FEDERAL UNIVERSITY OF SÃO PAULO

INSTITUTE OF SCIENCE AND TECHNOLOGY

PROFESSIONAL MASTER DEGREE OF TECHNOLOGICAL INNOVATION

AN AUDIT MODEL FOR SAFETY-CRITICAL SOFTWARE

TALITA MARQUES RUIZ SLAVOV

ADVISOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

CO-ADVISOR: PROF. DR. JOHNNY CARDOSO MARQUES

São José dos Campos - SP

April/2021

FEDERAL UNIVERSITY OF SÃO PAULO

INSTITUTE OF SCIENCE AND TECHNOLOGY

PROFESSIONAL MASTER DEGREE OF TECHNOLOGICAL INNOVATION

AN AUDIT MODEL FOR SAFETY-CRITICAL SOFTWARE

TALITA MARQUES RUIZ SLAVOV

Thesis presented to the Professional Master's Degree program in Technological Innovation of the Federal University of São Paulo, as part of the requirements for obtaining a Master's degree in Sciences, area of concentration: Process and Products technological.
Advisor: Dr. Luiz Eduardo Galvão Martins
Co-Advisor: Dr. Johnny Cardoso Marques

São José dos Campos - SP
April/2021

Elaborado por sistema de geração automática com os dados fornecidos pelo(a) autor(a).

Marques Ruiz Slavov, Talita
AN AUDIT MODEL FOR SAFETY-CRITICAL SOFTWARE / Talita Marques Ruiz Slavov
Orientador(a) Luiz Eduardo Galvão Martins; Coorientador(a) Johnny Cardoso Marques. - São José dos Campos, 2021.
203 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Mestrado Profissional Interdisciplinar em Inovação Tecnológica) - Universidade Federal de São Paulo - Instituto de Ciência e Tecnologia, 2021.

1. Audit. 2. Safety-critical software. 3. Software audit model. I. Galvão Martins, Luiz Eduardo, orientador(a). II. Cardoso Marques, Johnny , coorientador(a).

FEDERAL UNIVERSITY OF SÃO PAULO

INSTITUTE OF SCIENCE AND TECHNOLOGY

PROFESSIONAL MASTER DEGREE OF TECHNOLOGICAL INNOVATION

AN AUDIT MODEL FOR SAFETY-CRITICAL SOFTWARE

TALITA MARQUES RUIZ SLAVOV

Thesis presented to the Professional Master's Degree program in Technological Innovation of the Federal University of São Paulo, as part of the requirements for obtaining a Master's degree in Sciences, area of concentration: Process and Products technological.

Approved on 30 April 2021.

Approval Committee:

Prof. Dr. Luiz Eduardo Galvão Martins

(Advisor – Unifesp ICT)

Prof. Dr. Paulo Tadeu de Mello Lorenção

(Professor and Researcher – UNIFESP ICT)

Dr. Marcelo José Ruv Lemes

(Senior Software Specialist – EMBRAER)

Prof. Dr. Jéssyka Flavyanne Ferreira Vilela

(Professor and Researcher – UFPE)

São José dos Campos - SP
April/2021

Talita Marques Ruiz Slavov

Título da Dissertação: AN AUDIT MODEL FOR SAFETY-CRITICAL SOFTWARE

Dissertação apresentada à Universidade Federal São Paulo como requisito parcial para obtenção do título de Mestra em Profissional em Inovação Tecnológica.

Área de Concentração: Processos e Produtos Tecnológicos

Aprovada em 30 de Abril de 2021.

Presidente da Banca:

Prof. Dr. Luiz Eduardo Galvão Martins

Banca Examinadora:

Prof. Dr. Paulo Tadeu de Mello Lorenção

Prof. Dr. Marcelo José Ruv Lemes

Prof. Dr. Jéssyka Flavyanne Ferreira Vilela

This work is specially dedicated to my daughter **Pietra**, *my husband* **Rafael**, *my beloved mother* **Marilda**, *my cherished father* **Vagner** *and to my whole family Bruno, Pâmela, Lívia, Lara, Luísa, Pedro, Angélica, Bárbara, Tiago, Ursula, Catarina, Renato, Josiane, Ryan, Alice, Ricardo, Tatiane, Valentina, and Isadora.*

ACKNOWLEDGMENTS

First and foremost, I would like to express my deep and sincere gratitude to my daughter Pietra and my husband, Rafael, for understanding my absence and be patient with me during the moments dedicated to this research, as well as their dedication to me. This research is part of our life plans and hopes to bring excellent future achievements.

Thank you to my beloved mother, Marilda (*in memoriam*), who gave me the purest love of this world and taught me everything I know. I am profoundly grateful to have you as my example of a mother, woman, friend, and dedication, and for giving me your passion for books and the desire to learn.

Thank you to my whole family, especially my father, Vagner, and my parents-in-law, Pedro and Angélica, for understanding the period away from that I have dedicated to this research that had a proposal for our family, and it was not in vain.

Thanks to my co-advisor, Dr. Johnny Cardoso Marques, who spends his time with me to understand this research proposal and believed it was a good theme for a Master's Degree, and all support and knowledge during this journey.

Thanks to my advisor, Dr. Luiz Eduardo Galvão Martins, for having accepted the challenge and spent his time contributing to this work, directing my research to extract its full potential and generate this work and other fruits as well.

Finally, I want to thank God, who blessed my life, illuminated my thinking, and gave me courage, force, health, and knowledge to start and conclude this work.

There are no companies without failure, there are companies that are well or poorly audited.

Dr. Stephen Kanitz

RESUMO

Atualmente o uso de software considerados complexos e críticos está crescendo em diversos setores da indústria como a aeronáutica com seus diversos sistemas embarcados em aeronaves e a médica com seus dispositivos médicos cada vez mais avançados. Devido a isso, a quantidade de *standards* dedicados a esse tipo de desenvolvimento está crescendo nos últimos anos e autoridades regulamentadoras estão reconhecendo a sua aplicabilidade e, em alguns casos, tornando como parte dos requisitos obrigatórios de certificação ou aprovação. O intuito de uma auditoria de software é verificar que o software desenvolvido está de acordo com a norma aplicável, no entanto os modelos existentes não permitem o auditor ter a flexibilidade de adequar o modelo de auditoria às suas necessidades. Como parte dessa pesquisa, diferentes modelos de desenvolvimento software foram considerados, bem como *standards* da área aeronáutica (RTCA DO-178C) e área médica (IEC 62304) foram estudados quanto as suas recomendações e requisitos para desenvolvimento de software *safety*-crítico. Como objetivo dessa dissertação, um modelo de auditoria de software foi proposto com as atividades que são necessárias para a condução de auditoria de software *safety*-crítico, permitindo ao auditor aplicar o modelo de acordo com as atividades que precisam ser auditadas, dando a flexibilidade necessária para o escopo da auditoria, bem como um conjunto de perguntas para a auditoria de software desenvolvido utilizando RTCA DO-178C e IEC 62304 foi sugerido e avaliado por especialistas de software para garantir a maturidade e eficiência das perguntas propostas. Além da avaliação das perguntas, também foi conduzido um estudo de caso, em uma empresa aeroespacial, com duas instanciações para avaliar a maturidade do modelo de auditoria de software proposto.

Palavras-chave: auditoria, software *safety*-crítico, modelo de auditoria de software.

ABSTRACT

Nowadays, the use of software considered complex and critical is growing in several industry sectors, such as aeronautics with its various systems embedded in aircraft and the medical one with its increasingly advanced medical devices. Because of this, the number of standards dedicated to this type of development is growing in recent years, and regulatory authorities are recognizing its applicability and, in some cases, making it part of the mandatory certification requirements or approval. The software audit intent is to verify that the software developed complies with the applicable standard. However, the existing audit models do not allow the auditor to tailor the audit model to its audit necessities. As part of this research, the various software development models were considered, and standards in the aeronautical (RTCA DO-178C) and medical (IEC/ISO 62304) areas were studied regarding their guidelines and requirements for safety-critical software development. This thesis aims to propose a software audit model with the activities necessary for conducting a safety-critical software audit, giving the auditor the necessary flexibility in the audit execution without the need to achieve specific predetermined milestones. Additionally, a set of questions for software auditing developed using RTCA DO-178C and IEC 62304 has been suggested and evaluated by software experts to ensure the maturity and efficiency of the proposed questions. In addition to evaluating the questions, a case study was also conducted in an aerospace company, with two instances to evaluate the proposed software audit model's maturity.

Keywords: *audit, safety-critical software, software audit model.*

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AC – *Advisory Circular*

AHAA - *Agile, Hybrid Assessment Method for Automotive*

ANAC – *Agência Nacional de Aviação Civil (National Civil Aviation Agency)*

ATM – *Air Traffic Management*

CAE – *Claims-Arguments-Evidence*

CFR – *Code of Federal Regulations*

CNS – *Communication, Navigation, Surveillance*

COTS – *Commercial Off-The-Shelf*

DER – *Designated Engineering Representative*

DAL – *Development Assurance Level*

EASA – *European Union Aviation Safety Agency*

FAA – *Federal Aviation Administration*

FDA – *Food and Drugs Administration*

FMEA – *Failure Mode and Effects*

FTA – *Faut Tree Analysis*

FURPS – *Functionality, Usability, Reliability, Performance, Supportability*

GQM – *Goal Question Metrics*

GSN – *Goal Structuring Notation*

HAZOP – *Hazard and Operability Studies*

IEC – *International Electrotechnical Commission*

IEEE – *Institute of Electrical and Electronics Engineers*

ISO – *International Standard Organization*

LCROSS - *Lunar Crater Observation and Sensing Satellite*

MCAS – *Maneuvering Characteristics Augmentation System*

MEMS – *Method for Evaluating Middleware architectureS*

NASA – *National Aeronautics and Space Administration*

NDA - *Non-Disclosure Agreement*

ODC – *Orthogonal Defect Classification*

PEH – *Programmable Electronic Hardware*

PMA - *Premarket Approval*

QA – *Quality Assessment*

RQ – *Research Question*

RQD – *Research Question for Dissertation*

RTCA – *Radio Technical Commission for Aeronautics*

SAM – *Software Audit Model*

SAME – *Sofware Audit Model Evaluation*

SDLC – *Software Development Life Cycles*

SIL – *Safety Integrity Level*

SLR – *Systematic Literature Review*

SOI – *Stage of Involvement*

SOUP – *Software Of Unknown Provenance*

SP – *Software Product*

SQA – *Software Quality Assurance*

SysML – *System Modeling Language*

UML – *Unified Modeling Language*

SUMMARY

Chapter 1

INTRODUCTION

This chapter intends to present the research contextualization explaining its subject's relevance, followed by the objectives to be accomplished and how the text is organized.

1.1 Contextualization

The technological advances and the constant concern to ensure people's safety obliges the industry to increasingly use software for complex and critical functions for various systems embedded in their aircraft. Due to this, to comply with certification authorities' requirements and the system’s safety level, standards were created to guide the software development based on their contribution to systems failure conditions. Among the established norms is RTCA DO-178C [1] - Software Considerations in Airborne Systems and Equipment Certification and its supplements RTCA DO-330 - tool qualification [2], RTCA DO-331 - Model-based development [3], RTCA DO-332 – Object-oriented technology [4], and RTCA DO-333 – Formal methods [5].

Created in the 1980s by the Radio Technical Commission for Aeronautics, Inc. (RTCA), the standard RTCA DO-178 establishes the necessary procedures for aircraft manufacturers, developers, installers, and embedded software users be sure that their product was built in compliance with airworthiness requirements. The DO-178C is dated 2011, with its original design in 1982 (RTCA DO-178) and two later revisions, in 1985 (RTCA DO-178A) and 1992 (RTCA DO-178B) [6].

