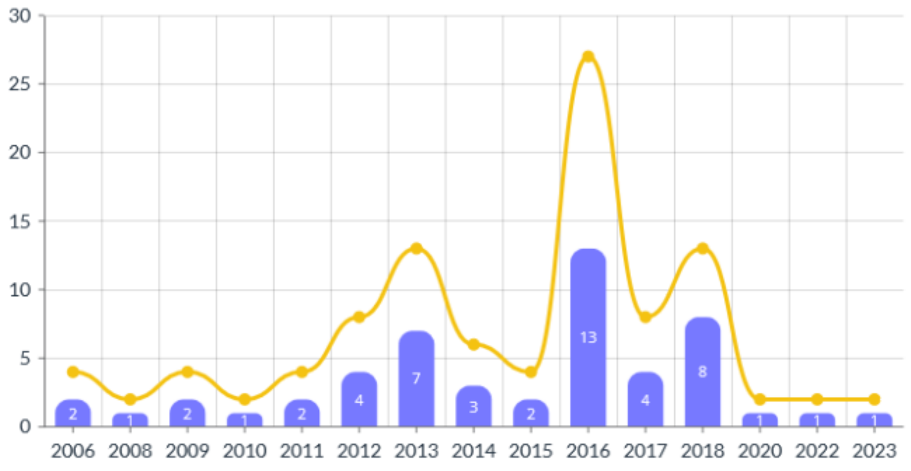


Figure 6 – Publication distribution through years

Figure 7 shows the main application domains that have developed safety-critical systems in an agile context. We identify six specific application domains (i.e., automotive, avionics, aerospace, medical, petrochemical, and railway.) and with general domain, i.e., they might be applied in any domain. Most publications are destined to this last category (general domain). Among specific application domains, medical is the most adopted and the least used is petrochemical. In the medical domain, 50% publications adopt Scrum for medical device software development and the other 50% are distributed among other methods/practices. For instance, Scrum was applied for the description of safety requirements specification, designing for safety, and testing quality improvements for the final product. In the case of the automotive domain, the industrial survey published annually about Agile Automotive (SCHWEIZER *et al.*, 2021) claims that, although there is little scientific research on the application of agile methods/practices in safety-critical systems, they are already adopted in the industry by 41% safety-critical applications types, such as braking systems and assistance/automatic driving systems. New research in this direction should be still made, supporting the industry to solve problems, such as conformance analysis between architecture and code, and agile safety requirements specification (KASAULI *et al.*, 2021). Hence, it is important to observe that safety-critical systems have been developed in an agile context. Thus, attention given to their architectures is essential to comply with the quality requirements of such systems.

2.4.2 RQ1: Mapping of sources by contribution and research facets

In this section, we summarize the results of our SMS to answer the following researchers sub-questions: "*RQ1.1 - How many studies present methods, techniques, tools, models, metrics, or processes for addressing safety in agile development of SCS?*", and "*RQ1.2 - What type of research have been applied in the studies in this area?*". Firstly, the type of contribution of each

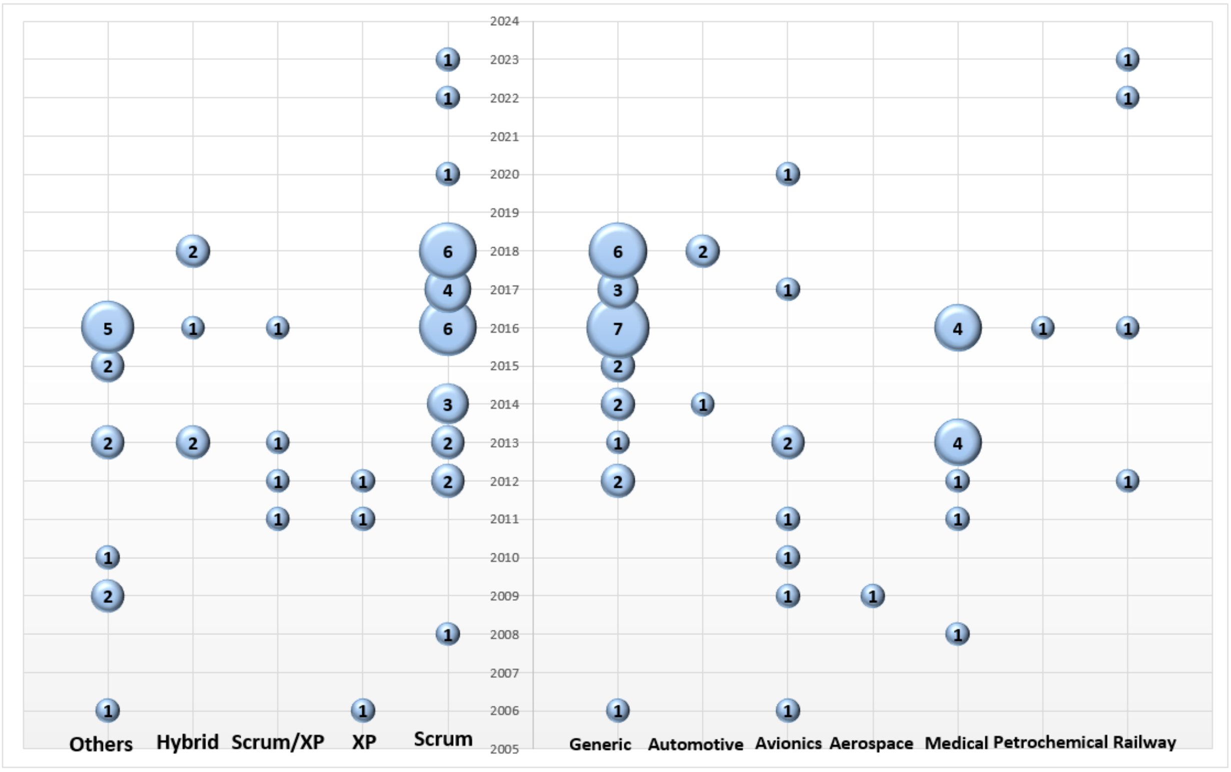


Figure 7 – Publication distribution through agile methods and domain

study was considered, which for example could be a process, method, tool, etc. These categories were derived from the keywords. These contributions were classified into three categories: (i) process or method, (ii) technique for an activity concern with safety lifecycle, and (iii) challenges or barriers to adopting agile in safety-critical system development, as shown in Figure 8. They are used in a bubble graph together with the research approach adopted by the authors of these studies. We choose an existing classification of research approaches presented by Wieringa *et al.* (2006). The central axis (or Y axis) is the publication year of the primary study; the X axis, on the left side, is the contribution facet, and, on the right side, the research approaches applied by the studies.

The majority of studies (43%) propose a new process or method related to the agile development of safety-critical systems. For example, Hanssen *et al.* (2018) presented SafeScrum, a process that aims to integrate the safety lifecycle into the Scrum process. In addition, McHugh *et al.* (2013) proposes the AV-Model process, which would be an adaptation of the V-Model on which the safety life cycle activities are conducted, but with some steps being carried out in an agile way. The studies that present new techniques for conducting safety-specific activities correspond to 33%, for instance, 'Agile Safety Analysis' proposed by Stålhane and Myklebust (2016), Test-Driven approach for safety software development' (ÖZçELIK; ALTILAR, 2015), and Safety Verification in Agile Development (WANG *et al.*, 2018). Finally, 24% of the studies discuss the challenges and barriers to applying agile methods/practices in the development of SCS. McHugh *et al.* (2013) and AAMI TIR45 (2012) state there are no external barriers to

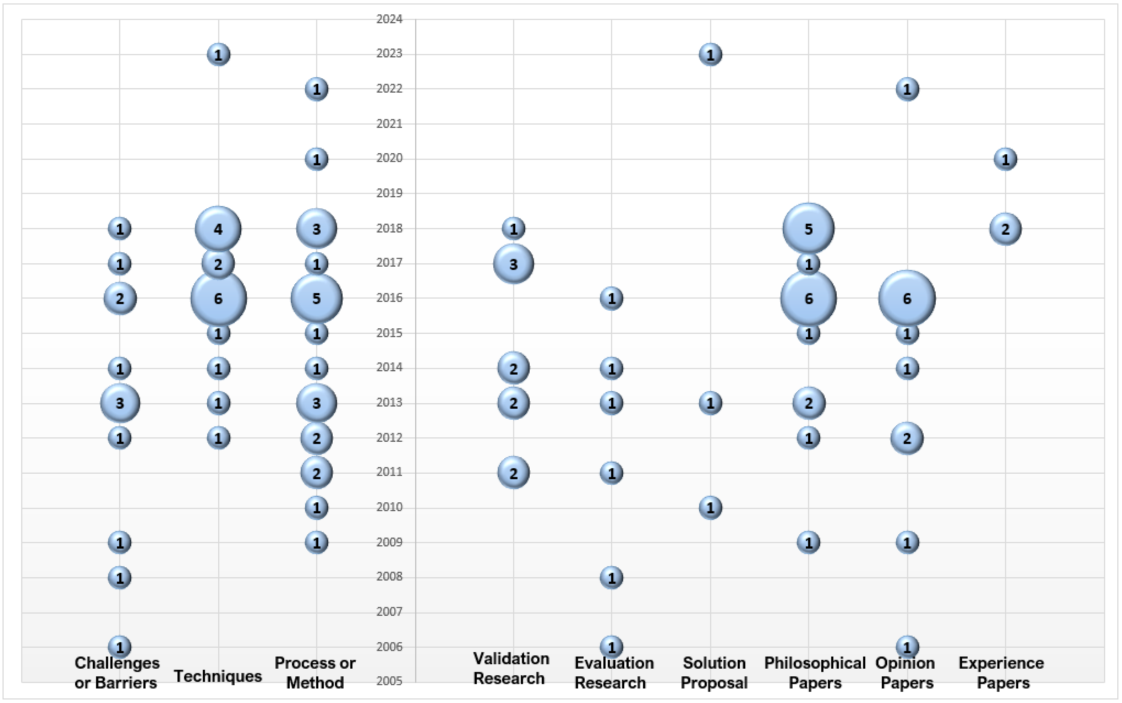


Figure 8 – Contribution and research facets

adopting agile methods when developing safety software. Existing barriers are mainly in-house ones (within the organizations), which could be overcome. In a general way, these challenges are related to task switching, i.e., people are participating in many projects at the same time and often need to jump from one task to another that might be in the context completely different. This directly impacts in the knowledge of the project parts and its development, as well as the commitment of the professional involved, and may affect the quality of the final product. Moreover, separation of concern and lack of end-to-end knowledge, quality of requirements, such as fuzzy requirements and/or vague requirements, can impair the understanding and translation of the design to its implementation (KATUMBA; KNAUSS, 2014).

On the right side, most studies (17 from 52) have presented philosophical papers, that have been published in the last years (after 2011), i.e., we can mention that agile for the development of safety-critical systems is a relatively new research topic. However, Rottier and Rodrigues (2008), Wang *et al.* (2018), Fitzgerald *et al.* (2013), Wang *et al.* (2017), and Islam and Storer (2020) are some examples of studies that have presented research approaches with empirical evidence, leading us to infer that there is a joint effort between industry and academy for attempting to solve challenges faced by the industry.

2.4.3 RQ2: Which phases, and activities of SCS development are being carried out in an agile context?

This section aims to answer our second question:"*RQ2 - According to safety lifecycle, which phases, activities of these systems development are being carried out in an agile context?*".

Figure 9 shows the distribution of 52 publications, which report these topics: (i) Traceability requirements; (ii) Safety Case; (iii) Safety Verification; (iv) Architecture; (v) design; (vi) Safety Requirements; (vii) Safety Plan; (viii) Failure modes; and (ix) Hazards and risk list. The main research topic is systems/software safety requirements, found in 44.7% of publications, followed by software safety verification specification found in 42.1% of them. Other topics are intermediates and appear from the initial development phase to the final, providing us with pieces of evidence that agile methods address relevant artifacts at all stages of safety-critical systems development. It is important to observe that safety validation specification is addressed in several publications. According to them, it is needed to incorporate safety validation practices into agile methods, since these are fundamental for safety-critical systems development (HANSSEN *et al.*, 2016) and (MCBRIDE; LEPMETS, 2016). It is noteworthy that architecture is considered the backbone in safety-critical development, 21% publications explicitly establish a systems/software architecture to develop safety-critical systems using agile methods/practices. In fact, it is difficult to define whether or not an up-front model is sufficiently accurate and detailed in an agile safety-critical software project (GE *et al.*, 2010).

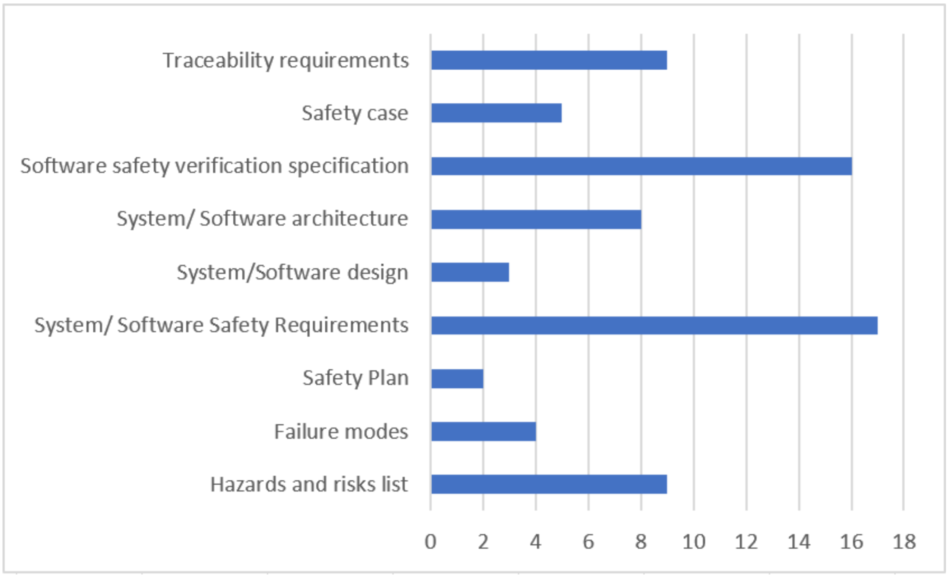
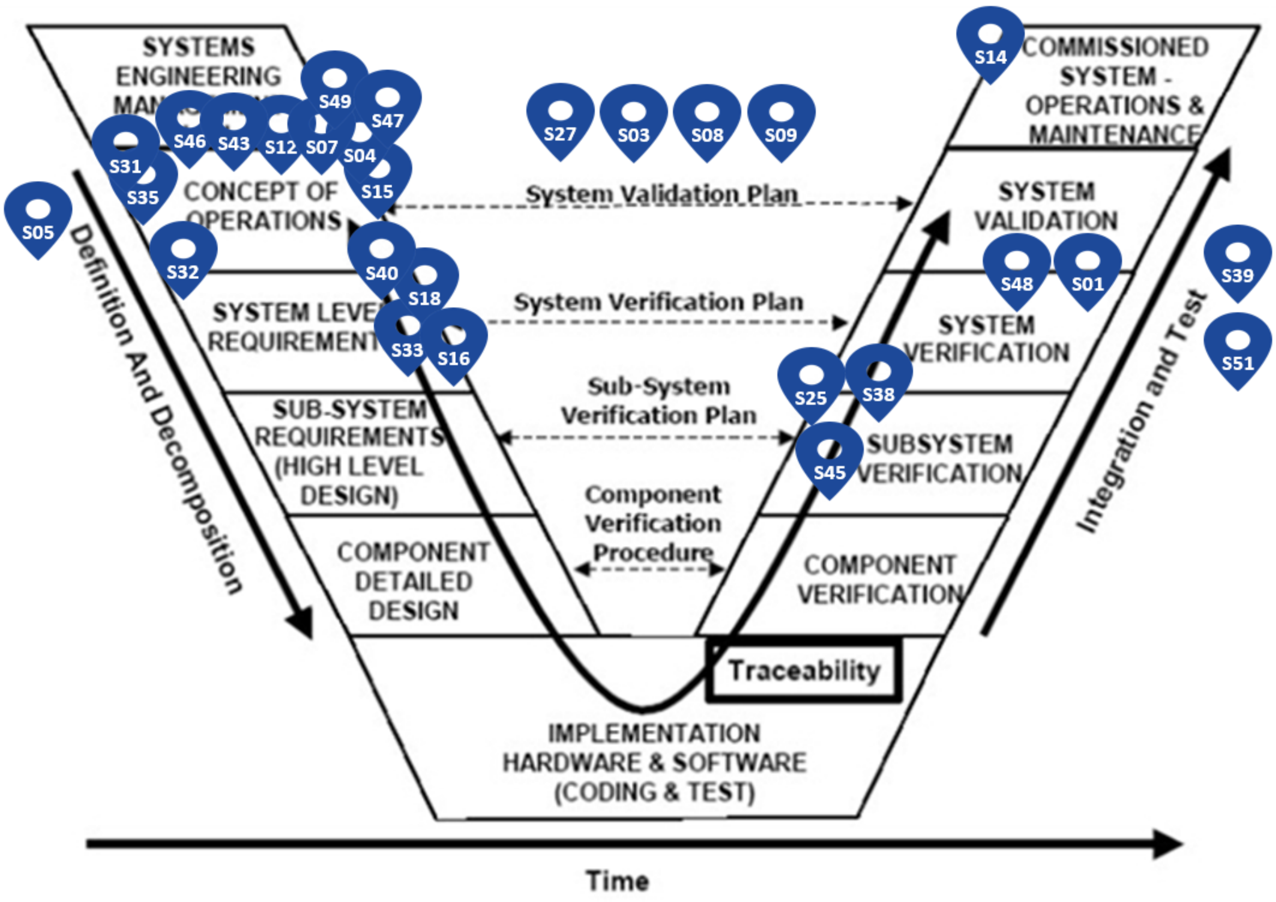


Figure 9 – Research topics on SCS development in agile context

In another perspective, Figure 10 depicts the studies distributed among the different phases of the safety lifecycle based on the V-model (ABDELAZIZ *et al.*, 2015). Not all studies were presented in the lifecycle, since their contributions refer to a new process, such as Hanssen *et al.* (2018) and Islam and Storer (2020), and therefore it is transversal to this lifecycle. It is possible to observe that the three main phases covered by the studies are: Hazard and Risk Analysis (20% studies), System Safety Requirements (10% studies), and Quality Assurance (8% studies). For example, Myklebust *et al.* (2017) proposes an Agile Hazard Log process based on Scrum, in which all safety management activities, such as hazards identified, decisions made, and solutions adopted should be collected and registered alongside the software development. Moreover, Stålhane and Myklebust (2016b) presents a safety requirements process in an agile setting based on the early top-level requirements, whereas Myklebust *et al.* (2016a) and Wang *et al.* (2017) suggest safety stories, safety epic, and agile safety plan can improve the safety-related communication in a Scrum development process in relation to maintain the safety assurance's capability (WANG *et al.*, 2017). Further, (BARBARESCHI *et al.*, 2022) proposes exploiting automatic generation of unit tests and coverage metrics to find any inconsistency between several regulations-required artifacts, including the requirements specification, the architectural specification, test specifications, and their implementation.


SYSTEMS ENGINEERING MANAGEMENT
CONCEPT OF OPERATIONS
SYSTEM LEVEL REQUIREMENTS
SUB-SYSTEM REQUIREMENTS (HIGH LEVEL DESIGN)
COMPONENT DETAILED DESIGN
IMPLEMENTATION HARDWARE & SOFTWARE (CODING & TEST)
COMPONENT VERIFICATION
SUBSYSTEM VERIFICATION
SYSTEM VERIFICATION
SYSTEM VALIDATION
COMMISSIONED SYSTEM - OPERATIONS & MAINTENANCE
System Validation Plan
System Verification Plan
Sub-System Verification Plan
Component Verification Procedure
Traceability
Definition And Decomposition
Integration and Test
Time
S05
S31
S35
S46
S43
S12
S07
S49
S47
S04
S15
S32
S40
S18
S33
S16
S27
S03
S08
S09
S14
S48
S01
S39
S51
S25
S38
S45
Figure 10 – Distribuition of studies among safefy lifecycle (adapted from Abdelaziz *et al.* (2015)).

2.4.4 RQ3: What safety-related information is considered throughout the agile development of SCS?

Section 2.4.3 presented the activities along the safety lifecycle that have been carried out in an agile context. In this sense, it is important to identify the safety artifacts that are addressed throughout the agile development of safety-critical systems. Then, this section presents the results of answering the '*RQ3 - What safety-related information is considered throughout the agile development of SCS?*'.

The main safety work product is systems/software safety requirements, found in 17 studies (i.e., 34.7% of them), followed by software safety verification specification found in 16 studies (i.e., 32.6% of them). Other work products are intermediates and appear from the initial development phase to the final, providing us evidence that agile methods are addressing relevant artifacts at all stages of safety-critical systems development.

Safety requirements are defined by Jonsson *et al.* (2012), Stålhane and Myklebust (2016b), Hanssen *et al.* (2017), Guo and Hirschmann (2012), Wang *et al.* (2017), Wang *et al.* (2017), and Hanssen *et al.* (2018) as user stories. Specifically, Wang *et al.* (2017) establishes a safety story pattern, which considers safety requirements specification from the unsafe control actions, whereas studies Wolff (2012) and Coe and Kulick (2013) advocate for a formal specification of these requirements, through a model-based agile process. In addition, study Guo and Hirschmann (2012) proposes that software safety requirements specification should be performed according to Scrum's sprints, as Stålhane and Myklebust (2016b) that proposes an agile process to identify new safety requirements from failure modes, hazard lists, and generic architectural patterns. As discussed in Jonsson *et al.* (2012), XP stories are not enough to capture safety requirements, and, hence, well-structured requirements engineering is also demanded by safety standards. Thus, we observe there is still no consensus regarding which agile requirements practices are most adequate to address safety requirements. This leads us to infer that more studies should be conducted to allow more widespread and efficient practices to specify safety requirements.

Further, we have also observed that safety validation specification is addressed in several studies: Hanssen *et al.* (2016), Ge *et al.* (2010), McBride and Lepmets (2016), McHugh *et al.* (2013), VanderLeest and Buter (2009), Hanssen *et al.* (2017), Marques *et al.* (2013), McCaffery *et al.* (2016), Guo and Hirschmann (2012), Rottier and Rodrigues (2008), Wang *et al.* (2017), Fitzgerald *et al.* (2013) and Myklebust *et al.* (2016c). For instance, Guo and Hirschmann (2012) describes an agile safety life cycle and proposes that software aspects of system safety validation should be considered throughout Sprint Review and Retrospective, for facilitating stakeholder involvement and continuous improvement of products. Besides that, in McCaffery *et al.* (2016), it is proposed to integrate agile practices into the MDevSPICE framework, amongst them iterative testing and TDD to assure that all safety implications are reflected in test cases. Therefore, according to authors of Hanssen *et al.* (2016) and McBride and Lepmets (2016), it is needed to incorporate safety validation practices into agile methods, since these are fundamental for safety-critical systems development (HANSSEN *et al.*, 2016; MCBRIDE; LEPMETS, 2016).

It is noteworthy that architecture is considered the backbone of safety-critical development, but only eight studies explicitly establish a systems/software architecture to develop safety-critical systems using agile methods/practices. An example can be seen in Stephenson *et al.* (2006) that proposes an iterative safety architecture to address safety concerns in the evolving design as soon as they arise. Although the establishment of a software architecture is not the focus in the safety-critical systems development with the application of agile methods/practices, at least a system architecture model is needed to start the design of a more detailed architectural description. In fact, it is difficult to define whether or not an up-front model is sufficiently accurate and detailed in an agile safety-critical software project (GE *et al.*, 2010).

Table 7 provides an overview of the included studies, which were classified as agile methods adopted in the safety-critical systems development, their applications domain, and work product that have been considered in the agile development.

Although there are several agile methods proposed in the literature, only Scrum and XP have been investigated and used in the development of safety-critical systems, as found in the 38 included studies and presented in Table 7. Agile methods have been adapted and/or combined with safety principles in order to maintain compliance with the software assurance requirements imposed by the application domain. We found nine studies (S5, S15, S17, S19, S20, S26, S28, S33, and S52), which are related to the adaptation of Scrum or XP for safety-critical systems. For instance, S5 proposes, based on XP, a risk-based iterative process using risk-based planning and verification, with safety analysis and safety argumentation activities. Other examples are S9, S33 and S52 which adopted Scrum. The former establishes software development and verification processes for RTCA/DO-178 (2012), a certification standard applied in the aerospace domain. The latter proposes SafeScrum, which makes use of the separation of concerns to effectively apply agile methods in the development, verification, and traceability of safety-critical systems; to do so, all requirements are split into safety-critical requirements and other requirements, and inserted into separate product backlogs (i.e., Backlog Splitting) that will be implemented. Hybrid models are applied in three studies (S10, S14, and S36), these studies combine agile and plan-driven development practices. S10 is applied in the medical domain and produces as work products a software architecture and a software safety verification specification. S14 proposes an approach based on agile practices to create the work-product safety case. Aerospace is the application domain adopted by S36 and the hazards and risks list together with systems safety requirements are the work products produced by it.

Specific safety-critical software development phases were also found in studies S4, S7, S8, S9, S18, S25, S29, S32, and S35, such as safety analysis, safety planning, and safety requirements specification. For example, S35 proposes an Agile Hazard Log process based on Scrum, in which all safety management activities, such as hazards identified, decisions made, and solutions adopted should be collected and registered alongside the software development. Moreover, S7 presents a safety requirements process in an agile setting based on the early top-level requirements, whereas S29 and S32 suggest safety stories, safety epic, and agile safety plans can improve the safety-related communication in a Scrum development process in relation to maintaining the safety assurance's capability (WANG *et al.*, 2017).

Besides the agile methods, agile practices have been also adopted in the development of safety-critical systems. To develop medical device software, S23 employs the practices of user stories, test-driven development (TDD), continuous integration, and sprint planning meetings, and its main work-products were system safety requirements and traceability specification. A

Table 7 – Overview of included studies.

Application domain							Safety Activities									Agile					
Railway	Petrochemical	Medical	Aerospace	Avionics	Automotive	Generic	Traceability requirements	Safety case	Software safety verification specification	System/ Software architecture	System/Software design	System/ Software Safety Requirements	Safety Plan	Failure modes	Hazards and risks list	Agile Practices	Hybrid	Scrum/ XP	XP	Scrum	Primary Studies
				√				√								√					S1
		√									√	√				√					S2
						√	√		√									√			S3
				√				√	√	√		√			√	√					S4
				√				√		√									√		S5
√							√		√			√							√		S6
						√						√		√	√	√					S7
						√		√							√					√	S8
		√							√							√					S9
		√							√	√							√				S10
										√										√	S11
		√													√			√			S12
			√						√							√					S13
						√		√									√				S14
						√				√				√	√				√		S15
		√					√			√								√			S16
						√						√								√	S17
						√					√	√								√	S18
				√			√		√	√		√								√	S19
				√					√											√	S20
		√							√			√				√					S21
		√										√						√			S22
		√					√					√				√					S23
					√						√	√				√					S24
√									√							√					S25
						√			√			√								√	S26
		√							√	√										√	S27
	√						√					√								√	S28
						√							√							√	S29
				√			√		√										√		S30
						√						√		√	√					√	S31
						√						√	√							√	S32
√						√	√		√			√								√	S33
		√													√	√					S34
						√								√	√					√	S35
			√									√			√		√				S36
		√					√		√											√	S37
						√			√											√	S38
						√			√			√			√	√					S39
						√	√	√			√	√		√	√					√	S40
				√								√					√				S41
						√	√	√									√				S42
					√							√			√		√				S43
						√		√				√	√		√		√				S44
						√			√					√		√					S45
						√						√		√	√					√	S46
						√		√				√		√	√					√	S47
						√			√			√		√						√	S48
						√						√		√	√		√				S49
√							√			√				√	√	√					S50
				√					√							√					S51
√							√		√			√								√	S52
5	1	11	2	8	2	24	13	9	21	9	4	27	3	11	17	15	8	4	4	21	Total

similar set of agile practices was also considered in study S6, mapping the main XP practices to CENELEC - EN 50128 (2011) requirements, such as pair programming, continuous integration, coding standard, TDD, and collective code ownership. S38 extended the S33 Backlog Splitting with adaptations to Safety-TDD, which is a modified use of TDD for developing safety-critical systems. The combination of Scrum and XP was also experimented in S3, S12, and S16. S3 combines Scrum practices, such as product backlog and sprint review, with the most common XP practices, i.e. TDD, regular work iterations, daily stand-ups, and continuous integration, for safety validation. Besides that, S12 investigates the risks introduced by the practice's product backlog, user stories, acceptance tests, and product release in safety-critical software development. As a result, the authors conclude neither Scrum nor XP practices were suitable for safety-critical projects in their strict form, but rather need to be tailored. A more detailed discussion can be found in S6, S12, S13, S23, S38, S41, S42, and S50.

The main safety work product is systems/software safety requirements, found in 17 studies (i.e., 44.7% of them), followed by software safety verification specification found in 16 studies (i.e., 42.1% of them). Other work products are intermediates and appear from the initial development phase to the final, providing us evidence that agile methods are addressing relevant artifacts at all stages of safety-critical systems development.

Safety requirements are defined by S6, S7, S19, S26, S31, S32, S33, S40, S46, AND S52 as user stories. Specifically, S32 establishes a safety story pattern, which considers safety requirements specification from the unsafe control actions, whereas studies S17 and S36 advocate for a formal specification of these requirements, through a model-based agile process. In addition, study S26 proposes that software safety requirements specification should be performed according to Scrum's sprints, as S7 proposes an agile process to identify new safety requirements from failure modes, hazard lists, and generic architectural patterns. As discussed in S6, XP stories are not enough to capture safety requirements, and, hence, well-structured requirements engineering is also demanded by safety standards. In S40, a safety story is defined by the EARS template (MAVIN *et al.*, 2009). Thus, we observe there is still no consensus regarding which agile requirements practices are most adequate to address safety requirements. This leads us to infer that more studies should be conducted to allow more widespread and efficient practices to specify safety requirements.

It is important to observe that safety validation specification is addressed in several studies: S3, S4, S9, S10, S13, S19, S20, S23, S26, S27, S32, S37, S38, S45, and S51. For instance, S26 describes an agile safety life cycle and proposes that software aspects of system safety validation should be considered throughout Sprint Review and Retrospective, for facilitating stakeholder involvement and continuous improvement of products. Besides that, in S23, it is proposed to integrate agile practices into the MDevSPICE framework, amongst them iterative testing and TDD to ensure that all safety implications are reflected in test cases. Therefore, according to authors of S3 and S9, it is needed to incorporate safety validation practices into agile methods, since these are fundamental for safety-critical systems development (HANSSEN *et al.*, 2016; MCBRIDE; LEPMETS, 2016).

It is noteworthy that architecture is considered the backbone of safety-critical development, but only eight studies explicitly establish a systems/software architecture to develop safety-critical systems using agile methods/practices. An example can be seen in S15 which proposes an iterative safety architecture to address safety concerns in the evolving design as soon as they arise. Although the establishment of a software architecture is not the focus in the safety-critical systems development with the application of agile methods/practices, at least a system architecture model is needed to start the design of a more detailed architectural description. In fact, it is difficult to define whether or not an up-front model is sufficiently accurate and detailed in an agile safety-critical software project (GE *et al.*, 2010).

We identify six specific application domains (i.e., automotive, avionics, aerospace, medical, petrochemical, and railway.) and studies with general domains, i.e., they might be applied in any domain. Most studies are destined to this last category (general domain). Among specific application domains, medical is the most adopted by studies and the least used is automotive and petrochemical. In the medical domain, we found S9, S10, S12, S16, S23, S27, and S37, of which 50% adopt Scrum for medical device software development, and the other 50% are distributed among other methods/practices. For instance, S16 uses Scrum for the description of safety requirements specification, designing for safety, and testing quality improvements for the final product. Both in the automotive and petrochemical domains, we found only one primary study, S24 and S28, respectively. In the case of the automotive domain, it is evident the a lack of research highlighting how safety has been considered by agile methods. The industrial survey published annually about Agile Automotive (SCHWEIZER *et al.*, 2021) claims that, although there is little scientific research on the application of agile methods/practices in safety-critical systems, they are already adopted in the industry by 41% safety-critical applications types, such as braking systems and assistance/automatic driving systems. New research in this direction should be still made, supporting the industry to solve problems, such as conformance analysis between architecture and code, and agile safety requirements specification (JONSSON *et al.*, 2012).

2.5 Discussion

Through the results of our SM, the existing knowledge about the adoption of agile methods/practices in the development of safety-critical systems was widely and thoroughly mapped. We found that agile methods and their practices have had a positive impact on safety-critical systems development. The main reason for adopting agile is that traditional software development processes are not flexible enough to address the emerging challenges in software development, e.g., high complexity increase, constantly changing requirements, and shorter time to market.

The return on experience is very positive (WEBER, 2015) (WILS *et al.*, 2006) (KAPPE, 2013). However, their use is still not widespread.

As far as using the knowledge presented in this work for further research is concerned, we identified important, interesting research lines that should be investigated soon:

- Investigation on how to better integrate agile methods/practices with safety and risk management practices;
- Conduction of qualitative and quantitative analysis regarding, for instance, cost/effort reduction and quality improvement resulting from the use of agile methods. Although agile methods highlight improving quality, the quality control mechanisms have not proven to be adequate to assure that the product is safe;
- Investigation on how to adapt agile methods when specific standards such as ISO 26262 must be followed, similarly to the one performed by AAMI TIR45 (2012), which mapped each of the identified suitable agile practice to the appropriate stage of development in IEC 62304 (2006); and
- Investigation on how agile methods impact different phases of the safety-critical systems life cycle, including hazard and risk analysis, safety requirements specification/analysis, safety design, safety validation and verification, safety assurance, and certification.

2.6 Threats to Validity

This section discusses what is considered to be the most important threats to the validity of this SM according to Wohlin *et al.* (2012). A threat to *internal validity* was related to missing of important primary studies. According to Wohlin (2014) two mapping studies of the same topic ended up with different sets of articles. To reduce this threat, we complemented the search with backward snowballing of all studies after full-text reading (see Figure 5). Moreover, researcher biases may appear during the selection and extraction of data. The study selection was conducted by an individual author, which is the main threat to validity. To reduce this threat and gain confidence in the results, we evaluated the study identification through the creation of a reference set of articles. This was done by the first author through snowballing technique. Only a small number of new studies have been obtained, indicating that the overall conclusions of this review would not change. In spite of our effort to include all relevant evidence, it is possible that primary studies were missed. The second threat, *to construct validity*, regards the data extraction and classification, during this phase researcher bias is also a threat. Kitchenham *et al.* (2015) states that it is useful to have one researcher to extract the data and another reviewing the extraction. To reduce the bias, the second author assessed all extractions made by the first author. However, given that this step involves human judgment, the threat cannot be eliminated. Furthermore, another threat, to *conclusions validity* relates to a researcher bias in the interpretation of data. The first author is a PhD student in Software Engineering and Safety and the other three are experienced researchers in Requirements Engineering, Software Engineering, or Safety-Critical Systems. Furthermore, aiming to increase the reliability of our SM, a safety expert reviewed the study's selection by applying the random choice strategy defined by Kitchenham and Charters (2007). However, it might be possible that relevant studies were excluded in the first stages due to the lack of important information in the title, abstract, keywords, introduction, and/or conclusions sections of these studies. Finally, to *external validity* is concerned with establishing the generalizability of the SM results. In order to mitigate external threats the search was conducted in five publication databases. In addition, we wanted to be as inclusive as possible, conforming to our inclusion and exclusion criteria.