In 2017, the Federal Aviation Agency (FAA), the American certification agency, recognized both the RTCA DO-178B and RTCA DO-178C as acceptable means for developing embedded software and approving systems and/or equipment using software through Advisory Circular (AC) 20-115D [7]. Although AC 20-115D allows other alternative methods, recent projects commonly use RTCA DO-178C as an acceptable mean of compliance [6]. AC 20-115D is an advisory document accepted by several certification authorities such as the European Union Aviation Safety Agency (EASA) and the National Civil Aviation Agency (ANAC - Brazil).

Not only the aeronautical world has concerns related to safety-critical software development, but in recent years the medical area also started to extensive use of sophisticated software for medical devices; it was identified the necessity to create a standard focused on this area. The International Electrotechnical Commission (IEC) started in 2006 the ISO/IEC 62304 [8] and named it Medical device software — Software lifecycle processes.

The Food and Drugs Administration (FDA) recognized in January 2019 the latest version of this standard. After that, a joint group of the International Electrotechnical Commission (IEC) released in 2006 the first version of the new international standard, resulting in harmonized use of this standard in the United States and Europe [9].

According to Peter et al. [10], strict regulations are used with medical devices to ensure the patient's safety and reduce the health risks produced by using the high technological products. The medical device's purpose is one aspect considered to determine the risk classification, and this classification defines the level of regulation applied in the safety-critical software approval process.

Besides, for the railway area, there is the standard IEC 62279 [11] named Railway applications - Communication, signaling and processing systems - Software for railway control and protection systems, created to regulate the development, deployment, and maintenance of railway application software [9].

Heinrich et al. [12] explain in their article that railway signaling networks are classified as critical infrastructure, and a security incident in the railway system can cause a considerable impact, due to this standards were created to guide the security concept ensuring the robustness of the railway signaling networks against cyberattacks. The IEC 62279 [11] determines a safety integrity level to classify the software application regarding its criticality and security level.

Order 8110.49 Chg 1 [13] was published in 2011 by FAA with some guidelines, and one of them is related to conducting the assessment and approval of software developed using the RTCA DO-178B [13]. Order 8110.49 is dated 2003, initially published with guidelines based on AC 20-115B and DO-178B to guide FAA personnel and their accredited.

As part of the assessment and approval process for safety-critical software, Order 8110.49 Chg 1 establishes Stages of Involvement (SOIs) to evaluate the software in stages, through audits, as shown in Table 1-1. Each of these steps requires a certain percentage of artifacts generated according to the software's life cycle to be evaluated. The best known and most suitable life cycle for the complete application of the audit model proposed in Order 8110.49 is waterfall development. For other types of software development, adaptation in the form of application is necessary. Based on each kind of software development's characteristics, the audits carried out using the guidelines of Order 8110.49 Chg 1 and DO-178C need to be customized. Possibly requiring more of one audit event due to the entire cover of the scope of available artifacts for their evaluation, repeat the audit due to a significant number of issues identified in the first event, or even join two phases (e.g., SOI- 2/3).

Table 1-1 – Stages of Involvement from Order 8110.49 Chg 1 [14]

Order 8110.49 Chg 1	Process Phase	Content	%
SOI-1	Planning	- Plans	100
SOI-2	Development	- Requirements - Design - Coding - Integration	50
SOI-3	Verification	- Tests - Analysis	50
SOI-4	Certification	- Certification artifacts	100

Regarding the safety-critical software developed for medical devices, the FDA has a classification process that determines the level of control necessary to assure a medical device's safety and effectiveness. Table 1-2 presents these classes and related regulatory controls. The Class helps determine the type of premarketing submission/application required for FDA clearance to market. Another information considered is the device's intended use and upon indications for use [15].

Table 1-2 – FDA Medical Device Classification [15]

Class	Identification	Risk
Class I	General controls • With exemptions • Without exemptions	Lowest
Class II	General Controls and Special Controls • With exemptions • Without exemptions	Medium
Class III	General Controls and Premarket Approval	Highest

Suppose it is identified during the classification that there is an exemption, in that case, the FDA provides a classification database of regulation numbers (medical specialty and regulation citation (21 CFR)) that helps the software developer classify its product once it covers most types of devices [16]. After that, if the product is considered a Class III, the applicant must follow the Premarket Approval (PMA), an FDA process of scientific and regulatory review to evaluate Class III medical devices' safety and effectiveness [17].

When developing safety-critical software, it is essential to follow the guidelines from applicable standards and to have a process to verify that it was correctly followed and there is compliance with their objectives, the process responsible for verifying this is called an audit. Mainly for safety-critical software that intends to be approved for certification authorities, like aeronautical or medical areas, the audit is essential to give confidence that the software is mature enough to be approved. By executing the audit before the certification authority, it is possible to identify the potential gaps early and solve them so the software is ready to be evaluated and approved.

Audit activity is not only used when necessary to get a safety-critical software approved by a certification authority. But a company may choose to assess its internal developments to evaluate the adherence to the defined process or even applicable standards, verify the correct development of software, verify the maturity of the established process, and even reduce the process execution gaps.

The most challenging topic in performing an audit is ensuring that all required concerns were covered by the auditor questions, thus reuniting evidence that the safety-critical software complies with the applicable standard and is mature enough.

Due to this, it is necessary to have a well-established audit process and a good knowledge of the standard used for safety-critical software development.

Even existing standards to develop safety-critical software in different areas and some of these areas have certification authorities or at least some organization that regulates software development, even though it has some accidents as a root cause of the software issues. Besides the standards and the recognition by certification authorities, accidents caused by an error on software implementation are another reason to have a team responsible for evaluating through audits, among other activities, the safety-critical software developed to be used in an aircraft, a medical device, or railway applications. This team is responsible for the management and performs assessments of the safety-critical software suppliers or internal developments.

This research intends to present a process with detailed guidelines related to safety-critical software audits, including guidelines to allow the auditor to create its own questions database and understand how to conduct an audit. Questions were proposed based on achieving compliance with RTCA DO-178C, recorded in Appendix B, and IEC 62304, recorded in Appendix C, and evaluated by software specialists in these standards, as a way to provide an example on how to generate the questions database. An additional analysis using traceability was performed to IEC 62304 to verify the proposed questions' equivalence and provide guidelines to update, insert, or remove questions.

Besides, as it is presented in CHAPTER 4, a systematic literature review was conducted to verify if there are techniques to conduct an audit, like the use of Goal Structuring Notation (GSN), Goal Question Metrics (GQM), Job Aid (FAA) or self-methodologies. However, two of them are not explicit in defining a process on how to prepare and conduct an audit, and for the Job Aid, FAA has discontinued and is no longer available to be consulted. Additionally, none of them provide a guideline on tailoring the process based on the auditor necessity and safety-critical software scope.

1.2 Objectives

This research proposes a safety-critical software audit model (SAM) that is flexible enough to be tailored to any safety-critical software area allowing the auditor to define the audit scope according to its necessity. The steps necessary to achieve this goal are:

1. Verify which are the audit methods proposed in the literature and how they are applied to safety-critical software.
2. Define and detail a safety-critical software audit model, including recommendations on creating a question database to guide the audit execution.
3. Using the proposed audit model, create a set of questions to verify two software compliance, one developed using RTCA DO-178C, and the other developed using IEC/ISO 62304, and evaluate these questions with software specialists.
4. Create guidance on how to adapt the audit model to other safety-critical standards.
5. Use the guidance to perform an adaptation of another safety-critical standard.

1.3 Organization of text

This research work is structured in eight chapters:

- **Chapter 2**: is related to the literature review and discusses some relevant topics in detail.
- **Chapter 3**: is related to the proposed methodology used to conduct the research.
- **Chapter 4**: is related to the systematic literature review conducted.
- **Chapter 5**: describes the safety-critical software audit model itself.
- **Chapter 6**: describes the SAM evaluation process, focusing on the RTCA DO-178C and IEC 62304 questions evaluation.

- **Chapter 7** describes the SAM process evaluation through a case study, focusing on the RTCA DO-178C and IEC 62304 questions evaluation.
- **Chapter 8**: brings the analysis and discussions of this research.
- **Chapter 9**: presents the research conclusion, contributions, social insertion, and future perspectives.

Chapter 2

LITERATURE REVIEW

This chapter intends to present a theoretical foundation and a systematic literature review to understand the main concepts used in this research and the academic research conducted to support the results of this research.

2.1 Theoretical Foundation

This section aims to define and explain the main concepts used in this research, like the safety-critical software audit model, software development model, software approval by certification authorities, embedded software, and safety-critical systems.

2.1.1 Safety-critical systems

According to Martins & Gorschek [20], a safety-critical system carries a high dependency level, demanding the systems to be available, reliable, safe, and secure. It is a system whose failure may cause injury or human being deaths [21]. As stated by Gallina, Sljivo & Jaradat [22], depending on the system's applicability, it must be approved, and to achieve this remark the system must demonstrate that it is acceptably safe. One of the means of compliance to accomplish this reliability is to use safety-critical standards, which define the processes that must be followed to develop a safety-critical system.

In the paper written by Gallina, Sljivo & Jaradat [22], the focus is on exploring a process line approach - which is a process that can be defined as a set of partially developed tasks that must be executed to develop a system. In the context of safety standards and based on ISO 26262 (automotive standard) [23] and EN 50126 (railway) [24], they compared these standards to identify their similarities and differences, once the integration of a system with a sub-system in different domains can lead to more than one standard to achieve the full compliance.

Srivastava & Schumann [25] propose software health management as a new discipline that has improvements for tasks in the entire software design, implementation, verification, and validation processes. The rationale for their proposal is that even taken the preventive measures already known by the industry, it was not possible to prevent some accidents like in:

- Aeronautical:
  - British Airways flight 027: error terrain collision and avoidance system,
  - Northwest flight 255: monitoring system disabled,
  - F-22 Raptors experience multiple computers crashes, and
  - A380: exploded engine.
- Satellites and spacecraft:
  - Mars Polar Lander: mission lost due to spurious sensor signals,
  - Mars rover Spirit", and
  - LCROSS: excessive fuel burn.

The safety-critical system is not only driven by functional requirements that are flow-down to software development but also the safety requirements, which require analysis like FTA (Fault Tree Analysis), FMEA (Failure Mode and Effects Analysis), HAZOP (Hazard and Operability Study), among others. These analyses are used to identify the failure conditions that can lead the system to an undesired state, which allows the system and software to be developed to avoid or at least mitigate these identified conditions [26].

Another crucial aspect is the testing activities for this type of system, once several standards pose specific requirements for testing (requirements-based testing and implementation-based testing). These tests have the intention to verify the degree of implementation and requirements coverage. However, due to these

systems' complexity, sometimes use only one of these techniques does not make it possible to ensure a reliable system [27].

Increasingly, tests related to the safety-critical system demand automatisms and greater integration among different systems. This integration ensures that the functional requirements are met and that the risks identified in the safety analysis were adequately mitigated by the system, allowing a high-level degree of confidence in the developed product. Also, many techniques, not only for testing but for assuring development, are being increasingly improved to ensure compliance with the various standards established by the certification authorities.

Many papers in the literature discuss mainly how to verify a safety-critical system or how to conduct the safety-assessment analysis. In their article, Hartonas-Garmhausen *et al.* [28] conducted a case study in formal verification of a complex real-world safety-critical system for the railway industry using a verification tool (Verus). Using STAMP – a model of accident causation – Leveson et al. [29] proposed designing and analyzing resilience by applying it to a safety culture. Besides, Gallina, Sljivo & Jaradat [22] presented a tool and its extension (SPEM 2.0 – Software Process Engineering Meta-model and vSPEM) to conduct safety analysis using a modeling tool.

2.1.2 Software development model

Software development is more than transforming an idea into lines of code. It is to design and build software that is reusable and easily maintainable. It is highly recommended to establish a software process and choose a development model that most attends to the software team strategy and necessity.

The software development process is responsible for identifying the main phases necessary to build software; usually, it is generically defined as planning, requirements, design, coding, integration, verification, and closure. Each of the demarcated phases is also accompanied by its related review phase, also known for the validation activities, and is responsible for checking if the produced artifacts are correct, complete, and following the applicable standard and regulation.