2.7 Conclusion

Safety-critical systems are becoming increasingly large and complex, and most of their functionalities have been intensively controlled by software components. It is essential to use a suitable development process model to develop high-quality software systems. Although agile methods seem initially not suitable for safety-critical systems, different application domains have already experimented with agile methods to reduce project risk and increase flexibility and even quality. In this paper, we present the results of an SM to characterize the state of the art concerning the current usage of agile methods in the context of safety-critical systems. This research area is relatively new and we found 52 studies, which are mainly concentrated in the last six years (i.e., 30 studies were published from 2015 to 2023).

As the main result, we observe there are already important, effective contributions exploring agility for the safety-critical domain. However, more attention should be given to this topic, especially regarding the establishment of a consensus among researchers about a way to become agile practices more suitable to safety and risk management practices. Based on these results, we also identified some lines of research that must be explored, yet. Furthermore, we intend this overview to open other new research lines, contributing to a more effective, successful development of safety-critical systems.

CHAPTER 3

An Approach to Support the Specification of Agile Artifacts in the Development of Safety-Critical Systems

[Published as: LEITE, A. An approach to support the specification of agile artifacts in the development of safety-critical systems. In: *Intl. Requirements Engineering Conference (RE)*. 2017. p. 526– 531.]

Abstract

Context: Providing a correct, complete, and unambiguous requirements specification is still one of the biggest challenges in software engineering. In the case of safety-critical systems, this challenge is even greater, since misinterpretations can lead to catastrophic damage to humans and to the environment. Agile development proposes minimizing the challenges in requirements specifications through short iterations, quick feedback, and active stakeholders. However, in safety-critical systems development, there is a gap (either geographical, cultural, educational, or temporal) between safety engineers and developers. **Problem:** Therefore, it is not possible to be assured by agile development teams that safety aspects are well understood by developers, and, if the latter are aware of the criticality of the problem, they can implement them accordingly. **Objective:** The proposed research aims to provide adequate support for more accurate specification of agile development artifacts in the development of safety-critical systems. In this regard, the first contribution of this research aims at defining an Agile Safety Process, whose purpose is to identify which artifacts or parts thereof are enough to specify failure detection and containment, as well as measures for taking the system to a safe state. The second contribution aims at providing a semi-automated methodology for supporting the specification of agile artifacts, taking into account safety aspects. **Expected results:** As a consequence, this

research will have a significant impact in terms of improving the creation of evidence to be submitted for certification in terms of timing and accuracy.

3.1 Introduction

Defining a requirements specification that is complete, correct, unambiguous, and understood by all stakeholders is one of the biggest challenges for safe software engineering. This challenge becomes more problematic in the context of safety-critical systems (e.g., medical devices, automotive, avionics, nuclear power plants) (HATCLIFF *et al.*, 2014). Safety-critical systems are systems where failures might result in human damage or even death, or damage to the environment (AVIŽIENIS *et al.*, 2004). One of the main causes of critical failures is the excessive amount of specification and its life-cycle, which ranges from requirements inception via system deployment and maintenance to retirement (AVIŽIENIS *et al.*, 2004).

Existing research in safety-critical systems, focusing on the analysis and design of these systems, e.g., Bauer (2016) and Antonino (2016), highlight that there is a gap between safety requirements specifications and their implementation. Actually, researchers are discussing that there are no guarantees that what is being specified will really be understood by developers and that the intent of an original requirement will be maintained throughout the development process (ANTONINO, 2016). This is also true for non-safety-critical systems development (De Lucia; QUSEF, 2010), but due to the criticality of safety-critical systems, it is more challenging for these systems.

Safety-critical systems must be certified by regulatory agencies (e.g., IEEE[1] or TüV[2] to ensure that the functional and quality requirements are met, particularly the safety-critical ones. However, these standards, as well as RTCA/DO-178 (2012) for the aviation industry and IEC 62304 (2006) for medical devices, do not recommend any particular software development methodology (CAWLEY *et al.*, 2010). This makes it possible for many companies to adopt agile practices to a certification process, bringing the benefits of agility to the safety-critical domain (PAGÈS *et al.*, 2012a).

In an agile development project, the communication between analysis, design, and development is performed through user stories, which are *"high-level definitions of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it"* (AMBLER, 2017).

However, such high-level specification is insufficient for properly communicating the challenges to be solved by the system in terms of safety (VUORI, 2011). Since the specification does not consider any safety aspects, these aspects will not be considered in the implementation either. It is thus important that developers are aware of a product's level of safety criticality,

[1] https://standards.ieee.org
[2] https://www.tuv.com

i.e., that it may cause harm to people, and they should base their actions on this awareness (NOTANDER *et al.*, 2013).

The primary goal of this research is to make the specification of agile development artifacts more precise in the context of safety-critical systems development. This will allow the safety product owner to provide evidence to certifiers and regulators that safety measures have been implemented and that there is end-to-end traceability of this set of agile artifacts.

The remainder of this paper is laid out as follows: Section 3.2 provides definitions of the key concepts used in this research abstract, while section 3.3 provides the research goals and research questions. Section 3.4 presents the literature review. Section 3.5 discusses the solution overview. Section 3.6 details the research method. Section 3.7 concludes the paper.

3.2 Definitions

In this section, we define the key concepts of this paper.

A. Safety Engineering:

Safety-Critical Systems (SCS) - Safety-critical systems are those systems whose failure could lead to consequences for the user(s) and/or the environment that are unacceptable (AVIŽIENIS *et al.*, 2004). There are many well-known examples in application areas such as medical devices, aircraft flight control, weapons, and nuclear systems.

Safety Requirements - A requirement that describes means for avoiding, detecting, or handling potential failures in a safety-critical system (ANTONINO, 2016). Safety requirements are suitable for addressing the safety goals obtained from the hazard analysis and risk assessment (BECKERS *et al.*, 2014). They include all assumptions necessary to address the respective safety goal.

Safety Engineer - A safety engineer is a member of the software development team, and a safety specialist; it is his/her responsibility to ensure that an acceptable level of safety is achieved and maintained throughout the life-cycle of the system being developed (LEVESON, 2012).

B. Agile Development:

User Stories - User stories are a common notation used for formulating requirements in agile development projects (LUCASSEN *et al.*, 2015). They follow a strict, compact template that captures in a simple manner who the user story is intended for, what it expects from the system, and why it is important.

Sprint - A sprint (or iteration) is the basic unit of development in SCRUM[3]. It normally lasts about two weeks. According to Stålhane *et al.* (2012), each sprint is a mini waterfall project or a mini V-model, and consists of planning, development, testing, and verification. For the

[3] www.scrum.org

development of safety-critical systems, traceability between the system/code and backlog items, including both functional requirements and safety requirements, is needed. The documentation and maintenance of trace information is introduced as a separate activity in each sprint.

Increment - The working product functionality at the end of each sprint (SCHWABER; SUTHERLAND, 2017). In safety-critical development, the selection of a set of new features and a hazard analysis of the new, planned system version form the essential starting point for an increment. Necessary tasks for safe delivery and deployment can be carried out in parallel during new development increments (VUORI, 2011).

Product Backlog - The full list of what is in the scope for your project, ordered by priority. Once you have your first requirement, you have a product backlog (SCHWABER; SUTHERLAND, 2017). In the development of safety-critical systems, the product backlog consists of all functional and safety-related system requirements. There exists some research, like (STÅLHANE *et al.*, 2012), that proposes using two product backlogs: one functional product backlog, which is typical for agile projects, and one safety product backlog, which is used to handle safety requirements.

3.3 Motivation and Research Questions

In order to address the goal of this research, which is to support more precise specification of agile artifacts in the development of safety-critical systems, and thereby improve the creation of evidence to be submitted for certification in terms of timing and accuracy, the research question (RQ) of this study is:

How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?

The agile team members that usually perform these activities are product owners and tech leads, who are responsible for managing the product backlog, respectively for leading a development team and ensuring the delivered solution meets the technical specifications and design requirements (SCHWABER; SUTHERLAND, 2017).

3.3.1 Motivation

To validate the research goal, we conducted an observational study with the following purpose:

Analyze how safety requirements are currently specified in agile development artifacts, with the purpose of observing how safety aspects are addressed or considered in the decomposition process from the point of view of a researcher (the author herself as a PhD candidate) in the context of agile development of safety-critical automotive systems.

1. Planning and Execution: The observational study was conducted with three different profiles (as depicted in Table 8): (i) Engineer responsible for designing and implementing systems; (ii) Developer; and (iii) Requirements Engineer. The experiment was performed with each participant individually, in two steps: Step 1: A brief presentation was made about the system context (Power Sliding Door (PSD) system), followed by a description of the task to be fulfilled by the participant, which was: "*Assuming the PO's cap, from the backlog containing all system requirements (functional and safety), how would you refine the safety requirement into User Stories?*". Step 2: A brief discussion was conducted with the participant about the outcome of the task, considering the following questions:

1. Did the participant have any difficulty understanding the requirement?
2. Did the participant have any difficulty refining the requirement? (if applicable)
3. Did the participant consider under which aspects to refine the requirement? (architectural, implementation, nonfunctional, etc.)
4. Did the participant consider any safety aspects in the decomposition?
5. Did the participant consider that the defined artifacts maintain the original intent of the safety requirement?

Table 8 – Description of the participants.

Profile	Experience
(i)	>= 15 years in Agile development of solutions for water power grid and geospatial systems
(ii)	>= 2 years in systems development
(iii)	>= 4 years in the analysis of military and avionics systems

2. Analysis of the experiments: All participants had problems understanding the safety requirements. Consequently, if the requirements were not well understood (w.r.t. safety), it is highly likely that the corresponding safety aspects would not be implemented. Furthermore, the derived user stories did not address any safety aspects.

Since the system designer/builder profile has a viewpoint focused on design, the decomposition of the user stories was expected to be about design decisions, but these were defined considering only functional aspects. This means that design problems (with regard to safety aspects) would be detected in the sprints (best case) or would not be addressed at all by the software (worst case).

The developer profile presented two different viewpoints: one viewpoint focused on requirement implementation, that is, the decomposed user stories addressed a solution idea to be implemented; and the other viewpoint focused on requirement validation, so the decomposed user stories addressed how the requirement could be validated. However, both sets of user stories did not address any safety measures for failure detection and containment.

The requirements engineer profile has an analytical viewpoint, so the decomposition of the user stories considered only the logical flow, and alternative flows were not addressed. As safety aspects are a part of the latter, they were not considered. Therefore, errors in the specification phase would be propagated to the next phases, requiring more time, cost, and effort to identify and correct them (if this is even possible).

Moreover, since safety aspects were not present in any of the user stories, it is not possible to ensure that tests would verify the safety properties. Tests play an important role in finding implementation and design errors. To get good quality test cases, the testers need to be knowledgeable about the system, the problem domain, and the development process (NOTANDER *et al.*, 2013). Hence, it is important to provide testers with ways to specify tests for safety properties based on the agile artifacts specification.

3. Conclusion: The observational study was conducted in the context of agile development of safety-critical systems, for three different profiles. Although governed by different perspectives, the results with regard to safety aspects were similar across the practitioners (system designer/builder, developer, requirement engineer). All had difficulties understanding and decomposing the requirements, and consequently, it would not be possible to ensure that the original intent of the requirements was implemented. Therefore, this study served to reinforce our research questions, based on the literature review, and demonstrates the need to enrich agile practices with safety and risk management practices, without sacrificing agility and still providing the necessary assurance level (MARTINS; GORSCHEK, 2016).

3.3.2 Research Questions

To meet the research goal, the main research question described at the beginning of Section 3.3 was refined into the following sub-questions:

RQ1 - How to identify which artifacts or parts thereof are really needed to provide a more precise specification regarding safety aspects in agile development?

- RQ1.1 Which safety aspects should be considered in the artifacts decomposition process?
- RQ1.2 Which set of artifacts is required for precisely specifying and implementing measures for failure detection [2], failure containment, and for taking systems to safe states?
- RQ1.3 Who provided which artifact(s) at which point of the agile process?

RQ2 - How to specify agile development artifacts that actually include the demands of safety requirements?

- RQ2.1 How to minimize the communication gap between stakeholders?

- RQ2.2 What traceability models or algorithms should be used to combine the development artifacts?
- RQ2.3 How to support the specification of agile development artifacts?
- RQ2.4 How to evaluate if the proposed specification is more precise in the context of safety-critical systems?

3.4 Related Work

As part of the related work study, this research involved searching and studying relevant literature in the fields of safety-critical system development and agile practices. The focus was on work where the two fields were found to intersect.

Hatcliff *et al.* (2014) points out the gap between systems engineering, safety engineering, and software engineering, whose processes, paradigms, and tools are not integrated. As a consequence, one problem is the inadequate quality of the requirements specifications for existing systems. Regarding the strong demand for increasing the efficiency of software development processes (in terms of effort, user satisfaction, and time) while still respecting the safety requirements imposed by relevant standards and regulations, Gorski and Lukasiewicz (2012) affirm that many companies have incorporated agile practices into safety-critical projects. However, for engineering safety when using agile methods, most researchers prefer combining agile methods with traditional development processes that rely on safety standards, like SafeScrum (STÅLHANE *et al.*, 2012), XP, and IEC 61508 (THORSEN, 2002), and not much emphasis is given to addressing or considering safety aspects throughout the development process.

Thorsen (2002) and Stålhane and Myklebust (2016b) suggest that safety requirements can be specified as user stories independent of functional requirements, and suggest placing these user stories into a separate document, whereas Paige *et al.* (2008) distinguish between user stories and safety stories, but only to be able to document the output from safety engineering. Also, the safety stories are specified from the user stories to address safety requirements. However, the software engineer has to be integrated into the safety process so that the problem domain is really understood. As mentioned above, such interaction is not always possible, meaning the understanding of the problem is often compromised. As a consequence, the system specification is inconsistent and incomplete.

However, according to Vuori (2011), the viewpoint of user stories is subjective and not sufficient for specifying safety requirements since these are objective and contain standard-defined design and implementation requirements. Furthermore, it is hard to fully express the complexity of the functionalities of these systems only with user stories and test cases (ANTONINO *et al.*, 2014).

Martins and Gorschek (2016) conducted a systematic literature review on requirements engineering for safety-critical systems. One of their findings was the lack of studies on what would enable the communication of safety requirements among the project team actors. This issue impacts directly on the certification process, which is mandatory in the SCS industry, and means that there is no guarantee that what is being specified is really understood by the developers and that the intent of an original requirement is maintained throughout the development.

3.5 Solution Overview

The solution for specifying agile artifacts more accurately in the context of safety-critical development to be developed during the Ph.D. activities is composed of two key contributions:

- An agile safety process, whose purpose is to identify which artifacts or parts thereof are really important and necessary for specifying failure detection and containment as well as measures for taking the system to a safe state, considering traceability requirements, standard-related constraints, and domain experts. In the context of safety-critical software, the evolution of artifacts is more challenging compared to general software development because whenever an artifact evolves, the safety analysis has to be performed again (TAROMIRAD; PAIGE, 2012). To this end, it is important to provide precise safety-related information to an agile development team about those parts of the software that are really critical and need to be understood well.

- A semi-automated methodology for supporting agile artifacts specification considering safety aspects. From the set of artifacts defined in the first step and based on the additional information provided by the interaction between stakeholders, the methodology should combine all this information (with combination algorithms and decision support) to support decision-making. To be more precise, the decision is actually the specification in this case. In order to provide agile artifacts for failure detection and containment, as well as for safe state achievement, the principle of user stories will be used, which are lightweight but precise specifications that can be implemented accurately and timely. Each of these agile artifacts is a subset of artifacts needed to specify measures either for detecting or containing failures, or for providing a safe state, and therefore for ensuring that the original intent of the safety requirement has been understood. One key benefit of this contribution will be, for instance, that the product owner will be able to provide evidence to certifiers and regulators that safety measures have been implemented and that there is end-to-end traceability of this set of agile artifacts.

For instance, consider the approach proposed by Stålhane *et al.* (2012), which focuses on adapting Scrum to safety-critical software development (as depicted in Figure 11) and which has already been applied in a number of real-life projects (STALHANE *et al.*, 2013).

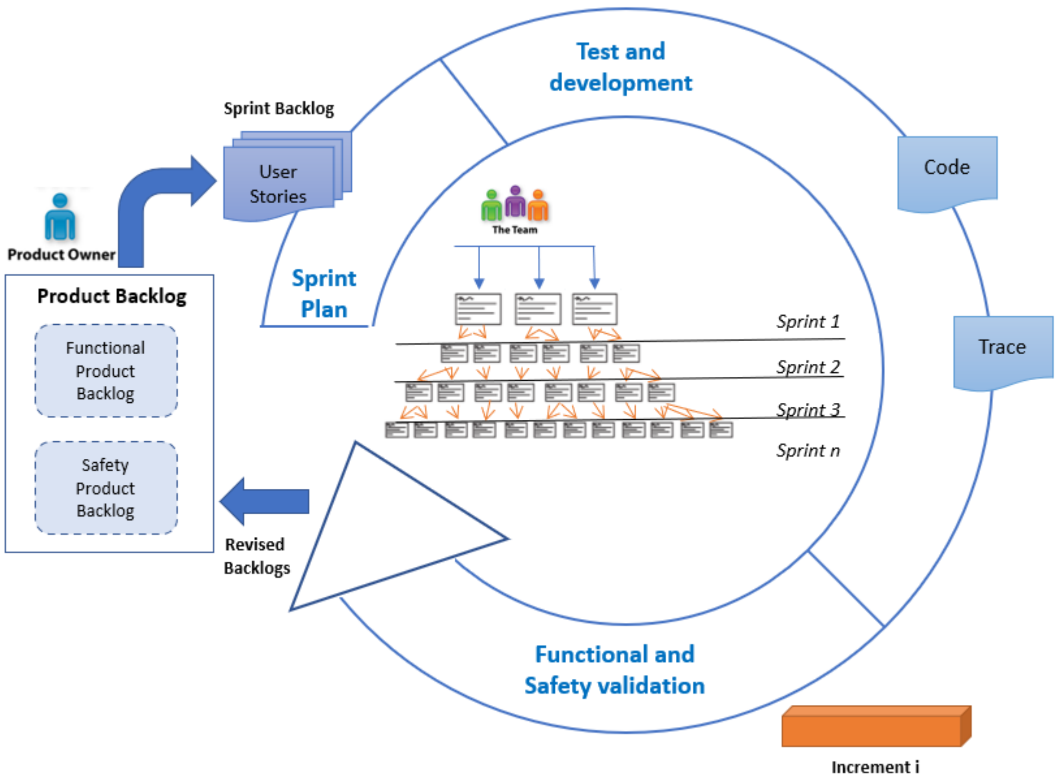


Figure 11 – The safe scrum model (adapted from Stålhane *et al.* (2012))

The method proposes that the safety requirements should also be refined into user stories. As mentioned above, due to the complexity of these user stories, they need to be decomposed into new user stories, which means that there is no guarantee that the requirement's original intent will be maintained throughout this decomposition process. Furthermore, due to gaps (communication, culture, different backgrounds) among system developers, software developers, and safety experts, the understanding of the problem is often compromised. As a consequence, an agile development team cannot ensure that the developers have actually considered the safety aspects.

The proposed approach (as depicted in Figure 12) aims to support the safety product owner in the decomposition of safety requirements into agile development artifacts to ensure that the specification of these artifacts considers the safety aspects. A top-level safety requirement must be broken down into functional and technical safety requirements and combined with other development artifacts. With the help of the agile safety process, it will be identified which artifacts (or which subset of them) will be considered, and who will provide each artifact, and at which point of the development. This will allow for more precise specifications of user stories for failure detection, failure containment, and safe state achievement. These stories will compose the sprint backlog. The product owner and the tech leader should determine at each sprint planning which of these stories should be implemented. When combining each sub-set of artifacts that make up a story - whether a detection or a safe state - a trace is created indicating that the safety

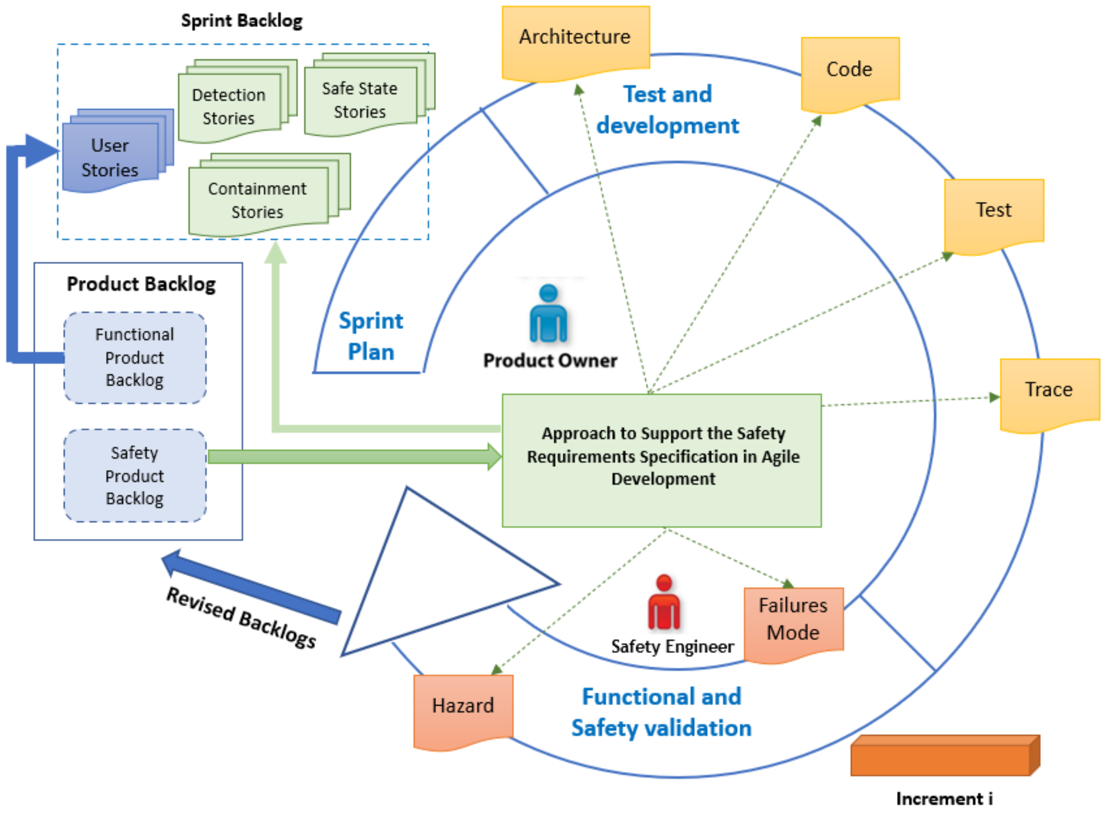


Figure 12 – Solution overview

aspects have been considered.

3.6 Research Design

Three main research activities are proposed. Each contributes to the outcomes of the other activities.

A. Systematic Literature Review (SLR) This review will follow the SLR process proposed by Kitchenham and Charters (2007) in order to collect evidence to answer research questions RQ1.1 and 1.2, and RQ2.2.

Defining the Agile Safety Process The second research activity involves identifying what is enough in the agile safety domain, based on traceability requirements and standard-related constraints. This activity aims to answer research question RQ1 (including RQ1.3) based on the evidence collected in the previous activity. Validation of the outcome will be carried out through a survey within the scope of the projects developed by Nucleus for Strategic Health Technologies NUTES[4], which is an initiative of the Brazilian Health Ministry for promoting the technological development of medical devices.

Specification Support for Agile Artifacts The third research activity involves develop-

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

ing a novel methodology for supporting the specification of agile artifacts considering safety aspects. This activity aims to address research question RQ2. An experiment or case study will be conducted with our existing industrial and government collaborators in order to apply and evaluate our approach in more realistic industrial environments.

3.7 Conclusion

Safety requirements play an important role in the life cycle of safety-critical systems. However, there is always a wide gap between their specification and implementation. Furthermore, in the context of agile development, more attention is given to functional requirements than to quality requirements. Due to the special nature of safety-critical systems, it is, however, necessary to ensure that agile artifacts address or consider safety requirements as well. Therefore, this research proposes an approach for supporting more precise specification of agile artifacts so that the developers get a better understanding of the original requirement and that this can be traced to the coding. The testers can then also understand the original requirement better, which enables them to specify test cases for safety properties. The safety engineers can obtain more accurate evidence that the safety requirements have been addressed. And the product owner can provide evidence to certifiers and regulators that safety measures have been implemented and that there is end-to-end traceability of this set of agile artifacts.

Regarding the next step in the context of this work, a systematic literature review will be conducted to identify which artifacts or parts thereof are really needed to provide a more precise specification regarding safety aspects in agile development.

CHAPTER

4

From Safety Requirements to Safety-Centered Architectural Solutions in Agile Context

[Published as: LEITE, A. I., Antonino, P. O., and Nakagawa, E. Y. *From Safety Requirements to Just-Enough Safety-Centered Architectural Solutions in Agile Contexts*. In XXXIV Brazilian Symposium on Software Engineering (SBES '20). 2020. p.766–771.]

Abstract

Context: Safety-critical systems can be found in many sectors of our lives, e.g., in medical equipments and vehicles. A agile practices have been increasingly incorporated into the development processes of these systems, mainly due to demands related to time-to-market and budget reduction. **Problem:** At the same time, recent accidents have shown that various failures have been caused by errors or faults introduced during development and resulted from misunderstandings of safety requirements by agile development teams. Moreover, there is still a lack of techniques for ensuring that safety requirements are properly addressed by both software architecture and implementation. **Objective:** To address this gap, this paper presents the SCA3DA metamodel, which leverages the understanding of safety requirements by agile teams, so that these requirements can be included in the just-enough safety-centered software architecture of safety-critical systems. To demonstrate the applicability of this metamodel, we used it in clinical safety interlock scenarios for infusion pumps. **Results:** Preliminary results indicate that adopting the SCA3DA metamodel is also feasible in industry projects.

4.1 Introduction

Safety-critical systems refer to systems whose failure could result in accidents that cause damage to the environment, financial losses, injury to people, and even the loss of lives (AVIŽIENIS *et al.*, 2004). These systems have increased in number and complexity; besides, software is becoming responsible for most critical functions in these systems (ABDULKHALEQ; WAGNER, 2015). Many key application domains (e.g., avionics, automotive, energy, manufacturing, and military) are highly safety-critical. One important domain in which safety-critical systems are present is healthcare. For example, when diabetes patients use a digital insulin pump to control and fine-tune the level of blood glucose, this is a safety-critical system (HEINEMANN *et al.*, 2015). The need for diabetes treatment with insulin is rising dramatically, and many more patients will be treated with digitized insulin injections and will use digitized blood glucose measurements in the future (FIRESMITH, 2004).

The development of safety-critical systems is typically rigorous and requires software certification (RUSCIO *et al.*, 2014) by regulatory agencies (such the FDA[1] and TuV[2]).However, these certifications, such as DO-178B (RTCA/DO-178, 2012) for the aviation industry and IEC 62304 (IEC 62304, 2006) for medical devices, do not recommend any particular software development methodology (MCHUGH *et al.*, 2013). Recent accidents, such as the Boeing 737 crash (2019) (BBCNEWS, 2019), a woman struck and killed by one of Uber's autonomous cars (2018) (ISAAC *et al.*, 2018), a train crash in Singapore (2017) (PARK, 2017), the crash of an Airbus A400M (2015) (HEPHER; MORRIS, 2015), and accidents caused by malfunction of Nissan's airbags sensor detectors (2014) (JENSEN, 2014), have shown the vulnerabilities of software-intensive safety-critical systems development. According to data from the U.S. Food and Drug Administration (FDA), software failures are responsible for 24% of all medical device recalls (WIZEMANN, 2010). Currently, researchers are discussing that there are no guarantees that what is being specified will be understood by developers and that the intent of the original/correct requirements will be maintained throughout the development process (MARTINS; GORSCHEK, 2016). This is also true for non-safety-critical systems development (De Lucia; QUSEF, 2010), but due to the criticality of safety-critical systems, it is more challenging here.

To manage the challenges imposed by large-scale safety-critical systems development, different application domains have already experimented with agile methods to reduce project risk and increase flexibility and even quality. As mentioned in the Medical Device Development Report (2018) (PERFORCE, 2019), important growth can be observed in the use of agile methods/practices in medical systems development. Likewise, the Agile in Automotive Report (2018) (MULLER; BESEMER, 2018) pointed out that more than 18% of companies apply agile methods/practices in more than 75% of their projects.

Recent research about safety-critical systems Antonino (2016), Tommila and Alanen (2015), Wu and Kelly (2006), Beckers *et al.* (2017) has focused on the analysis and design of these systems. This research highlights a gap between safety requirement's specifications and their implementation (HATCLIFF *et al.*, 2014). In turn, safety requirements play an important

[1] https://www.fda.gov/
[2] https://www.tuv.com

role in the life cycle of safety-critical systems, as they describe means for avoiding, detecting, or handling potential failures in these systems (ISO26262, 2018). Furthermore, each standard or regulation, such as ISO/IEC 26262 (ISO26262, 2018) for the automotive domain, requires safety requirements related to software and hardware to be derived from system safety requirements. All these requirements should be traceable, but an *ad-hoc* way is usually adopted in development based on experience and background of engineers/architects.

In the context of agile development, the communication among the analysis, design, and development teams is performed through user stories, which refer to high-level specifications of requirements, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement them (BECK; WEST, 2014). However, such specifications are insufficient for properly communicating the challenges to be solved by the system in terms of safety (VUORI, 2011). Moreover, when face-to-face communication is not possible among agile development teams across multiple functional areas and physical locations, each team may have an inconsistent interpretation of the system (LO; CHEN, 2017). Hence, the main communication challenge is to go beyond word-of-mouth requirements, architectures, and systems. Besides that, according to IEC 61508 (2010), from a safety viewpoint, software architecture is where the basic safety strategy is developed in a system.

In summary, the main problem addressed in this work is that safety-critical systems have been increasingly built using agile development processes, but such these processes have not been enough to assure that safety requirements are addressed in these systems. It is also observed that there is a lack of guidance on how to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well.

This work aims to contribute to improving the understanding of how safety requirements should be addressed in safety-critical systems development by agile teams. For this, we propose the SCA3DA metamodel, which is a standard-independent metamodel that serves as a guide for deriving just-enough safety-centered architectures, i.e., architectures that contain a minimal set of relevant information for specifying strategies for both failures detection and containment (ISO26262, 2018). This work was developed and evaluated using a design science methodology (WIERINGA, 2014) in cooperation with research partners (Fraunhofer IESE[3] and NUTES[4]) on projects that directly benefit industry. We have observed that metamodel can assure explicit consideration of safety requirements and reflect them in the software architecture of safety-critical systems. Our findings also indicate that the SCA3DA metamodel is suitable for adoption by agile teams, including in large-scale industrial projects.

The remainder of this work is structured as follows: Section II discusses related work, while Section III presents the SCA3DA metamodel. Section IV introduce a running example and Section V discusses the preliminary results. Finally, Section VI concludes with a summary of

[3] https://www.iese.fraunhofer.de/
[4] http://nutes.uepb.edu.br/

our findings and an outlook on future work.

4.2 Related Work

We conducted a systematic mapping (according to (KITCHENHAM; CHARTERS, 2007) and (PETERSEN *et al.*, 2015)) to identify the state of the art of how have agile methods are being applied in the development of safety-critical systems, and, more specifically, how safety requirements are being addressed in agile methods/practices.

Regarding the strong demand for increasing the efficiency of software development processes (in terms of effort, user satisfaction, and time), while still respecting the safety requirements imposed by relevant standards and regulations, Górski and Łukasiewicz (GÓRSKI; LUKASIEWICZ, 2013) affirm that many companies have incorporated agile practices into safety-critical projects. However, for engineering safety when using agile methods, most researchers prefer combining agile methods with traditional development processes that rely on safety standards, like SafeScrum (HANSSEN *et al.*, 2018), XP and IEC 61508 (THORSEN, 2002), and not much emphasis is given to addressing software and hardware safety requirements throughout the development process.

Thorsen (THORSEN, 2002) and Stalhane (STÅLHANE; MYKLEBUST, 2016b) suggest that safety requirements can be specified as user stories independent of functional requirements, and suggest placing these user stories into a separate document, whereas Paige et al. (PAIGE *et al.*, 2008) distinguish between user stories and safety stories, but only to be able to document the output from safety engineering. Besides, the safety stories are specified from the user stories to address safety requirements. However, software engineers should be integrated into the safety process so that the problem domain is really understood. However, such interaction is not always possible, meaning the understanding of the problem is often compromised (HATCLIFF *et al.*, 2014). As a consequence, the system specification is inconsistent and incomplete. Cleland-Huang (CLELAND-HUANG; VIERHAUSER, 2018) propose specifying safety stories using the Easy Requirements Specification (EARS) format, and stating they should be traceable to the design definition, which is a solution for addressing safety stories at the system level. Likewise, Wang et al. (WANG *et al.*, 2017) propose a new pattern for specifying safety stories, but safety background knowledge is needed in the development team to help write them.

According to Vouri (VUORI, 2011), the viewpoint of user stories is subjective and not sufficient for specifying safety requirements since these are objective and contain standard-defined design and implementation requirements. Furthermore, it is hard to fully express the complexity of the functionalities of these systems only with user stories and test cases (ANTONINO *et al.*, 2014). In turn, Martins and Gorschek (MARTINS; GORSCHEK, 2016) conducted a systematic literature review on requirements engineering for safety-critical systems. One of their findings was the lack of studies on what would enable the communication of safety requirements among the project team actors. This issue impacts directly the certification process, which is mandatory in the safety-critical systems industry, and means that there is no guarantee that what is being specified is understood by the developers and that the intent of an original requirement is maintained throughout the development.

4.3 SCA3DA Metamodel

The SCA3DA metamodel aims to improve the understanding of safety requirements by agile teams and support architectural design decisions. The use of this metamodel allows building the necessary common knowledge in agile teams to adequately consider safety issues and, create more precise evidence for certification bodies that safety issues were, in fact, implemented, without burdening agile development. We followed a design science methodology (WIERINGA, 2014) in order to establish the metamodel properly.

The SCA3DA metamodel is a safety-standard-independent metamodel, whose aim is to support agile teams in identifying a minimal set (aligned to the agile principles) of architecturally relevant information for specifying safety-critical failure detection and containment strategies. Agile teams can then make design decisions that satisfy safety stories. For instance, according to the safety-integrity level of safety stories, there are safety measures that must be considered, but they are often left aside. Hence, in safety-critical projects, it is necessary to be more deliberate about how safety stories should be addressed in architectural solutions (CLELAND-HUANG; VIERHAUSER, 2018). Based on that, software architecture views are essential to ensure that safety is properly addressed. Our metamodel provides guidance for the specification of just enough safety-centered architectural solutions so that agile teams can identify what is important at each level of architecture abstraction (context, function, software, and hardware) to address safety stories. As a consequence, the development team's understanding of those critical parts that deserve attention is improved.

In the remainder of this section, we will introduce the concepts present in the metamodel (as depicted in Figure 13) considering the abstraction levels from top to bottom. At the **context level**, *safety concerns* (hazards, risks, failures) are identified from the hazard and risk assessment and, for each hazardous event, a *safety epic* (safety goal or top-level safety requirements) should be determined, as well as the *safety integrity level (SIL)* associated with it (ISO26262, 2018). At the **functional level**, each safety epic should be refined by *safety stories*, which record strategies for realizing safety measures. These strategies should take into account the SIL, which specifies the necessary requirements and safety measures to avoid unreasonable residual risks (IEC 61508, 2010). These safety measures are related to failure detection, containment, or avoidance. Hence, each safety story can be refined into a *failure detection story* and a *failure containment story*, as recommended by safety standards (such as (IEC 61508, 2010), (ISO26262, 2018) and (IEC 62304, 2006). Moreover, we specified traces between safety story and *system function* and the

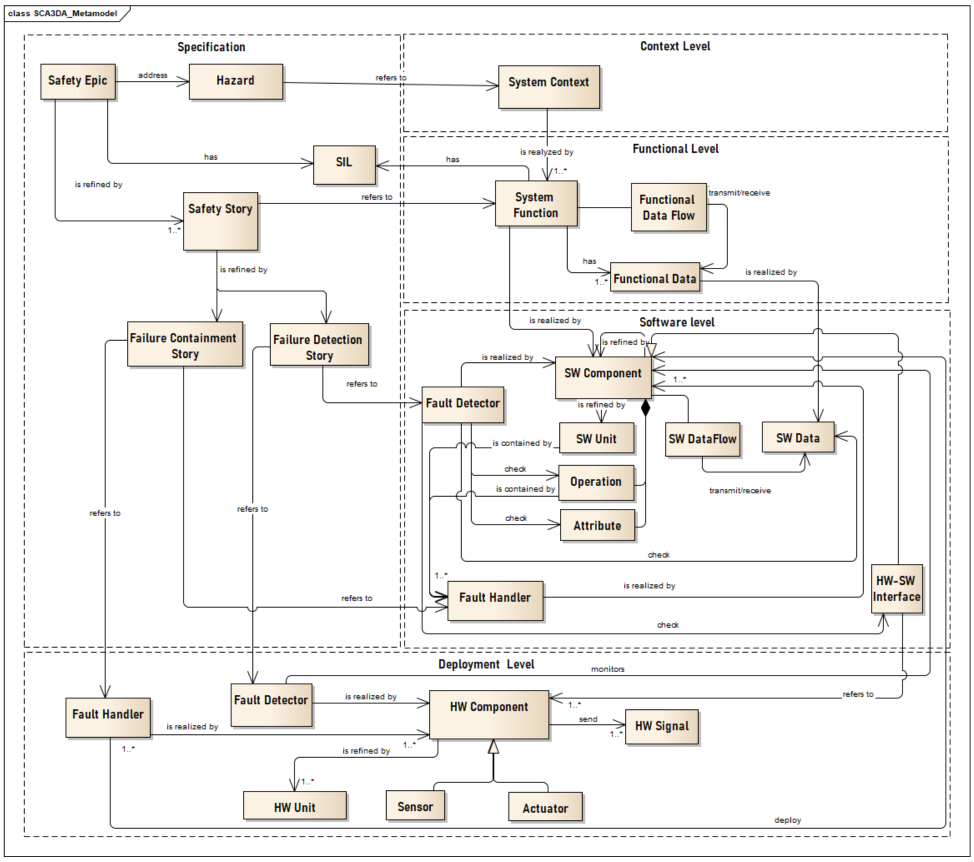


Figure 13 – SCA3DA metamodel.

dependencies that address it. A system function represents a service of the system that is provided either to an external entity or to another function of the system. Once that function is realized by *software component* (**at software level**), it is necessary to identify this component, its *provided and required interfaces*, as well as *software data flow*. Once failure detection stories and failure containment stories are realized by the software, at this software level, it is essential to make design decisions according to the SIL, and communicate explicitly which failure detection or containment strategy should be implemented. According to each strategy, it is necessary to identify the software elements necessary to realize it. Regarding failure detection, the SIL should induce different fault detection strategies. For instance, the *fault detector component* can be realized by another software component, in this case, the software component might either be deployed in the same hardware or not, or it may be implemented in a software unit and may be achieved by checks in the value domain (*software data*) or in the time domain. Likewise, for failure containment, the *handler component* prevents fault propagation across defined boundaries. Besides, it is necessary to define the level at which the containment will occur, whether at the component or the software unit level. Handler components according to the SIL should be deployed in different hardware or modules. Failure detection stories and failure containment stories can alse be realized by hardware (**at the hardware level**); in this case, it is essential to define, according to the SIL, the detection or containment strategy to be implemented and to identify the hardware components needed to realize that strategy.

4.4 Running Example

We adapted the Integrated Clinical Environment (ICE) from (GOLDMAN, 2009; LEITE *et al.*, 2017) to exemplify the use of the SCA3DA metamodel. The purpose of the ICE is to allow bolus doses for patients using patient-controlled analgesia in a controlled loop, called "*enable infusion*". Patients may require extra doses of opioids (e.g., heparin, dopamine, and fentanyl) administered through a portable device provided by the infusion pump. Patients press the button and the infusion pump gives them an extra dose (referred to as a bolus) of painkiller. To avoid overdosage, the ICE must assess the patient's vital signs and allow bolus doses to be administered to patients who may need them. In particular, the standard ASTM 2761 (ASTM F2761, 2013) presents the implementation of clinical safety interlock scenarios for infusion pumps, as well as the control loop for preventing overdosage. The main objective of the scenario "*stop the infusion with vital signs*" is to stop the infusion pump based on parameters, such as respiratory rate, heart rate, and blood saturation. If any parameter moves outside the pre-defined ranges, the ICE must send a stop message to the infusion pump; in addition, it must issue an alarm to the caregiver.

Figure 14 depicts the context diagram of the ICE, which shows the main input variables to be received are the patient's respiration rate and SPO2 (oxygen saturation). Assuming that, along with the safety analysis of the system's preliminary architecture and requirements, the safety engineers identified that, if the ICE does not verify whether the patient's vital signs are within an acceptable range, the safety of the patient might be compromised, because the patient might receive a higher infusion than required. Based on that, the hazard, the safety epic, and the safety story should be addressed by the system implementation (as depicted in Figure 15). More explicitly, both failure detection and containment strategies must be specified, as presented in Figure 16.

The next step is to identify which ICE functions and their dependencies are associated with that safety story. At this point, the functional view might only present information related to the critical function associated with the safety story *"The ICE system should not allow infusion when SPO2 data is lower than 90%"*, so that the agile team can focus on meeting current needs, following the agile mantra of "no big design upfront". In this context, the *Vital Signs Computation* function receives the patient's vital signs and verifies whether the data is below the pre-set threshold (cf. Figure 16).

Next, it should be noted whether this function is implemented via software or hardware. In our example, this function is realized by the software component *Oxygen Calculation* (cf. Figure 17). In the software view, failure detection and containment stories specify safety strategies that should be implemented by the agile team. In, both cases, the SIL must be observed to define how the strategy will be addressed[5]. According to IEC 61508, SIL 3 (in the context of our example) recommends using both homogeneous and heterogeneous redundancy. As

[5] (due to the lack of space, we detailed only the failure containment story)

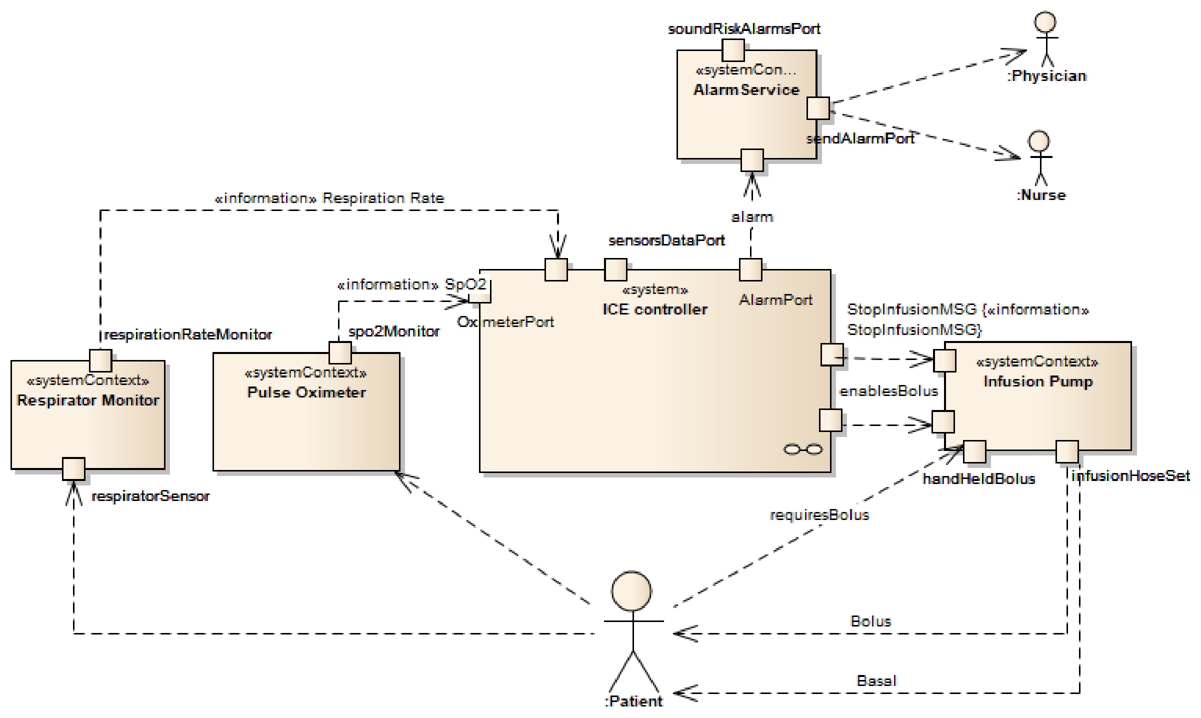


Figure 14 – Context diagram of the ICE system.

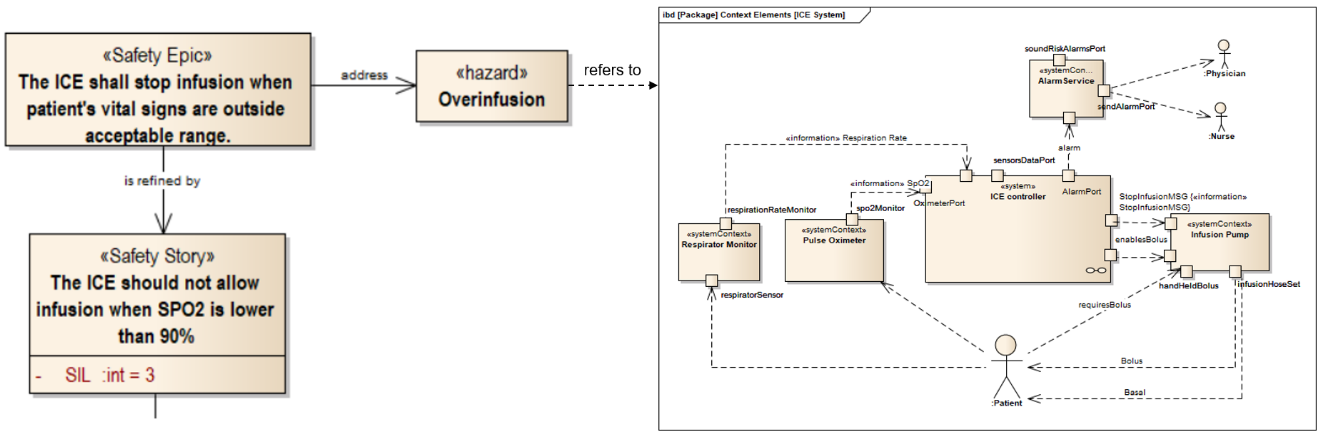

«Safety Epic»
The ICE shall stop infusion when patient's vital signs are outside acceptable range.
address
«hazard»
Overinfusion
refers to
is refined by
«Safety Story»
The ICE should not allow infusion when SPO2 is lower than 90%
- SIL :int = 3
ibd [Package] Context Elements [ICE System]

Figure 15 – Safety story refinement.

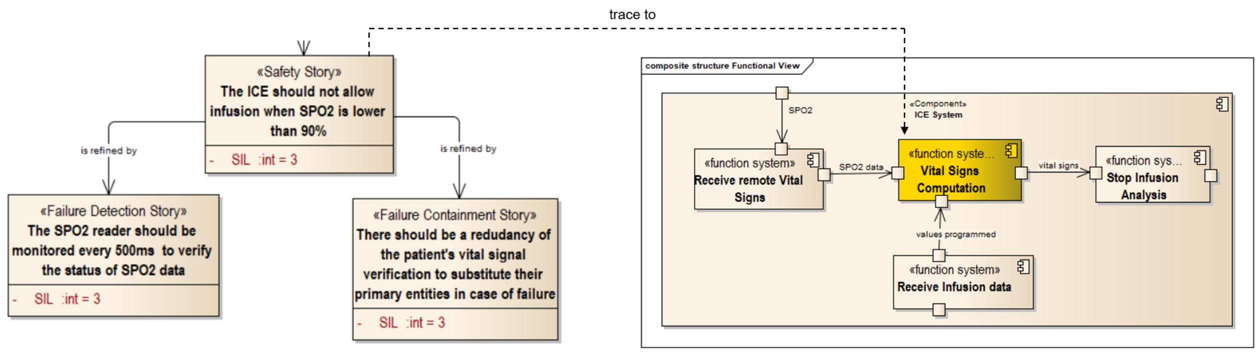

trace to
«Safety Story»
The ICE should not allow infusion when SPO2 is lower than 90%
- SIL :int = 3
is refined by
is refined by
«Failure Detection Story»
The SPO2 reader should be monitored every 500ms to verify the status of SPO2 data
- SIL :int = 3
«Failure Containment Story»
There should be a redudancy of the patient's vital signal verification to substitute their primary entities in case of failure
- SIL :int = 3
composite structure Functional View
SPO2
«Component»
ICE System
«function system»
Receive remote Vital Signs
SPO2 data
«function syste...
Vital Signs Computation
vital signs
«function sys...
Stop Infusion Analysis
values programmed
«function system»
Receive Infusion data

Figure 16 – Preliminary functional model of ICE system.

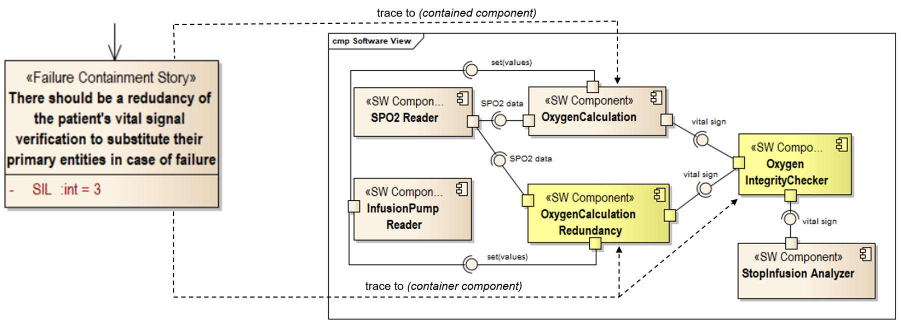


Figure 17 – Container component specified at component level.

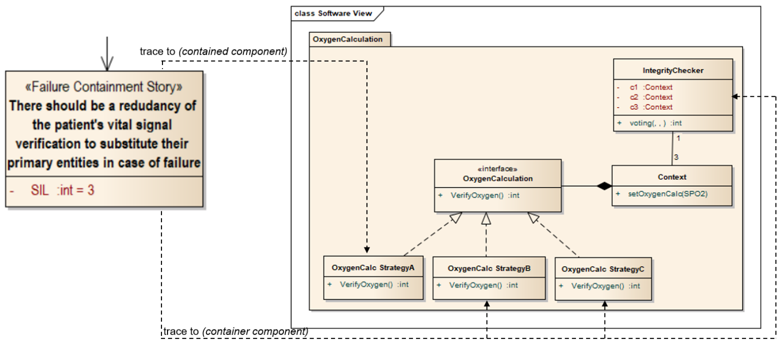


Figure 18 – Container component specified at software unit level.

heterogeneous redundancy is used here, the handler component can be specified at the component level (cf. Figure 17) or at the unit level (cf. Figure 18). Depending on the level, different implementations are defined and, using a voting strategy, the solution that presents the correct (expected) execution is identified. Therefore, from each safety story at the system level, the metamodel provides guidance for tracing and deriving failure detection and containment stories (at the software or hardware level) and then supports the derivation of design decisions addressing these safety stories. As a result, the just-enough safety-centered architecture solution depicted in Figures 15, 16 and 17 is specified, which gives the development team a better understanding of the real intention of the safety requirements, and how they can be implemented.

4.5 Discussion

Considering the increasing complexity and use of software-intensive safety-critical systems, a better understanding of safety requirements is essential to adequately build, and evolve such systems. Existing research has focused on specifying safety requirements in the sense of "what should be done". However, both the state of the art and the state of the practice emphasize that addressing only the "what" is insufficient, especially when these systems are developed using agile development. Besides, most developers have no background or even experience in safety engineering, compromising the understanding of the actual intention of safety requirements and, as a consequence, resulting in inaccurate and inconsistent safety specifications. Therefore, the "how and why it should be done" is also essential. The SCA3DA metamodel provide a guidance on "how should be done" so that resulting just-enough safety-centered architecture solutions can encompass all safety concerns. We believe using this metamodel agile teams could leverage how they deal with software and hardware safety requirements and therefore, realize more accurate architecture solutions. Agile teams can also use these solutions to improve team communication and ensure a unique understanding of system criticality and a more accurate interpretation of safety requirements.

It is worth highlighting that resulting architectural solutions do not intend to address all details of how software components must be built up-front, but they do address, according to the context, the minimal set of architecturally relevant information for specifying safety-critical failures detection and containment strategies. These solutions can be further refined during agile sprints. We also recognize that the SCA3DA metamodel has a limitation. It is not possible to ensure that decisions made during requirements specification and architecture design are the most correct ones; therefore, we recommend adopting the SCA3DA metamodel together with validation/verification approaches to increase the confidence of the decisions.

4.6 Final Remarks and Future work

The dependence of many sectors of your lives on safety-critical systems brings important challenges for the development of these systems, especially if safety must be assured when this development applies agile practices. This work positions itself in this scenario by introducing the SCA3DA metamodel, which provides a means for dealing with safety requirements and for reflecting these requirements in the architectures of these systems.

For future work, the community of safety-critical systems development should invest in initiatives like the SCA3DA metamodel, proposing other approaches or even experimenting with this metamodel. In particular, we intend to investigate how this metamodel could be incorporated into agile development, identifying who the involved stakeholders are, and which artifacts should be considered and/or created. Moreover, we intend to apply the SCA3DA metamodel in real-world scenarios and collect results to refine it. Hence, we could mature it so this work could ultimately bring the safety team and the agile team closer together and enable them to cooperate in developing systems that are trustworthy in terms of safety.

CHAPTER

5

An Investigation of Knowledged Gaps of Graduate Students Regarding Safety-Critical Systems Development: A Controlled Experiment

[Published as: LEITE, A. I. M., et al. An investigation of knowledge gaps of graduate students regarding safety-critical systems development: a controlled experiment. In: *IEEE Transactions on Education* 65.1 (2021): 64-72.]

Abstract

Contribution: This paper details the conduction of an experiment to investigate the knowledge that computer science graduate students have about safety-critical systems development, in particular, safety requirements specification. Future research directions are also discussed. **Background:** Safety-critical systems have been increasingly used in many critical domains, requiring a rigorous development process as well as certification by regulatory agencies. However, it is not clear whether computer science courses deliver adequate education for students to develop such systems. Experimental research with these students could provide evidence of whether they have the minimum knowledge necessary to develop safety-critical systems. This type of research presents several advantages, such as reproducibility, high-level control, and specific conclusions, but has not been widely adopted to verify learning issues. **Research Question:** How to measure the understanding of students on safety-critical systems development? **Methodology:** A controlled experiment was conducted with computer science graduate students in which they prepared safety requirements specifications. Three variables, i.e., understandability of the tasks, accuracy of the specifications, and effort required, were analyzed. **Findings:** This experiment made it possible to explicitly measure the effect of understandability on accuracy and

effort. Our experimental scenario revealed that most graduate students do not know about safety engineering. Hence, it is paramount to rethink computer science curricula to include basic safety engineering concepts and how software-engineering-specific knowledge should be considered in the development of these systems. Besides, software architecture must receive special attention, as it is the backbone of all software-based systems, including safety-critical systems.

5.1 Introduction

Safety-critical systems have been increasingly used in highly critical application domains, such as aerospace, medical devices, transportation, and energy, to mention but a few. At the same time, software has become more and more responsible for the most critical functions in these systems (MAURYA; KUMAR, 2020). Safety is defined as freedom from unacceptable levels of risk of harm to people (CENELEC - EN 50128, 2011). According to Leveson (2012), safety is basically a system characteristic that defines that, to an acceptable extent, the system does no harm, e.g. to human beings, does not damage property, and does not cause environmental pollution. For example, diabetes patients have used digital insulin pumps with an embedded safety-critical system to control and fine-tune the level of blood glucose (HEINEMANN *et al.*, 2015). As the need for diabetes treatment is rising dramatically, many more patients will require the use of these pumps, meaning that the demand for so-called safety-critical medical systems, including such pumps, will increase soon.

The engineering journey of a safety-critical system is very rigorous, requiring certification by regulatory agencies, such as FDA[1] and TuV[2], to ensure that it is safe for use (HOBBS, 2019). Actually, each standard or regulation, such as IEC 62304 (2006) for medical devices and ISO26262 (2018) for the automotive sector, describes strict recommendations to be included, for instance, in the safety requirements specification. These requirements describe means to avoid, detect, or handle potential failures in safety-critical systems (ISO26262, 2018). They include all assumptions necessary to address the respective safety goals obtained from a hazard analysis and risk assessment (BECKERS *et al.*, 2014).

In 2009, software professionals complained that undergraduate courses did not teach students the skills they need to get their jobs done efficiently; for instance, software engineering topics were found to be usually taught rather superficially (WANGENHEIM; SILVA, 2009). Later, in 2014, many senior decision-makers in companies stated that they did not have the required background to lead the development and/or certification of safety-critical systems (HATCLIFF *et al.*, 2014). The Curricular Guidelines for Graduation (IEEE/ACM) (ACM/AIS/IEEE-CS, 2005) have sought to mitigate software engineering teaching gaps in recent years (ALARIFI *et al.*, 2016; GAROUSI *et al.*, 2019), but there is currently still doubt whether the topics taught in computer science undergraduate courses introduce the students to the development of safety-

[1] https://www.fda.gov
[2] https://www.tuv.com

critical systems (HATCLIFF *et al.*, 2014). There is also a lack of empirical evidence about the knowledge of these students concerning such development. At the same time, experimental research has not been widely used to investigate learning issues. Experimental research refers to empirical inquiries where one or more factors of the studied setting are manipulated and different subjects apply different treatments, while other variables are kept constant, to measure the effects on outcome variables (WOHLIN *et al.*, 2012). The main advantage of these experiments is that they allow eliminating uncertainty about the significance of the results.

In this scenario, we are interested in answering the following research question: "How to measure the understanding of students on safety-critical systems development?". Hence, the main goal of this work is first to investigate the knowledge of computer science students about safety-critical systems development, in particular regarding the specification of safety requirements, which are the key elements for the success of such development. The second goal is to present how a controlled experiment can be applied to measure learning issues. For this, an experiment was planned and conducted with graduate students who solved safety requirements specification tasks of two projects in different domains: automotive and medical devices. Two groups (with and without training) were measured in terms of the subjects' understanding of the tasks, projects, and safety issues, as well as the accuracy of the solutions proposed by the subjects and the effort required. As the main result, it was found that different domains did not affect effort. Even a short training could significantly affect the subjects' understanding; however, both groups still had difficulties solving the tasks, which significantly affected the level of accuracy. This proved that experiments can be considered an important means to assess students' knowledge.

The remainder of this paper is structured as follows. Section 5.2 presents the background, especially the main concepts associated with safety-critical system development. Section 5.3 summarizes the experimental design and execution, while Section 5.4 reports the results. Section 5.5 discusses the main findings and discusses areas for future investigations. Section 5.6 concludes this work.

5.2 Background

Safety is a system property defined as '*absence of catastrophic consequences on the user and the environment*' (AVIŽIENIS *et al.*, 2004). According to IEC 61508 (2010), which is a basic functional safety standard applicable to all kinds of industries, a safety system (safety-related system) is a designated system that implements the required safety functions necessary to achieve or maintain a safe state. Besides, it is intended to achieve, on its own or in conjunction with other safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions.

Safety-critical systems are an aggregate of hardware, software, and firmware (software embedded in hardware devices) and play an important role in many functions of critical systems. For instance, if a braking system, a stability control system, an automatic train stop system, or an insulin pump fails, the failure can result in unacceptable consequences (MAURYA; KUMAR, 2020), (Nascimento *et al.*, 2020). In IEC 61508 (2010) and IEC 61508 (2010) safety standards, these failures can be random and systematic failure, common cause failure, safe failure, or dependent failure. Hence, safety-critical systems should encompass safety strategies to avoid and/or control those failures.

According to Leveson (2012), IEC 61508 (2010), RTCA/DO-178 (2012), safety-critical development (as shown in Figure 19), starts with hazard analysis and risk assessment. This step is performed during the very early phases of the development process, at the latest when the system requirements are available. The safety goals are defined as top-level safety requirements that have to be incrementally refined during the safety life cycle. Once the safety goals have been defined, the system development continues through different phases, like the definition of system functions, the system and software architecture, the design, and finally the implementation of the system. Besides that, the validation and verification of the system should be conducted in parallel to the development, and the next steps in the safety development process should also be performed in parallel. To this end, the available development artifacts are used as input to safety analyses in order to identify potential causes of system failures. From that, the safety engineer and architect owner derive a specification of the safety requirements, their allocation to architectural elements, and the interaction necessary to achieve the safety goals and information associated with these requirements (HATCLIFF *et al.*, 2014). Besides, as the developers incrementally refine the system over the different development phases, the safety engineer analyzes the refined development artifacts step by step and refines the safety requirements accordingly.

Safety requirements play an important role in the life cycle of safety-critical systems, as they describe means for avoiding, detecting, or handling potential failures in these systems. Furthermore, according to ISO26262 (2018) '*the software safety requirements shall address each software-based function whose failure could lead to a violation of a technical safety requirement allocated to software*'. These software-based functions are related to failure detection and containment and enable the system to achieve a safe state. Besides, the software and hardware safety requirements should provide sufficient information to enable software architecture design and implementation activities. Also, *' software developers must be able to demonstrate traceability of designs against requirements'* (RTCA/DO-178, 2012).

5.3 Experimental Design end Execution

The well-known guidelines found in Wohlin *et al.* (2012), Jedlitschka *et al.* (2008) were used to design was used. In Appendix C, the complete experimental design and execution are available. For the design, a balanced factorial design with two factors of interest with two treatments

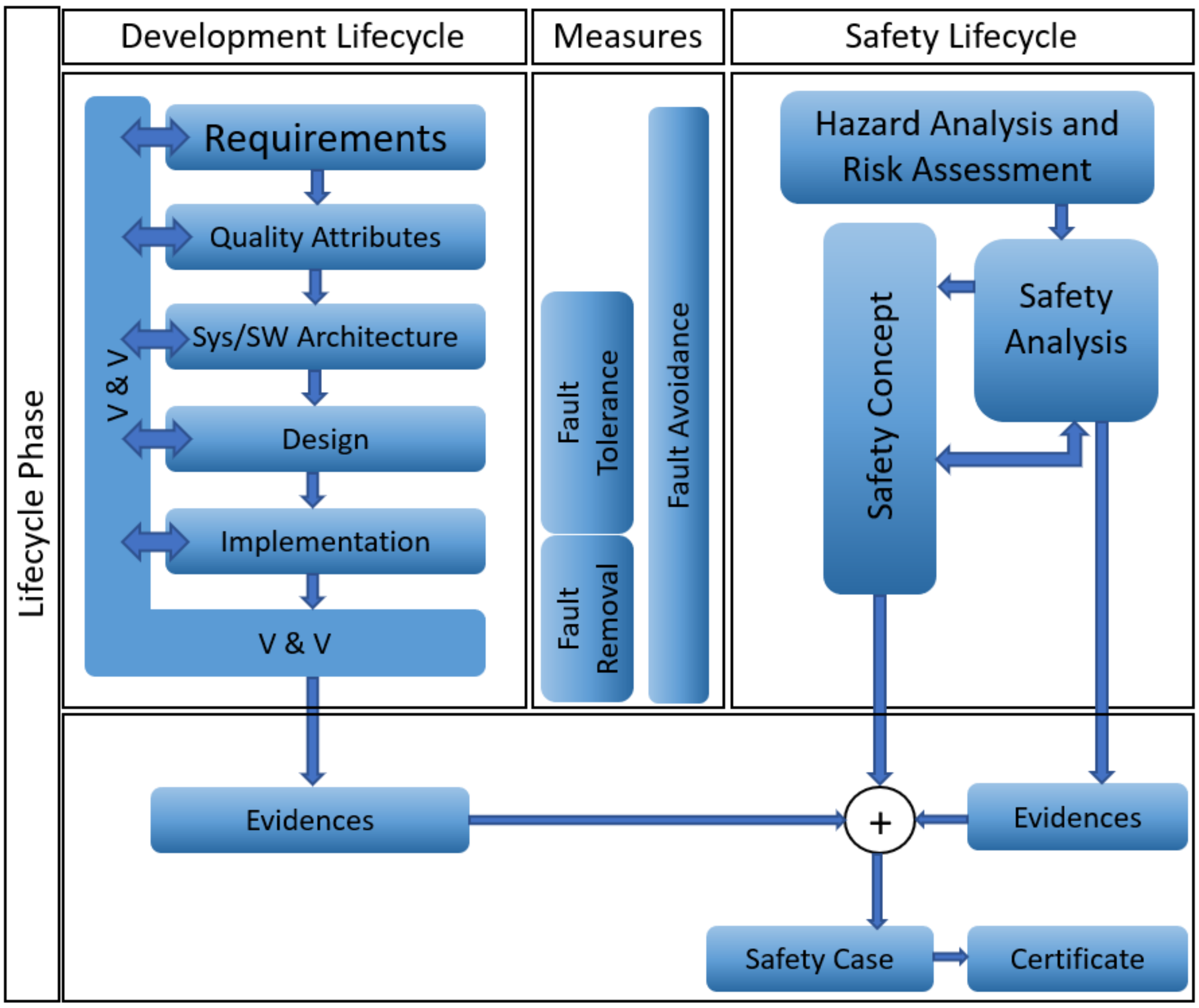


Figure 19 – Safety engineering life cycle (adapted from Trapp and Schneider (2011)).

(with/without training and two different application domains). Regarding the domains, two projects from different domains were used, a power assist control system for a sliding door system (referred to hereafter as Automotive) and an integrated clinical environment controller (referred to as Medical). The following research questions were addressed in this study:

- **RQ1:** To what extent can the subject understand the safety requirements in order to derive specifications from them?
- **RQ2:** How accurate are the specifications of the safety requirements provided by the subjects?
- **RQ3:** How much effort do subjects need to expend to specify safety requirements?

Three dependent variables were then considered: understanding, accuracy, and effort. For each of these, the following hypotheses were defined:

Table 9 – Self-assessment about background and experience by subjects

	Descriptive Results				
Self-Assessment on 5-point Likert scale	**Mode**	**Median**	**MAD**	**Min**	**Max**
Background in software development	2	3	1	2	5
Experience in software development	4	2	1	1	4
Background in safety software development	1	1	0	1	3
Experience in safety software development	1	1	0	1	2

- **H1:** Computer graduate students are able to understand safety requirements and then provide specifications that address these requirements independent of the training and application domain. (H1.0 is the null hypothesis and H1.1 is the alternative hypothesis).

- **H2:**: Computer graduate students are able to prepare accurate specifications from safety requirements. (H2.0 is the null hypothesis and H2.1 is the alternative hypothesis).

- **H3:** The effort expended by computer graduate students to understand safety requirements and then provide specifications that address these requirements is independent of the safety application domain and of whether the participants receive training or not. (H3.0 is the null hypothesis and H3.1 is the alternative hypothesis).

This experiment involved a total of 16 participants/subjects (11 Ph.D. students, four Master's students, and one post-doctoral researcher), all of them from the graduate program in Computer Sciences at the University of Sao Paulo (USP), one of the highest-quality programs in Brazil. Concerning the subjects' profile, which was collected through a pre-experiment questionnaire (presented in Appendix C), most of them had know-how and experience in software engineering: 3 of 16 subjects had experience as requirements engineer, 7 as a developer, 5 as a tester, and 2 as an architect. Two knew about safety engineering and only one specifically about safety-critical systems development. Moreover, the subjects rated their background and experience (in software development and safety software development) on an ordinal scale (1 = very poor to 5 = very good), whose mode, median, median absolute deviations (MAD), minimum, and maximum scales are shown in Table 9. Overall, the subjects had a medium level of background and experience in software development and a low level of background and experience in safety engineering. The deviation in the subjects' answers regarding their self-assessment for all responses was 1 or lower; hence, we had a homogeneous group. The subjects were randomly assigned to four groups (A, B, C, and D), with and without training and dealing with two different systems, as presented in Table 10.

Table 10 – Design of the four groups of subjects.

Domain	**Training**	
	Without	**With**
Medical	Group A	Group C
Automotive	Group B	Group D

The experiment included two tasks: one task for the Medical project (given to Groups A and C) and one task for the Automotive project (Group B and D). Both tasks addressed the same activity of architectural specification; i.e., the description of software safety requirements. Subjects were asked to describe the necessary changes in the architectural design to encompass those requirements. To do that, subjects had to understand the task to be solved, explore the project documentation (e.g., functionalities and architectural description to identify parts to be changed), and report their solution.

The experiment was conducted over two days. On the first day, the reason for the experiment was explained to all subjects, and the documentation (i.e., consent form, analyst survey, and project documentation) was delivered. Then the participants were randomly shuffled into four groups, as shown in Table 10. On this first day, without getting any training, Groups A and B worked on the assigned tasks and, in the end, filled in the task report and the post-experiment questionnaire (as shown in C). On the second day, the subjects received training in safety-critical systems development, in particular, regarding safety life cycle, safety requirements, and safety standards, i.e., IEC 61508, ISO 26262, IEC 62304, DO-178C, and ISO 14971 (Risk Management). The training was done in a 2-hour session using a system — Airbag System — unlike those the participants used later during the experiment. This session consisted of 30 minutes for the theoretical presentation and 90 minutes for practicing. At the end of the training, the participants received feedback on their performance and an example of a correct specification of the software safety requirements of that system. This feedback allowed the participants to understand which safety-related information should be explicitly considered in the specification of software safety requirements. Then groups C and D performed the tasks and filled in the questionnaire. Other data was also collected during the execution of the experiment, i.e., the time required to perform the tasks and the tasks' level of difficulty, and a data analysis was conducted with appropriate descriptive statistics. To answer RQ1, the strategy presented in Carew *et al.* (2005) was applied, which requires participants to read artifacts and provide answers. The participants were then asked to answer a questionnaire to provide their feedback regarding their level of understanding of the task, the project description, and the safety requirements. The response time and accuracy of the answers reflect the understanding of the artifacts. Furthermore, the measures of the central tendency of the ordinal data were calculated for each question of the post-experiment questionnaire: mode, median, and MAD for their dispersion. Also, the reliability of the answers to the post-experiment questionnaire was measured using Cronbach's α reliability measure, which is a common measure for internal consistency. A value of 0.7 or above to Cronbach's α is sufficient for internal consistency (RITTER, 2010). To answer RQ2, the solutions provided by the subjects (i.e., the safety requirements specifications) were evaluated and, based on a baseline (correct solution provided by experts in safety-critical systems development and certification), were analyzed by comparing them to the essential elements (i.e., failure detection, containment measures, and how to take the system to a safe state) present in the baseline. The level of accuracy represents how close the solutions are to the baseline. In cases of doubt, experts who supported our experiment determined the accuracy of the solutions.