The waterfall model is considered the first software development model, initially documented by Benington [29] and later adapted by Royce [30] in the 1970s. It is a sequential and linear development model in which the phases are well defined: System Requirements, Software Requirements, Analysis, Design, Coding, Testing, and Operations.

The name waterfall was not initially given for this type of development but became popular due to its diagram [30]. This development's main characteristic is that one process phase can start only when its previous phase is complete. The customer will also receive the software at the end of the lifecycle [31], as represented in Figure 2-1.

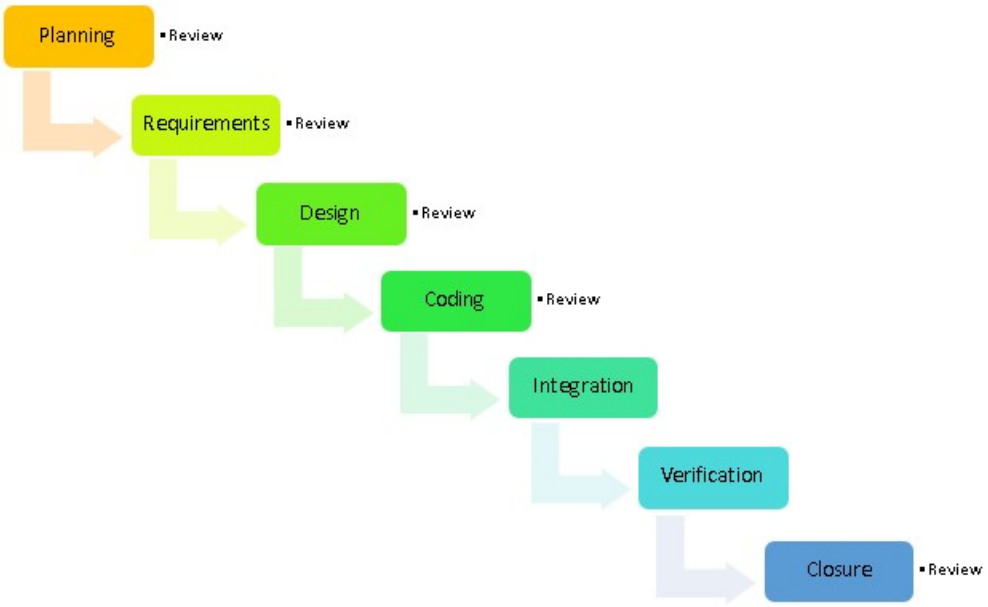


Figure 2-1 - Waterfall software life cycle adapted from [30]

The V-Model was created by NASA (National Aeronautics and Space Administration), the American space agency, which is also a variation of the waterfall model. It has this name because the sequence of activities is presented in a "V" diagram, where the descent (left leg) is related to decomposition and definition (requirement, design, code, and integration). The rise (right leg) is related to integration and verification (unitary, integrated, and functional tests) [32]. Figure 2-2 represents the V-Model.

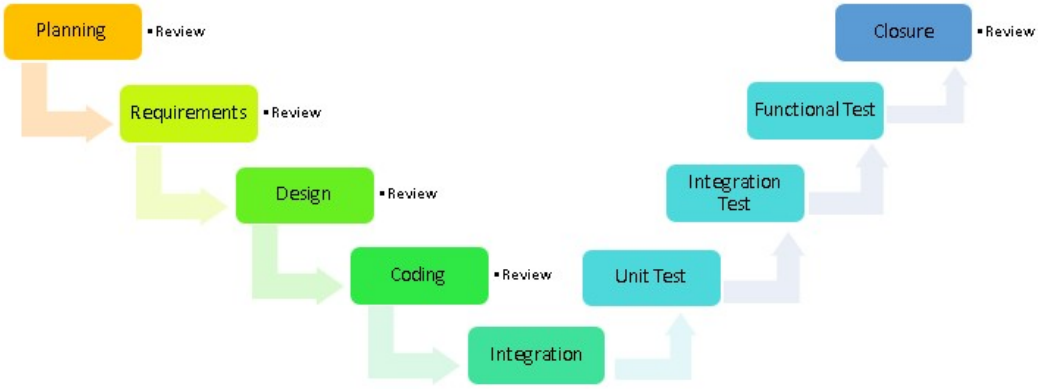


Figure 2-2 - V-Model software life cycle adapted from [33]

The agile model has as its main characteristics the ability to avoid scope changes and feature creep by breaking a project into smaller sub-projects. The development occurs in short intervals, and software releases are made to capture small incremental changes [32], allowing the customer to evaluate the release and provide feedback if necessary.

Besides, according to Ruparelia [32]:

> Applying Agile to large projects can be problematic because it emphasizes real-time communication, preferably on a personal, face-to-face basis. Also, Agile methods produce little documentation during development (requiring a significant amount of post-project documentation) while de-emphasizing formal process-driven steps.

The Spiral model results from the modification performed by Boehm in the waterfall model in 1986 [34] by introducing several iterations that spiral out from small beginnings. An oft-quoted idiom describing the spiral method's philosophy is to start small, think big [32]. Figure 2-3 is a representation of the Spiral model by demonstrating the three software versions release.

Incremental Software Development has as main characteristics the customer participation by providing feedback and comments during the software development by releasing intermediate versions. The complete set of requirements is specified at the beginning of the development, while design, coding, and integration processes occur concurrently, with customers' involvement [31]. This lifecycle data iteration with the customer is represented in Figure 2-4.

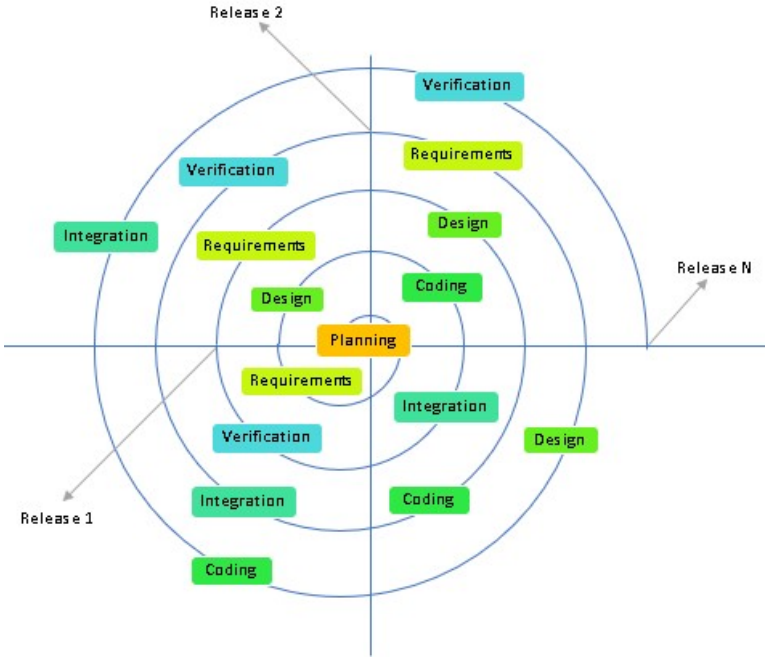
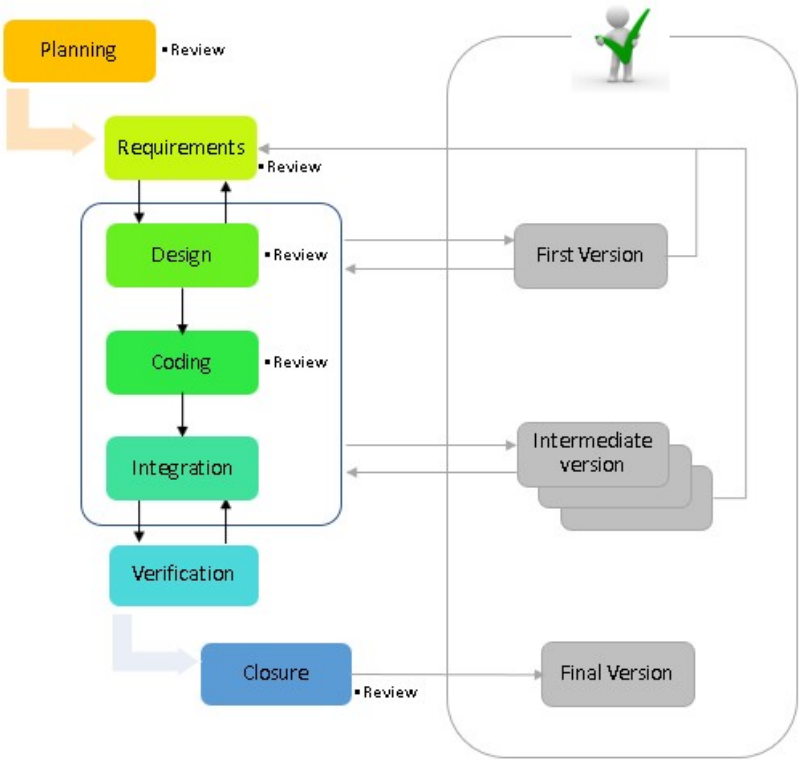


Figure 2-3 -Spiral software life cycle adapted from [34]



2.1.3 Figure 2-4 -Incremental software life cycle adapted from [35]Embedded software

As stated by Voelter *et al.* [36], embedded software is used to control hardware devices, commonly related to time and memory constraints aspects, no matter if it is simple hardware or a complex and sophisticated one. It is possible to notice the increase in the number of devices using embedded software in different industries and its challenges. Some examples of embedded software applications are telephones, medical systems diagnosis, climate control, aircraft, among others [37], which prevents creating a single specialized platform [38].

Due to an increase in the embedded software complexity, standards were created to provide a guide to develop it, e.g., in the automotive industry, there is the IEC 61131-3 – Programming Industrial Automation System, which is related to how to program a Programmable Logical Controller (PLC). In the aeronautical industry, there is the RTCA DO-178C - Software Considerations in Airborne Systems and Equipment Certification and RTCA DO-278A - Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance.

According to a systematic review conducted by Shen *et al.* [39], the six main characteristics to develop an embedded software are (i) software developers skills, which requires a high-level development ability and knowledge of the hardware; (ii) hardware dependencies, where the target is usually developed concurrently with the software; (iii) competitive pressure, which brings challenges to attend time-to-market, control the budget, and project buffer; (iv) limited resources, which lead to resource constraints like memory space, processing power, execution time; (v) changes originated from the requirement, hardware dependencies, and customers' need; (vi) performance requirements regarding the high-performance to attend constraints related to real-time and safety issues.

Veerabahu [40] identifies some embedded software development characteristics in his article, where the number of lines of code is not exactly the most important in this type of software but develops software in such a way it can control the hardware perfectly. This necessity can demand a lot of time and effort, which makes the ability to control and manage the system (or hardware) an essential

aspect instead of the algorithm itself. The software developed does not run in personal computers, instead of on another platform like a printed circuit board with a microcontroller, and to debug, it is sometimes necessary to use tools like oscilloscopes and logic analyzers and multimeters.

The embedded software allied with the hardware platform intends to exploit the system's full potential when associated with safety analysis and a very-well requirements elicitation, allowing to improve the extraction of the benefits from the hardware and the software developed. These characteristics require low-level programming languages for control algorithms, like C, which is good at low-level algorithms, produces efficient binaries, and is beneficial to address the embedded software's relevant aspects. However, it has some limitations that can be circumvented to meet all the software's needs under development, as stated by [36].

Kusters, Van Solingen & Trienekens [41] discuss in their paper the non-functional requirements (quality characteristics) that should be explicitly specified, like usability, maintainability, portability, and reliability. Also, he explains in detail the approaches commonly identified in the literature to elicit these non-functional requirements, e.g., asking, comparing, analysis, trial and error, multi-party, and questionnaire-based. Always making clear that the methods to obtain non-functional requirements are directly related to the characteristics of the companies developing the software.