Finally, to answer RQ3, the subjects' effort was measured by considering the time required to perform the tasks.

5.4 Results

Section 5.4.1 presents the data collected and processed concerning the three dependent variables, while Section 5.4.2 presents the hypothesis testing. Section 5.4.3 discusses the answers to our RQs and Section 5.4.4 the threats to the validity of this experiment and how these were mitigated.

5.4.1 Descriptive Statistics

A. Understandability

Table 11 shows the reliability statistics for the dependent variable understandability calculated using the Cronbach's α reliability measure, which demonstrated high reliability and internal consistency, as it ranged from 0.71 to 0.79, i.e., it exceeded 0.70.

Table 11 – Cronbach's α calculated for the dependent variable understandability from data collected in the post-experiment questionnaire.

Dependent Variable	**Understandability**		
	Task	Project	Training
Cronbach's α measure	0.79	0.71	0.77

Going into more detail, seven questions (Q3–Q9) were asked to collect the subjects' opinions about the understandability of the description of the task to be performed, the safety-critical project documentation, the safety requirement to be addressed, as well as the specification to be developed. Table 12 summarizes the results. In particular, Q3 asked whether they found the task description easy to understand. The results show that they agreed that the task description was easy to understand. Q4 was on whether the subjects had difficulty understanding the project documentation. There was a neutral agreement for the automotive project, but the subjects disagreed on the medical project. The deviation here was low, with a MAD between 0 and 1. Q5 was on whether the subjects had difficulty understanding the safety requirements. We observed that this understanding received neutral agreement for the automotive project, whereas the subjects disagreed on the medical project. It could also be observed that there was no difference in perception between those who had received training and those who had not. Q6 was on whether the subjects had difficulties specifying the safety requirements. The results show that they agreed that it was difficult to provide the specifications, with a deviation in all responses of 1 or lower. Q7–Q9 specifically aimed at collecting the opinions of the subjects who received training (Groups C and D); therefore, they were not answered by the other groups. Q7 was on whether training contributed to improving their understanding of what safety requirements are. The subjects were mostly neutral, with a small deviation (MAD: 1). Q8 was on whether during training it was made clear how to accomplish the task. The results show that they agreed that the training helped them to accomplish the task. The deviation was again low, between 0 and 1. Finally, Q9 was on whether training helped them to specify the safety requirements. The results revealed that they were neutral for the automotive project, while they agreed for the medical project. The deviation of all responses was 1 or lower.

B. Accuracy and Effort

The accuracy of the solutions provided by each subject as well as the effort required were calculated. Both were analyzed for normality; Table 13 shows the descriptive statistics (mean, median, and standard deviation) for them. The results show that accuracy and effort (in minutes) had a higher mean for the groups that received training. Additionally, Figure 20 compares the time that the subjects spent on the tasks with and without training as box plots for both projects (left) and separately for the automotive project (middle) and the medical project (right). Similarly, Figure 21 compares the overall accuracy's level achieved and the accuracy's per project as bar plots. Each bar indicates the percentages of the solutions according to their accuracy level (0% to 100%).

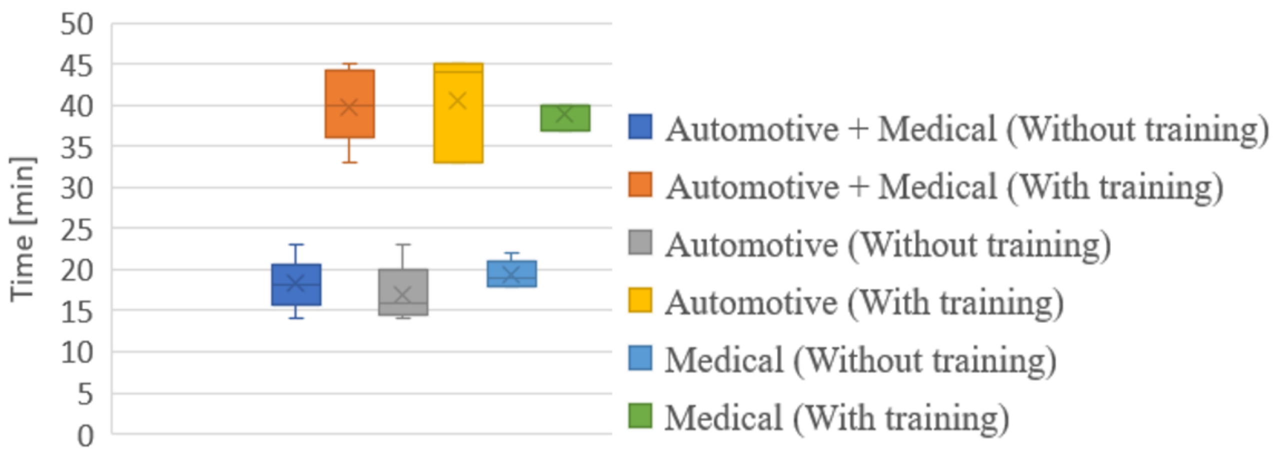


Figure 20 – Effort measured through time spent by subjects to perform tasks.

Table 12 – Descriptive statistics for the dependent variable understandability calculated from data collected in the post-experiment questionnaire.

		Understandability					
Project		**Both**		**Automotive**		**Medical**	
Training		No	Yes	No	Yes	No	Yes
Q3	Median	4	3	3	2	4	4
	Mode	4	2	3	2	4	4
	MAD	1	1	0	0	1	0
	Min	2	2	3	2	2	4
	Max	5	5	5	3	5	5
Q4	Median	2	3	3	3	1	2
	Mode	1	2	4	3	1	2
	MAD	1	1	1	1	0	0
	Min	1	2	1	1	1	1
	Max	4	5	4	4	3	2
Q5	Median	2	2	3	3	1	2
	Mode	1	2	4	3	1	2
	MAD	1	0	1	0	0	0
	Min	1	1	1	2	1	1
	Max	4	3	4	3	2	2
Q6	Median	4	4	4	4	4	5
	Mode	5	5	5	4	4	5
	MAD	0	0	0	1	0	1
	Min	3	3	3	3	3	3
	Max	5	5	5	5	5	5
Q7	Median	*	3	*	2	*	4
	Mode	*	2	*	2	*	4
	MAD	*	1	*	1	*	1
	Min	*	1	*	2	*	1
	Max	*	5	*	4	*	5
Q8	Median	*	4	*	3	*	4
	Mode	*	4	*	3	*	4
	MAD	*	1	*	1	*	0
	Min	*	2	*	2	*	4
	Max	*	5	*	5	*	4
Q9	Median	*	3	*	3	*	4
	Mode	*	3	*	3	*	4
	MAD	*	0	*	0	*	1
	Min	*	3	*	3	*	3
	Max	*	5	*	4	*	5

[3] * Not applicable.

Table 13 – Descriptive statistics calculated for the dependent variables accuracy and effort.

Project (P)	**Training (T)**	**Accuracy [%]**			**Time [min]**		
		Mode	Median	Standard Deviation	Mean	Median	Standard Deviation
Automotive+Medical	Without	0	0	0.24	18.2	18	2.89
	With	50	25	0.24	39.83	30	3.36
Automotive	Without	0	0	0.32	17	16	3.53
	With	-	25	1	40.66	40.66	4.7
Medical	Without	0	0	0.14	19.4	19	1.67
	With	50	50	0.28	39	39	1.22

Table 14 – Summary of analysis of variance (n = 16).

Source of variation	**Df**	**Understandability**			**Accuracy**			**Time**		
		F	p-value	Fc	F	p-value	Fc	F	p-value	Fc
Independent Variables										
Project (P)	1	2.27	0.15	4.49	0.06	0.80	4.49	0.069	0.79	4.49
Training (T)	1	26.27	0.0001	4.49	1.56	0.22	4.49	240.2	4.67	4.49
Interaction										
P x T	1	7.36	0.015	4.49	0.56	0.49	4.49	2.12	0.16	4.49

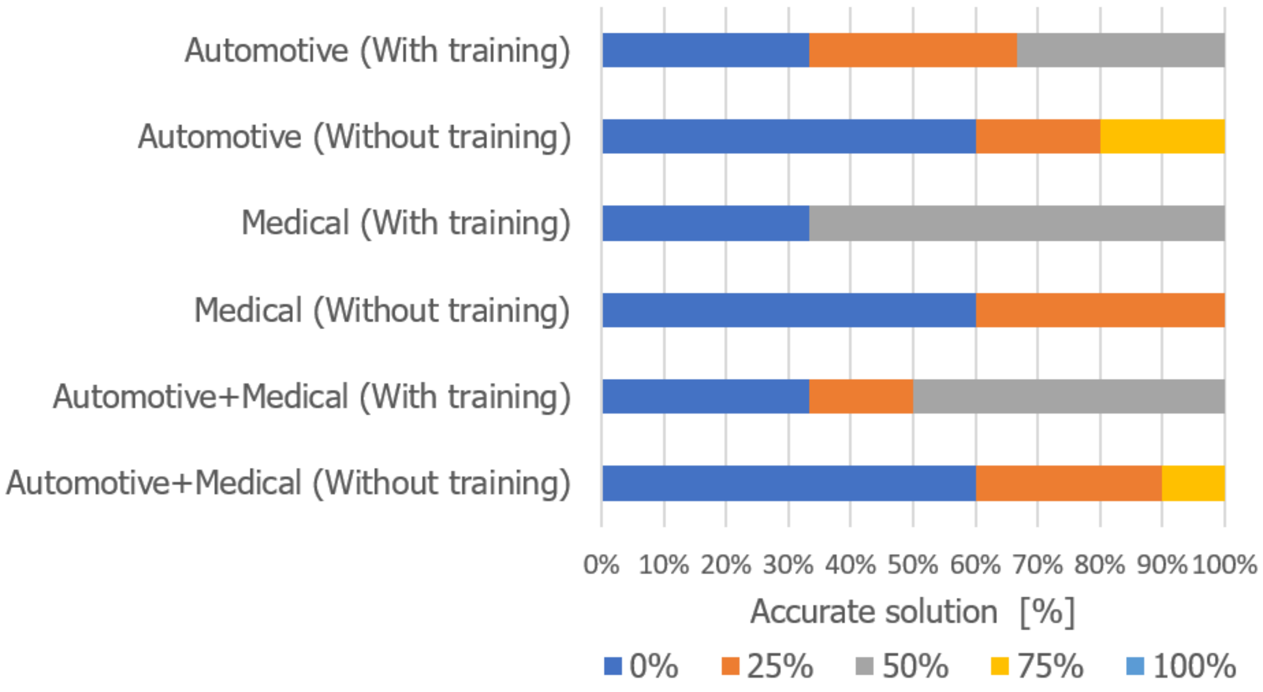


Figure 21 – Accuracy of the solutions provided by subjects.

5.4.2 Hypothesis Testing

To test the three hypotheses defined above, the two-way analysis of variance (ANOVA) with post-hoc analysis at a confidence level of 0.05 (SCHEFFE, 1999) was used. ANOVA was applied to determine the effect of independent variables on the dependent variables understandability, accuracy, and effort, whose results[4] are presented in Table 20. Due to the use of more than one independent variable, it was necessary to consider whether there was an effect of the interaction of these variables on the dependent variables (WOHLIN *et al.*, 2012). Thus, ANOVA was also computed to analyze the influence of interaction between project domain and training on understandability, accuracy, and effort, which was necessary to answer the three hypotheses. For hypothesis H1, the test revealed that the project domain (Medical or Automotive) did not significantly affect understandability. Regarding the training, the test revealed that it significantly affected the understandability of safety requirements. So the null hypothesis H1.0 was accepted with p-values $(p < 0.05)$. For H2, the statistical tests showed that the project domain (Medical or Automotive) did not affect accuracy (see row Project (P) of Table 20: $F < Fc$, where F represents *F distribution* and Fc limits the rejection region (WOHLIN *et al.*, 2012)). This is also true for the row Training (T) of Table 20: $F < Fc$, which did not significantly affect the accuracy of the solution. Hence, the null hypothesis H2.0 with p-values $(p > 0.05)$ was accepted. Regarding H3, the test revealed that the project domain (Medical or Automotive) did not significantly affect effort. In contrast, training, which aimed at providing guidelines for solving the task, affected effort, but not significantly $(p > 0.05)$; hence, the alternative hypothesis $(H3.1)$ was accepted.

To enhance confidence in the results, we also performed the Kruskal-Wallis non-parametric test (WOHLIN *et al.*, 2012), which makes no assumptions about the type of underlying distribution. We also used the R software to perform this test and calculated the H' value and the corresponding p-value, as shown in Table 15. According to the p-value, there is no statistically significant difference in the accuracy level between the two projects (Automotive and Medical), nor for groups that received training and those that did not. This information was already expected since we had a small sample. However, it corroborated the results presented by ANOVA and improved our confidence in the results.

Table 15 – Kruskal-Wallis data analysis

Kruskal-Wallis Test for Equal Medians	
H (chi^2)	1.521
H_c (tie corrected)	1.777
p(same)	0.62

[4] All statistics were calculated using R environment (http://www.r-project.org).

5.4.3 Answering Research Questions

Based on the experiment results, the RQs were answered. To answer RQ1, the variable in the analysis was understandability, to investigate the understanding of safety requirements by the subjects. This question is very important because when requirements are not really understood by software engineers, there are no guarantees that they will be addressed throughout the software development process. For this experiment, training was fundamental for improving this understanding, since 64% of the subjects who received training claimed to have a better understanding of the safety requirements. On the other hand, only 16% of the subjects who did not receive training claimed to have understood the safety requirements. Furthermore, even though the subjects claiming to have understood a safety requirement's intention, most of them (68%) said they had difficulties in providing a specification that addresses that requirement, which was also impacted by their lack of deep background and experience in developing software, including safety-critical systems.

For RQ2, the accuracy of the solutions designed by the subjects analyzing two different scenarios was measured, the first with or without training and the second for the automotive or the medical project. As shown in Table 13, accuracy had a mean and median of 25% only in the scenario with training (meaning 16% of the solutions designed by the subjects who received training had an accuracy level of 25%), while the best accuracy level was 50% in the medical application with training. 60% of the solutions designed by the subjects who did not receive training did not reach 25% accuracy. In summary, accuracy did not get high values even for groups with training. It is important to clarify that the training was on safety requirements specification (for instance, it was aimed to identify precise safety-related information about those parts of the software that are really critical and need to be understood well). However, the main difficulties reported by the subjects were on how to correctly design such specifications, which should be done as an architectural design. 56% of the solutions had a textual format and only 20% were architectural designs. As previously highlighted in Alarifi *et al.* (2016), Garousi *et al.* (2019), it was also observed that the subjects did not have knowledge on some software engineering topics, such as software architecture and specification of quality attributes. In fact, the specification of safety requirements must be fundamentally done at the architecture level, making it mandatory to use architecture description languages (ADL) like UML and SysML to correctly specify these requirements. Therefore, without this basic knowledge of architecture design, it becomes even more difficult to ensure that safety requirements have been understood and implemented in safety-critical systems.

To answer RQ3, the effort was measured, i.e., the time that the subjects spent to perform the task. As Table 13 shows, subjects who received training took more time to perform the task, with the highest mean of 40.66 minutes in the automotive project; however, the lowest mean of 16 minutes was also observed in the automotive project. Effort is directly related to the task's understandability and difficulty as perceived by the subjects. The group without training indicated that the task was easy, but also had low accuracy of their solutions, at 25%; hence, due to a lack of understanding of what should really be done, the effort they undertook was also lower. Obviously, effort did not indicate an increase in accuracy, but the group with training had a better understanding of the task.

5.4.4 Threats to Validity

This section discusses the most important threats to the validity of the experiment according to Wohlin *et al.* (2012). A threat to *internal validity* is related to the language used; hence, all materials were translated into Portuguese, since all subjects were Brazilian, to avoid misunderstanding. Another threat was that lessons learned from training might influence the results; so, the subjects were randomly associated to four groups, two with training and two without, which performed their tasks on two different systems (Automotive and Medical), and no communication among them was allowed during the experiment execution.

The decision of what to teach and how much time participants need to learn about the task is a key experimental design consideration. This suggests that there is an inherent bias in controlled experiments toward tasks that can be quickly learned, and against those that require significant practice. There is no one right approach to training. However, to mitigate this threat, we tested whether the participants had learned everything necessary in the training, for instance, we asked them to explain the materials back to the experimenters (the three first authors), and we had a question and answer session that ended when the experimenters were confident that they understood the material.

Besides that, an introductory tutorial on safety requirements management was provided to all participants to decrease the difference in knowledge among the participants. After this tutorial, the subjects were asked whether they had questions before starting the experiment and it was found that, except for a few organizational questions, they felt ready. This suggests this tutorial was sufficient.

The *external validity* refers to the approximate truth of conclusions involving generalizations within different contexts. When designing the experiments, we took into account possible threats to validity coming from the involvement of students as the participants (CARVER *et al.*, 2003), (FALESSI *et al.*, 2017). Our first concern was to select groups of participants widely representative of software professionals. The participants had knowledge of UML and dynamic modeling as the majority of graduate students from Brazil's universities. In fact, these students will be soon integrated into the software industry, so they can be considered as widely representative of young professionals (FALESSI *et al.*, 2017). A threat that might affect the external validity concerns the size and complexity of the tasks used. We decided to use relatively small tasks since a controlled experiment requires that participants complete the assigned tasks in a limited amount of time. To confirm or contradict the achieved results, we plan to conduct case studies with larger and more complex tasks.

Possible threats to statistical *conclusion validity* include violations of the assumptions underlying statistical procedures, low statistical power, and low effect size. The group design with randomized treatment assignments and equal group size helped to eliminate these impacts. Before making inferences from the statistical test, the ANOVA assumptions were verified, and a Cronbach's α reliability measure was used to check the consistency of answer variance. Regarding *construct validity*, the experiment aimed at investigating the knowledge of computer science students about safety requirements specification. In this regard, the knowledge was assessed by having the subjects perform specification tasks, and the performance was operationalized by the time spent (effort) and the accuracy of the solutions. If a subject understands the real intention of safety requirements, then they should perform better on a specification task, and/or the solution should be more correct. If the subject has a better understanding of the requirements and the task to be done, the time spent on the task should be lower. The experiment therefore focused on these measures.

5.5 Discussion

As an overall conclusion, graduate students, including those from high-score graduate programs, do not appear to be prepared for developing safety-critical systems and are even lacking an understanding of safety requirements and architectural design. Although the training provided to participants had significantly increased participants understanding of software safety requirements, it was not sufficient to enable them to provide more accurate software design solutions. As a cause of this, the participants pointed out the gap in knowledge on topics regarding software architecture, including architecturally significant requirements, model-driven software development, and design decisions that address quality attributes. These topics are especially important for safety-critical systems development.

This experiment corroborated the impression that a step forward has to be taken toward improving the teaching of safety engineering basics (such as the concepts of failure, fault, error, mistake, failure mode, and effect analysis) and software architecture, which is a software engineering topic considered fundamental for the development of safety-critical systems. Similarly, studies reported in Alarifi *et al.* (2016), Garousi *et al.* (2019) analyzed the knowledge gap concerning the teaching of software engineering topics both in undergraduate and vocational training and indicated a lack of dedication to certain software engineering topics considered important, such as software architecture, which in practice is considered an essential basis, not only for software engineers but for any software professional. In particular, safety-critical systems development often focuses on constructing a distinct safety architecture that detects and mitigates hazards. According to the standard IEC 61508 for safety-critical systems (IEC 61508, 2010), software architecture is the vehicle for incorporating and assuring critical safety strategies and safety requirements in these systems. The other claim is that much experience and knowledge can be acquired when students are introduced to industry projects, but institutions/universities should provide the initial knowledge about the development of such current systems in line with the industry's claims. To achieve this, computer science undergraduate curricula should be brought up to date. Furthermore, we present an alternative to assess the understandability in the controlled experiment. Understandability constitutes an aspect of ease of use, namely the effort required to read and correctly interpret some information. The interpretation comprises identifying a particular connection of constructs and assigning meaning to it. Unfortunately, cognitive processes like interpretation and mental effort cannot be observed. To face such unobservability, we defined three dependent variables: effort, accuracy, and understandability. For the latter as a qualitative variable, we present a methodology to carry out its quantitative analysis, which can be relevant both in future research and development of course materials.

With respect to the use of experiments to verify learning issues, the main observations made were: (i) Experiments allow applying systematic means to observe the complicated effects/relationship that can exist among different factors/variables, in this case, understandability, accuracy, and effort; (ii) experiments with a small number of people could be used to discover whether a given training, and associated training materials, for a large number of people would be effective; (iii) training materials could be improved based on the results of experiments; and (iv) experiments can usually be replicated precisely in other scenarios to verify whether the results are the same. For experiments similar to the one presented here, other means for collecting data could be used. Tools such as Google Forms[5], Learning Space Tool Kit[6], and Gimlet[7] could be adopted to exactly capture the time spent by the subjects as well as their behavior in each part/task performed. Besides that, the accuracy of the solutions could be checked automatically using supporting tools. In the case of architectural design, tools such as Enterprise Architect[8] and MagicDraw[9] provide functionalities that make it possible to compare two or more similar solutions. With regard to the subjects' effort, the time spent could be associated with the number of correct artifacts produced. Teams that spend more time often produce better solutions, perhaps because good solutions demand more time to be done. To measure the complex factor of understandability, complementary means could be used in addition to self-assessment; e.g., the interactions among the subjects and/or with the instructors during the experiment could also be considered and additional tests could be done on the topics to be addressed.

For this work, a controlled experiment was conducted. In scenarios where it is not possible to assure random assignment of subjects and resources or the ability to exercise full control over the study conditions, quasi-experiments can be conducted (WOHLIN *et al.*, 2012). From an empirical software engineering point of view, other empirical strategies also exist such as case studies, which can offer an in-depth understanding of how and why certain phenomena occur and which can reveal the mechanisms by which cause-effect relationships occur (WOHLIN

[5] https://tinyurl.com/yabp62ll
[6] https://learningspacetoolkit.org
[7] https://www.gimlet.us
[8] https://www.sparxsystems.eu
[9] https://www.magicdraw.com

et al., 2012); however, one weakness of case studies is that data collection and analysis are more open to interpretation and that, unlike in the case of quasi-experiments and controlled experiments, research bias is an issue.

Finally, it is important to highlight that there was concern in detailing the experiment design and making it available as supplementary material, whose aim is to support the experiment reproducibility by other institutions/universities from different cultures and regions around the world, as well as can be used as an example to be reproduced in other contexts involving learning issues.

5.6 Conclusions

In the era of increasing dependence of all of society on complex, large safety-critical systems (e.g., modern cardiac pacemakers, insulin pumps, flight control systems, car braking systems), adequate preparation of graduates to enable them to deal with these systems should be a priority concern, as also stated in Hatcliff *et al.* (2014), Alarifi *et al.* (2016), Garousi *et al.* (2019). To accomplish this, the curricular guidelines for computer science degree programs should be updated continually considering the dynamism of the computing area in terms of new technologies, new challenging critical domains, and new concepts and approaches for the development and evolution of such systems. In particular, as corroborated in this experiment, the teaching of software architecture is paramount for graduates to enable them to adequately deal with safety-critical systems.

This work could serve as a motivation to other researchers and educators to adopt experiments as a systematic, rigorous means for measuring the complicated equation that is the discovery of the real knowledge of students and, ultimately, contribute to better preparing the next generation of computing professionals.

CHAPTER

6

SCA3DA: A Method for Specifying Software Safety Requirements in Agile Context

[Submitted as: LEITE, A. I., Antonino, P. O., and Nakagawa, E. Y. *SCA3DA: A Method for Specifying Software Safety Requirements in Agile Contexts*. In Journal of Systems and Software (JSS), 2023.]

Abstract

Context: Software-intensive safety-critical systems are present in several sectors of society like medicine and vehicles, while agile practices have been increasingly incorporated into the development processes of these systems mainly due to demands related to time-to-market and budget reduction. At the same time, recent accidents have shown that several failures have been caused by errors or faults introduced during development and resulted from the misunderstandings of safety requirements by agile development teams. Moreover, there is still a lack of techniques to ensure that safety requirements are properly addressed by both software architecture and implementation. **Objective:** To address this gap, this paper presents the SCA3DA method, which leverages the understanding of safety requirements by the agile teams, so that these requirements can be included in the just-enough safety-centered software architecture of software-intensive safety-critical systems. **Method:** The contributions of our work have been developed and evaluated through a design science methodology, whose purpose was to present the SCA3DA method to support software safety requirements specification and safety-centered architectural solutions that address them in safety-critical systems development in agile contexts. The SCA3DA method is composed of a safety standard-independent metamodel and a guideline for the specification of safety strategies in terms of software and hardware components. **Results:** Our findings indicate the feasibility of adopting the proposed method in industry projects, particularly, enriching semantically the understanding of safety requirements implementation by agile teams.

6.1 Introduction

Safety-critical systems are those whose failures could result in accidents and cause damage to the environment, financial losses, injury to people, or even loss of lives (AVIŽIENIS *et al.*, 2004). Diverse critical domains have increasingly used these systems, while the software embedded in these systems is rising in complexity and becoming responsible for most of the critical functions of systems (Nascimento *et al.*, 2020). For example, when diabetes patients use a digital insulin pump to control and fine-tune the blood glucose level, it is a safety-critical system (SCHIDEK; TIMINGER, 2022). The need for diabetes treatment with insulin pumps is advancing at an extraordinary rate and has the potential to improve diabetes outcomes for individuals of all ages (BERGET *et al.*, 2019). Many other key applications are also critical for safety, such as avionics, transportation, automotive, manufacturing, and energy.

Safety requirements play an essential role in the life cycle of safety-critical systems, as they describe means for avoiding, detecting, or handling potential failures in these systems (ISO26262, 2018). They often result from architectural analysis (in particular, safety analysis of system architectures) and, ultimately, elements of the architecture must address them (ANTONINO, 2016). According to IEC 61508 (2010), architecture is where the safety strategies are incorporated into a system. Moreover, errors that are introduced during the specification of safety requirements are usually more difficult and expensive to solve than errors introduced later in the life cycle (HATCLIFF *et al.*, 2014), (BECKERS *et al.*, 2014), and (Critical Software, 2020). In this scenario, although safety standards present strict recommendations for the specification of safety requirements, no guideline also exists to derive such requirements. In particular, each standard or regulation, such as ISO26262 (2018) for the automotive domain, requires that software/hardware safety requirements are derived from the system safety requirements and are also traceable. However, such derivation has been usually conducted in an *ad hoc* way, relying on the understanding, experience, and background of engineers/architects (KASAULI *et al.*, 2021).

It is worth mentioning that recent accidents, such as Boeing 737 crash (2019) (BBC-NEWS, 2019), Nissan airbag's sensory detectors (2014) (JENSEN, 2014), crash of an Airbus A400M (2015) (HEPHER; MORRIS, 2015), train crash in Singapore (2017) (PARK, 2017), Uber's autonomous cars struck and killed a woman (2018) (ISAAC *et al.*, 2018), show the vulnerabilities of safety-critical systems development. For instance, incomplete and vague software specifications are a major reason (i.e., 60%) for medical recalls (ELOFF; BELLA, 2017). Researchers have then argued there are no guarantees that developers will understand what is specified and that the intent of original/correct requirements will be maintained throughout the development process (MARTINS; GORSCHEK, 2016). This is also true for non-safety-critical systems development (De Lucia; QUSEF, 2010) but much more challenging for safety-critical systems.

To manage the challenges that safety-critical systems development has imposed, different application domains have already experimented with agile methods to reduce project risk and increase flexibility and even quality (DEMISSIE *et al.*, 2018). As mentioned in the Medical Device Development Report (2019) (PERFORCE, 2019), the adoption of agile and hybrid development is on the rise in the medical device industry. Likewise, the Agile in Automotive Report (2021) (SCHWEIZER *et al.*, 2021) pointed out that agile development covers more than 52% of the safety-critical automotive sector, for instance, to develop braking systems and sensors.

In agile development, the communication among analysis, design, and development teams usually occur through user stories, which refer to high-level specifications of requirements, containing just enough information so that developers can produce a reasonable estimate of the effort to implement them (BECK; WEST, 2014). However, such specifications are insufficient for properly communicating the safety requirements (ELOFF; BELLA, 2017; CLELAND-HUANG; VIERHAUSER, 2018). Moreover, when face-to-face communication among teams from multiple functional areas and physical locations is not possible, each team may interpret the system differently resulting in inconsistencies (LO; CHEN, 2017). Hence, the main challenge is to go beyond word-of-mouth requirements.

In this scenario, the main problem addressed in this work is that while safety-critical systems have been increasingly built using agile methods, such methods cannot assure that safety requirements are properly addressed in the systems and, particularly, in their architectures. There is also a lack of guidance on how to provide the agile teams with precise safety-related information about those safety-critical parts of the software. Hence, the main research questions guided our work: RQ1: *How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?*.

The main goal of this work is to contribute to agile teams that can suitably understand safety requirements, and these requirements can be properly included in the architecture of software-intensive safety-critical systems. For this, we present SCA3DA, a method composed of a safety standard-independent metamodel and guidelines for the requirements specification to further support agile teams for deriving safety-centered architectures, i.e., architectures that contain a set of necessary information but still aligned to the principles of agile methods. We adopted the design science methodology (WIERINGA, 2014) to develop and evaluate SCA3DA, together with the research cooperation with Fraunhofer IESE[1] (which has led several safety-critical industry projects) and NUTES[2] (which has competence in embedded software development, usability assessment, and conformity analysis of medical devices). We evaluated SCA3DA through three complementary approaches: semi-structured interviews, focus groups, and controlled experiments. We first conducted semi-structured interviews and focus groups with practitioners to analyze the suitability of SCA3DA in agile development as well as its

[1] https://www.iese.fraunhofer.de/
[2] http://nutes.uepb.edu.br/

completeness regarding elements/concepts that the method describes. Following, a controlled experiment analyzed the effectiveness of SCA3DA to specify software safety requirements. Our findings have shown that the application of the SAC3DA method is promising in terms of providing positive support to better understand the software safety requirements specification and that safety-centered architectural solutions derived led to a reduction in the time taken for their analysis, with no loss of requirements understandability.

The remainder of this paper is laid out as follows: Section 6.2 presents the background, especially regarding the main concepts associated with safety-critical systems, and discusses related work. Section 6.3 presents the research methodology applied to establish the SCA3DA Method. Section 6.4 presents the SCA3DA method properly. Section 6.5 describes its evaluation. Section 6.6 discusses the main findings and discusses areas for future investigations. Finally, Section 6.7 summarizes our conclusions.

6.2 Background and Related Work

Safety is a system property defined as absence of catastrophic consequences on the user and the environment (AVIŽIENIS *et al.*, 2004). According to IEC 61508 (2010), which is a basic functional safety standard applicable to all kinds of industries, a safety system (or safety-related system) implements the required safety functions necessary to achieve or maintain a safe state. This system also intends to achieve, on its own or in conjunction with other safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions.

Safety-critical systems aggregate hardware and software elements and play an essential role in many contexts. For instance, if a braking system, a stability control system, an automatic train stop system, or an insulin pump fail, failures can result in unacceptable consequences (MAURYA; KUMAR, 2020)(Nascimento *et al.*, 2020). Hence, safety-critical systems should encompass *safety strategies* to avoid and/or control those failures, which can be random, systematic, common cause, safe, or dependent, according to the safety standards IEC 61508 and CENELEC EN 50126.

According to IEC 61508 (2010), Leveson (2012), RTCA/DO-178 (2012), safety-critical development (i.e., *safety life cycle*) starts with Hazard Analysis and Risk Assessment, as shown in Figure 22. This step is performed during the very early phases of the development process, at the latest when the system requirements are available. The safety goals are also defined as top-level safety requirements that are incrementally refined during the safety life cycle. Once the safety goals are defined, the system development continues through different phases, i.e., the definition of quality attributes, system and software architecture, design, and finally system implementation. Besides that, system validation and verification (*V&V*) are conducted in parallel to the development, and the next steps in the safety development process are also performed in parallel. To this end, Safety Analysis uses the available development artifacts as input to identify potential causes of system failures. From that, the safety engineer and architect owner derive a specification of the safety requirements, their allocation to architectural elements, and the interaction necessary to achieve the safety goals and information associated with these requirements (HATCLIFF *et al.*, 2014). Besides, as the developers incrementally refine the system over the different development phases, the safety engineer analyzes the refined development artifacts step by step and refines the safety requirements accordingly. The whole system development should comply with standards and regulatory guidelines and externalize evidences and arguments as part of safety cases to support third-party certification.

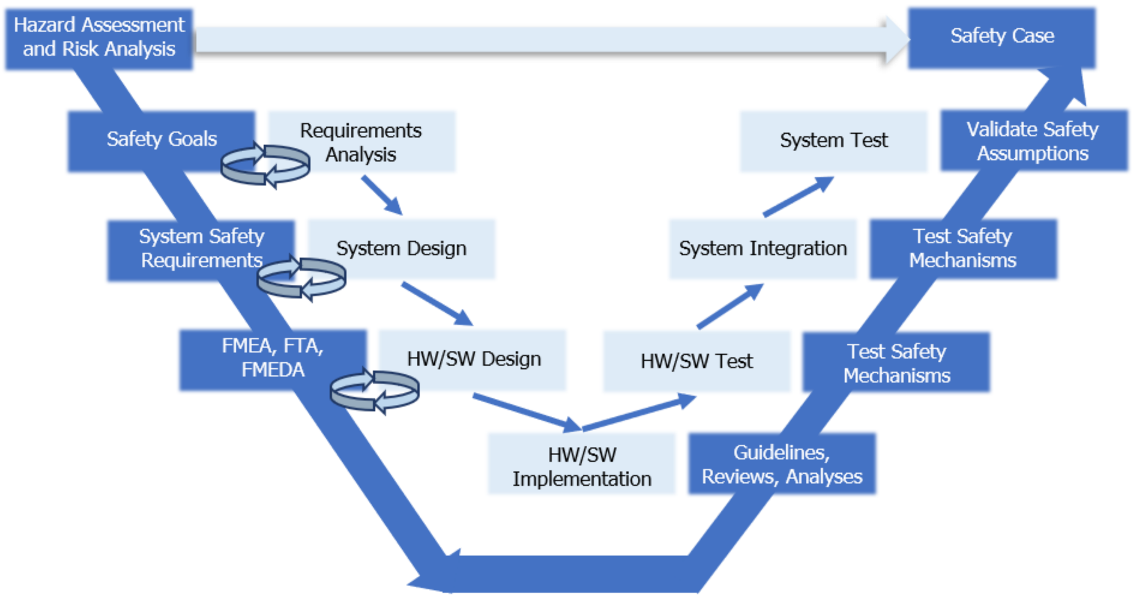


Figure 22 – Overview of the safety development process (adapted from IEC 61511 (2021)).

Safety requirements are essential to develop accordingly safety-critical systems and address all software-based functions whose failures could lead to a violation of technical safety requirements (ISO26262, 2018). Safety requirements must also provide sufficient information necessary to software architecture design and implementation. Besides, designers must be able to demonstrate traceability from design to requirements (RTCA/DO-178, 2012). In this regard, the following aspects should be addressed when specifying safety requirements (ISO26262, 2018; ANTONINO, 2016; BECKERS *et al.*, 2014): (i) it is recommended to break down the specification in smaller and sub-coordinated entities; (ii) traceability from requirements to architecture needs to be explicit; and (iii) the decomposition break-down structure should be traceable to: (a) the failure mode(s)/propagation path(s) that describe failures that motivate the existence of the safety requirement; and (b) architecture elements from multiple abstraction levels – from the high-level functional specification to the software and hardware elements that realize them – that address the needs described in the safety requirements.

In order to find related work, we conducted a systematic mapping (according to Kitchenham and Charters (2007) and Petersen *et al.* (2015)) to identify the state of the art of how agile methods have been applied in the development of safety-critical systems and, more specifically, how safety requirements have been addressed.

Górski and Łukasiewicz (GÓRSKI; LUKASIEWICZ, 2013) mention that many companies have incorporated agile practices into safety-critical projects, which have combined such practices with traditional development processes that rely on safety standards, like SafeScrum (HANSSEN *et al.*, 2018) and IEC 61508 (THORSEN, 2002). However, not much emphasis is given to address or consider safety aspects throughout the development process (GÓRSKI; LUKASIEWICZ, 2013; HEEAGER; NIELSEN, 2018; KASAULI *et al.*, 2018).

Thorsen (THORSEN, 2002) and Stalhane (STÅLHANE; MYKLEBUST, 2016b) specify safety requirements using user stories separately from functional requirements and also in a separate document. Paige et al. (PAIGE *et al.*, 2008) distinguish between user stories and safety stories. Safety stories are specified from user stories to address safety requirements. However, software engineers need to be integrated into the safety process so that the problem domain is really understood. But such interaction is not sometimes possible, meaning the understanding of the problem is often compromised (HATCLIFF *et al.*, 2014), resulting in inconsistent and/or incomplete system specification. Cleland-Huang (CLELAND-HUANG; VIERHAUSER, 2018) specifies safety stories using Easy Requirements Specification (EARS) format, and these should be traceable to the design definition, which addresses safety stories at the system level. Likewise, Wang et al. (WANG *et al.*, 2017) propose a new pattern to specify safety stories, but the development team needs to have background and knowledge on safety. Although there are various studies suggesting the adoption of user stories, according to Vouri (VUORI, 2011), they are subjective and not sufficient for specifying safety requirements, which are rather objectives and contain standard-defined design and implementation requirements. Furthermore, according to (ANTONINO *et al.*, 2014), it is hard to fully express the complexity of the functionalities of these systems only with user stories.

Martins and Gorschek (MARTINS; GORSCHEK, 2016) conducted a systematic literature review on requirements engineering for safety-critical systems. One of their findings was the lack of studies on communication of safety requirements among the project team members. This issue impacts directly on the certification process, which is mandatory in the safety-critical systems industry, and means that there is no guarantee that the developers will really understand what is specified, and that the intent of original requirements is maintained throughout the development.

The lack of studies that deal adequately with the specification of safety requirements motivated us to conduct this work.

6.3 Methodology

This section presents the design science methodology (WIERINGA, 2014), as shown in Figure 23, to adequately establish the SCA3DA (**S**afety-**C**entered **A**rchitectural **D**rivers and **D**ecisions

Derivation), a method composed by safety standard-independent metamodel and guidelines for the software safety requirements specification.

We conducted three main steps: (i) Step 1: Identification of information sources; (ii) Step 2: Development of artifacts; and (iii) Step 3: Evaluation. First, the design science methodology guided the *identification of information sources* through various relevance cycles including an observational case study about how safety requirements are addressed by agile teams. Then, *the development and evaluation* of SCA3DA was conducted through multiple design cycles with the stakeholders in the problem domain. More specifically, The **SCA3DA metamodel and guidelines** were conceived to support agile teams (including team leaders and architects) for deriving safety-centered design decisions that satisfy safety stories and are necessary to address the critically inherent safety requirements. The artifacts were iteratively studied and improved through direct support from our industry collaborator. Finally, to fulfill the rigor cycle in our methodology, we conducted a controlled experiment in the medical domain and reflected on how our research results are transferable to other settings. These steps are discussed below:

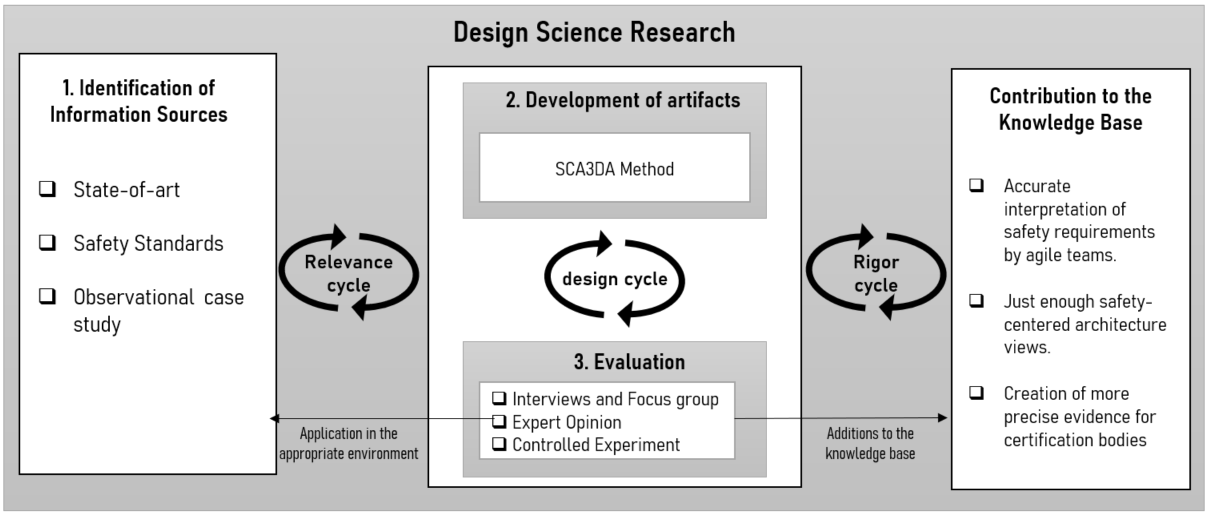

Design Science Research
1. Identification of Information Sources
State-of-art
Safety Standards
Observational case study
Relevance cycle
Application in the appropriate environment
2. Development of artifacts
SCA3DA Method
design cycle
3. Evaluation
Interviews and Focus group
Expert Opinion
Controlled Experiment
Rigor cycle
Additions to the knowledge base
Contribution to the Knowledge Base
Accurate interpretation of safety requirements by agile teams.
Just enough safety-centered architecture views.
Creation of more precise evidence for certification bodies

Figure 23 – Our design science methodology (adapted from Wieringa (2014)).

Step 1 - Identification of information sources: We considered three information sources: (i) related studies; (ii) safety standards; and (iii) case study. We first conducted a systematic mapping following the guidelines proposed by (KITCHENHAM; CHARTERS, 2007) and (PETERSEN *et al.*, 2015) to find related studies and gather knowledge of the state of the art of how has safety been addressed throughout agile development. As the main results of this SM, we found that architecture is considered the backbone in safety-critical development, but few publications (i.e. 4 studies) explicitly establish a systems/software architecture to develop safety-critical systems using agile methods/practices. As well, we observe there is still no consensus regarding which agile requirements practices are most adequate to address safety requirements. This leads us to infer that more studies should be conducted to allow more widespread and efficient practices to specify software safety requirements. Further, we did not find guidelines to provide software architectural information/artifacts that are needed to specify safety strategies, such as failure detection, containment, and avoidance.

Another information source was safety standards. We analyzed them to identify what is necessary to specify software safety requirements. These standards were considered because compliance with safety standards is required in safety-critical systems development. We first analyzed IEC 61508, which is a basic functional safety standard applicable to all kinds of industries. This standard describes appropriate techniques and strategies to specify safety requirements according to safety integrity levels (SIL). SIL is used for specifying the safety integrity requirements (of the safety functions in safety-critical systems), where SIL 1 has the lowest level of safety integrity and SIL 4 has the highest (IEC 61508, 2010). Hence, for each SIL, we map the recommendations of the standard regarding safety strategies for fault detection and containment.

We also analyzed other domain-dependent safety standards, in particular, ISO 26262 for the automotive domain, IEC 62304 for the medical domain (IEC 62304, 2006), and DO-178C for the avionics domain (RTCA/DO-178, 2012). Then, we systematically defined those recommendations that are domain-dependent, had already been mapped, or had not yet been identified. As a result, we defined a checklist with the information recommended to specify safety strategies for fault detection and containment and take the system to a safe state. For instance, the recommendation *"traceability between the software safety requirements specification and software architecture"* is required in safety-critical systems whose SIL is 3 or 4. *"Semi-formal notations for software and hardware safety requirements specification"* is highly recommended in systems whose SIL is 3 or 4. *"Software safety requirements shall address monitor techniques with separation between the monitor component and monitored component"* is recommended for SIL 3 and is highly recommended for SIL 4.

We also conducted an observational case study (LEITE, 2017) that analyzed how safety requirements are specified in agile development and whether safety strategies are addressed in the software requirements specification. This study was conducted with three different profiles (engineer responsible for designing and implementing systems, developer, and requirements engineer) in the context of different projects at Fraunhofer IESE. We found that all participants had difficulties understanding safety requirements and deriving them at software and hardware detailed levels. Hence, it would not be possible to ensure that the original intent of the safety requirements was implemented. This study served to reinforce the need to enrich agile practices with safety and risk management practices, without sacrificing agility and still providing the necessary assurance level.

Step 2 - Development of SCA3DA: We dedicated this step to developing SCA3DA, i.e., the two artifacts (metamodel and guidelines) that compose our method. The first version was based on safety standards, the related studies (gathered from the systematic mapping), and a case study. After evaluations (summarized in Step 3 and detailed in Section 6.5), we obtained the final version of SCA3DA, presented in Section 6.4.

Step 3 - Evaluation: We dedicated this step to evaluating SCA3DA (metamodel and guidelines). Two focus groups of about 2 hours were organized with eight members (developers and architects) of agile projects, and three safety experts from NUTES. We first presented the identified concepts and relationships, safety stories definition, and safety-centered architectural views. We then discussed the guidelines to understand the software safety requirements specification. Additionally, evaluated SCA3DA using semi-structured open-ended interviews (according to (ROBSON; MCCARTAN, 2015)) with five members of agile industry projects from Fraunhofer IESE. We presented the metamodel (its concepts and relationships), the guidelines, and how they could be used throughout agile development. We asked the participants about the understandability, completeness, and suitability of SCA3DA and its implications and relevance for researchers and practitioners. After that, we returned to Step 2 to refine SCA3DA considering the answers of the interviewees. We also conducted a controlled experiment with practitioners from NUTES and Atlântico Institute[3] where we evaluated if practitioners consider that SCA3DA can improve the understandability of software safety requirements specifications. Section 6.5 details this controlled experiment.

6.4 SCA3DA Method

SCA3DA is a method for specifying software safety-critical requirements, in particular, requirements associated with the minimal set of architecturally relevant information required to specify failure detection and containment strategies to take systems to a safe state. Section 6.4.1 presents an overview of the metamodel, while Section 6.4.2 presents our safety story definition. Section 6.4.3 details the SCA3DA method as a whole. Then, Section 6.4.4 presents SCA3DA integrated into the agile method.

6.4.1 SCA3DA Metamodel

The SCA3DA metamodel was presented in Leite *et al.* (2020). It is a safety-standard-independent metamodel and provides guidance for the specification of just-enough safety-centered architectural views so that agile teams can identify what is important at each level of architecture abstraction (context, function, software, and hardware) to address safety stories (as depicted in Figure 24). As a consequence, the development team's understanding of those critical parts that deserve attention is improved.

The metamodel provides a guide to identify which safety-related information should be considered, in each abstraction level, for specifying safety requirements at the software/hardware level. As already stated by (CLELAND-HUANG; VIERHAUSER, 2018), specifying the safety story alone is insufficient for the agile teams to understand the real intention of the requirement; information about the design is also necessary. Therefore, the metamodel meets the compliance

[3] https://www.atlantico.com.br

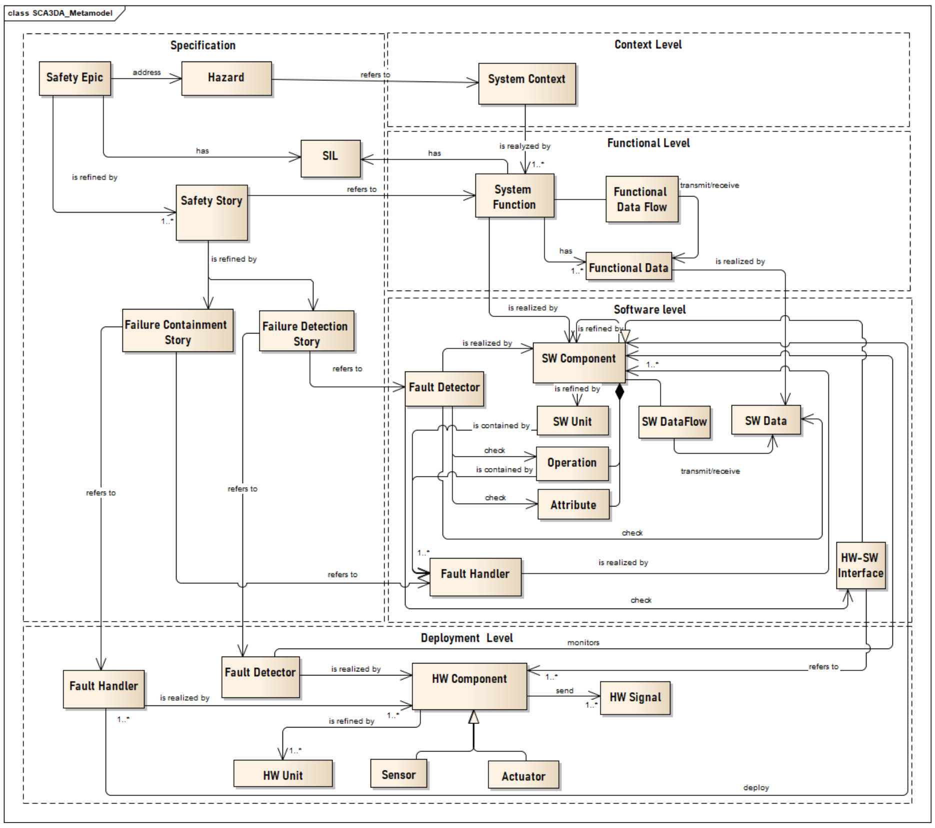


Figure 24 – SCA3DA metamodel (LEITE *et al.*, 2020).

requirements of safety standards, proposing traceability between the specification and the design, and guiding the derivation of the safety goal (system level) to failure detection and containment stories (software/hardware level), which are described more detailing in next section.

6.4.2 Safety Story Definition

According to Cleland-Huang and Vierhauser (2018), safety stories are equivalent to safety requirements, as described by Firesmith (2004) as a *"requirement that specifies a minimum, mandatory amount of safety in terms of a system-specific quality criterion and a minimum level of an associated metric"*. Different patterns to describe safety stories have been presented, for instance, Myklebust and Stalhane (2016) defines a safety story like a user story template (i.e., *To keep <function> safe, the system must <achieve or avoid something>.*). While (WANG *et al.*, 2017) modified that pattern before considering also the unsafe control actions (i.e., *To keep <the control action> safe, the system must <achieve or avoid something>.*). Both reinforce that linking safety needs to software design is already required by IEC 61508; however, they do not specify which information should be considered. Cleland-Huang and Vierhauser (2018) advances in this direction and describes design definitions that represent solutions for addressing the respective safety story. These solutions share three common aspects: (i) an agile team highly integrated with the safety team and effective communication; (ii) reliance on the architect's experience and background; and (iii) specification defined at the system level. The development of safety-critical systems (SCS) does not always have this effective communication, and without an explicit context of how to derive the software safety requirement, it is not possible to ensure that its real intention is understood by the development team and therefore implemented.

In SCA3DA metamodel (depicted in Figure 24), we presented our safety story refinement in compliance with generic safety standard (IEC 61508, 2010), and specific domains (ISO26262, 2018), (IEC 62304, 2006), (AAMI TIR45, 2012), (RTCA/DO-178, 2012), and (CENELEC - EN 50128, 2011). Safety epics illustrate the top-level safety requirements (i.e., safety goals in the automotive domain), which, in turn, exist to address hazards identified in the hazard analysis and risk assessment (LEVESON, 2012) (TRAPP; SCHNEIDER, 2014). Moreover, a safety epic has a safety integrity level (SIL), that corresponds to a range of safety integrity values, where SIL 4 has the highest level of safety integrity and SIL 1 has the lowest level (IEC 61508, 2010), (ISO26262, 2018), (ANTONINO, 2016). In this regard, the SIL needs to be indicated explicitly throughout the specification to ensure safety integrity compatibility between safety epic, safety stories, and software architecture design. Next, a safety epic can be refined in many safety stories, which should provide a description of the strategies used for realizing a functional safety requirement (ISO26262, 2018) (FIRESMITH, 2004). Hence, each safety story must be traced back to its respective system function in the architecture at the functional level.

Safety standards recommend that safety requirement specifications describe means for detecting and containing failures, involving fault tolerance techniques that allow "living" with systems that are susceptible to faults (IEC 61508, 2010), (ISO26262, 2018), (IEC 62304, 2006). And, conform to Antonino (2016), Hatcliff *et al.* (2014), and Mäder and Gotel (2012), the safety engineer has more focus on the general safety strategy, and failed to describe strategies for detection and containment in precise terms. By explicitly describing detection and containment strategies, the engineer can indicate how to detect and contain the failures considering elements of the software architecture (already existing or to be introduced). Concerning that, each safety story is refined in the **failure detection story** and **failure containment story**. The detection and containment measures described in the failure detection and containment stories should refer to elements in the software architecture that will address these measures. If the elements already exist, these stories just need to reference them. If they are not part of the software architecture, they have to be included. In addition, we understand that it is necessary to ensure that the safety story, failure detection and containment stories, and software architecture are compatible in terms of their integrity level.

6.4.3 SCA3DA Step-to-Step

The SCA3DA method aims to improve the understanding of safety requirements by agile teams and support safety-centered architectural design decisions. Using this method allows building the necessary common knowledge in agile teams to adequately consider safety issues and, in fact, create more precise evidence for certification bodies that safety issues were implemented, without burdening agile development.

As software architecture views are essential to ensure that safety is properly addressed in the system, the SCA3DA method provides thought metamodel and guidelines support to specify just-enough safety-centered views. These views make it possible for agile teams to identify what (i.e., context, function, software, and hardware) is important in the architectural abstraction level to address safety stories as described in Section 6.4.1. These views can also communicate to the development teams those critical parts. Figure 25 presents the SCA3DA method.

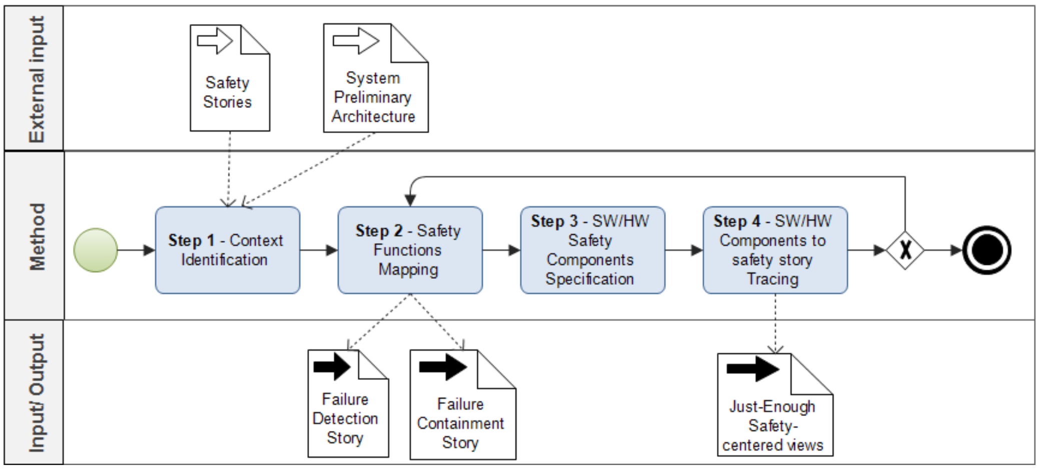

External input
Safety Stories
System Preliminary Architecture
Method
Step 1 - Context Identification
Step 2 - Safety Functions Mapping
Step 3 - SW/HW Safety Components Specification
Step 4 - SW/HW Components to safety story Tracing
Input/ Output
Failure Detection Story
Failure Containment Story
Just-Enough Safety-centered views

Figure 25 – Overall view of SCA3DA method

- ***Step 1 - Context Identification:*** Based on system safety stories and the preliminary specification of system architecture, the safety concerns (hazards, risks, failures) are identified from the hazard and risk assessment and, for each hazardous event, a safety epic (safety goal or top-level safety requirements) and the SIL associated with it are determined.

- ***Step 2 - Safety Functions Mapping:*** In this step, each safety epic is refined by safety stories, which record strategies for realizing safety strategies. These strategies take into account the SIL, which specifies the necessary requirements and safety strategies to avoid unreasonable residual risks (IEC 61508, 2010). These safety strategies are related to failure detection, containment, or avoidance. Hence, each safety story can be refined into a *failure detection story* and a *failure containment story*, as recommended by safety standards (such as (IEC 61508, 2010), (ISO26262, 2018), and (IEC 62304, 2006)). Moreover, it is necessary to specify traces between the safety story and the system safety function and the dependencies that address it. A system function represents a service of the system that is provided either to an external entity or to another function of the system.

- ***Step 3 - SW/HW Safety Components Specification:*** Once that function is realized by the software component (at the software level) or hardware component (at the hardware level), it is necessary to identify this component, its provided and required interfaces, as well as software data flow. Once failure detection stories and failure containment stories are realized by the software, at this software level, it is essential to make design decisions according to the SIL and communicate explicitly which failure detection or containment strategy will be implemented. According to each strategy, it is necessary to identify the software elements necessary to realize it. Likewise, failure detection stories and failure containment stories can be realized by hardware (at the hardware level); in this case, it is essential to define, according to the SIL, the detection or containment strategy to be implemented and to identify the hardware components needed to realize that strategy.

- ***Step 4 - SW/HW Components to Safety Story Tracing:*** Regarding failure detection, the SIL induces different fault detection strategies. For instance, the *fault detector component* can be realized by another software component, in this case, the software component might either be deployed in the same hardware or not, or it may be implemented in a software unit and may be achieved by checks in the value domain (software data) or in the time domain. Likewise, for failure containment, the *handler component* prevents fault propagation across defined boundaries. Besides, it is necessary to define the level at which the containment will occur, whether at the component or the software unit level. Handler components according to the SIL are deployed in different hardware or modules.

6.4.4 SCA3DA Method integrating to Agile Process

In order to better illustrate how SCA3DA method can be positioned in the context of agile development, we considered the SCRUM framework, as depicted in Figure 26. We chose SCRUM because several safety-critical systems projects have already adopted SCRUM (HEEAGER; NIELSEN, 2017; WEBER, 2015; ALBUQUERQUE *et al.*, 2012; CAWLEY *et al.*, 2010), which can be also combined with other agile methods, e.g., eXtreme Programming (HEEAGER, 2014), AV-Model (MCHUGH *et al.*, 2013), Kanban (MULLER; BESEMER, 2018)). In summary, Scrum presents three major phases (SCHWABER; SUTHERLAND, 2017): (i) pregame (i.e., definition of system requirements); (ii) game (development and testing cycles); and (ii) postgame (i.e., quality assurance, validation, and packaging).

When applying SCA3DA method during *pregame*, activities related to safety are carried out, such as risk and hazards analysis, the definition of system preliminary architecture, fault analysis, and specification of system safety requirements over several iterations. Throughout this phase, requirements (functional and architectural) are specified in user stories, in the case of safety in safety stories as a state by (STaLHANE; MYKLEBUST, 2018), (CLELAND-HUANG; VIERHAUSER, 2018), and (WANG *et al.*, 2018). These stories should be prioritized and allocated to the product backlog. In this phase, SCA3DA supports product owners and safety engineers to define more rigorously a safety story. Thus, safety-centered stories should consider the SIL explicitly, as well as failure detection and containment strategies as recommended in (ISO26262, 2018) and (IEC 61508, 2010).

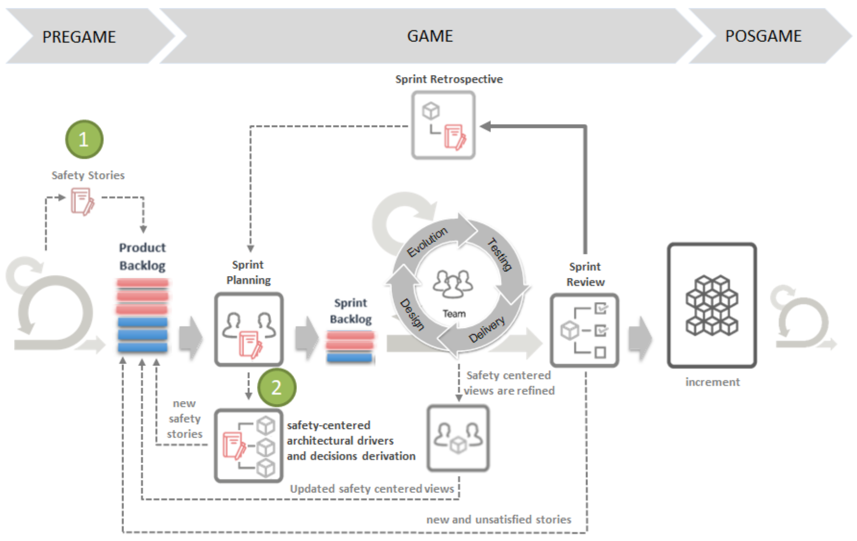


Figure 26 – SCA3DA in the SCRUM framework.

The *game* consists of several sprints until all implemented requirements have been approved. At the beginning of each sprint, the planning of which requirements will be allocated/implemented in that sprint is carried out, those with the highest priority should be implemented first, given the safety criticality, safety stories have higher priority. During planning, each safety story is decomposed into new stories, which can be a user story (a new functional requirement), an architectural story (a new architectural requirement, regarding performance, for instance), or a new safety story. During the decomposition, the product owner together with the architect defines which safety story will be implemented at the software level (safety software story) or the hardware level (safety hardware story), this decision is made considering the system's preliminary architecture. SCA3DA, which is composed of metamodel and guidelines, allows the architect to specify the safety software/hardware stories, as well as the decisions that address them, resulting in just-enough safety-centered architectural solutions, which are used as a guide by the development team. During the development cycle, these just-enough safety-centered architectural solutions can be refined by them. At the end of the sprint, new stories will compose the product backlog. The refinements on safety-centered architectural solutions will compose the system's preliminary architecture for the next sprint.

The next section presents the evaluation of SCA3DA.

6.5 Evaluation

We systematically evaluated the SCA3DA Method, as mentioned in Section 6.3. Section 6.5.1 presents the first evaluation activity (in the design phase), which sought to address the question of whether the SCA3DA method is suitable for agile development, and the completeness of the concepts described in the metamodel. And, Section 6.5.2 presents the second evaluation activity, a controlled experiment (in the evaluation phase), which aimed to analyze the effectiveness (with respect to understandability, accuracy, and effort) of the SCA3DA method to support the specification of software safety requirements.

6.5.1 Interviews and focus group with practitioners

The purpose of the interviews and focus groups section, according to GQM (BASILI *et al.*, 2002), was *to assess the suitability of the SCA3DA method in agile development, and the completeness of the concepts described it; from the point of view of the agile team (Product Owner, architect, developer) and Safety Engineer, in the context of agile development of safety-critical systems*. To this end, we formulated three research questions:

- **(RQ1)** Does the proposed method meet practitioners' needs? This RQ aimed at studying the suitability of the SCA3DA method in an agile context, by analyzing if practitioners consider that the method proposed supports their current practices for specifying failure detection and containment strategies as well as strategies for taking the system to a safe state and avoids or mitigates possible challenges. We regard these characteristics as the main ones that the method must ensure in order to enable its application in the industry. If the characteristics are not ensured, then the method will not meet practitioners' needs;

- **(RQ2)** Are main concepts (described in the method) easy to identify, understand, and follow by practitioners? This RQ aimed at studying whether the main concepts described in the SCA3DA method are complete enough for to practitioners specify failure detection and containment strategies as well as strategies for taking the system to a safe state. If practitioners can not understand and follow the concepts using methods for specifying and communicating safety strategies, among other aspects, then it is unlikely that the method will be complete.

- **(RQ3)** Do practitioners find benefits in using the proposed method? This RQ aimed at studying if the guidelines are considered useful and thus can help in mitigating or avoiding issues in the specification of failure detection and containment strategies as well as strategies for taking the system to a safe state. If practitioners do not find benefits in using models for specifying and communicating safety strategies, among other aspects, then it is unlikely that the method will be adopted in the industry.

6.5.1.1 Data Collection

Data was collected during three sessions: the first session was conducted with members of agile project teams from IESE Fraunhofer (5 participants). The second was with members of agile project teams from NUTES (7 participants). And finally, with safety engineers from IESE and NUTES (3 participants).

In the first and third session we conducted semi-structured open-ended interviews, which lasted around 45 minutes each one. We started with a short presentation regarding to purpose of the SCA3DA method, its concepts, and practical considerations. Then, the metamodel and guidelines were introduced and an example of how to use the method was presented. We asked the interviewees whether they agreed with the findings, whether any information was missing, and what conclusions or implications they thought of. The second session took the form of focus groups (TREMBLAY *et al.*, 2010), first, we presented an introduction regarding the research problem, and the goals of evaluation. Then, the attendees received a description of the method with the metamodel and guidelines, and examples were presented. We then followed short question-and-answer discussions about the method to understand the differences between participants' perspectives and the method's suitability to agile context. This stage took 1.5 hours. During the focus group, the attendees also made other comments on the SCA3DA method and its resulting models. We noted these comments and later analyzed them and updated the metamodel accordingly.

After each session, we prepared an anonymous questionnaire to gather quantitative and qualitative data (as depicted in Appendix B). The questionnaire included a 5-point Likert scale (FOWLER, 2013) in order to measure to what degree practitioners agreed with the findings and so they were able to describe strengths and drawbacks.

6.5.1.2 Results

To answer RQ1, the results shown in Figure 27 provide evidence that the SCA3DA method can meet practitioners' needs. There was an overall agreement that the SCA3DA method is aligned with the precepts of agile development, and its main concepts are clear, as well as, the detail's level. For RQ2, the results showed that most practitioners agree upon the ease of understanding and the ease of identification of the main concepts, which are enough (as depicted in Figure 27). Besides, there was agreement on the ease to follow the metamodel, whose objective is clear. Regarding RQ3, the practitioners find benefits, especially in safety requirement specification, understanding of safety requirement intention, and understanding of the relationships between safety concepts. Most respondents indicated that the metamodel was easy to understand, they would very probably use the method to help in understanding the safety requirement, and they found the metamodel enough to use for communication with agile teams.

6.5.2 Controlled Experiment

The purpose of the controlled experiment was to evaluate the SCA3DA method's effectiveness to support the specification of software safety requirements. For this, an experiment was planned and conducted, considering the pilot experiment applied and validated with students (LEITE *et al.*, 2022), and well-known guidelines (WOHLIN *et al.*, 2012; JEDLITSCHKA *et al.*, 2008), with practitioners from NUTES (5 participants) and Atlantico Institute (7 participants). They solved

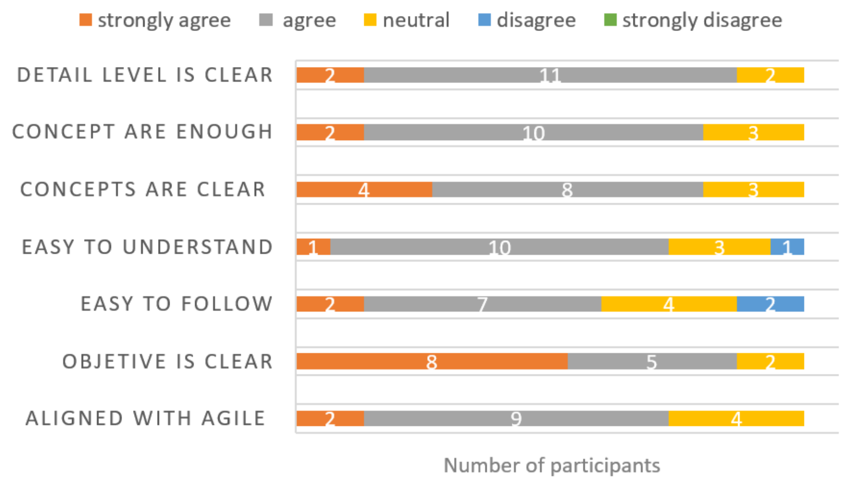


Figure 27 – Feedback on SCA3DA method.

safety requirements specification tasks in the medical domain. Two groups (with and without training) were measured in terms of the subjects' understanding of the tasks, and safety issues, as well as the accuracy of the solutions proposed by the subjects and the effort required. For the design, a balanced factorial design with one factor of interest with two treatments (with/without training) was used. The following research questions were addressed in this study:

- **RQ1:** To what extent can the subject understand the safety requirements in order to derive specifications from them?
- **RQ2:** How accurate are the specifications of the software safety requirements provided by the subjects?
- **RQ3:** How much effort do subjects need to expend to specify software safety requirements?

Three dependent variables were then considered: understanding, accuracy, and effort. For each of these, the following hypotheses were defined:

- **H1:** Practitioners are able to understand safety requirements and then provide specifications that address these requirements independent of the training. (H1.0 is the null hypothesis and H1.1 is the alternative hypothesis).
- **H2:** Practitioners are able to prepare accurate specifications from safety requirements independent of the training. (H2.0 is the null hypothesis and H2.1 is the alternative hypothesis).
- **H3:** The effort expended by Practitioners to understand safety requirements and then provide specifications that address these requirements is independent of whether the participants receive training or not. (H3.0 is the null hypothesis and H3.1 is the alternative hypothesis).

Table 16 – Self-assessment about background and experience by subjects

	Descriptive Results				
Self-Assessment on 5-point Likert scale	**Mode**	**Median**	**MAD**	**Min**	**Max**
Background in software development	4	4	0	3	5
Experience in software development	4	4	0	3	5
Background in safety software development	2	3	1	2	4
Experience in safety software development	2	3	1	2	5

This experiment involved a total of 12 participants/subjects. Concerning the subjects' profile, which was collected through a pre-experiment questionnaire (as presented in B), most of them had know-how and experience in software engineering: 3 of 12 subjects had experience as a Product Owner, 8 as a developer, and 3 as an architect. Ten knew about safety engineering and seven specifically about safety-critical systems development. Moreover, the subjects rated their background and experience (in software development and safety software development) on an ordinal scale (1= very poor to 5 = very good), whose mode, median, median absolute deviations (MAD), minimum, and maximum scales are shown in Table 16. Overall, the subjects had a high level of experience in software development and a low level of experience in safety engineering. The deviation in the subjects' answers regarding their self-assessment for all responses was 1 or lower; hence, we had a homogeneous group. The subjects were randomly assigned to two groups (A and B), with and without training respectively.