Another compelling characteristic is the possibility to use model-driven engineering, thus based on a defined architecture and its related textual requirements. It is possible to implement the requirements through graphical notations, giving a better abstraction level than programming languages. It also enables automatic code generation through specific tools that can also be qualified to ensure the executable object code. In some cases, the modeling languages cannot be used only to implement requirements and to specify the requirements themselves through tools and languages designed for this (e.g., Unified Modeling Language (UML), System Modeling Language (SysML)).

2.1.4 Software audit model

In the aviation industry, the RTCA DO-178C [1] uses the term Software Quality Assurance (SQA) to designate a process and their activities related to ensuring that the software lifecycle data was produced based on the defined process and conforms to related standards and applicable regulations, as well as if the configuration management system is correctly established and followed [42].

Besides, the SQA team can also identify improvements in the defined software process and plans and standards once issues on safety-critical software development are identified, and sometimes these issues are justified as necessary deviations. However, this information can only be recognized if the SQA team has a process established, which includes a software development audit process.

According to Elliott, Dawson & Edwards [43], companies usually establish an internal auditing department and company to comply with their applicable standards and regulations. Some companies seeking to, or wishing to remain certified, for example, to ISO 9001:2000 [44], are required to conduct internal audits at planned intervals to determine whether the management system and its related implementation comply with the standard and is effective. A similar situation occurs in the aeronautical world; however, the intent is not to certify the company in a standard but certify an aircraft with a safety-critical software embedded that must comply, according to its criticality, with several standards objectives.

According to Elliott, Dawson & Edwards [43], the certification process can be classified into three types: 3rd-party audit, 2nd-party audit, and 1st-party audit. The first type is used to indicate that an independent company conducts the audit, the second type is undertaken by a customer, and the last one is being performed by the company's employees. This research focuses on the software audit for safety-critical software, initially proposed to conduct in the suppliers that developed it, but can also be adapted to evaluate internal developments.

In his paper, Oman [45] conducts a case study using the SQA audit model proposed by "IEEE Standard for Software Reviews and Audits", intending to demonstrate an example of how it can be used emphasizing metrics and measurements. In the paper conclusion, it is indicated some points to keep in mind:

- *Audits should not be undertaken by anyone unfamiliar with the applicable standards. (An audit without standards makes no sense.);*
- *Analyses of the overall code structure, organization, maintainability aspects and code change traceability, traceability of the V&V process and documentation, and compliance with applicable standards;*
- *Looking at software management administrative lines of control, documents comprising the software management plan, software configuration control mechanism, and the maturity of their in-place software development and maintenance processes.*

In their paper, Ortega & Rojas [46] designed a Systematic Quality model based on the most relevant aspects of the various models studied, like McCall, Boehm, FURPS, ISO 9126, Dromey, and Systemic Quality. Their model took into consideration a process and product dimension, and by validating it on two software products, it was possible to ascertain that their proposed model was a useful tool for analyzing product quality.

It is possible to identify papers in the literature related to the software audit model, but almost all of them are not associated with aeronautics standards like RTCA DO-178C. Tuohey [47], in his paper, performed a comparison of several software engineering standards, describing the implications and benefits that flow from them. He also concludes that Software Level C of RTCA DO-178B is a valuable reference for software developments. When not considering the certification process, once to comply with Level C objectives, it is required to have an effective quality management system.

In his paper, Rushby [48] discusses new aircraft certification challenges and includes two approaches known by the literature. The Claims-Argument-Evidence (CAE) is a framework to create claims, arguments, and evidence for assessing safety systems; and the standard-based approach, where the evaluation occurs based on the applicable standard guidelines. Also, in his article, an analysis of the advantages and disadvantages of each method is discussed. Simultaneously, the CAE framework is focused on safety aspects, and the standards-based incorporates the accumulated experience and community wisdom.

Fulton [43] already shares his practical experiences working as a Designated Engineering Representative (DER) for the FAA with the standard set of Job Aid worksheets. These worksheets provide several recommendations on evaluating a

software development based on DO-178B and are used to audit a project series of four stages of involvement (SOI) audits. Jiménez, Merodio & Sanz [49] propose a two-level checklist based on RTCA DO-178C and RTCA DO-278A for assessing, through inspection, the software developed through the checklist approach.

Besides, Graydon [50] apud [51] discusses in his paper the overarching properties, which is based on defining a "set of properties that are sufficient to warrant receiving approval for [an entity's] use on an aircraft". Bernhart *et al.* [52] discuss in their paper an evaluation method focused on the Code Review Process for a development based on RTCA DO-278 and company-specific organizational requirements. The proposed process is based on a peer desk-check or pass-around method.

One example is that by performing internal audits and even external audits (i.e., supplier), it is possible to identify the process deviations and the lack of compliance with applicable standards and regulations and technical issues in the product under development.

2.1.5 Standards for safety-critical software development

According to Marques & Cunha [53], apud Münch *et al.* [54], the number of organizations that need to check adherence to regulatory standards has increased. Software Standards usually prescribe rules for the development, verification, validation, configuration, and other disciplines available in the Software Engineering field. The demonstration of compliance must ensure that activities carried out during the project are repeatable and traceable within a chosen life cycle.

According to Marques *et al.* [55], the typical standardization presented in any safety-critical domain does not define how the software development life cycle must be conducted. There are some standards for safety-critical software development, such as RTCA DO-178C (aviation), RTCA DO-278A (air traffic control), IEC 62304 (medical), and IEC 62279 (railways).

2.1.5.1 RTCA DO-178C

It is a standard created by Radio Technical Commission for Aeronautics (RTCA) and named RTCA DO-178C - Software Considerations in Airborne Systems and Equipment Certification. It provides a guide summarized into 10 tables with objectives to each of them focusing on planning, development (requirements, design, coding), verification (SW unit, SW-SW tests, and SW-HW test), verification of verification, configuration management, quality assurance, and certification, that intends to develop a software with a level of confidence in the safety and complies with the airworthiness requirements.

It is considered the primary means of compliance to obtain civil aircraft software approval after being recognized by FAA through the AC 20-115D and EASA. Besides, the RTCA DO-178C has supplements divided into four standards: RTCA DO-330 – Tool qualification[2], RTCA DO-331 – Model-based development [3], RTCA DO-332 – Object-oriented technology, and RTCA DO-333 – Formal methods [4].

2.1.5.2 RTCA DO-278A

It is a standard created by Radio Technical Commission for Aeronautics (RTCA) and named RTCA DO-278A - Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems [56]. It provides a guide to performing the assurance of software contained in non-airborne CNS/ATM, and equally to RTCA DO-178C, intends to develop a software with a level of confidence in safety through a set of tables with objectives. In the software industry, these standards are known as containing similar guidelines; however, they focus on their specific software type.

Another relevant aspect is due to the wide adoption of commercial off-the-shelf (COTS) in the development of CNS/ATM projects; it also provides fourteen additional objectives related to COTS and guidelines on how to achieve compliance

and ensure the same level of confidence when using this type of software during the development.

2.1.5.3 IEC 62304

It is a standard created by the International Electrotechnical Commission (IEC) and named Medical device software — Software lifecycle processes, a standard created for the medical area. Like the aircraft area, the extensive use of sophisticated software for medical devices, the Food and Drugs Administration (FDA) created a specific software regulation. After a joint group of the International Electrotechnical Commission (IEC), a new international standard was released in 2006, resulting in harmonized use of this standard in the United States and Europe [9].

This standard describes five processes: software development, software maintenance, software risk management, software configuration management, and software problem resolution. The software level classification is based on the potential hazard that could result in injury for the patient in the system's abnormal function. There are three classes for the software: Class C the highest rigor, a failure could lead to death or serious injuries and must comply with all the objectives. Class B does not need to comply with all objectives. However, it has more objectives than Class A. While Class B can cause non-serious injuries, Class A does not cause injuries or damages to health [9].

2.1.5.4 IEC 62279

It is a standard created by International Electrotechnical Commission (IEC) and named Railway applications - Communication, signaling and processing systems - Software for railway control and protection systems, that is a standard created for railway area. This standard was created to regulate the development, deployment, and maintenance of railway application software. It discusses subjects related to requirements of the developing organization (roles and competencies), life cycle

(phases, documentation, and methods), and software assurance (testing, verification, validation, and quality assurance and evaluation) [9].

The software classification is called Safety Integrity Level (SIL) and is based on potential danger that could harm the user in case of abnormal system behavior. It is divided into five levels from 0 to 4. Its guidance is related to eleven phases: planning, system development, requirements, architecture design, component design, implementation, testing, integration, validation, maintenance, and evaluation [9].

2.1.5.5 IEC 61508

It is a standard created by the International Electrotechnical Commission (IEC) and named Functional safety of electrical/electronic/programmable electronic safety-related systems, a standard created for providing safety standards for electrical, electronic, or programmable electronic systems and products. This standard was created to regulate the development of electronically-based safety systems, usually in the automotive area [57].

This standard can be used for the automotive industry and others that have concerns with the safety aspects of their safety-related systems. It is focused on the analysis of the potential risks or hazards of a system or device. The categories, divided into four categories of system integration level (SIL), allow determining the potential risks and their consequences if it happens [57].

2.1.5.6 IEC 61511

It is a standard created by International Electrotechnical Commission (IEC) and named Functional safety - Safety instrumented systems for the process industry sector, a standard created to guide on how to use the instrumentations process to ensure the safety of systems and products. This standard was created to establish the process implementation of IEC 61508 [58].

This standard is responsible for defining the design and requirements management processes, including initial concept, design, implementation, operation, and maintenance through to decommissioning. It also uses the SIL categorization as a guide to define and develop the safety instrumented system based on the required process.

2.1.6 Software approval by certification authorities

As defined by Rushby [59], “certification is intended to provide stakeholders and society at large with an assurance that deploying a given system does not pose an unacceptable risk of adverse consequences.”. Besides this assurance, another relevant aspect is the increasing complexity of the software developed for the aeronautical industry and the medical, railway, and others.

Standards and regulations were created to establish a guideline for companies that develop safety-critical software, thus allowing to establish what are the necessary certification requirements so that software can be approved and, e.g., in the case of the aeronautical industry, it can be embedded, thus making sure that the software installed is in compliance with the certification authorities requirements.

Regarding the aeronautical industry, the software approval by certification authorities process is guided, evaluated, and assured by certification authorities like Agência Nacional de Aviação Civil (ANAC) from Brazil, Federal Aviation Administration (FAA) from the USA, European Union Aviation Safety Agency (EASA) from Europe, among others. They are responsible for conducting software audits during the safety-critical software development and provide approval for the final software version allowing them to be embedded in the aircraft.

As part of the process of evaluating and approving embedded safety-critical software, Order 8110.49 Chg 1 [60] establishes Stages of Involvement (SOIs) to evaluate the software in stages, through audits events. Each of these steps requires a certain percentage of artifacts generated according to the software's life cycle to be evaluated, where certification authorities establish this percentage based on the safety-critical software characteristics and criticality.

The certification authorities' most used method to perform their assurance activities is called regulation-based once they intend to verify compliance with regulations like AC 20-115D, which recognizes the RTCA DO-178C [1] and its supplements [59] as a mean of comopliance. To perform this assurance, the FAA created the Job Aid (Conducting Software Reviews Prior to Certification) with several questions for each event established by Order 8110.49 Chg 1. However, nowadays, the Job Aid has been discontinued.

Each audit event is placed at strategic points in the software lifecycle development to decrease the amount of effort in evaluating the lifecycle and reduce the risk of failing at the final certification authority audit when the cost to correct the issue is high as well its impact on the software product schedule and release. An early indication of a potential certification failure is essential to guarantee that the software process is not heading in the wrong direction. An audit failure usually requires reworking an artifact before repeat the audit. The software engineers need to indicate how appropriately the software development process adheres to the certification requirements before performing a SOI audit [61]. Some companies utilize conducting an internal audit using similar maturity criteria to SOI before formal audits with certification authorities.

Recent studies indicate that aeronautical certification authorities are discussing the use of safety assurance cases, which is a practice that relies on goal-oriented or claim-based safety arguments. Additionally, this is a practice already used by medical areas that are requesting to related industries to submit the safety-assurance cases as part of the approval process [62].

Chapter 3

METHODOLOGY

This chapter intends to present the research questions and steps used to achieve this thesis objective and propose a safety-critical software audit model.

3.1 Research Questions

Table 3-1 describes the research questions proposed for this research. It was used to perform all the research, including the systematic literature review from CHAPTER 4. All four questions were answered by the analysis conducted, although the conclusion was that most of the proposed methods are not strictly related to a specific standard. It was identified at least eleven different audit methods, even though a considerable number of papers were related to safety assessment. Even though the literature presents some software audit methods, very few focus on models that can be adapted for different environments using safety-critical applications. Most of them are customized for a specific scenario, for example, aeronautical [49] or railway [62].

Table 3-1 – Research questions for dissertation

ID	Research Question for Dissertation
RQD1	What are the audit models proposed in the literature which take into consideration safety-critical standards?
RQD2	What initiatives are proposed in the literature to conduct an audit on critical embedded software to approve software by certification

ID	Research Question for Dissertation
	authorities?
RQD3	Which models allow flexibility of the audit to be conducted?
RQD4	To what extent does the proposed audit model help companies concerned with the certification process?
RQD4.1	How does the proposed model advance or improve previous models?
RQD4.2	What are the main benefits for the companies?

3.2 Research Steps

Figure 3-1 depicts the research steps used to achieve this thesis's objectives. It starts with the research questions definitions, which were presented in the previous section. The questions were proposed thinking about what information would be necessary and relevant for the research so that, in the end, their findings would help propose a software audit model that would meet the needs of the industries that develop safety-critical software.

Once the research questions were considered mature and aligned with the objectives, a literature review was carried out to find the answers to the research questions and even seek information from the existing software audit models. If there was a model already proposed, be able to identify its points of improvement, and if not, be able to understand the points of need to create a model capable of meeting the most significant number of safety-critical software development.

The literature review was divided into two activities: the systematic literature review and the theoretical foundation. The first executed was the systematic literature review because this type of research would allow identifying any existent software audit model, what is discussed about this subject in the literature, and possible gaps that to be covered in this research. The results and findings of this activity are detailed in CHAPTER 4.

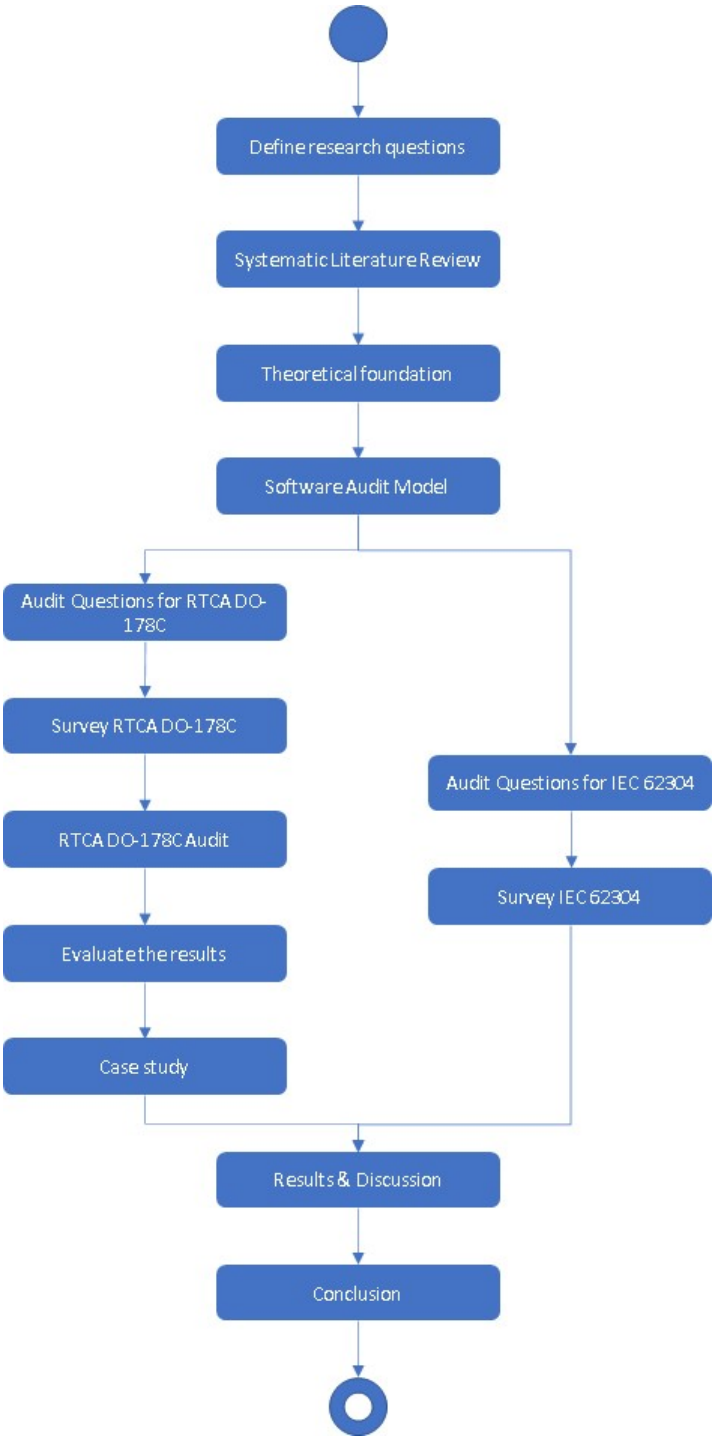


Figure 3-1 – Research workflow

After identifying aspects covered in the literature, as well as the existing gaps for a software audit model, the next step was to search for theoretical bases on the topics that were also relevant and are around the central theme of this research so that it could find a theoretical basis for the relevance and applicability of the research as a whole. For these activities, academic databases, like Elsevier, IEEE, ACM

Digital, were consulted, as well the Google Scholar, as a means to search the papers and improve the research string. The academic databases were not limited to the one cited previously, as well the data extraction also used books, magazines, and industry standards to compose the theoretical foundation and related argumentation. The theoretical literature is detailed in Section 2.1.

As per the extensive research in the academic databases and the researcher's experience, it was possible to describe the proposed safety-critical software audit model in detail, including its activities and related questions database. The process was defined as focusing on the audit process itself, allowing the customization to any safety-critical software development that applies a standard, aeronautical, medical, railway, automotive. The process includes the related SAM requirements, roles and responsibilities, preparation for the audit, evaluate documentation, perform the audit, follow-up, recommendation on questions creation, and general recommendations.

Once the safety-critical software audit process was defined, a way to make its assessment viable would be proposing an initial question database so that it could then be used in two audits performed through a case study with actual safety-critical software development. Thus, the first standard selected was the RTCA DO-178C due to the extensive use in the aeronautical industry, being recognized as a means of compliance by aeronautical regulation authorities, which means that there are several applications in the industry to evaluate the proposed process and be the standard of most practical knowledge of the researcher.

The RTCA DO-178C process is divided into planning, requirements, design, coding, verification, certification liaison, and integral process composed of the quality assurance and configuration management phases, which shall be executed throughout all the other processes. Each phase has specific guidelines and related objectives that must be accomplished or not by the final software product, considering the software development assurance level. This standard's questions were created from scratch to cover all the standard’s concerns, based on its guidance and objectives. So that at the end of the audit, it is possible to evidence the compliance of the software development with the applicable standard and identify the gaps that must be corrected for the software to be approved by the certification authorities.

Software specialists then evaluated the proposed questions through a SAME to ensure their maturity and relevance. This validation step with a software specialist

is essential once the standard itself was created and updated through the years by reuniting specialists from several companies, regulation authorities, and even airlines to discuss the necessary guidance to be used to develop an airborne software. Some of these specialists made up the organizational committees that RTCA created to create and update the standard according to the demand and applicants' needs.

With that in mind, it was clear the gain in conducting an assessment of the proposed questions with these software experts, as they have the necessary knowledge to assess each of the questions and indicate whether it adequately address each of RTCA DO-178C objectives and concerns, thereby giving the necessary feedback to ensure the maturity of the questions that would be used to conduct the audits associated with this standard. The process used to elaborate and evaluate the proposed questions for RTCA DO-178C is described in Section 6.1.

A second standard was selected to create questions in such a way to increase confidence in the proposed SAM and demonstrate its flexibility. The additional intention was to demonstrate that the questions could be adapted and reused from the ones already suggested another standard. The second standard chosen was the IEC 62304, due also guides safety-critical software development, applicability on medical devices and is recognized by the Food and Drugs Administration (FDA).

Like RTCA DO-178C, IEC 61304 also has a definition in phases: planning, requirements, design, software unit implementation and verification, software integration and integration testing, software system testing, and release. It also has additional processes: maintenance, risk management, configuration management, problem resolution, and a reference for another standard that threatens the quality assurance aspects. Another similarity between them is that IEC 62304 also has a system for categorizing software development - CLASS (A, B, C) - which is also used to indicate which sections of the standard the final software product must comply with.

Questions for IEC 62304 were suggested by adapting the ones created for RTCA DO-178C, considering these similarities, and creating new ones for those aspects specific from IEC 62304. A software audit model evaluation (SAME) was also conducted to evaluate the questions with software specialists to ensure their maturity and relevance, the process used to elaborate and evaluate the proposed questions for IEC 62304 is described in detail in Section 6.2.

After the questions database was mature enough, a case study was planned and conducted with safety-critical software developed for an aeronautical company, and the proposed questions were used based on the scope defined for this development. This step was necessary to evaluate the software audit model proposed with actual software development and assess its adaptability, as this is one of the benefits provided by the proposed process. After the audit was finished, an interview was conducted with the auditor to collect his impressions of using the safety-critical software audit model and provide feedback on improvements that could be implemented.

CHAPTER 8 presents the results and discusses the safety-critical software audit in light of the proposed research questions, answering them with the information researched and learned from the research proposed in this dissertation, and concludes the proposed research methodology.

Chapter 4

SYSTEMATIC REVIEW LITERATURE

This chapter intends to present the Systematic Literature Review (SLR) conducted to investigate: (i) what are the methods proposed to conduct an audit in safety-critical safety-critical software to approve the software; (ii) how it is considered the software development methods, and (iii) what are the pros and cons of the proposed approaches. Appendix G presents the list of papers selected.

4.1 Related work

According to Elliott, Dawson & Edwards [43], companies usually establish an internal auditing department and company to comply with their applicable standards and regulations. Some companies seeking to, or wishing to remain certified, for example, to ISO 9001:2000 [8], are required to conduct internal audits at planned intervals to determine whether the management system and its related implementation comply with the standard and is effective. A similar situation occurs in the aeronautical world. However, the intent is not to certify the company in a standard but to certify an aircraft with a safety-critical software embedded that must comply with several standards objectives according to its defined criticality.

In their paper, Ortega & Rojas [46] designed a Systematic Quality model based on the most relevant aspects of the different models studied, like McCall, Boehm, FURPS, ISO 9126, Dromey, and Systemic Quality. Their model took into consideration the process and product characteristics. By validating it on two

software products, it was possible to ascertain that their proposed model was a useful tool for analyzing product quality.

It is possible to identify papers in the literature related to the software audit model related to different standards like RTCA DO-178C, RTCA DO-248C, RTCA DO-330, which may indicate the software audit model is not necessarily related to the applicable standard. Tuohey [47], in his paper, performed a comparison of several software engineering standards, describing the implications and benefits that flow from them. He also concludes that Software Level C of RTCA DO-178B is a helpful reference for software developments. When not considering the certification process, once to comply with Level C objectives, it must have an effective quality management system.

In his paper, Rushby [48] discusses new challenges in aircraft certification and includes two approaches known by the literature. The Claims-Argument-Evidence (CAE) is a framework to create claims, arguments, and evidence for assessing safety systems; and the standard-based approach, where the evaluation occurs based on the applicable standard guidelines. Also, in his article, an analysis of the advantages and disadvantages of each method is discussed. Simultaneously, the CAE framework focuses on safety aspects, and the standards-based incorporates the accumulated experience and community wisdom.