The experiment included one task in the medical domain, which addressed the description of software safety requirements. Subjects were asked to describe the necessary changes in the architectural design to encompass those requirements. To do that, subjects had to understand the task to be solved, explore the project documentation (e.g., functionalities and architectural description to identify parts to be changed), and report their solution.

The experiment was conducted over one day, in two sessions. In the first session, the reason for the experiment was explained to all subjects, and the documentation (i.e., consent form, analyst survey, and project documentation) was delivered. Then the participants were randomly shuffled into two groups. Then, without getting any training, Group A worked on the assigned task and, at the end, filled in the task report and the post-experiment questionnaire. In the second session, the subjects from Group B received training in safety-critical systems development, in particular, regarding software safety requirements specification supported by the SCA3DA method. The training was done in a 2-hour session using a system — Airbag System — unlike those the participants used later during the experiment. This session consisted of 30 minutes for the theoretical presentation and 90 minutes for practicing. At the end of the training, the participants received feedback on their performance and an example of a correct specification of the software safety requirements of that system. This feedback allowed the participants to understand which safety-related information should be explicitly considered in the specification of software safety requirements. Then group B performed the tasks and filled in the questionnaire. Other data was also collected during the execution of the experiment, i.e., the time required to perform the tasks and the tasks' level of difficulty, and a data analysis was conducted with appropriate descriptive statistics.

To address RQ1, we applied the strategy outlined in (Carew *et al.*, 2005), which involved participants reading artifacts and providing responses. Subsequently, participants were given a questionnaire to gauge their understanding of the task, project description, and safety requirements. The response time and accuracy of their answers served as indicators of their comprehension of the artifacts. Additionally, we calculated the central tendency strategies for each question in the post-experiment questionnaire, including mode, median, and MAD to assess their dispersion. Furthermore, the reliability of the post-experiment questionnaire responses was evaluated using Cronbach's α, a common measure of internal consistency, with a value of 0.7 or higher considered sufficient (RITTER, 2010).

To address RQ2, we evaluated the solutions proposed by the participants, which included the safety requirements specifications. These solutions were compared to a baseline, which consisted of correct solutions provided by experts in safety-critical systems development and certification. We analyzed the participants' solutions by comparing them to the essential elements found in the baseline, such as failure detection, containment strategies, and methods to bring the system to a safe state. The accuracy level indicates how closely the solutions align with the baseline. In cases of uncertainty, experts who assisted in our experiment determined the accuracy of the solutions.

Finally, to answer RQ3, the subjects' effort was measured by considering the time required to perform the tasks.

6.5.2.1 Results

Section 6.5.2.2 presents the data collected and processed concerning the three dependent variables, while Section 6.5.2.3 presents the hypothesis testing. Section 6.5.2.4 discusses the answers to our RQs and Section 6.6.3 the threats to the validity of this experiment and how these were mitigated.

6.5.2.2 Descriptive Statistics

Understandability: Table 17 shows the reliability statistics for the dependent variable understandability calculated using the Cronbach's α reliability measure, which demonstrated high reliability and internal consistency, as it ranged from 0.71 to 0.79, i.e., it exceeded 0.70.

Table 17 – Cronbach's α calculated for the dependent variable understandability from data collected in the post-experiment questionnaire.

Dependent Variable	**Understandability**		
	Task	Project	Training
Cronbach's α measure	0.79	0.71	0.77

Delving further, we posed seven questions (Q3–Q9) to gather the participants' opinions regarding the understandability of the task description, safety-critical project documentation, the safety requirement to be addressed, and the intended specification to be developed. Table 18 summarizes the results. Specifically, Q3 inquired whether participants found the task description easy to comprehend. The results indicated unanimous agreement that the task description was clear. Q4 focused on whether participants encountered challenges in understanding the project documentation. Participants uniformly disagreed, and there was no deviation, indicated by a MAD of 0. Q5 explored whether participants faced difficulty understanding the safety requirements. Participants disagreed, and interestingly, this perception was consistent across both trained and untrained individuals. Q6 probed whether participants struggled with specifying the safety requirements. The responses indicated agreement that providing specifications was challenging, with all responses showing a deviation of 1 or lower. Questions Q7-Q9 were specifically designed to gather feedback from participants who underwent training (Group B) and were not applicable to Group A. Q7 assessed whether training improved their understanding of safety requirements. Participants in Group B mostly agreed, with a small deviation (MAD: 1). Q8 investigated whether the training clearly outlined how to complete the task. The results indicated unanimous agreement that the training was helpful in task completion, with no deviation. Finally, Q9 inquired whether training aided in specifying the safety requirements. The majority strongly agreed, with all responses showing a deviation of 1 or lower.

The accuracy of solutions provided by each participant and the effort required were calculated and analyzed for normality. Table 19 displays their descriptive statistics, including mean, median, and standard deviation. The results indicate that both accuracy and effort (measured in minutes) had a higher mean for the group that received training. Additionally, Figure 28.(a) compares the time spent by participants on tasks with training (orange line) and without training (blue line) using line plots. Similarly, Figure 28.(b) contrasts the overall accuracy achieved (shown as bar plots) and the accuracy per project. Each bar represents the percentage of solutions based on their accuracy level (ranging from 0

6.5.2.3 Hypothesis Testing

To test the three hypotheses outlined earlier, a one-way analysis of variance (ANOVA) with post-hoc analysis at a confidence level of 0.05 was conducted (SCHEFFE, 1999). ANOVA was utilized to assess the impact of the independent variable on the dependent variables: understandability,

Table 18 – Descriptive statistics for the dependent variable understandability calculated from data collected in the post-experiment questionnaire. (* Not applicable.)

Understandability										
Project Medical										
Training	**Q3**					**Q4**				
	Median	Mode	MAD	Min	Max	Median	Mode	MAD	Min	Max
No	4	4	1	2	5	1	1	0	1	3
Yes	4	4	0	4	5	2	2	0	1	2
	Q5					**Q6**				
	Median	Mode	MAD	Min	Max	Median	Mode	MAD	Min	Max
No	1	1	0	1	2	4	4	0	3	5
Yes	2	2	0	1	2	5	5	1	3	5
	Q7					**Q8**				
	Median	Mode	MAD	Min	Max	Median	Mode	MAD	Min	Max
No	*	*	*	*	*	*	*	*	*	*
Yes	4	4	1	1	5	4	4	0	4	4
	Q9									
	Median	Mode	MAD	Min	Max					
No	*	*	*	*	*					
Yes	4	4	1	3	5					

Table 19 – Descriptive statistics calculated for accuracy and effort dependent variables.

Project (P)	**Training (T)**	Accuracy [%]			Time [min]		
		Mode	Median	Standard Deviation	Mean	Median	Standard Deviation
Medical	Without	25	25	0.25	22.8	19.5	8.54
	With	50	63	0.24	39.8	40	4.44

Table 20 – Summary of analysis of variance (n = 12).

Source of variation	**Df**	Understandability			Accuracy			Time		
		F	p-value	Fc	F	p-value	Fc	F	p-value	Fc
Independent Variable										
Training (T)	1	26.27	0.0001	4.96	6.63	0.02	4.96	18.69	0.001	4.96

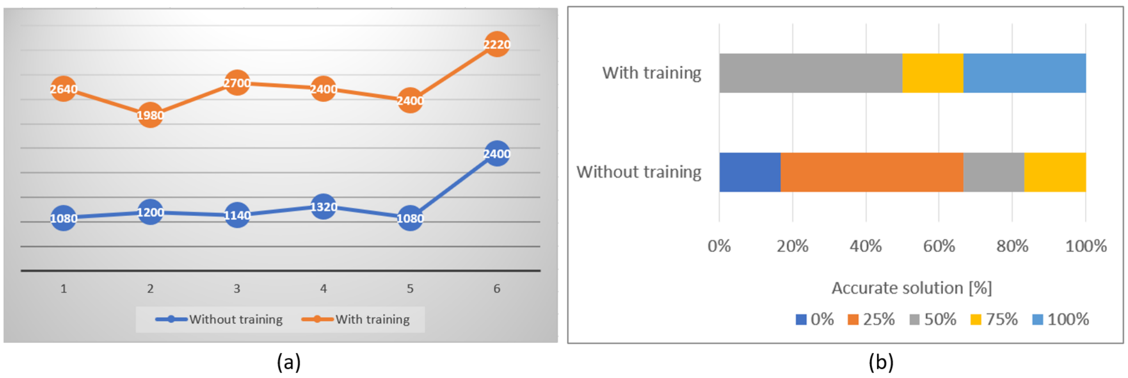


Figure 28 – (a) Effort measured through time spent by subjects to perform tasks. (b) Accuracy of the solutions provided by subjects.

accuracy, and effort. The results of these analyses[4] are detailed in Table 20. For hypothesis H1, the test revealed that the training significantly affected the understandability of safety requirements. So the alternative hypothesis H1.1 was accepted with p-values $(p < 0.05)$. For H2, the statistical tests showed that the training significantly affected the accuracy's level (see row Training (T) of Table 20: $F < Fc$, where F represents *F distribution* and *Fc* limits the rejection region (WOHLIN *et al.*, 2012)). Hence, the alternative hypothesis H2.1 with p-values $(p > 0.05)$ was accepted.

For H2, the statistical tests indicated that training had a significant impact on the accuracy level (as indicated by the Training (T) row in Table 20: $F < Fc$, where F represents the F distribution and Fc limits the rejection region (WOHLIN *et al.*, 2012)). Consequently, the alternative hypothesis H2.1 with p-values $(p > 0.05)$ was accepted. Regarding H3, the test revealed that the training, which aimed at providing guidelines for solving the task, affected significantly effort. Hence, the alternative hypothesis $(H3.1)$ was accepted.

6.5.2.4 Answering Research Questions

Based on the experiment results, the Research Questions (RQs) were addressed. To answer RQ1, the variable analyzed was "understandability," investigating the participants' comprehension of safety requirements. This question is pivotal because if software engineers do not genuinely grasp the requirements, there is no assurance that these requirements will be properly addressed throughout the software development process. In this experiment, training proved crucial for enhancing this understanding. A significant 72% of participants who received training reported a better understanding of safety requirements. In contrast, only 21% of those without training claimed to comprehend the safety requirements.

Moreover, even though participants asserted they understood the intention behind a safety requirement, a majority (78%) admitted they faced challenges in providing a specification that adequately addressed that requirement. This difficulty was also influenced by their lack of in-depth background and experience in developing safety-critical systems.

For RQ2, the accuracy of the solutions proposed by the participants was assessed. As depicted in Table 19, accuracy had a mean and median of 63% only in the scenario with training (meaning 50% of the solutions designed by the participants who received training had an accuracy level above 75%). Additionally, in this scenario, a 100% accuracy level was achieved. On the other hand, 20% of the solutions developed by the participants who did not receive training did not exceed 25% accuracy. In summary, accuracy did not reach high values for the group without training. The main challenges reported by the participants revolved around correctly specifying these requirements, which should be accomplished through architectural design. However, most of them provided pseudo-code solutions for the requirements without considering safety aspects. Notably, 64% of the solutions focused solely on functional aspects, while 20% failed to produce a solution altogether. As highlighted in previous studies (HATCLIFF *et al.*,

[4] All statistical calculations were performed using the R environment (http://www.r-project.org).

2014; CLELAND-HUANG; VIERHAUSER, 2018; GÓRSKI; LUKASIEWICZ, 2013), it was also observed that the participants lacked knowledge about the context of safety requirements, making it more challenging to ensure that safety requirements were understood and implemented in safety-critical systems.

To address RQ3, we measured the effort, i.e., the time the participants spent on the task. As displayed in Table 19, participants who received training took longer to complete the task, with the highest mean time of 40 minutes. Effort directly correlates with the task's perceived understandability and difficulty by the participants. The group without training found the task easy but exhibited a low solution accuracy of 25%; therefore, due to a lack of understanding of what needed to be done, their effort was also minimal. Notably, increased effort did not translate into higher accuracy. However, the trained group demonstrated a better understanding of the task.

6.6 Discussion

Considering the increasing complexity and use of software-intensive safety-critical systems, a better understanding of safety requirements is essential to adequately build, and evolve such systems. Existing research has focused on specifying safety requirements in the sense of "what should be done". However, both the state of the art and the state of the practice emphasize that addressing only the "what" is insufficient, especially when these systems are developed using agile development. Besides, most developers have no background or even experience in safety engineering, compromising the understanding of the actual intention of safety requirements and, as a consequence, resulting in inaccurate and inconsistent safety specifications. Therefore, the "how and why it should be done" is also essential. The SCA3DA method provide a guidance on "how should be done" so that resulting just-enough safety-centered architecture solutions can encompass all safety concerns. We believe using this method agile teams could leverage how they deal with software safety requirements and therefore, realize more accurate software safety specifications. Agile teams can also use these solutions to improve team communication and ensure a unique understanding of system criticality and a more accurate interpretation of safety requirements.

6.6.1 Lessons learnt

The process of validating and evaluating the SCA3DA method, via the analysis of safety standards, interviews, and controlled experiments with practitioners has allowed us to gain insights into the practicalities of conceiving the SCA3DA method. We present the main practicalities by means of the following lessons learnt.

- *Decomposition of safety stories*: Safety stories can be specified with different granularity levels. A safety story can correspond to a system safety requirement and be later decomposed into other detection and containment safety story corresponding to software detailed design. These detection and containment safety stories are directly related to a design decision that conforms to the SIL. A key characteristic of the SCA3DA is that explicitly represents the traceability needs to specify failure detection and containment measures in order to obtain compliance with safety standards, and the entire knowledge for correct interpretation of safety requirements.

- *Variability in safety-centered architecture views*: There exists the possibility that different members of the agile team will derive safety-centered architecture views in different ways. The possible variability in the SCA3DA method is a consequence of design decisions made by the agile team. The guidelines help in reaching an agreement on these variations and thus lead to the creation of a common, shared interpretation of safety standards recommendation compliance needs.

6.6.2 Limitations

It is worth highlighting that resulting architectural solutions do not intend to address all details of how software components must be built up-front, but they do address, according to the context, the minimal set of architecturally relevant information for specifying safety-critical failure detection and containment strategies. These solutions can be further refined during agile sprints. We also recognize that the SCA3DA method has a limitation. It is not possible to ensure that decisions made during requirements specification and architecture design are the most correct ones; therefore, we recommend adopting the SCA3DA method together with validation/verification approaches to increase the confidence of the decisions.

6.6.3 Threats to Validity

This section discusses the most important validity threats in qualitative research, according to (WOHLIN *et al.*, 2012).

The first threat *external validity*, refers to the approximate truth of conclusions involving generalizations within different contexts. We conducted our work together with more than 12 participants from three different organizations, in different projects. As our design-science methodology consists of several phases, the participants were involved in different parts of the data collection. Validating the method in several steps with projects in the automotive and medical domains helps to mitigate threats to generalizability. We expect our findings to be transferable to other industries, especially in large-scale agile contexts.

The second threat, to *internal validity*, is related to the noise in the data discovered during the iterative evaluating cycles of our design science methodology; To mitigate bias and preconceptions developed during the study, we critically discussed our findings with other researchers. We also thoroughly evaluated our findings using the methods described in Section 6.3.

The third threat, to *construct validity*, refers to having established correct operational measures for the constructs being studied. In this regard, the effectiveness of the method was assessed by having the subjects perform specification tasks, and the performance was operationalized by the time spent (effort) and the accuracy of the solutions. If a subject understands the real intention of safety requirements, then they should perform better on a specification task, and/or the solution should be more correct. And if the subject has a better understanding of the requirements and the task to be done, the time spent on the task should be lower. The examination of the results showed that all the subjects were able to provide more precise software safety specifications.

The fourth threat, to *researcher bias*, refers to any kind of negative influence of the researcher's knowledge, or assumptions, of the study. The researchers' background, values, and preconceptions influence the conclusions of any qualitative study. A thorough evaluation of our findings helped to mitigate this threat. We included a diverse sample of participants in order to get different perspectives on the topic.

Possible threats to *reliability*, which refers to the consistency of certain measurements; It is expected that replications of the experiment should offer results similar to that presented in Section 6.5. Of course, concrete measured values will differ from those presented here as they are specific to subjects, but the underlying trends and implications should remain unchanged as the analysis discussed in Subsection 8.4.3 showed that the experience of the subject did not influence the results. Furthermore, pilot studies executed showed similar results and supported the discussed findings.

6.6.4 Future works

For future work, the community of safety-critical systems development should invest in initiatives like the SCA3DA, proposing other approaches or even experimenting with this method. Moreover, we intend to apply the SCA3DA method in other safety domains, such as financial and avionics, and collect results to refine it. Hence, we could mature it so this work could ultimately bring the safety team and the agile team closer together and enable them to cooperate in developing systems that are trustworthy in terms of safety.

6.7 Conclusion

Safety requirements play an important role in the life cycle of safety-critical systems. However, there is always a wide gap between their specification and implementation. Furthermore, in the context of agile development, more attention is given to functional requirements than to quality requirements. In the era of the increasing dependence of all of society on complex, large safety-critical systems (e.g., modern cardiac pacemakers, insulin pumps, flight control systems, and car braking systems), it is necessary to ensure that agile artifacts address or consider safety requirements as well. Therefore, approaches that facilitate a more precise specification of agile artifacts so that the developers get a better understanding of the safety software requirement and that this can be traced to the coding. Likewise, the safety engineers can obtain more accurate evidence that the safety requirements have been addressed. The product owner can provide evidence to certifiers and regulators that safety strategies have been implemented and that there is end-to-end traceability of this set of agile artifacts.

This paper has presented the SCA3DA method, which aims is support the agile team in identifying a minimal set of architecturally relevant information for specifying safety-critical failure detection and containment strategies to take the system to a safe state. The method is composed of a safety standard-independent metamodel and provides guidelines for the specification of safety strategies in terms of software and hardware components.

The SCA3DA method has been systematically evaluated by modeling and analyzing safety and agile architectural drivers and in a controlled experiment both with practitioners. The results from the evaluation make us confident in the suitability and effectiveness of the method in an agile context. They indicate that the SCA3DA method can be used for the specification of just enough safety-centered architectural solutions, and practitioners find benefits regarding the understandability of system criticality and a more accurate interpretation of safety requirements.

CHAPTER 7

Conclusions

This chapter concludes this thesis by providing answers to the research questions and discusses the directions for future research on software safety requirements specification in agile contexts.

Chapter 1 states the main research problem, which formed the basis for the work presented in this thesis:

The misunderstood software safety requirements is one key reason for their inaccurate architecture solutions and implementation in agile development.

To address this problem, Chapter 1 defined five research questions to guide us through our research that were answered in Chapters 2 to 6. Table 21 summarizes the research questions, the chapters in which questions were addressed, and the main contributions made regarding the research questions. Each research question is revisited and briefly answered in Section 7.1, while Section 7.2 presents the limitations of this work and the directions for future research.

7.1 Main Contributions

In our studies, we addressed the aforementioned questions, gaining valuable insights, making contributions, and drawing important lessons. Following, we summarize our main contributions and answer the research questions based on our findings about the development of safety-critical systems in agile contexts.

RQ1 - How safety is currently considered and specified in the agile context?

Before investigating the key problems in the development of safety-critical systems in agile contexts, we first needed to obtain a comprehensive understanding of the current state of the research on safety-critical systems development in agile contexts (RQ1). This could help us to build a solid understanding of safety in agile contexts and inspire us to come up with appropriate approaches for the safety requirements specification in agile contexts. To answer

Table 21 – Research Questions and Contributions.

RQ ID	Research Questions	Answered in	Contribution
RQ1	How safety is currently considered and specified in the agile context?	Chapter 2	Overview of the current development of safety-critical systems in agile contexts.
RQ2	How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?	Chapter 3	A case study providing evidence on the misunderstood of safety requirements by agile teams.
RQ3	How to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well?	Chapter 4	(i) The SCA3DA metamodel, a standard-independent metamodel, serves as a guide for deriving just-enough safety-centered architectures. (ii) A mind map of safety standards recommendations regarding safety measures for detecting and containing failures according to safety integrity level.
RQ4	How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?		
RQ4.1	How to measure the understanding of students on safety-critical systems development?	Chapter 5	(i) A controlled experiment providing evidence on the gap in knowledge on topics regarding software architecture and safety requirements by graduate students. (ii) An alternative (quantitative) approach to assess the understandability in the controlled experiment.
RQ4.2	Is the method proposed effective and suitable in agile contexts?	Chapter 6	(i) A SCA3DA method comprised of a metamodel and guidelines to support the software safety requirement specification. (ii) An extension of the safety stories concept. (iii) A controlled experiment providing evidence on the effectiveness of this SCA3DA method.

RQ1, we conducted a systematic mapping study on safety-critical systems development in agile contexts.

The main results of this mapping study are summarized as follows. (1) Both academia and industry paid significant attention to research on safety-critical systems development in agile contexts. (2) Several safety-critical domains have developed safety-critical systems in an agile context. We identify six specific application domains (i.e., automotive, avionics, aerospace, medical, petrochemical, and railway.) and with general domains, i.e., they might be applied in any domain. Most studies are destined to this last category (general domain). Among specific application domains, medical presents more results. (3) Scrum has been the most widely used agile method in safety-critical domains, as well as its combination with other methods (e.g., Scrum + Kanban, Scrum + XP), or even some Scrum ceremonies (e.g., daily meetings, product backlog) integrated with Kanban and TDD. (4) Agile methods have addressed relevant artifacts at all stages of safety-critical systems development. Traceability requirements, Safety Case, Safety Verification, Architecture, Safety Requirements, Safety Plan, Failure modes, and Hazards and risk list received the most attention. Among these, safety verification was the most studied. (5)

Most studies report on activities carried out at the system level. Three main activities identified were: Hazard and Risk Analysis, System Safety Requirements, and Quality Assurance. (7) User story, safety story pattern, and model-based requirement are the most frequently used approaches for specifying System Safety requirements. (8) Traceability, safety requirements specification, and requirements documentation are the key challenges reported by researchers as central issues to adopt agile methods/practices in the safety-critical systems development.

In summary, there are already important, effective contributions exploring agility for the safety-critical domain. The agile methods and their practices have had a positive impact on safety-critical systems development. The main reason for adopting agile is that traditional software development processes are not flexible enough to address the emerging challenges in software development, e.g., high complexity increase, constantly changing requirements, and shorter time to market. However, we observe there is still no consensus among researchers regarding which agile requirements practices are most adequate to address safety requirements.

RQ2 - How can agile teams ensure that agile development artifacts will address not only functional requirements but safety aspects as well?

After having gained an understanding on the state of the art of safety-critical systems development in agile contexts, we seek to characterize the problem based on the state of the practice. To achieve this, we conducted an observational case study, whose aim was to investigate how safety requirements are currently specified or considered by agile teams in the context of agile development of safety-critical automotive systems. The results with regard to safety aspects were similar across the practitioners (system designer/builder, developer, and requirement engineer). All had difficulties understanding and decomposing the requirements, indeed only functional flow was considered. For practitioners who used user story cards to specify/refine the safety requirement, these stories did not address any safety measures. Consequently, it would not be possible to ensure that the original intent of the requirements was implemented. In a general way, this issue is related to task switching, i.e., people are participating in many projects at the same time and often need to jump from one task to another that might be in the context completely different. This directly impacts in the knowledge of the project parts and its development, as well as in the commitment of the professional involved, and may affect the quality of the final product. Moreover, separation of concern and lack of end-to-end knowledge, quality of requirements, such as fuzzy requirements and/or vague requirements, can impair the understanding and translation of the design to its implementation. Due to the critically of safety-critical systems, it is more challenging.

RQ3 - How to provide precise safety-related information to agile teams about those parts of the software that are critical and need to be understood well?

Having delved into the state-of-the-art and state-of-practice of safety-critical systems development in agile contexts, our focus on this question is to understand and identify a minimal set (aligned to the agile principles) of architecturally relevant information for specifying safety-critical failure detection and containment strategies. Existing approaches focus on specifying safety requirements at the system level. The specification of the software safety requirement is a critical part that should potentially shape or bound the design of the software later on along the process, but, in practice that is done in an *ad hoc* way, relying on experience and background of the engineers/architects. The safety standards require that software and hardware safety requirements are derived from the system safety requirements and they should be traceable.

To address these issues, first, we elaborate a mind map of safety standards recommendations concerning failure detection and containment strategies according to safety integrity level. For each safety standard, we defined a checklist with the information recommended to specify safety strategies for failure detection and containment and take the system to a safe state. This checklist provides support to agile teams in making safety-centered design decisions in compliance with safety standards, which is required in safety-critical systems development. Then we developed the SCA3DA metamodel, a standard-independent metamodel, that serves as a guide for deriving just-enough safety-centered architectures. A real-world scenario example was used to illustrate the use of the metamodel. Agile teams using the SCA3DA metamodel can leverage how they deal with software and hardware safety requirements and therefore, realize more accurate architecture solutions. They can also use these solutions to improve team communication and ensure a unique understanding of system criticality and a more accurate interpretation of safety requirements.

RQ4 - How to improve the understanding of software safety requirements by agile teams in safety-critical systems development?

After defining the SCA3DA metamodel, we structured the SCA3DA method considering the findings, feedback, and insights from RQ1 e RQ2, for instance, the systematic mapping revealed that all safety-critical domains already apply agile methods in some activities of safety-critical system development. Thus, the proposed SCA3DA method was developed domain-independent. Besides that, there is no unanimity regarding which agile method/practice is best for SCS development. In this regard, the SCA3DA method can be applied regardless of the agile method/practice used by the project.

Another fundamental point observed is that from a safety perspective, user stories are not sufficient, as safety requirements are specific goals that follow designs defined by standards and have implementation requirements. To address this issue, we extended the concept of a safety story found in the literature, providing an explicit guide to describe failure detection and containment measures according to the safety integrity level associated with the safety goal.

To evaluate the effectiveness of the proposed method, we initially conducted a controlled experiment with students. At this stage, our objective was to validate the protocol and all supplementary materials for the experiment. Most importantly, we aimed to define a quantitative evaluation of participants' understanding of the topic of interest. We structured this quantitative evaluation of understandability based on accuracy, effort, and qualitative understanding.

Subsequently, we conducted a controlled experiment with industry practitioners in the medical domain, following the validated and refined protocol. The results of the controlled experiment demonstrated that the SCA3DA method significantly enhanced participants' understanding of software safety requirements. Furthermore, the method provides guidance on "how it should be done," allowing resulting just-enough safety-centered architectural solutions to encompass all safety concerns. Practitioners found benefits related to the comprehensibility of system criticality and a more accurate interpretation of safety requirements.

7.2 Limitations and Future Directions

This section describes the limitations of this research and concludes by pointing out future directions for software safety specification in agile contexts.

One limitation pertains to SCA3DA method. It is not possible to ensure that decisions made during requirements specification and architecture design are the most correct ones, the resulting safety-centered architecture solutions do not intend to address all details of how software components must be built up-front, but they do address, according to the context, the minimal set of architecturally relevant information for specifying safety-critical failures detection and containment strategies. These solutions can be further refined during agile sprints. Therefore, we recommend adopting the SCA3DA method together with validation/verification approaches to increase the confidence of the decisions.

Another limitation is the relatively small sample size used in the controlled experiment, restricting the use of data for more advanced statistical analyses. A larger sample size, including participants from different domains (e.g., automotive, medical, avionics) and levels of experience (e.g., safety engineer, product owner, architect, devOps, quality analyst, tester) would help increase the reliability of the findings.

Many research ideas emerged during the research process of this thesis. Some of them are related directly to the limitations above and others come from the necessity to confirm the findings in other safety-critical domains. Therefore, interesting directions for the future research are:

- *support the solution adequacy check*, i.e., verifying whether the safety story is properly addressed by safety-centered architectural solution, that can be realized by simulation tools (such as FERAL (KUHR *et al.*, 2013)) or apply machine learning techniques.

- *analysis of the impact of the use of the SCA3DA method in different safety domains*, such as financial and avionics. Hence, we could mature it so this work could ultimately bring the safety team and the agile team closer together and enable them to cooperate in developing systems that are trustworthy in terms of safety.

- *exploration of other quality concerns* (e.g., security, reliability) to be incorporated in the SCA3DA method.
- *exploration of the use of artificial intelligence and machine learning* to support the derivation of safety-centered architecture solutions from SCA3DA metamodel and guidelines.

7.3 Final Remarks

Safety requirements play an important role in the life cycle of safety-critical systems. However, there is always a wide gap between their specification and implementation. Furthermore, in the context of agile development, more attention is given to functional requirements than to quality requirements. In the era of increasing dependence of all of society on complex, large safety-critical systems (e.g., modern cardiac pacemakers, insulin pumps, flight control systems, car braking systems), it is necessary to ensure that agile artifacts address or consider safety requirements as well. Therefore, approaches that facilitate a more precise specification of agile artifacts so that the developers get a better understanding of the safety software requirement and that this can be traced to the coding. Likewise, the safety engineers can obtain more accurate evidence that the safety requirements have been addressed. And the product owner can provide evidence to certifiers and regulators that safety strategies have been implemented and that there is end-to-end traceability of this set of agile artifacts.

In this thesis, a new method is proposed to address the current challenges of software safety specification in agile contexts. Hence, we proposed the SCA3DA method, which aims is support the agile team in identifying a minimal set of architecturally relevant information for specifying safety-critical failure detection and containment strategies to take the system to a safe state. The method is composed of a safety standard-independent metamodel and provides guidelines for the specification of safety strategies in terms of software and hardware components.

The SCA3DA method has been systematically evaluated by modeling and analyzing safety and agile architectural drivers and in a controlled experiment both with practitioners. The results from the evaluation make us confident in the suitability and effectiveness of the method in an agile context. They indicate that the SCA3DA method can be used for the specification of just-enough safety-centered architectural solutions, and practitioners find benefits regarding the understandability of system criticality and a more accurate interpretation of safety requirements.

This thesis contributes toward developing effective communication in an agile context. The solutions derived from the SCA3DA method serve as a guide for communicating safety-related needs to the agile team, thereby promoting cooperation in conflict resolution and decision-making. A major challenge encountered in defining the method is to make the real need (intention) of the safety requirement explicit in the agile context. While existing approaches have focused on "what should be done," this project seeks to introduce the concept of "how and why it should be done". By doing so, understanding becomes clearer, and incorrect assumptions are avoided. We believe that this work provides valuable insights into the importance of improving the understanding of safety requirements specification. Therefore, agile teams can realize more accurate software safety specifications, and also use these solutions to improve team communication and ensure a unique understanding of system criticality and a more accurate interpretation of safety requirements.

References

AAMI TIR45. **- Guidance on the use of AGILE practices in the development of medical device software**. 2012. Citations on pages 26, 35, and 87.

ABDELAZIZ, A. A.; El-Tahir, Y.; OSMAN, R. Adaptive software development for developing safety critical software. In: **ICCNEEE'15**. Khartoum, Sudan: IEEE, 2015. p. 41–46. Citations on pages 15, 23, 28, and 29.

ABDULKHALEQ, A.; WAGNER, S. A controlled experiment for the empirical evaluation of safety analysis techniques for safety-critical software. In: **Intl. Conf. on Evaluation and Assessment in Software Engineering (EASE)**. 2015. p. 1–10. Citation on page 50.

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. Agile software development methods: Review and analysis. **CoRR**, VTT Publication, 2002. Citation on page 15.

ACM/AIS/IEEE-CS. **Joint Task Force for Computing Curricula - The Overview Report**. 2005. Citation on page 60.

AL-SAQQA, S.; SAWALHA, S.; ABDELNABI, H. Agile software development: Methodologies and trends. **Int. J. Interact. Mob. Technol.**, v. 14, p. 246–270, 2020. Citation on page 3.

ALARIFI, A.; ZAROUR, M.; ALOMAR, N.; ALSHAIKH, Z.; ALSALEH, M. Secdep: Software engineering curricula development and evaluation process using swebok. **Information and Software Technology**, v. 74, p. 114–126, 2016. Citations on pages 60, 71, 73, and 75.

ALBUQUERQUE, C.; ANTONINO, P.; NAKAGAWA, E. An Investigation into Agile Methods in Embedded Systems Development. In: **Intl. Conference on Computational Science and Its Applications (ICCSA)**. Salvador, Brazil: Springer, 2012. p. 576–591. Citations on pages 18, 89, and 124.

AMBLER, S. **User Stories: An Agile Introduction**. 2017. Available: <http://www.agilemodeling.com/artifacts/userStory.htm>. Citation on page 38.

ANTONINO, P.; KEULER, T.; GERMANN, N.; CRONAUER, B. A non-invasive approach to trace architecture design, requirements specification and agile artifacts. In: **Australian Software Engineering Conference (ASWEC)**. Milsons Point, Australia: IEEE, 2014. p. 220–229. Citations on pages 15, 43, 52, and 82.

ANTONINO, P. O. **Improving Consistency and Completeness of Safety Requirements Specifications**. Phd Thesis (PhD Thesis) — TU Kaiserslautern, 2016. Citations on pages 2, 38, 39, 50, 78, 81, and 87.

ASTM F2761. **- Medical Devices and Medical Systems - Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) - Part 1: General requirements and conceptual model**. 2013. Citation on page 55.

AVIŽIENIS, A.; LAPRIE, J.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **IEEE Trans. on Dependable and Secure Computing, 2004**, v. 1, p. pp.11–33, 2004. Citations on pages 1, 15, 38, 39, 50, 61, 78, and 80.

BARBARESCHI, M.; BARONE, S.; CARBONE, R.; CASOLA, V. Scrum for safety: an agile methodology for safety-critical software systems. **Software Quality Journal**, Springer, v. 30, n. 4, p. 1067–1088, 2022. Citations on pages 16, 23, and 29.

BARBARESCHI, M.; BARONE, S.; CASOLA, V.; TORCA, S. D.; LOMBARDI, D. Automatic test generation to improve scrum for safety agile methodology. In: **The 18th International Conference on Availability, Reliability and Security**. Benevento, Italy: ACM, New York, NY, USA, 2023. (ARES'23), p. 6 Pages. Citation on page 23.

BASILI, V.; CALDIERA, G.; ROMBACH, H. **The Goal Question Metric (GQM) Approach**. : John Wiley Sons, 2002. Citation on page 91.

BAUER, T. **Enabling Functional Integration Testing by Using Heterogeneous Models**. Phd Thesis (PhD Thesis) — TU Kaiserslautern, 2016. Citation on page 38.

BBCNEWS. **Boeing 737 Max Lion Air crash caused by series of failures**. 2019. Available: <https://www.bbc.com/news/business-50177788>. Citations on pages 4, 50, and 78.

BECK, K.; BEELDE, M.; BENNEKUM A. COCKBURN, A. van; CUNNINGHAM, W.; FOWLER, M. **Manifesto for Agile Software Development**. : Agile Alliance, 2001. Citation on page 2.

BECK, K.; WEST, D. User stories in agile software development. scenarios, stories, use cases: Through the systems development life-cycle. In: ____. : John Wiley Sons, 2014. p. 265–279. Citations on pages 6, 51, and 79.