Based on these statements and SLR's intent, some selected works discuss the details of the audit process for safety-critical software. Dodd & Habli [61] proposed and empirically evaluated a statistical method for supporting software audits to obtain the approval of certification authorities based on collecting and analyzing data about the software throughout its lifecycle. Steele & Knight [63] proposed a method called the Filter Model. The certification process is a safety-critical system where an incorrectly certified system is an accident.

Linling, wenjin & Kelly [64], Poreddy & Corns [65], Hawkins *et al.* [66], Ruiz, Larrucea & Espinoza [67], and Schwierz & Forsberg [68] conducted their assurance cases proposing an audit method that uses the Goal Structuring Notation (GSN) method, which is a graphical argument notation that can be used to document explicitly the elements and structure of an argument and the argument's relationship to evidence. GSN, calls claims of the argument as goals, and items of evidence are valid solutions[69].

Habli & Kelly [70] associated the GSN method with Goal Question Metrics (GQM) method, which is a top-down measurement framework for defining goals related to products and processes. A goal is interpreted using a set of questions whose answers are associated with objective or subjective metrics. It is specified relative to four elements: issue (e.g., reliability, security, or cost), object (e.g., software, platform, or process), viewpoint (e.g., independent, or internal auditing), and purpose (e.g., failure rate reduction, or certification). Questions are derived from these elements, and subsequently, their answers estimate the achievement of the top goal using primitive metrics (e.g., faults detected, failure classifications, or objectives satisfied).

Silva & Vieira [71] defined a framework and a process to map the system assessment issues (empirical data) to their root causes and act upon those root causes. Davila-Nicanor & Mejia-Alvarez [72] proposed a methodology to evaluate the quality of web-based software systems using statistical web testing techniques. The framework predicts performing an analysis, using statistical modeling techniques, studying the system's behavior, and detecting software defects. In his article, Fulton [42] indicates the application of Job Aids that is a FAA standard set of worksheets used to audit a project in a series of four SOI audits. Liu, Li & Zhang [73] developed a Method for Evaluating Middleware architectureS (MEMS), which measures middleware architectures by rating multiple quality attributes.

Jiménez, Merodio & Sanz [49] defined two level-checklists for compliance with RTCA DO-178C and RTCA DO-248A standards, where a set of questions are proposed and distributed in two types of checklists that the auditor must use to seek the evidence with the objectives of the related standards. Graydon [50] discusses in his work the concept of overarching properties, which "constitute a set of properties that are sufficient to warrant receiving approval for [an entity’s] use on an aircraft" [51], applied to safeguard system associated with the assurance-cases method to review the process.

The 46% (18) of the papers selected for SLR research were related to assessing software from a safety perspective. The other 54% (21) of the papers discuss other subjects, like 76% (16) related to approving software, and the additional 24% was related to another topic.

4.2 Research Methodology

The SLR was conducted based on the research methodology of Biolchini *et al.* [74], Kitchenham & Charters [75], and Martins & Gorschek [76]. Figure 4-1 shows the research methodology adopted, and the following sections describe its steps. The first chapter of this paper presents the reason for this SLR (Step 1 of Figure 4-1). It was verified the existence of a similar SLR. It was performed this verification at Compendex, Google Scholar, and EBSCOhost digital libraries (accessed on 20 May 2019) using the following string within the titles, abstracts, and text:

(Safety AND/OR Critical) AND Assessment AND (Review OR SLR OR Research overview)

The retrieved studies from this initial verification were related to safety in its various aspects of concern. None of them were directly related to the SLR on safety-critical software assessment and could not answer all the Research Questions (RQ) proposed in this thesis. It was elaborated a defined three research questions, allowing what to extract from the selected studies (Step 2 of Figure 4-1). These questions are presented in Table 4-1. A real case of an aerospace application that has several safety-critical software suppliers from different technologies, which must be approved by certification authorities to allow the aircraft certification was the motivation for RQ1 [42]. The purpose of this research question is to map all methods proposed by the literature, no matter the software development area, and to verify which one could be implemented or adapted in the aerospace application.

According to Marques & Cunha [31], the creation of many Software Development Life Cycles (SDLC) occurred over the years. Although they define different strategies for executing software development, validation, and verification, it is observable that such models differ on (i) granularity; (ii) involvement of the user; and (iii) software product evaluation. The RQ2 intends to verify how the software evaluation considers them due to various software development methods.

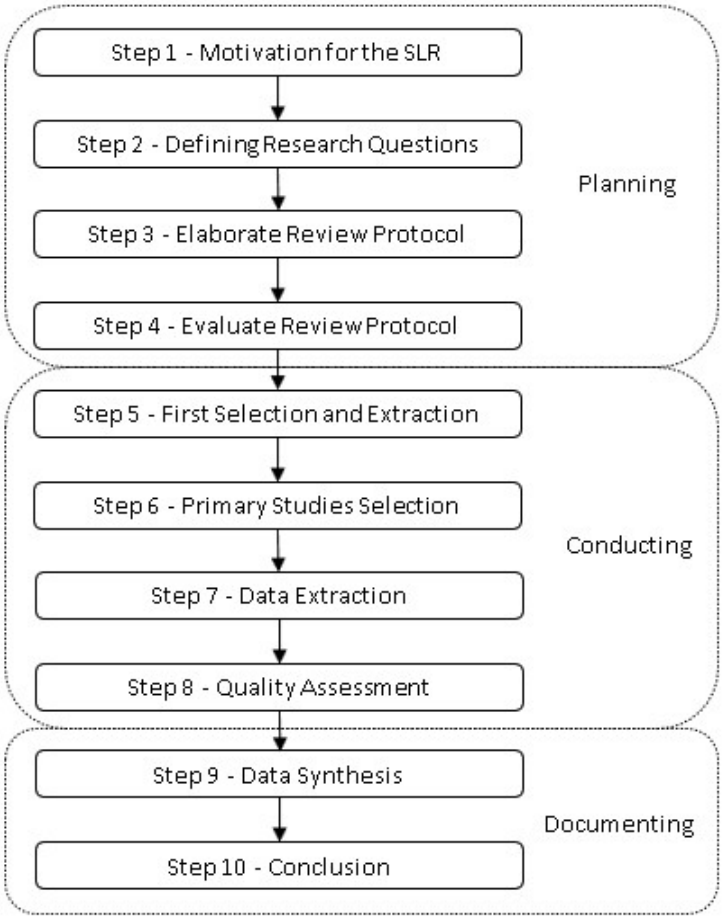


Figure 4-1 - Systematic review steps adapted from [74]**,** [75] **and** [76]

Table 4-1 - Research questions for SLR

ID	Research Question	Aim
RQ1	What methods have been proposed to conduct an audit in safety-critical software to approve the software?	To identify and classify which have been the proposed initiatives to conduct an audit in safety-critical software to have the software approved by a certification authority.
RQ2	How have software development methods been taken into consideration?	To identify if the audit methodology has been proposed using cascade, v-model, or incremental development methods.
RQ3	What are the pros and cons of the approaches proposed for?	To analyze the advantages and disadvantages of each method of applying to an audit.

4.2.1 Search strategy

Figure 4-2 presents the search strategy for the identification of papers as proposed by Martins & Gorschek [76]. The keywords defined for the search string were extracted from the research questions to contribute to the research to find out works related to safety-critical software assessments. We performed some tryouts to ensure the relevance of the proposed search string, on Science Direct and Springer Link databases (Step 5 of Figure 4-1), due to the number of studies related to safety assessment in the literature.

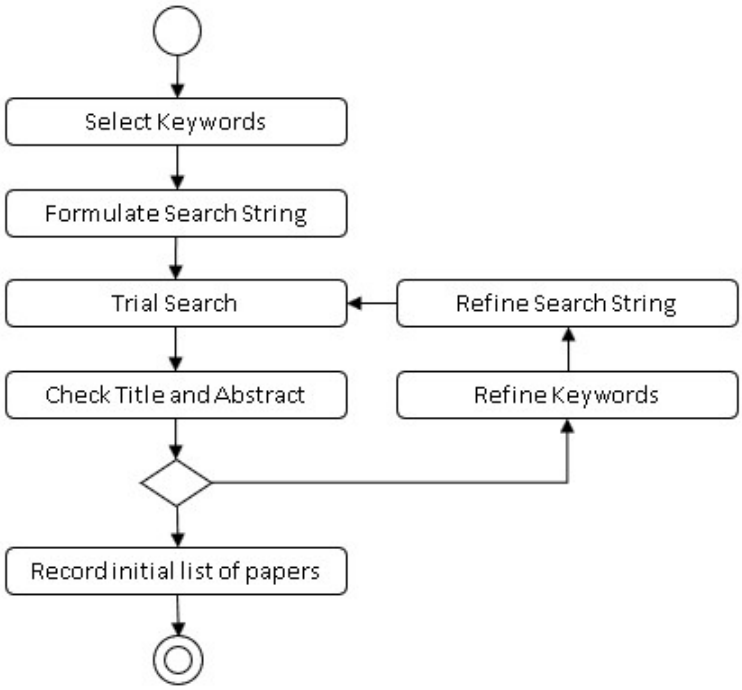


Figure 4-2 - Detailed search strategy adapted from [74]**,** [75] **and** [76]

After testing a different combination of words (such as audit, assessment, software, safety-critical software, embedded, evaluation, certification, quality assurance, "Safety-Critical" System, aircraft, aerospace) and does not find too many papers to conduct the SLR, it was decided to propose the search considering the following information in each database considered relevant due to safety-critical software subject:

- **ScienceDirect**: titles and abstracts.
- **Springer Link**: all fields.

- **IEEE Xplore**: selected the option "Full Text & Metada" and each word from the search string in each field concatenated by an AND.

The four databases were used the advanced search tool and considered the range of 1999 to 2019 in the year field, to select only the past two decades. The search string used in the SLR is specified below:

Assessment AND "Safety-Critical" Software AND "Airborne"

Table 4-2 - Search string justification

Term	Justification
Assessment	It is also used to indicate the activity of evaluating a process or product.
Safety-Critical	It is used to designate software that, when associated with a system, its failure may cause injury or humans death, usually requires approval before it can be used, which requests an assessment.
Software	The literature discusses the safety-critical software and the safety-critical systems, and the focus of this research is the software assessment.
Airborne	It is a term that helps to direct research to software that is developed to be associated with safety-critical systems.

In addition to Table 4-2, another reason to consider the word "assessment" in the search string instead of the word "audit" was to expand the papers selected related to this topic because, during the tryouts, we noticed that few papers returned from the search when using the word “audit” and some of them returned as a result using word “assessment”.

4.2.2 Review protocol

We developed a review protocol (Step 3 of Figure 4-1) with the main elements: the **selected databases** chosen were Science Direct, Springer Link, IEEE Xplore, and ACM Digital. We applied the defined **search method** on research through web search engines available on digital libraries, and when necessary, we used the available filters to refine the research. The **population** considered only publications peer-reviewed reporting methodologies for assessing safety-critical software; the

reason for the **intervention** was to verify the methods used and what they took into consideration.

Table 4-3 - Inclusion/exclusion criteria

Number	Inclusion Criterion
1	Primary studies.
2	Studies which subject was related to safety-assessment (using the perspective of audit), audit, certification, quality assurance, and frameworks to conduct an assessment.
3	Studies published from 1999 until May 2019.
4	Studies related to the certification process.
Number	**Exclusion Criterion**
1	Secondary studies.
2	Studies that do not bring any discussion about approaches for conduct an audit/assessment for any software
3	Studies not written in English.
4	Duplicated studies.
5	Studies whose text was not available (through search engines or by contacting the authors).

Table Table 4-3 provides a summary of the exclusion criterion and inclusion criterion. A tryout has been conducted on two digital databases (Science Direct and Springer Link) to review the protocol (Step 4 of Figure 4-1). The number of papers was collected and validate with the two other participants. Both agreed with the defined protocol and initial results. Additionally, in the first moment, only titles and abstracts were read to select the first samples. For those studies that the title and abstract were not enough to evaluate the study selection, we included the introduction and conclusion as part of the read criteria.

4.2.3 Procedure for studies selection

As presented in Figure 4-3, the search string was applied in the database, read the title and abstract, and verified whether the study satisfied the inclusion criteria or exclusion criteria, both presented in Section 4.2.2. If, based on title and abstract was not possible to classify the study, the next step was to read the

introduction and conclusion and, based on both, recheck the inclusion and exclusion criteria.

After running the research string and the review protocol based on Figure 4-3, some papers were selected for the data extraction (Step 6 of Figure 4-2). Since this research aimed to verify what the literature has about audit safety-critical software, we chose the ScienceDirect, Springer Link, IEEE Xplore, and ACM Digital database due to its relevance on computer engineering publications.

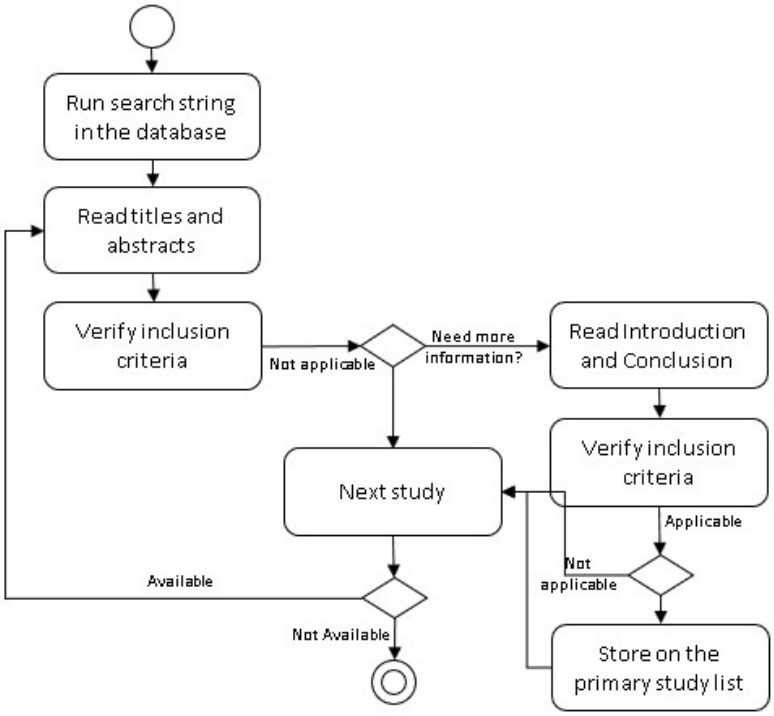


Figure 4-3 - Procedure for primary studies selection (Step 6)

4.2.4 Data extraction

The first phase, which consisted apply the search string in four databases selected, resulting in 780 papers selected for the data extraction (Step 7 of Figure 4-1), distributed in 209 papers from ScienceDirect, 85 papers from Springer Link, 382 papers from IEEE Xplore, and 104 papers from ACM Digital. The second phase was to verify which of these 780 papers would attend to the defined inclusion criteria. Only twelve papers from Science Direct passed through the inclusion criteria, and the same occurred for four papers from Springer Link, twenty papers from IEEE Xplore, and three papers from ACM Digital, making 39 papers, as represented in Figure 4-4.

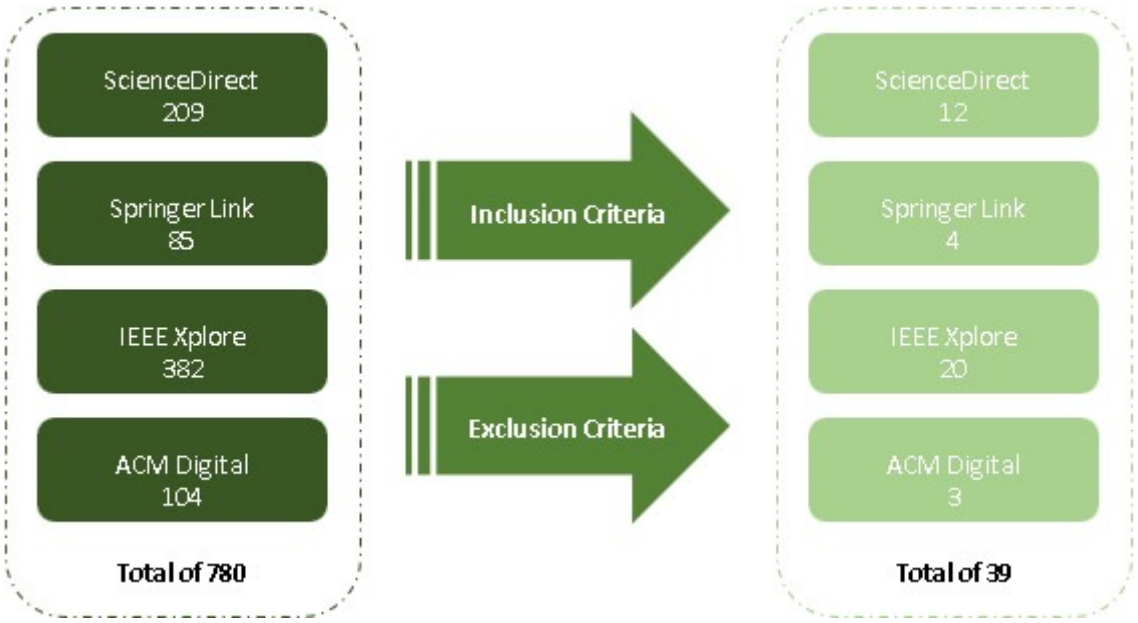


Figure 4-4 – Summary of papers selection

During the data extraction, it was defined questions and data that could help extract valuable data. Table 4-4 provides the complete set of questions and data.

Table 4-4 - Extraction data

Number	Study Data	Description	Relevant RQ
1	Study identifier	Unique identification for the paper selected	Paper Overview
2	Title, Year, Author, Country	Identify basic information from papers	Paper Overview
3	Study source	ScienceDirect, Springer Link, IEEE Xplorer, and ACM Digital	Paper Overview
4	Type of paper	Review article, article, conferences, chapter, research article	Paper Overview
5	Area	Aerospace, Automotive, Air traffic system, Military, Railway systems, and Medical devices	Paper Overview
6	Research Method	Case Study, Conceptual Analysis, Example, Industry Report, SLR, Survey	Paper Overview
7	Assessment methods	What methods have been proposed to conduct an audit in safety-critical software approved by a certification authority?	RQ1
8	Regulation standards	What are the standards used, if any?	RQ1
9	Assessment methods description	Is there any assessment method described?	RQ1
10	Certification	Do this paper is certification-related?	RQ1

11	Influence of development methods	How have software development methods been taken into consideration?	RQ2
12	Development methods used	What are the development methods used?	RQ2
13	Development methods indicated	Does the paper indicate any development method?	RQ2
14	Pros / Cons	What are the pros and cons of the approaches proposed for?	RQ3

4.2.5 Research Method

The studies were categorized (see list of papers selected in Appendix G) according to the applied research method and partially adopted from Martins & Gorschek [76] considering the following categories:

- **Case study**: the study empirically evaluates an approach or a theoretical concept in an industrial setting, having a clearly defined goal [77], [78], and [79].
- **Conceptual analysis**: the study aims to discuss a theoretical concept or a new approach, but without validating it with a case study or an experiment [76];
- **Example**: the study presents a new approach and conducts a preliminary discussion about it based only on the authors' assertions, or it provides some examples. Such a category complies with the "Assertion" method described in [77];
- **Industry Report**: the study aims to report industrial experience without declaring research questions or theoretical concepts [78].
- **SLR**: the study has as objective conduct a systematic literature review.
- **Survey**: the study collects data based on a questionnaire or interviews [79].

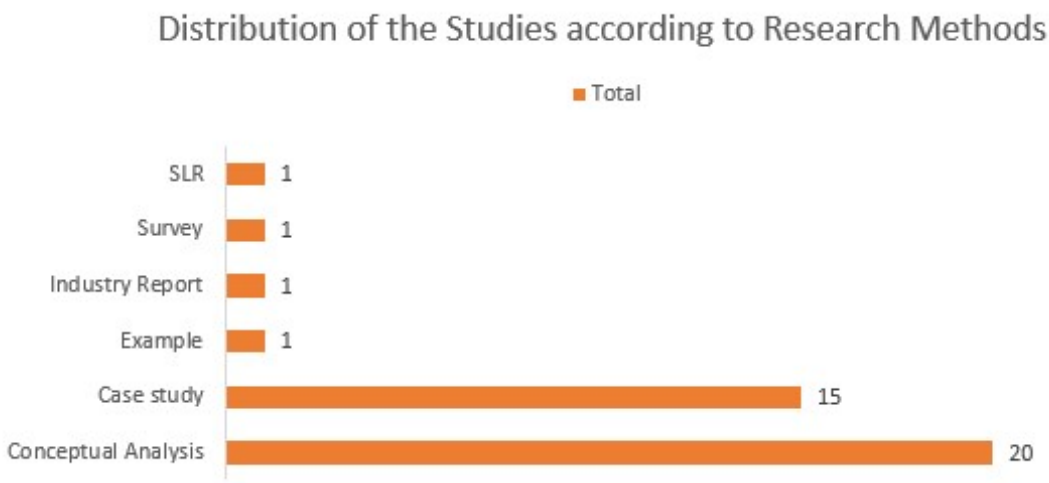


Figure 4-5 - Distribution of studies by research method

According to Figure 4-5, it is possible to verify that 90% of the papers selected used as research method Case Study or Conceptual Analysis. These research methods type indicates increasing the number of studies in the literature about audit and certification of safety-critical software and even the applicability of these studies in the industry application in the last two decades. Besides case study and conceptual analysis, the other categories defined presented only one paper each. Identifying only one SLR indicates that few of them are conducted in this area, and the one identified was more related to safety-critical software them the assessment process itself, nor surveys and industry report to help understand what the practices in the industry are, nor examples introducing discussions based on the authors' assertions.

4.2.6 Study quality assessment

The data extraction is a relevant part of the process but ensuring the selected papers' relevance is crucial. In this case, the study quality assessment (Step 8 of Figure 4-1) is significant to assess the quality of the selected primary studies, in a way to guide to allow the interpretation of findings and determine the maturity of the data chosen and to guide recommendations for further research [75].

Based on the recommendation of Kitchenham & Charter [75] and Unterkalmsteiner *et al.* [78], three (3) questions were created (see Table 4-5). These questions support the classification of Yes (it thoroughly answers the question), Partially (it has some relevant information, but it does not fully answer the questions, based on the researcher understanding), and No (it does not answer the question) to identify the relevance of each paper for this SLR.

Question QA1 aims to check if the papers are clearly understandable and if was clearly explain the proposed method. Question QA2 checks if the paper's goal is clearly defined and understandable. Question QA3 intends to verify if the discussion of the relevant findings occurred in the paper. Question QA4 verifies if the discussion of regulation standards occurred from the software assessment method proposed in the paper. Finally, question QA5 intends to verify if the paper increased the knowledge about safety-critical software assessment. Based on the results, we found a lack of discussion about the audit of safety-critical software in the academic environment. However, this is starting to change since it is already possible to find a few papers that discuss this theme more clearly.