BECKERS, K.; CôTÉ, I.; FRESE, T.; HATEBUR, D.; HEISEL, M. Systematic derivation of functional safety requirements for automotive systems. In: **Intl. Conference on Computer Safety, Reliability, and Security (SAFECOMP)**. Florence, Italy: Springer, 2014. p. 65–80. Citations on pages 2, 15, 39, 60, 78, and 81.

BECKERS, K.; HEISEL, M.; FRESE, T.; HATEBUR. Deriving safety requirements according to iso 26262 for complex systems: A method applied in the automotive industry. In: ____. **Innovative Produkte und Dienstleistungen in der Mobilität**. : Springer, 2017. p. 211–221. Citation on page 50.

BERGET, C.; MESSER, L.; FORLENZA, G. A Clinical Overview of Insulin Pump Therapy for the Management of Diabetes: Past, Present, and Future of Intensive Therapy. **Diabetes spectrum : a publication of the American Diabetes Association, 2019**, 2019. Citation on page 78.

Carew, D.; Exton, C.; Buckley, J. An Empirical Investigation of the Comprehensibility of Requirements Specifications. In: **Intl. Symposium on Empirical Software Engineering (ISESE)**. 2005. p. 1–10. Citations on pages 65 and 95.

CARVER, J.; JACCHERI, L.; MORASCA, S.; SHULL, F. Issues in using students in empirical studies in software engineering education. In: **Works. on Enterprise Networking and Computing in Healthcare Industry**. Sydney, Australia: , 2003. p. 239–249. Citation on page 72.

CAWLEY, O.; WANG, X.; RICHARDSON, I. Lean/Agile Software Development Methodologies in Regulated Environments – State of the Art. In: **Intl. Conference on Lean Enterprise Software and Systems (LESS)**. Helsinki, Finland: Springer, 2010. p. 31–36. Citations on pages 16, 38, and 89.

CENELEC - EN 50128. **Railway Applications - Communication, Signalling And Processing Systems - Software For Railway Control And Protection Systems**. : European Committee for Electrotech. Standardization, 2011. Citations on pages 33, 60, and 87.

CLELAND-HUANG, J.; VIERHAUSER, M. Discovering, analyzing, and managing safety stories in agile projects. In: **Intl. Requirements Engineering Conference (RE)**. Banff, Canada: IEEE, 2018. p. 262–273. Citations on pages 23, 52, 53, 79, 82, 85, 86, 89, and 99.

COE, D.; KULICK, J. A model-based agile process for do-178c certification. In: **SERP'13**. Las Vegas, USA: Inspec, 2013. Citations on pages 22 and 30.

COMAR, C.; GASPERONI, F.; RUIZ, J. Open-do: An open-source initiative for the development of safety-critical software. In: **SaRS'09**. London, UK: IET, 2009. p. 1–5. Citations on pages 14 and 20.

Critical Software. **The Role of Software in Medical Device Failures**. 2020. Available: <https://www.criticalsoftware.com/multimedia/critical/pt/xvhdnSb3o-CSW_White_Paper_The_Role_of_Software_in_Medical_Device_Failures.pdf>. Citations on pages 5 and 78.

De Lucia, A.; QUSEF, A. Requirements engineering in agile software development. **Journal of Emerging Technologies in Web Intelligence, 2010**, 2010. Citations on pages 38, 50, and 78.

DEMISSIE, S.; KEENAN, F.; MCCAFFERY, F. Investigating the suitability of using agile for medical embedded software development. In: **SPICE'16**. Dublin, Ireland: Springer, 2016. p. 409–416. Citation on page 22.

DEMISSIE, S.; KEENAN, F.; ÖZCAN-TOP, Ö.; MCCAFFERY, F. Agile Usage in Embedded Software Development in Safety Critical Domain - A Systematic Review. In: **Intl. Conference on Software Process Improvement and Capability Determination**. : Springer, 2018. p. 316–326. Citation on page 79.

DIEBOLD, P.; DAHLEM, M. Agile practices in practice: A mapping study. In: **EASE '14**. London, UK: ACM, 2014. p. 30:1–30:10. Citation on page 15.

DU, J.; WANG, J.; FENG, X. A safety requirement elicitation technique of safety-critical system based on scenario. In: **ICIC**. Taiyuan, China: Spriger, 2014. p. 127–136. Citation on page 14.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Inf. Softw. Technol.**, 2008. Citations on pages 18, 124, 125, and 127.

ELGHARIANI, K.; KAMA, N. Review on agile requirements engineering challenges. In: **ICCOINS'16**. Kuala Lumpur, Malaysia: IEEE, 2016. p. 507–512. Citations on pages 18 and 124.

ELOFF, J.; BELLA, M. **Software Failures: An Overview**. : Springer, 2017. 7-24 p. Citations on pages 78 and 79.

FALESSI, D.; JURISTO, N.; WOHLIN, C.; AL. et. Empirical software engineering experts on the use of students and professionals in experiments. **Empirical Software Engineering**, v. 23, p. 452–489, 2017. Citation on page 72.

FIRESMITH, D. Engineering safety requirements, safety constraints, and safety-critical requirements. **Journal of Object Technology, 2004**, v. 3, p. pp. 27–42, 2004. Citations on pages 5, 50, 86, and 87.

FITZGERALD, B.; STOL, K.; O'SULLIVAN, R.; O'BRIEN, D. Scaling agile methods to regulated environments: An industry case study. In: **ICSE '13**. San Francisco, USA: IEEE, 2013. p. 863–872. Citations on pages 22, 27, and 30.

FOWLER, J. **Survey Research Methods**. : Sage Publications, 2013. Citation on page 92.

GALLINA, B.; MURAM, F. U.; ARDILA, J. P. C. Compliance of agilized (software) development processes with safety standards: A vision. In: **XP '18**. Porto, Portugal: ACM, 2018. p. 1–6. Citation on page 23.

GARG, A. Agile software development. **DRDO Science Spectrum**, 2009. Citation on page 16.

GAROUSI, V.; GIRAY, G.; TUZUN, E. Understanding the knowledge gaps of software engineers: An empirical analysis based on swebok. **ACM Transactions on Computing Education**, v. 20, p. 1–33, 2019. Citations on pages 60, 71, 73, and 75.

GARY, K.; ENQUOBAHRIE, A.; IBANEZ, L.; CHENG, P.; YANIV, Z.; CLEARY, K.; KOKOORI, S.; MUFFIH, B.; HEIDENREICH, J. Agile methods for open source safety-critical software. **Softw. Practice Experience**, 2011. Citations on pages 14, 17, 21, and 124.

GE, X.; PAIGE, R. F.; MCDERMID, J. A. An iterative approach for development of safety-critical software and safety arguments. In: **AGILE '10**. Florida, USA: IEEE, 2010. p. 35–43. Citations on pages 14, 16, 21, 28, 30, 31, and 34.

GOLDMAN, J. **Medical Devices and Medical Systems - Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) - Part 1: General requirements and conceptual model**. 2009. Citation on page 55.

GORSKI, J.; LUKASIEWICZ, K. Assessment of risks introduce to safety critical software by agile practices. **Computer Science**, 2012. Citations on pages 22 and 43.

GÓRSKI, J.; LUKASIEWICZ, K. Towards agile development of critical software. In: **Intl. Workshop on Software Engineering for Resilient Systems (SERENE)**. 2013. p. 48–55. Citations on pages 21, 52, 82, and 99.

GUO, Z.; HIRSCHMANN, C. An integrated process for developing safety-critical systems using agile development methods. In: **ICSEA'12**. Lisbon, Portugal: IARIA, 2012. Citations on pages 22 and 30.

HANSSEN, G.; STåLHANE, T.; MYKLEBUST, T. Adapting safescrum®. in: Safescrum®–agile development of safety-critical software. In: ____. 2018. p. 153–165. Citations on pages 16, 22, 26, 28, 30, 52, and 82.

HANSSEN, G. K.; HAUGSET, B.; STÅLHANE, T.; MYKLEBUST, T.; KULBRANDSTAD, I. Quality assurance in scrum applied to safety critical software. In: **XP'16**. Edinburgh, UK: Springer, Cham, 2016. p. 92–103. Citations on pages 21, 28, 30, and 34.

HANSSEN, G. K.; WEDZINGA, G.; STUIP, M. An assessment of avionics software development practice: Justifications for an agile development process. In: **XP'17**. Cologne, Deutschland: Springer, 2017. p. 217–231. Citations on pages 21 and 30.

HATCLIFF, J.; WASSYNG, A.; KELLY, T.; COMAR, C.; JONES, P. Certifiably safe software-dependent systems: Challenges and directions. In: **Future of Software Engineering Proceedings (FOSE)**. Hyderabad, India: ACM, 2014. p. 182–200. Citations on pages 2, 38, 43, 50, 52, 60, 61, 62, 75, 78, 81, 82, 87, and 99.

HEEAGER, L. T. **How Can Agile and Documentation-driven Methods Be Meshed in Practice?** Rome, Italy: Springer, 2014. 62–77 p. Citations on pages 22 and 89.

HEEAGER, L. T.; NIELSEN, P. A. **Agility in Development of Safety-Critical Software: A Conceptual Model**. 2017. Citation on page 89.

HEEAGER, L. T.; NIELSEN, P. A. A conceptual model of agile software development in a safety-critical context: A systematic literature review. **Information and Software Technology, 2018**, v. 103, p. 22–39, 2018. Citations on pages 16 and 82.

HEINEMANN, L.; FLEMING, G.; PETRIE, J.; HOLL, R.; BERGENSTAL, R.; PETERS, A. Insulin pump risks and benefits: A clinical appraisal of pump safety standards, adverse event reporting, and research needs a joint statement of the european association for the study of diabetes and the american diabetes association diabetes technology working group. **Diabetes Care**, p. 716–722, 2015. Citations on pages 50 and 60.

HEPHER, T.; MORRIS, S. **Airbus shares fall after A400M military plane crash - Available at: https://www.reuters.com/article/uk-spain-crash-airbusshares/airbus-shares-fall-after-a400m-military-plane-crash-idUKKBN0NW0HJ20150511**. : Reuters, 2015. Citations on pages 4, 50, and 78.

HOBBS, C. **Embedded software development for safety-critical systems**. : CRC Press, 2019. Citation on page 60.

IEC 61508. **Functional safety of electrical/ electronic/ programmable electronic safety-related systems**. 2010. Citations on pages 1, 6, 10, 14, 51, 53, 61, 62, 73, 78, 80, 84, 87, 88, and 89.

IEC 61511. **Functional safety - Safety instrumented systems for the process industry sector**. 2021. Citations on pages 15 and 81.

IEC 62304. **(ANSI/AAMI/IEC 62304): Medical device Software - Software Life Cycle Processes, Assoc. Advancement Medical Instrumentation**. 2006. Citations on pages 10, 15, 35, 38, 50, 53, 60, 84, 87, and 88.

ISAAC, M.; WAKABAYASHI, D.; CONGER, K. **Uber's Vision of Self-Driving Cars Begins to Blur - Available at: https://www.nytimes.com/2018/08/19/technology/uber-self-driving-cars.html?rref=collection%2Ftimestopic%2FAccidents and Safety**. : The New York Times, 2018. Citations on pages 4, 50, and 78.

ISLAM, G.; STORER, T. A case study of agile software development for safety-critical systems projects. **Reliability Engineering System Safety**, v. 200, p. 106954, 2020. Citations on pages 23, 27, and 28.

ISO26262. **International Organization for Standardization - Road Vehicles - Functional Safety**. 2018. Citations on pages 2, 5, 10, 15, 51, 53, 60, 62, 78, 81, 87, 88, and 89.

JEDLITSCHKA, A.; CIOLKOWSKI, M.; PFAHL, D. Reporting Experiments in Software Engineering. In: Guide to advanced empirical software engineering. In: ____. : Springer, 2008. p. 201–228. Citations on pages 10, 62, and 92.

JENSEN, C. **Nissan Recalls 990,000 Vehicles for Air Bag Malfunction - Available at: https://www.nytimes.com/2014/03/27/automobiles/nissan-recalls-990000-cars-and-trucks-for-air-bag-malfunction.html**. : The New York Times, 2014. Citations on pages 4, 50, and 78.

JONSSON, H.; LARSSON, S.; PUNNEKKAT, S. Agile practices in regulated railway software development. In: **ISSRE Workshops 2012**. Dallas, TX, USA: IEEE, 2012. p. 355–360. Citations on pages 14, 21, 30, and 34.

KAPPE, B. **Agile in an FDA Regulated Environment - Pathfinder White Paper**. 2013. Available: <http://www.himss.org/agile-fda-regulated-environment-pathfinder-white-paper>. Citation on page 35.

KASAULI, R.; KNAUSS, E.; HORKOFF, J.; LIEBEL, G.; de Oliveira Neto, F. Requirements engineering challenges and practices in large-scale agile system development. **Journal of Systems and Software**, v. 172, 2021. Citations on pages 2, 25, and 78.

KASAULI, R.; KNAUSS, E.; KANAGWA, B.; NILSSON, A.; CALIKLI, G. Safety-critical systems and agile development: A mapping study. In: **Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. : IEEE, 2018. p. 470–477. Citations on pages 16 and 82.

KATUMBA, B.; KNAUSS, E. Agile development in automotive software development: Challenges and opportunities. In: **PROFES'14**. Helsinki, Finland: Springer, 2014. p. 33–47. Citations on pages 22 and 27.

KITCHENHAM, B.; BUDGEN, D.; BRERETON, P. **Evidence-based software engineering and systematic reviews**. : CRC press, 2015. Citations on pages 18 and 35.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. 2007. Citations on pages 36, 46, 52, 81, 83, 123, and 127.

KITCHENHAM, B.; PRETORIUS, R.; BUDGEN, D.; Pearl Brereton, O.; TURNER, M.; NIAZI, M.; LINKMAN, S. Systematic literature reviews in software engineering – a tertiary study. **Information and Software Technology**, n. 8, p. 792 – 805, 2010. Citation on page 125.

KUHR, T.; FORSTER, T.; BRAUN, T.; GOTZHEIN, R. Feral — framework for simulator coupling on requirements and architecture level. In: **Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign**. USA: IEEE Computer Society, 2013. (MEMOCODE '13), p. 11–22. Citation on page 107.

LAYTON, M.; OSTERMILLER, S.; KYNASTON, D. **Scrum For Dummies**. : Wiley, 2022. Citation on page 3.

LEITE, A. An approach to support the specification of agile artifacts in the development of safety-critical systems. In: **Intl. Requirements Engineering Conference (RE)**. 2017. p. 526–531. Citations on pages 12 and 84.

LEITE, A.; ANTONINO, P.; NAKAGAWA, E. From Safety Requirements to Just-Enough Safety-Centered Architectural Solutions in Agile Contexts. In: **Brazilian Symposium on Software Engineering (SBES)**. Natal, Brazil: ACM, 2020. p. 766–771. Citations on pages 15, 12, 85, and 86.

LEITE, A.; LAGOA, L.; LOPES, S.; BRAGA, R.; ANTONINO, P. O.; NAKAGAWA, E. Y. An investigation of knowledge gaps of graduate students regarding safety-critical systems development: A controlled experiment. **IEEE Transactions on Education**, v. 65, n. 1, p. 64–72, 2022. Citations on pages 12 and 92.

LEITE, F.; ADLER, R.; FETH, P. Safety assurance for autonomous and collaborative medical cyber-physical systems. In: **SAFECOMP'17**. Trento, Italy: Springer, 2017. p. 237–248. Citations on pages 6 and 55.

LEVESON, N. **Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)**. : The MIT Press, 2012. Citations on pages 1, 14, 39, 60, 62, 80, and 87.

LI, N.; GUO, J.; LEI, J.; LI, Y.; RAO, C.; CAO, Y. Towards agile testing for railway safety-critical software. In: **XP Workshops'16**. Edinburgh, UK: ACM, 2016. p. 1–4. Citation on page 22.

LIAN, S. Sw process tailoring practice in medical device industry. In: **ICSSP'14**. New York, USA: ACM, 2014. p. 193–194. Citation on page 14.

LO, S. C.; CHEN, N. IEEE 42010 and Agile Process- Create Architecture Description through Agile Architecture Framework. In: **Intl. Conference on Software Engineering Research and Practice (SERP)**. Las Vegas, USA: CSREA Press, 2017. p. 149–155. Citations on pages 6, 51, and 79.

LUCASSEN, G.; DALPIAZ, F.; Van Der Werf, J.; BRINKKEMPER, S. Forging high-quality User Stories: Towards a discipline for Agile Requirements. In: **RE'15**. Ottawa, Canada: IEEE, 2015. p. 126–135. Citation on page 39.

LUKASIEWICZ, K.; GÓRSKI, J. Agilesafe - a method of introducing agile practices into safety-critical software development processes. In: **FedCSIS'16**. Gdańsk, Poland: IEEE, 2016. p. 1549–1552. Citations on pages 16 and 21.

MäDER, P.; GOTEL, O. Towards automated traceability maintenance. **Journal of Systems and Software**, 2012. Citations on pages 2 and 87.

MARQUES, J. C.; YELISETTY, S. M. H.; CUNHA, A. M. D.; DIAS, L. A. V. Card-rm: A reference model for airborne software. In: **ITNG'13**. New York, USA: IEEE, 2013. p. 273–279. Citations on pages 22 and 30.

MARSDEN, J.; WINDISCH, A.; MAYO, R.; GROSSI, J.; VILLERMIN, J.; FABRE, L.; AVENTINI, C. Ed-12c/do-178c vs. agile manifesto – a solution to agile development of certifiable avionics systems. In: **ERTS 18**. Toulouse, France: INRIA, 2018. Citation on page 23.

MARTINS, L.; GORSCHEK, T. Requirements engineering for safety-critical systems: A systematic literature review. **Information and Software Technology, 2016**, 2016. Citations on pages 42, 44, 50, 52, 78, and 82.

MAURYA, A.; KUMAR, D. Reliability of safety-critical systems: A state-of-the-art review. **Quality and Reliability Engineering, 2020**, v. 36, p. 2547–2568, 2020. Citations on pages 60, 62, and 80.

MAVIN, A.; WILKINSON, P.; HARWOOD, A.; NOVAK, M. Easy approach to requirements syntax (ears). In: **17th IEEE International Requirements Engineering Conference**. : IEEE, 2009. p. 317–322. Citation on page 33.

MCBRIDE, T.; LEPMETS, M. Quality assurance in agile safety-critical systems development. In: **QUATIC**. Lisbon, Portugal: IEEE, 2016. p. 44–51. Citations on pages 21, 28, 30, and 34.

MCCAFFERY, F.; LEPMETS, M.; TREKTERE, K.; TOP, O. Özcan; PIKKARAINEN, M. Agile medical device software development: Introducing agile practices into mdevspice. **International Journal On Advances in Life Sciences**, 2016. Citations on pages 22 and 30.

MCDERMID, J.; GE, X.; PAIGE, R. An iterative approach for development of safety-critical software and safety arguments. In: **AGILE Conference**. Orlando, USA: IEEE, 2010. p. 35–43. Citations on pages 17 and 124.

MCHUGH, M.; CAWLEY, O.; MCCAFFERY, F.; RICHARDSON, I.; WANG, X. An agile v-model for medical device software development to overcome the challenges with plan-driven software development lifecycles. In: **Intl. Workshop on Software Engineering in Health Care (SEHC)**. 2013. p. 12–19. Citations on pages 16, 21, 26, 30, 50, and 89.

MCHUGH, M.; MCCAFFERY, F.; CASEY, V. Adopting agile practices when developing software for use in the medical domain. **Journal of Software: Evolution and Process**, 2014. Citation on page 20.

MEHTA, K.; SOOD, V. M. Agile software development in the digital world – trends and challenges. In: ____. **Agile Software Development**. : Wiley, Ltd, 2023. p. 1–22. Citation on page 2.

MULLER, M.; BESEMER, F. **Perspectives of Agile Product Development Processes in the Automotive Ecosystem**. 2018. Citations on pages 50 and 89.

MYKLEBUST, T.; BRAINS, R.; HANSSEN, G. The agile hazard log approach. In: **ESREL'17**. Portoroz, Slovenia: ESRA, 2017. Citations on pages 22 and 28.

MYKLEBUST, T.; ERIKSEN, J.; HELLANDSVIK, A.; HANSSEN, G. The agile fmea approach. In: **SSS'18**. York, UK: SCSC, 2018. p. 1–17. Citation on page 23.

MYKLEBUST, T.; STALHANE, T. Safety stories–a new concept in agile development. In: **Fast Abstracts at International Conference on Computer Safety, Reliability, and Security**. 2016. (SAFECOMP'2016). Citation on page 86.

MYKLEBUST, T.; STåLHANE, T.; HANSSEN, G. Use of agile practices when developing safety-critical software. In: **ISSC 2016**. Florida, USA: Springer, 2016. Citations on pages 23 and 29.

MYKLEBUST, T.; STåLHANE, T.; LYNGBY, N. An agile development process for petrochemical safety conformant software. In: **RAMS 2016**. Tucson, USA: IEEE, 2016. p. 1–6. Citation on page 22.

MYKLEBUST, T.; STåLHANE, T.; LYNGBY, N. The agile safety plan. In: **PSAM 13**. Seoul, Korea: , 2016. Citations on pages 22 and 30.

Nascimento, A.; Vismari, L.; Molina, C.; Cugnasca, P.; Camargo, J.; Almeida, J.; Inam, R.; Fersman, E.; Marquezini, M.; Hata, A. A systematic literature review about the impact of artificial intelligence on autonomous vehicle safety. **IEEE Transaction on Intelligent Transportation System, 2020**, v. 21, p. 4928–4946, 2020. Citations on pages 62, 78, and 80.

NOTANDER, J.; HÖST, M.; RUNESON, P. Challenges in flexible safety-critical software development - an industrial qualitative survey. In: **Intl. Conference on Product Focused Software Process Improvement (PROFES)**. 2013. p. 283–297. Citations on pages 21, 39, and 42.

ÖZçELIK, O.; ALTILAR, D. Test-driven approach for safety-critical software development. **Journal of Software**, 2015. Citations on pages 23 and 26.

PAGÈS, F.; BORGERS, E.; ILKIEWICZ, M.; ATTWOOD, K. **Open Platform for EvolutioNary Certification of Safety-Critical Systems Baseline for the Common Certification Language**. 2012. Citation on page 38.

PAGÈS, F.; BORGERS, E.; ILKIEWICZ, M.; ATTWOOD, K. **Open Platform for EvolutioNary Certification Of Safety - critical Systems Baseline for the Common Certification Language**. 2012. Citation on page 14.

PAIGE, R.; CHARALAMBOUS, R.; GE, X.; BROOKE, P. Towards agile engineering of high-integrity systems. In: **Intl. Conference on Computer Safety, Reliability, and Security (SAFECOMP)**. : Springer, 2008. p. 30–43. Citations on pages 43, 52, and 82.

PAIGE, R. F.; GALLOWAY, A.; CHARALAMBOUS, R.; GE, X.; BROOKE, P. J. High-integrity agile processes for the development of safety critical software. **Int. J. Crit. Comput.-Based Syst.**, p. 181–216, 2011. Citations on pages 16, 17, 21, and 124.

PARK, K. **Singapore Train Collision Leaves 25 People Injured - Available at: https://www.bloomberg.com/news/articles/2017-11-15/singapore-train-collides-with-stationary-one-straits-times-says-ja0gqxj6**. : Bloomberg, 2017. Citations on pages 4, 50, and 78.

PARSIFAL. **Support to Perform Systematic Literature Reviews**. 2020. Available: <https://parsif.al/>. Citations on pages 18 and 125.

PEFFERS, K.; TUUNANEN, T.; ROTHENBERGER, M.; CHATTERJEE, S. A design science research methodology for information systems research. **Journal of Management Information Systems**, M. E. Sharpe, Inc., v. 24, n. 3, p. 45–77, 2007. Citations on pages 15, 8, and 9.

PERFORCE. **State of Medical Device Development in 2019**. 2019. Citations on pages 4, 50, and 79.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering. **Information and software technology, 2015**, 2015. Citations on pages 8, 14, 16, 52, 81, 83, and 123.

RITTER, N. Understanding a widely misunderstood statistic: Cronbach's. In: **SERA**. Montreal, Canada: , 2010. p. 1–17. Citations on pages 66 and 95.

ROBSON, C.; MCCARTAN, K. **Real World Research**. : Wiley, 2015. Citations on pages 11 and 85.

ROTTIER, P. A.; RODRIGUES, V. Agile development in a medical device company. In: **Agile Conference'08**. Toronto, Canada: IEEE, 2008. p. 218–223. Citations on pages 22, 27, and 30.

RTCA/DO-178. **Software Considerations in Airborne Systems and Equipment Certification, RTCA**. 2012. Citations on pages 2, 10, 31, 38, 50, 62, 80, 81, 84, and 87.

RUSCIO, D.; MALAVOLTA, I.; PELLICCIONE, P. The role of parts in the system behaviour. In: **Intl. Workshop on Software Engineering for Resilient Systems (SERENE)**. 2014. p. 24–39. Citations on pages 14 and 50.

SANKHE, P.; MATHUR, S.; REHMAN, T. B.; DIXIT, M. Review of an agile software development methodology with scrum extreme programming. In: **Intnl Conference on Current Development in Engineering and Technology (CCET)**. 2022. p. 1–6. Citation on page 2.

SCHEFFE, H. **The analysis of variance**. : John Wiley & Sons, 1999. Citations on pages 70 and 96.

SCHIDEK, A.; TIMINGER, H. Agilization of technical development processes for medical devices. In: IEEE. **2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference**. 2022. p. 1–9. Citations on pages 4 and 78.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game**. 2017. Citations on pages 40 and 89.

SCHWEIZER, C.; YAZER, C.; STRUBE, D.; KNEISEL, F.; BAJAN, M. **Agile Automotive - State of Practice 2021**. 2021. Citations on pages 15, 3, 4, 25, 34, and 79.

SCRUM.ORG. **The Scrum.Org - The Home of Scrum**. 2023. Citation on page 3.

SMITH, D. J.; SIMPSON, K. G. L. **Safety Critical Systems Handbook**. : Elsevier Ltd., 2011. Citation on page 15.

STÅLHANE, T.; HANSSEN, G. K.; MYKLEBUST, T.; HAUGSET, B. Agile change impact analysis of safety critical software. In: **SAFECOMP'14**. Florence, Italy: Springer, 2014. p. 444–454. Citation on page 21.

STÅLHANE, T.; MYKLEBUST, T. The agile safety case. In: **SAFECOMP'16**. Trondheim, Norway: Springer, 2016. p. 5–16. Citation on page 21.

STÅLHANE, T.; MYKLEBUST, T. Early safety analysis. In: **Proceedings of the Scientific Workshop XP2016**. 2016. p. 1–6. Citations on pages 21, 28, 30, 43, 52, and 82.

STÅLHANE, T.; MYKLEBUST, T.; HANSSEN, G. The application of safe scrum to IEC 61508 certifiable software. In: **PSAM 11**. Helsinki, Finland: Curran Associates, Inc., 2012. p. 6052 – 6061. Citations on pages 15, 17, 39, 40, 43, 44, 45, and 124.

STALHANE, T.; MYKLEBUST, T.; HANSSEN, G. Safety standards and scrum - a synopsis of three standards. In: **PSAM'13**. Seoul,Korea: IAEA INIS, 2013. Citation on page 44.

STEPHENSON, Z.; MCDERMID, J.; WARD, A. Health modelling for agility in safety-critical systems development. In: **Conference on System Safety**. London, UK: IET, 2006. p. 260–265. Citations on pages 21 and 30.

STaLHANE, T.; MYKLEBUST, T. Agile safety analysis. **SIGSOFT Softw. Eng. Notes**, 2016. Citations on pages 23 and 26.

STaLHANE, T.; MYKLEBUST, T. Hazard Stories, HazId and Safety Stories in SafeScrum. In: **Intl. Conference on Agile Software Development: Companion**. 2018. p. 1–7. Citations on pages 23 and 89.

TAROMIRAD, M.; PAIGE, R. F. Agile requirements traceability using domain-specific modelling languages. In: . New York, USA: ACM, 2012. (XM '12), p. 45–50. Citation on page 44.

TGA Health Safety Regulation. **Actual and potential harm caused by medical software: A rapid literature review of safety and performance issues**. 2020. Citation on page 5.

THORSEN, L. **Extreme Programming in safety-related systems**. 2002. Citations on pages 43, 52, and 82.

TOMMILA, T.; ALANEN, J. Conceptual model for safety requirements specification and management in nuclear power plants. **VTT-VTT Technology, 2015**, 2015. Citation on page 50.

TRAPP, M.; SCHNEIDER, D. Safety assurance of open adaptive systems - a survey. In: **Models@run.time**. : Springer, 2011. p. 279–318. Citations on pages 15 and 63.

TRAPP, M.; SCHNEIDER, D. Safety assurance of open adaptive systems – a survey. In: ____. : Springer, 2014. Citation on page 87.

TREMBLAY, M.; HEVNER, A.; BERNDT, D. The Use of Focus Groups in Design Science Research. In: Design research in information systems. In: ____. 2010. p. 121–143. Citation on page 92.

VANDERLEEST, S. H.; BUTER, A. Escape the waterfall: Agile for aerospace. In: **DASC'09**. Florida, USA: IEEE, 2009. p. 3–16. Citations on pages 17, 21, 30, and 124.

VERSIONONE. **16th Annual State of Agile Development Survey**. 2022. Citation on page 3.

VÖST, S.; WAGNER, S. Keeping continuous deliveries safe. In: **ICSE-C'16**. Buenos Aires, Argentina: IEEE, 2016. p. 259–261. Citation on page 23.

VUORI, M. **Agile Development of Safety-Critical Software**. : Tampere University of Technology, 2011. Citations on pages 6, 14, 38, 40, 43, 51, 52, and 82.

WANG, Y.; BOGICEVIC, I.; WAGNER, S. A study of safety documentation in a scrum development process. In: **Proceedings of the XP2017 Scientific Workshops**. 2017. p. 22:1–22:5. Citations on pages 16, 22, 29, 30, 31, 52, 82, and 86.

WANG, Y.; DEGUTIS, D.; WAGNER, S. Speed up BDD for Safety Verification in Agile Development: A Partially Replicated Controlled Experiment. In: **Intl. Conference on Agile Software Development: Companion**. 2018. p. 1–8. Citations on pages 23, 26, 27, and 89.

WANG, Y.; RAMADANI, J.; WAGNER, S. An exploratory study on applying a scrum development process for safety-critical systems. In: **PROFES'17**. Innsbruck, Austria: Springer, 2017. p. 324–340. Citations on pages 22, 27, and 30.

WANGENHEIM, C.; SILVA, D. **A Survey on the Relevance of Topics in Computer Science Education**. 2009. Citation on page 60.

WARG, F.; GASSILEWSKI, M.; TRYGGVESSON, J.; IZOSIMOV, V.; WERNEMAN, A.; JOHANSSON, R. Defining autonomous functions using iterative hazard analysis and requirements refinement. In: **Computer Safety, Reliability, and Security**. Porto, Portugal: Springer International Publishing, 2016. p. 1–6. Citation on page 23.

WEBER, S. **Agile in Automotive – State of Practice 2015**. 2015. Citations on pages 14, 35, and 89.

WIERINGA, R. **Design Science Methodology for Information Systems and Software Engineering**. : Springer Berlin Heidelberg, 2014. Citations on pages 15, 8, 9, 51, 53, 79, 82, and 83.

WIERINGA, R.; MAIDEN, N. A. M.; MEAD, N. R.; ROLLAND, C. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. In: **RE'06**. Minnesota, USA: Springer, 2006. p. 102–107. Citation on page 26.

WILS, A.; BAELEN, S. V.; HOLVOET, T.; VLAMINCK, K. D. Agility in the avionics software world. In: **XP'06**. Oulu, Finland: Springer, 2006. p. 123–132. Citations on pages 22 and 35.

WIZEMANN, T. **Public Health Effectiveness of the FDA 510 (k) Clearance Process: Measuring Postmarket Performance and Other Select Topics**. 2010. Citation on page 50.

WOHLIN, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In: **Intl. Conference on Evaluation and Assessment in Software Engineering (EASE)**. London, England: ACM, 2014. p. 1–10. Citations on pages 8, 20, and 35.

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. **Experimentation in Software Engineering**. : Springer, 2012. Citations on pages 10, 35, 61, 62, 70, 72, 74, 75, 92, 98, 100, and 126.

WOLFF, S. Scrum goes formal: Agile methods for safety-critical systems. In: **FormSERA '12**. Zurich, Switzerland: IEEE, 2012. p. 23–29. Citations on pages 21 and 30.

WU, W.; KELLY, T. Deriving safety requirements as part of system architecture definition. In: **Intl. System Safety Conference, System Safety Society (ISSC)**. Dublin, Ireland: IET, 2006. Citation on page 50.

ZELLER, M. Towards continuous safety assessment in context of devops. In: **Computer Safety, Reliability, and Security**. : Springer, 2021. (SAFECOMP, 2021), p. 145–157. Citation on page 2.

APPENDIX A

Systematic mapping protocol

In this appendix, we present the planning and execution of the Systematic Mapping (SM) conducted to identify the state of the art of how agile methods have been applied in the safety-critical systems development. The research method adopted in this study follows the guidelines proposed by (KITCHENHAM; CHARTERS, 2007) and (PETERSEN *et al.*, 2015), which is a technique for finding and summarizing available pieces of evidence on a particular research topic. This Systematic Mapping is divided in three phases: (i) planning: in this phase, the research objectives and the systematic review protocol are defined. (ii) conduction: during this phase, primary studies are identified, selected, and evaluated according to the inclusion and exclusion criteria. For each selected study, data are extracted and synthesized; and (iii) reporting: in this phase, a final report is organized and presented. Thus, this appendix describes the first and second phases in more details.

A.1 Planning

During the planning stage, the objectives and the protocol of the SM are defined. This protocol contains: (i) research questions; (ii) search string and databases; (iii) selection criteria (i.e., inclusion and exclusion criteria); (iv) procedures for the studies selection; and (v) data extraction and synthesis method.

A.1.1 Research Objectives

The main goal of this study is to systematically map, review and synthesize the state-of-the-art for providing a panorama of how safety (measures, requirements, aspects) has been addressed throughout agile development.

Keyword	**Synonym**
Agile	Agil, agility, "extreme programming“, xp, "feature driven“, scrum, crystal, kaban, lean, ”iterative development”, adaptative, evolucionary
Safety	
Critical	critical system, complex system, critical software, critical application, critical equipment, critical device

A.1.2 Research Questions

Aiming to reach the research objectives, three research questions (RQs) were created.

RQ1 : Mapping of sources by contribution and research facets:

RQ1.1 : *How many studies present methods, techniques, tools, models, metrics, or processes for addressing safety in agile development of SCS?*

RQ1.2 : *What type of research methods have been used in the studies in this area?*

RQ2 : According to safety lifecycle, which phases, activities of these systems development are being carried out in an agile context?

RQ3 : What safety work products (artifacts) are considered throughout agile development?

A.1.3 Search Strategy

To establish the search strategy for answering the research questions, we developed the search string by specifying the main terms of the phenomena under investigation. A number of pilot searches were performed to refine the keywords in the search string using a control group for our SM. We used five previously known studies ((MCDERMID *et al.*, 2010) and (PAIGE *et al.*, 2011), (GARY *et al.*, 2011), (VANDERLEEST; BUTER, 2009), and (STÅLHANE *et al.*, 2012)) that were suggested by a knowledge expert. They were our baseline to check whether our search string was properly defined, i.e., if our string was able to find these studies in the publication databases. We removed terms whose inclusion did not yield additional papers in the automatic searches. After several iterations, we settled on the following search string. This search string, which is expressed as a conjunction of three parts, was used to search within keywords, title and abstract.

The first part of the search string relates to agile methods. The synonymous were extracted from of systematic literature reviews, such as: (ELGHARIANI; KAMA, 2016), (ALBUQUERQUE *et al.*, 2012) and (DYBÅ; DINGSØYR, 2008) . The second part of the search string captures keywords related to criticality of applications (i.e. critical software, critical system, critical equipment, critical application). The third and final part of the search string concerns safety (i.e. safety certification, safety requirements, safety case, safety assurance).

We performed an automatic search performed on a set of five recognized scientific database libraries, namely ACM Digital Library, IEEE Xplore, ScienceDirect, Scopus, and Web of Science to find publications relevant to the scope of the review. According to (DYBÅ;

DINGSØYR, 2008) and (KITCHENHAM *et al.*, 2010) , these publication databases are the most relevant sources in the computer science area. The search string presented previously was used to search within keywords, title and abstract.

A.1.4 Selection Criteria

The selected primary studies needed to be assessed for their relevance, enabling the inclusion of studies that provide evidence for the research questions as well as the exclusion of studies that do not. This is achieved by the definition of Inclusion Criteria (IC) and Exclusion Criteria (EC). Thus, our criteria were:

IC_1: The study addresses agile methods/practices in safety-critical systems development;
IC_2: The study presents benefits, challenges, and limitations of agile methods in safety-critical systems development;
EC_1: The study is out of the SM objective;
EC_2: The study does not present any abstract or it is not available for further reading;
EC_3: The primary study is a table of contents, short course description, tutorial, keynotes, copyright form or summary of an event (e.g., a conference or a workshop);
EC_4: The study is a previous version of a more complete one the same research, of the same authors;
EC_5: The study is written in a language other than English;
EC_6: The study compiles results of other studies.

A.2 Conduction

Our SM was conducted from February to July 2018. During the conducting phase, primary studies were identified, selected, and evaluated using the inclusion and exclusion criteria. For each selected study, data were extracted and synthesized according to the protocol presented in Section A.1. We adopted the Parsifal tool (PARSIFAL, 2020) to support the protocol definition and SM conduction. We prepared digital forms to accurately record any information needed to answer the research questions. Figure 29 shows the results for each step described below.

- *First selection*: The search string was customized and applied to the selected publication databases. As a result, we obtained 2166 primary studies and removed duplicate ones (i.e., 568 studies), remaining 1598 studies for analysis. Further, we excluded all editorials, prefaces, discussions, workshops, panels, and article summaries as well as studies that were clearly not about agile development of safety-critical systems. As an example, because our search strategy included the term "lean", we got several "hits" on articles related to *Lean Manufacturing*. Articles with titles and abstract that indicated clearly that the articles

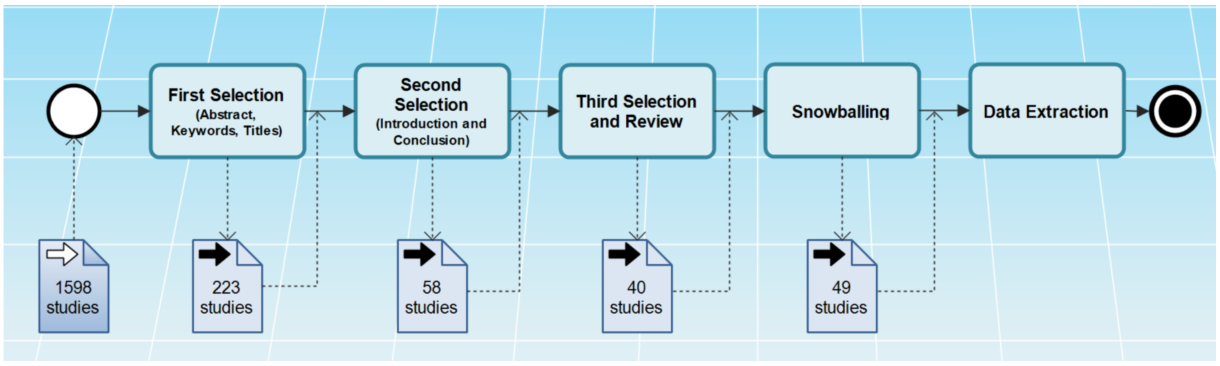


Figure 29 – Search conduction results.

were outside the scope of this systematic mapping were excluded. As result of this first selection activity, a total of 223 studies were included for detailed inspection.

- *Second selection*: The introduction and conclusion of the 223 primary studies was read and the selection criteria were again applied. As a result, 48 primary studies were included and 175 studies were excluded.

- *Third selection and Review*: Each of the 48 studies that remained after second selection activity was reviewed in depth. At the end of the selection activity, a safety expert performed the selection review. First, he inspected all studies excluded in the third selection (i.e., 21 of them). Afterwards, he checked the studies included by reading the abstract and any disagreements were discussed. As a result of this discussion, a total of 40 studies were were finally included as relevant to our SM.

- *Reference list Review (Snowballing)*: We used the backward snowballing technique (WOHLIN *et al.*, 2012) intending to cover the whole research area. This technique allowed us to identify and examine works referenced in the studies selected in the previous activity. Among all works evaluated, we selected 9 relevant primary study, which had not been previously identified. Finally, a set of 49 studies was selected and considered to answer the research questions.

- *Data extraction*: During this stage, all necessary data for our mapping study was extracted based on a predefined extraction form. This form allowed extracting all data with all details needed for the analysis of the research questions. Apart from the bibliographic information (title, authors, year, publication database and type), we extracted from each study the application domain in which the system under assessment or certification was used, the underlying safety standard(s) used to show compliance, the agile method/practice that was used in the respective project, the information, artifact, tool, or technique related to safety activities, the stakeholders involved, at which point of the agile development, safety was considered and the needs and challenges addressed about agile development of safety-critical systems. The dataset gathered from these forms supported the synthesis of the results.

A.3 Threats to validity

The main threats identified to the validity of this SM are described as follows:

Missing of important primary studies: The search for agile methods in the safety-critical systems development was conducted in five publication databases. According to Dyba et al. (DYBÅ; DINGSØYR, 2008) and Kitchenham and Charters (KITCHENHAM; CHARTERS, 2007), these publication databases are the most relevant sources in the computer science area. In addition, we were as inclusive as possible; thus, no limits were placed on publication date and we avoided imposing restrictions (i.e., filters by title, abstract, and keywords) on the primary study selection. Aiming at not missing any important evidence, we also adopted the snowballing technique using the reference list of the selected primary studies. In spite of our effort to include all relevant evidence, it is possible that primary studies were missed.

Selection reliability: Aiming at ensuring an unbiased selection process, we defined our research question in advance, performed a multi-stage process, and documented all inclusions and exclusions. Moreover, the first author is a PhD student in Software Engineering and Safety and the other three are experienced researchers in Requirements Engineering, Software Engineering or Safety-Critical Systems. Furthermore, aiming to increase the reliability of our SM, a safety expert reviewed the studies selection by applying the random choice strategy defined by Kitchenham and Charters (KITCHENHAM; CHARTERS, 2007). However, it might be possible that relevant studies were excluded in the first stages due to the lack of important information in the title, abstract, keywords, introduction and/or conclusions sections of these studies.

Data Extraction: During data extraction, we created a data sheet to fill and save all answers from each study. However, since not all information was explicitly available in the studies, some data had to be interpreted. To minimize this threat, discussions among the authors were developed.

APPENDIX

B

Mind map on safety recommendations

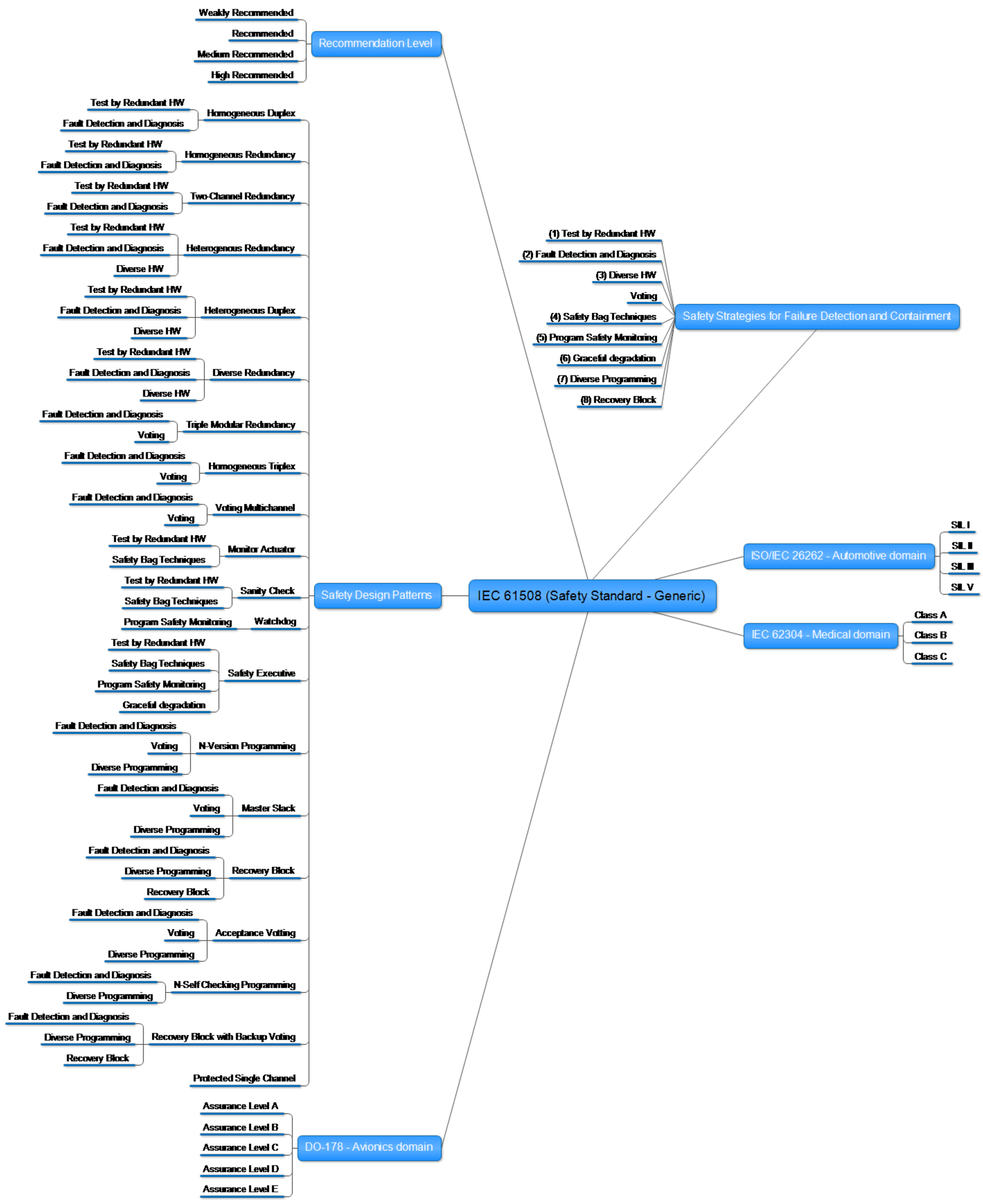
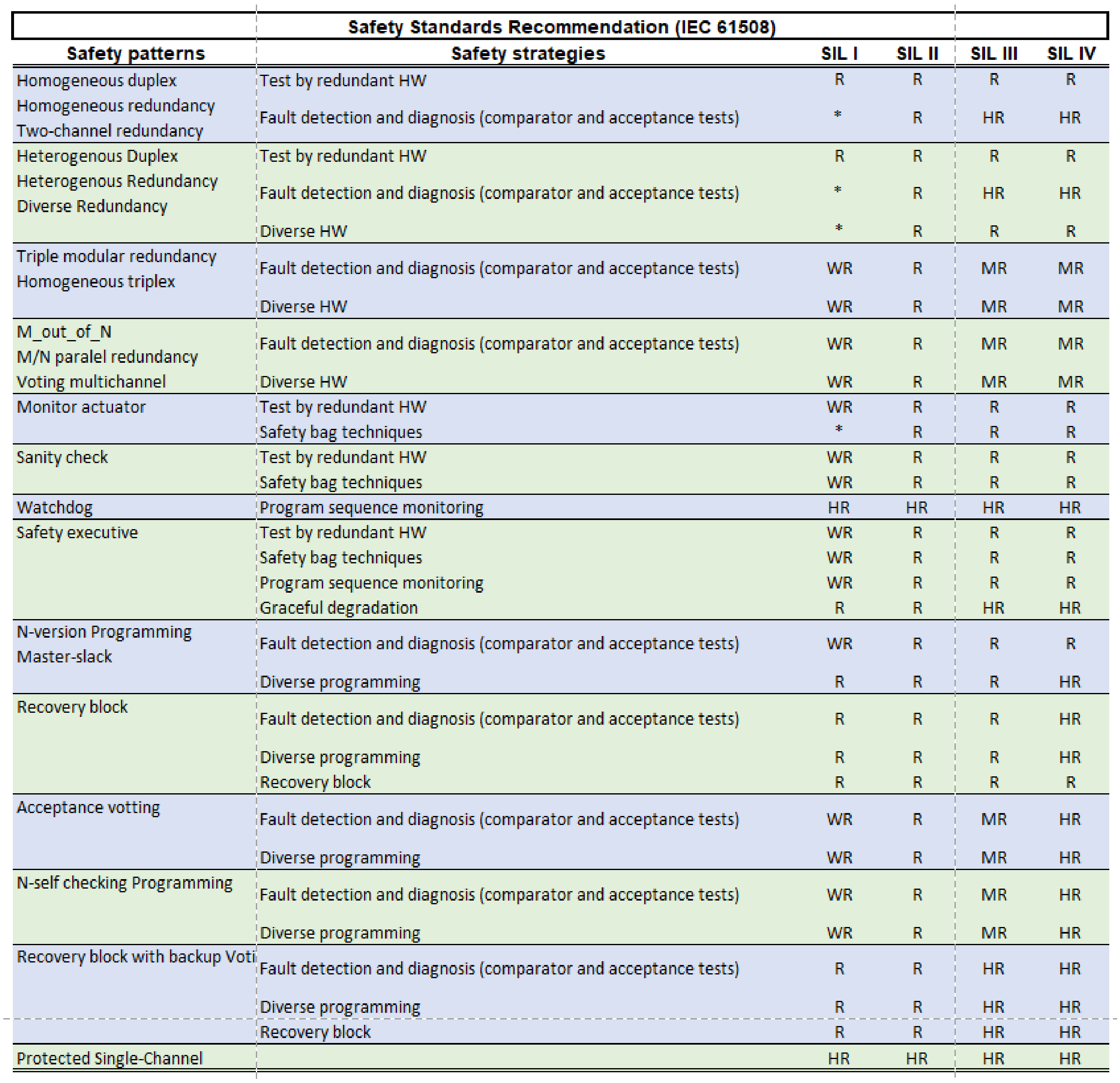


Figure 30 – Mind map of safety standards' recommendations.



	Safety Standards Recommendation (IEC 61508)				
Safety patterns	**Safety strategies**	**SIL I**	**SIL II**	**SIL III**	**SIL IV**
Homogeneous duplex	Test by redundant HW	R	R	R	R
Homogeneous redundancy Two-channel redundancy	Fault detection and diagnosis (comparator and acceptance tests)	*	R	HR	HR
Heterogenous Duplex	Test by redundant HW	R	R	R	R
Heterogenous Redundancy	Fault detection and diagnosis (comparator and acceptance tests)	*	R	HR	HR
Diverse Redundancy	Diverse HW	*	R	R	R
Triple modular redundancy	Fault detection and diagnosis (comparator and acceptance tests)	WR	R	MR	MR
Homogeneous triplex	Diverse HW	WR	R	MR	MR
M_out_of_N M/N paralel redundancy	Fault detection and diagnosis (comparator and acceptance tests)	WR	R	MR	MR
Voting multichannel	Diverse HW	WR	R	MR	MR
Monitor actuator	Test by redundant HW	WR	R	R	R
	Safety bag techniques	*	R	R	R
Sanity check	Test by redundant HW	WR	R	R	R
	Safety bag techniques	WR	R	R	R
Watchdog	Program sequence monitoring	HR	HR	HR	HR
Safety executive	Test by redundant HW	WR	R	R	R
	Safety bag techniques	WR	R	R	R
	Program sequence monitoring	WR	R	R	R
	Graceful degradation	R	R	HR	HR
N-version Programming Master-slack	Fault detection and diagnosis (comparator and acceptance tests)	WR	R	R	R
	Diverse programming	R	R	R	HR
Recovery block	Fault detection and diagnosis (comparator and acceptance tests)	R	R	R	HR
	Diverse programming	R	R	R	HR
	Recovery block	R	R	R	R
Acceptance votting	Fault detection and diagnosis (comparator and acceptance tests)	WR	R	MR	HR
	Diverse programming	WR	R	MR	HR
N-self checking Programming	Fault detection and diagnosis (comparator and acceptance tests)	WR	R	MR	HR
	Diverse programming	WR	R	MR	HR
Recovery block with backup Voti	Fault detection and diagnosis (comparator and acceptance tests)	R	R	HR	HR
	Diverse programming	R	R	HR	HR
	Recovery block	R	R	HR	HR
Protected Single-Channel		HR	HR	HR	HR

Figure 31 – Correlation between safety patterns, safety strategies, and safety recommendation.

APPENDIX

C

Controlled experiment material

Experiment Planning

IDENTIFICATION

TITLE: Safety-Centered Architectural Drivers and Decisions Derivation

THEME: Verifying a method that aims to support the derivation of safety-centered architectural drivers and decisions (SCA3DA).

TECHNICAL AREA: Software Architecture Specification

AUTHORS: Ana Isabella Muniz Leite – isabella.muniz@usp.br
Lucas Lagôa – lucaslagoa@usp.br
Samuel de Souza Lopes – samuel.lopes@usp.br

AFFILIATION: University of São Paulo (USP/ICMC) – Brazil

LOCAL: ICMC/USP

1. INTRODUCTION

The Safety Standards require that SW and HW safety requirements are derived from the system safety requirements and they should be traceable. But, in practice, that is done in an ad hoc way. Agile itself is already ad hoc, so the agile methods have to be adapted to address the safety standards requirements.

The SCA3DA (Safety-Centered Architectural Drivers and Decisions Derivation) Method aims to support systematic derivation of safety-centered architectural drivers and decisions to address them (improve the understandability). For that, it assumes as input, a preliminary specification of system architecture and safety story (which is a safety requirement specification at system level). From these inputs, the method provides a metamodel and guidelines to support the architect in decisions making, providing evidences, for instance, according to SIL (Safety Integrity Level) of safety story, there are some strategies that should be considered, but software architects often are unaware of or do not care about the SIL. The expected output is just-enough safety-centered architecture solution, which will be used as a guide by the development team.

Problem Statement: There exists no empirical data in terms of SCA3DA efficiency (specification is created faster) and efficacy (specification is more accurate).

2. CHARACTERIZATION OF EXPERIMENT

TYPE: In Vitro.

DOMAIN: Safety-Critical System Development in Agile Context.

LANGUAGE: Portuguese.

PARTNERS (INSTITUTIONS): UEPB/NUTES (Paraíba, Brasil)
Fraunhofer IESE (Kaiserslautern, Alemanha)
Instituto Atlântico (Ceará, Brasil)

LINK: Not defined yet.

ESTIMATED ACCOMPLISHING: Not defined yet.

3. EXPERIMENTAL STUDY DEFINITION

OBJECT OF STUDY: SCA3DA Method.

OBJECTIVE:

- To evaluate SCA3DA efficacy and efficiency for specifying safety-centered architectural solutions in relation to ad hoc methods.
- **Specific aims:**

**Analyze ………………………….. ** SCA3DA Method
For the purpose of ………… evaluation
with respect to ……………….. Efficacy and Efficiency
from the perspective of …… agile team (architect, Product Owner, and tech leader), safety engineer
in the context of ……………… agile development of safety-critical systems.

QUALITY FOCUS: Safety-centered architectural solution.

RESEARCH CONTEXT: The experiment will be conducted in pilot with masters and doctorate students in Software Engineering. Then with an agile team (architect, Product Owner, and tech leader) from NUTES (UEPB) and Instituto Atlântico. In the context of agile development of a safety-critical system, it will be carried out a real problem:

- Medical cyber-physical system: ICE (infusion pump)
- Automotive safety-critical system: (Power Sliding Door System) *(in pilot experiment only)*

RESEARCH QUESTIONS:

- RQ1: Is the approach capable of providing safety-centered architectural solutions at a level of efficacy comparable to ad hoc in agile development?

- RQ2: How much effort can be saved by using the approach and how dependent are these savings on the understanding of the real intent of safety requirements?

METRICS:

The efficacy (Ea) refers to creating more accurate safety-centered architectural solutions in an agile context. To be able to determine the accuracy of these architectural solutions depends upon having an agreed upon baseline. Here, the baseline refers to what would have been a correct and required solution to a safety system requirement and preliminary system architecture. For this experiment, the baseline is individual for each system description and each one was provided by an expert in the respective domain. These baselines will be used to analyze the architectural solution accuracy, determined after the experiment by the experimenters in a very intensive procedure. In cases of uncertainty, the expert (in the respective domain) shall decide about the correctness or incorrectness of the architectural solution. According to the baseline, three types of solutions can be distinguished and represent the dependent variables captured to answer research question 1:

Δc: correct solutions according to the baseline;
Δi: incorrect solutions according to the baseline;
Δm: missing solutions (in case of subjects given up on performing the task, these results should be classified as missing solutions).

$$Ea = \Delta c \div (\Delta c + \Delta i + \Delta m)$$

The efficiency refers to creating accurate architectural solutions faster. As an indicator of efficiency (*Ef*), we calculate the average of correct solutions by total time spent by subjects in responding to a task. These represent the dependent variables recorded in order to answer research question 2:
Δc: correct solutions according to the baseline;
ki: time spent (in hours) used by subject *i*;
n: number of subjects in the group.

$$Ef = [\Delta c \div (\Sigma ki)] \div n$$

4. PLANNING (Detailed)

HYPOTHESIS FORMULATION:

- H1: Individuals using SCA3DA method provide more accurate safety-centered architectural solutions than individuals using ad hoc.

Table 1 - Test Hypotheses to answer research question 1

Dependent variable	Null Hypothesis H0	Alternative Hypothesis H1
Efficacy	$Ea(SCA3DA) = Ea(ad-hoc)$	$Ea(SCA3DA) > Ea(ad-hoc)$

- H2: Individuals using SCA3DA method are more efficient than individuals using an ad-hoc method.

Table 2 - Test Hypotheses to answer research question 2

Dependent variable	Null Hypothesis H0	Alternative Hypothesis H1
Efficiency	$Ef(SCA3DA) = Ef(ad-hoc)$	$Ef(SCA3DA) > Ef(ad-hoc)$

- H3: There is no difference in the SCA3DA method efficiency and efficacy regarding individuals who used the medical system description and those who used the automotive system description (Interaction effect):

Table 3 - Test Hypotheses to answer research question 1 e 2

Dependent variable	Null Hypothesis H0	Alternative Hypothesis H1
Efficacy	$Ea(SCA3DA/Medical) \neq Ea(SCA3DA/Automoti$	$Ea(SCA3DA/Medical) \simeq Ea(SCA3DA/Automoti$
Efficiency	$Ef(SCA3DA/Medical) \neq Ef(SCA3DA/Automoti$	$Ef(SCA3DA/Medical) \simeq Ef(SCA3DA/Automotiv$

Process conformance measured with questionnaire (on an ordinal 5-point Likert scale (1=strongly disagree to 5=strongly agree)):

Q4.How much experience did the subjects have with architectural specification?
M4.Experience of subject(EXP)

Q5.How well did the subjects understand the Task?
M5.Understanding of task of subject(UNDERSTTK)

Q6.How well did the subjects understand the Method?
M6.Understanding of method of subject(UNDERST)

Q7.How well were the subjects motivated?
M7.Motivation of subject (MOTIV)

Q8.How well did the subjects follow the Method?
M8.Following of guidelines of subject (FOLLOW)

Q9.Did the subjects have enough time to complete their work?
M9.1.Enough time for subject (TIME)
M9.2.Percentage of task finished (PERCENT)

VARIABLES SELECTION:

- **Independent Variables:**
  - Safety-Centered Architectural Drivers and Decisions Derivation Methods/ Guidelines: we have two alternatives: The SCA3DA and ad-hoc methods.
  - Target Systems:
    - Project A: Medical cyber-physical system: ICE (infusion pump)
    - Project B: Automotive safety-critical system: (Power Sliding Door System)*(in pilot experiment only)*

- **Dependent Variables:**
    - Efficiency (architectural solution is created faster)
    - Efficacy (architectural solution is more accurate)

EXPERIMENT DESIGN:

Design chosen based on considerations regarding:

Target System:

- A: Medical cyber-physical system;
- B: Automotive safety-critical system *(in pilot experiment only)*.

Specification sequence:

- Each document should be used with both methods to control for differences between documents
- Order of techniques -> two treatments are applied in parallel

Training:

- only training for SCA3DA Method

Design:

- 2x2 factorial design
- Follow the principles of blocking and balancing;
- Two factors and two treatments for each factor
- Each subject will apply a method and use a target system.

- **Factor A:** Methods
- **Treatments:** SCA3DA and ad-hoc

- **Factor B:** Target system
- **Treatments:** Medical and Automotive systems documents

	Medical System (Document)	**Automotive System (Document)**	
Ad-hoc	Group 1	Group 2	First Day
SCA3DA method	Training		Second Day
	Group 3	Group 4	

SUBJECTS SELECTION:

- Pilot Experiment: Masters and PhD students from the Software Engineering Laboratory at University of São Paulo
- Agile team (architect, product owner, and tech leader) from NUTES (UEPB) and Instituto Atlântico.
- The sample used will be defined by the Convenience Sampling

INSTRUMENTATION:

- Artifacts:
  - Presentation of the Experiment
  - Consent form
  - Analyst Survey
  - List of participants
  - Training Material
  - Systems Description
    - Project A: Medical cyber-physical system: ICE (infusion pump)
    - Project B: Automotive safety-critical system: (Power Sliding Door System)*(in pilot experiment only)*
  - Feedback Questionnaire

RESULTS VALIDITY:

- Internal validity
  - Language: To avoid bias due to the language, the Portuguese language will be used in all the artifacts distributed to the subjects in Brazil;
  - Learning: As the sample has no sufficient knowledge about the domain of the systems used in the experiment, subjects may have difficulty in understanding some aspects of these systems. However, this can be beneficial, since none of the subjects has superior knowledge to the other subjects about the domain of the systems;
  - Process conformance of the subjects: There is no guarantee that subjects are deriving architectural drivers and decisions that address them in an ad hoc way or applying the SCA3DA Method. To avoid hindrances to this, all subjects will specify the requirements using only one approach (ad hoc or SCA3DA). Thus, it is avoided that one approach influences the application of another.
- External validity:
  - Pilot experiment: As the target audience of the object of study is composed of professionals from the software industry, the results obtained cannot be generalized since the sample used is not representative. However, as the main goal of this experiment is to conduct a pilot study in order to validate the experiment protocol, the sample used can be appropriate, and it will contribute to define parameters (duration of the training, for example) for conduction of experiment with a sample composed by practitioners of medical domain later;
  - Due to the limited number of subjects available to participate in the experiment, the sample used may not be statistically valid to generalize the results. However, the results can be beneficial in the context of the proposed pilot study.

5. TRAINING

- Definition and procedure:
    - The training will be done in a face-a-face session with the subjects (groups 3 and 4)
    - Steps of the training:
        - Step 1: SCA3DA Method will be explained.
        - Step 2: Execute the application of the method using an example.
        - Step 3: Clarify doubts about the method.
- Instructor: Ana Isabella Muniz Leite
- Participants: Masters and PhD students *(in pilot experiment only)*
- Agile team (architect, product owner, and tech leader)
- Artifacts:
    - List of participants
    - Training Material

6. EXECUTION PROCEDURE

As illustrated in Fig. 1, on the first day the subjects receive some introductory explanation and then are randomly shuffled within two groups. Both groups will be asked to derive safety-centered architectural drivers and decisions with their usual technique, the group 1 will work on the medical system, while group 2 on the automotive system. On the second day, the new subjects are trained in SCA3DA Method and afterwards as on the previous day, they will be randomly shuffle within two groups, then these are asked to perform the same task, but now they should use SCA3DA Method.

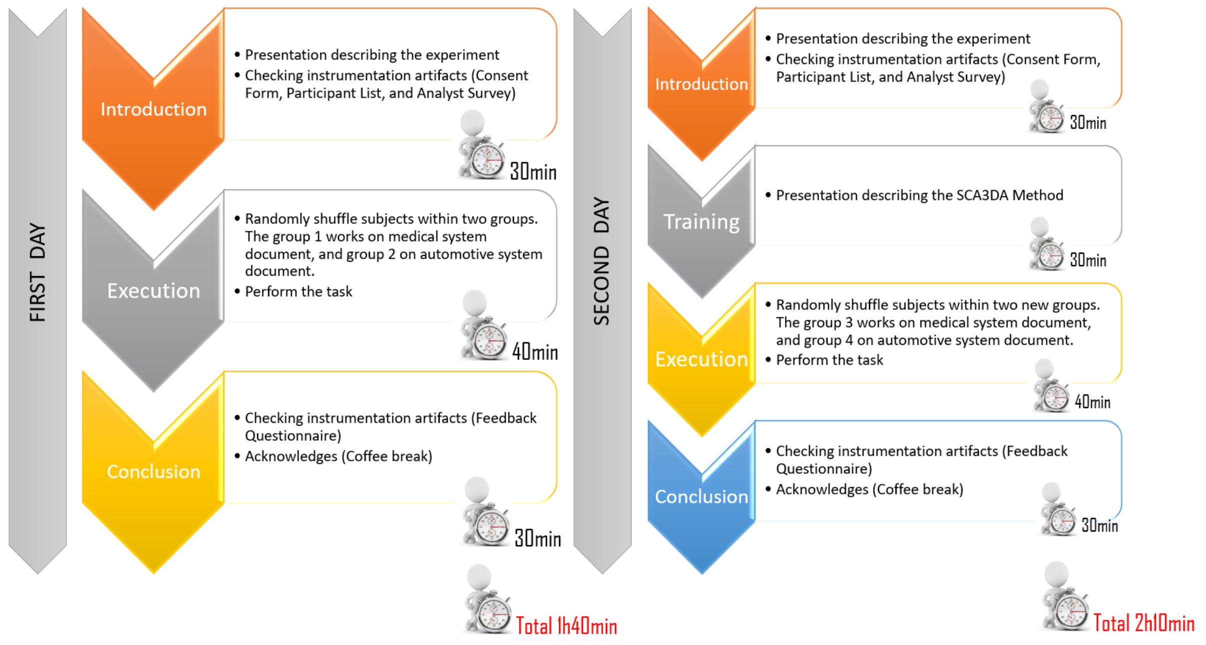

FIRST DAY
Introduction
• Presentation describing the experiment
• Checking instrumentation artifacts (Consent Form, Participant List, and Analyst Survey)
30min
Execution
• Randomly shuffle subjects within two groups. The group 1 works on medical system document, and group 2 on automotive system document.
• Perform the task
40min
Conclusion
• Checking instrumentation artifacts (Feedback Questionnaire)
• Acknowledges (Coffee break)
30min
Total 1h40min
SECOND DAY
Introduction
• Presentation describing the experiment
• Checking instrumentation artifacts (Consent Form, Participant List, and Analyst Survey)
30min
Training
• Presentation describing the SCA3DA Method
30min
Execution
• Randomly shuffle subjects within two new groups. The group 3 works on medical system document, and group 4 on automotive system document.
• Perform the task
40min
Conclusion
• Checking instrumentation artifacts (Feedback Questionnaire)
• Acknowledges (Coffee break)
30min
Total 2h10min

Fig.1 - Execution procedure

Analyst Survey

This form aims to gather some information about your profile and experience.

Participant: ____ Date: ____/_____/ ____

Part 1: General Information

1. Postgraduate.

   □ Master student

   □ PhD student

   □ Post Doctoral student

   Other one: _________________________

2. Professional Experience.

Role	How many months/years?
□ Analyst	
□ Developer	
□ Tester	
□ Architecture	
□ Project Manager	
Other one: _____________________	

Part 2: Specific Information

3. How do you rate your experience with respect to the agile context (agile practices, agile methods)?

□ Great	□ Good	□ Regular	□ Low	□ None
I participated in several projects of companies that use this type of approach	I participated in a project of a company that uses this type of approach	I participated in an academic project with a real client, in which this type of approach was used	Only in university discipline projects	I have never developed or participated in projects that have applied this type of approach

4. How do you rate your experience with respect to software-based safety-critical systems development?

☐ Great	☐ Good	☐ Regular	☐ Low	☐ None
I participated in several projects related to the development of critical systems	I participated in a project in a company involving critical systems	I participated in an academic project with a real client, related to this type of system	Only in university discipline projects	I have never developed or participated in projects related to critical systems

5. How do you rate your knowledge with respect to software architecture?

☐ Great	☐ Good	☐ Regular	☐ Low	☐ None
I participated in several projects, in which we need to detail the entire system architecture	I participated in a project with real customers, in which the entire system architecture was described	I participated in academic projects, in which I needed to define some architectural specification of the system	Only theoretical, in subjects at the university	I have never studied, I don't know what it is

6. How do you rate your knowledge with respect to safety requirements?

☐ Great	☐ Good	☐ Regular	☐ Low	☐ None
I participated in the process of specifying such requirements	I participated in projects in which such requirement was specified	I am aware of the rules, although I have never participated in a real project	University subjects only	I have never studied, I don't know what it is

7. How comfortable are you at specifying a software safety requirement?

☐ Great	☐ Good	☐ Regular	☐ Low	☐ None
Extremely comfortable	Comfortable	Moderate	Uncomfortable	Extremely uncomfortable

Thanks for your cooperation!

Your participation has been essential for the conduction of this experiment!

Feel free to contact us!

Post-Questionnaire Feedback

The purpose of this form is to gather your feedback on the experiment. All information collected on this form is confidential. Please, feel free to answer as you think its correct.

Part 1: Review the item that you think is most appropriate for the following statements

1. The task description was clear enough.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

2. The time dedicated to finish the task was sufficient.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

3. The training was sufficient to understand how to use the guidelines for performing the task.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

4. During the training, the objectives and steps for performing the task were properly clarified.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

5. I had difficulties in understanding the project documentation.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

6. I had difficulties in understanding the safety requirements.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

7. I had difficulties in specifying the software safety requirement.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

8. I had difficulties in following the guidelines presented along training, as consequence I had difficulties to accomplish the task.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

9. After using the guidelines (presented in the training), it was possible to better understand the intention of the safety requirement of the system.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

10. I felt motivated to perform the task.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

11. The attending in this experiment can help to understand software engineers' perception about safety-critical systems..

☐	☐	☐	☐	☐
Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree

Part 2: We would like to have your feedback on the following points:

1. What are the strengths perceived throughout this experiment?

____

____

____

2. What weaknesses were perceived throughout this experiment?

____

____

____

____

____

3. Would you have any suggestions for improvements in conducting this experiment?

____

____

____

____

____

Thanks for your cooperation!

Your participation was essential for the conduct of this experiment!

Feel free to contact us!