Table 4-5 - Study quality assessment results adapted from [78]

ID	Quality Assessment Questions	Yes	Partially	No
QA1	Is the presented approach clearly explained?	0 (0%)	20 (51%)	19 (49%)
QA2	Is the goals of the research clearly defined?	37 (95%)	0 (0%)	2 (5%)
QA3	Is there a discussion of the relevant findings?	34 (87%)	0 (0%)	5 (13%)
QA4	Are the regulation standards discussed in the light of the safety-critical software assessment method proposed in the study?	18 (46%)	5 (13%)	16 (41%)
QA5	Is the study significantly increase the knowledge about safety-critical software assessment? (adapted from [80] apud [81].	12 (31%)	8 (21%)	19 (49%)

4.2.7 Threats to validity

According to Kitchenham & Charters [75], publication bias refers to the problem that positive results are more likely to be published than negative results. The concept of positive or negative results sometimes depends on the viewpoint of the researcher. The selected search string was carefully tested against two digital databases to avoid an unreasonable search. It chose the option which brings more results to allow increasing the number of selected papers. Also, once the research string was defined, two digital databases were added to ensure that various scientific media vehicles, including the conferences and journals, could be considered for this SLR.

4.3 Results and Discussion

The main objective of this SLR was to verify what the literature discusses regarding auditing safety-critical software and which methods they are proposing. Initially, it was expected to verify (or discover) different methodology concepts to improve the actual safety-critical software audit method. Nevertheless, only 5% of the papers describe any safety-critical software audit method, and 69% of the selected literature partially addresses one of the research questions and being that most of them the RQ1.

On the data analysis (Step 9 of Figure 4-1) performed, we verified the lack of discussion about the assessment subject for conducting an audit. Instead of it, safety assessment corresponds to 46% of the papers. Safety assessment is a method to evaluate safety-critical software. The information was more related to safety analysis and techniques to conduct it, but this was not the audit method expected for this SLR. Besides, it indicates that the more restrictive terms in the search string were “assessment” and “certification”.

Another relevant finding is that 41% of the papers discussed the certification/approval topic extensively. Which is relevant information for this SLR because the purpose of certification/approval activity in the aeronautical industry is to

establish a confident level of safety that must be checked for compliance with aircraft regulation [82], and a way to perform this check is conducting an audit. This concern exists in the aeronautical industry and other regulated environments governed by standards.

Although it was defined as inclusion criteria for a period from 1999 until May 2019, it was observed in Figure 4-6 that the assessment/certification topic is increasing over the years as a consequence of the increase in the extensive software usage for aircraft, engines, and airborne equipment [83] (apud [84]). Also, certification authorities like US FAA and the Union Aviation Safety Agency (EASA) have issued some standards, guidelines, and reports related to certification and other aspects of software assurance, such as licensing, qualification, or validation, in their specific areas of interest [85].

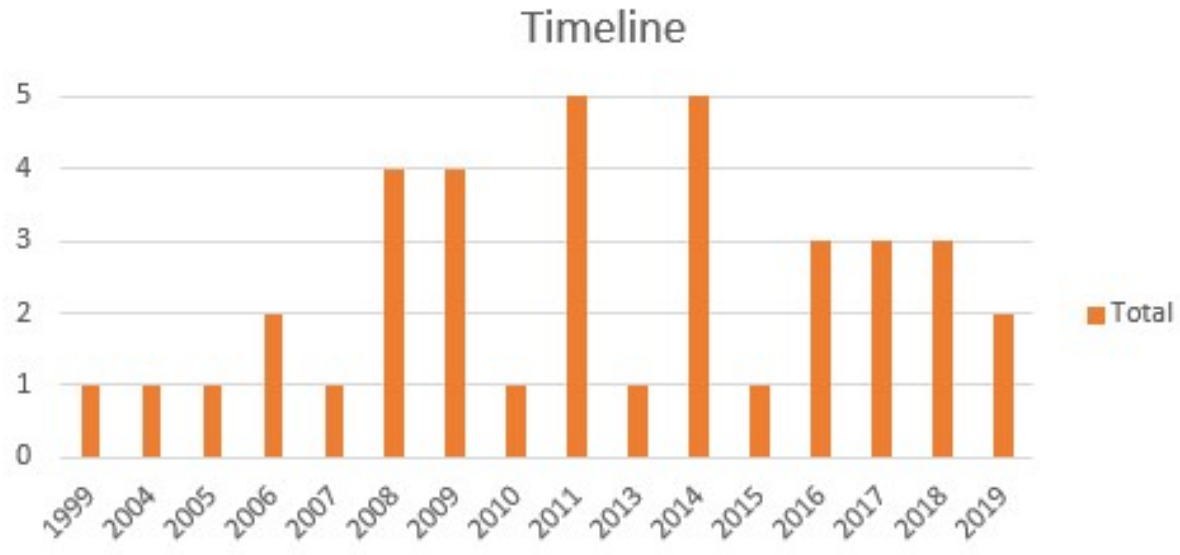


Figure 4-6 – Papers publication timeline

Another analysis performed was the authors' nationality to verify where this subject has high relevance. Based on Figure 4-7, it is possible to notice that 33% of the authors are from the USA, which is comprehensive once the FAA is one of the primary references to discuss standards and approval process. Except for Australia, the following author's country after the USA is from Europe (UK, Sweden, Spain, Germany, and Portugal) with 49% of the papers, which is also justified once EASA is another primary reference to discuss standards and certification rules.

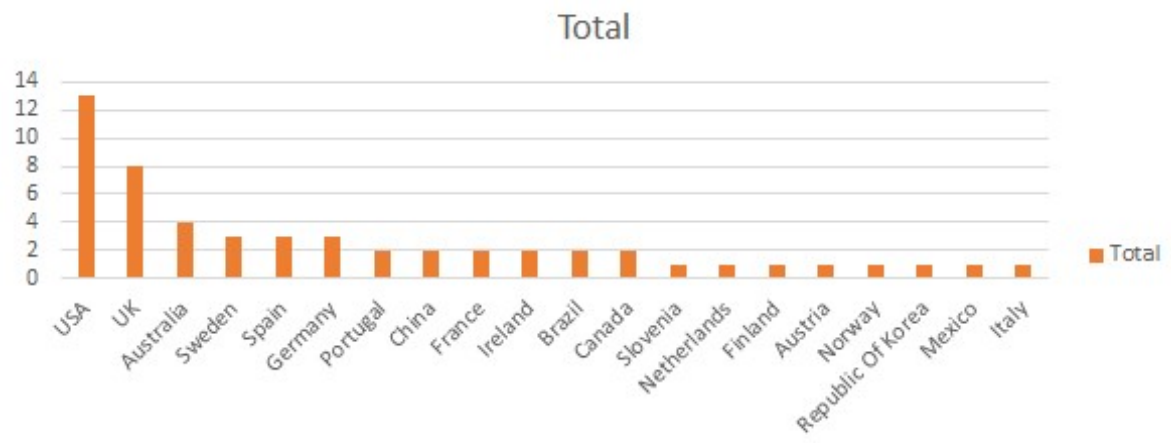


Figure 4-7 - Distribution of author's nationality

Figure 4-8 presents an analysis to verify if the papers that indicated one or more assessment methods also indicated that this would be used to certify or approve a software product, considering the possibility of not indicating an assessment method but discussing the process certify or approve a software.

According to this figure, it is possible to evaluate that although 48% of the papers selected do not have defined an assessment method, 42% of them have any relationship with the certification process/method. These numbers allow an understanding that it is unnecessary to have a defined audit method to discuss the certification process. However, 35% of the papers that discuss the certification/approval process also presented an assurance method. It is impossible to state which audit method is the most appropriate, as we found eleven different methods mentioned.

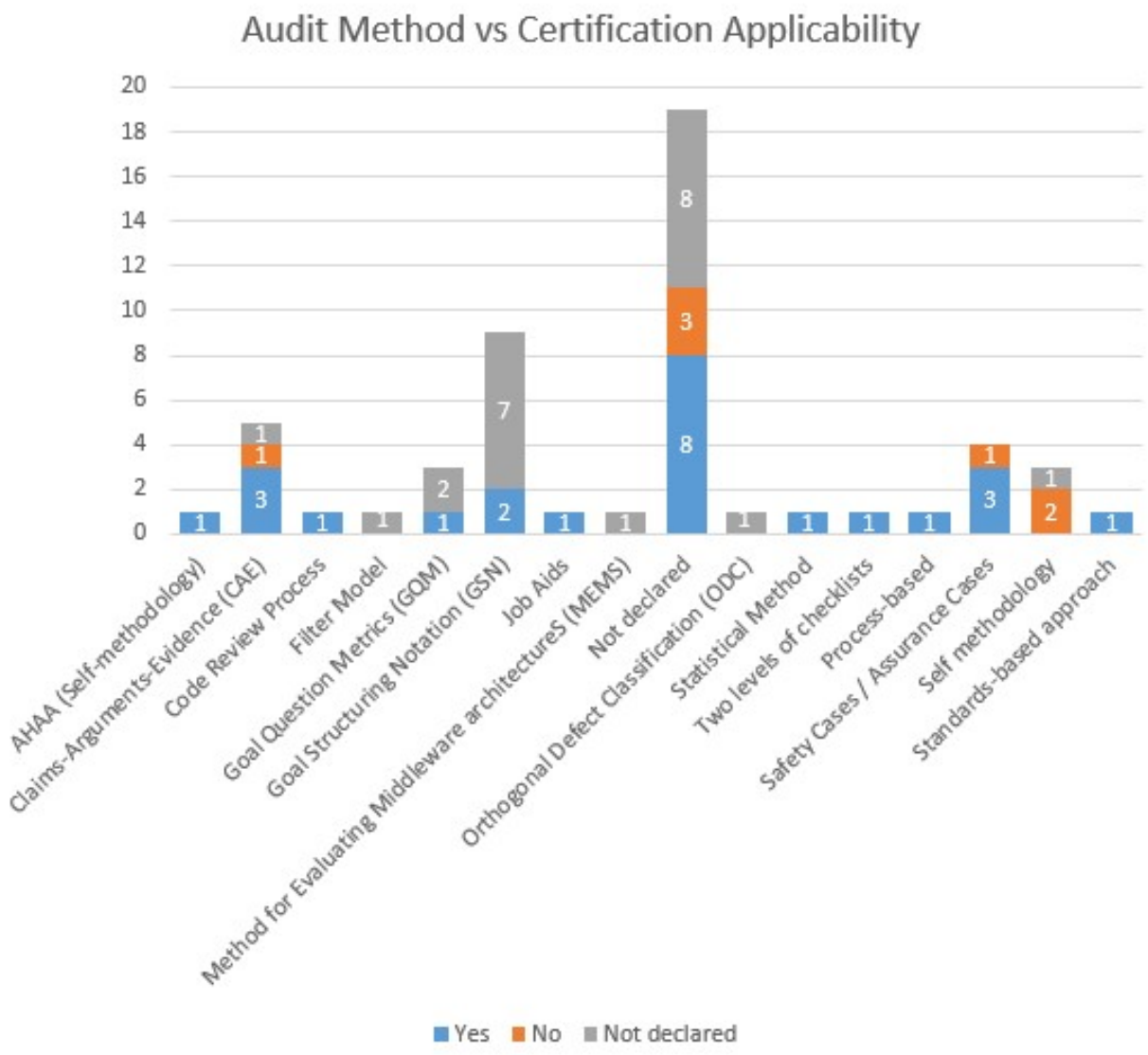


Figure 4-8 - Distribution of audit method per paper's certification applicability

4.3.1 Approaches for safety-critical software audit (RQ1)

The research question RQ1 aims to search and identify the methods proposed in the literature to conduct an audit of safety-critical software. From the 39 papers selected in the SLR, it was possible to identify 11 different methods (Figure 4-9) to conduct an assessment. However, in 40% of the papers, no audit method was cited. The most adopted technique was the GSN language, found in 17% of the selected papers for this SLR. The GSN language was associated with another method (e.g., CAE framework or GQM approach), in some cases, as a way to complement the

assessment. Due to this association with the GSN language, the CAE framework is the third approach most used with 8% of the papers. Also, the safety cases/assurance cases appear in 8% of the papers as a third approach most used too.

The GSN language, CAE framework, and GQM approach are techniques used to create the goals and arguments used by the safety cases/assurance cases method. It was identified separately in Figure 4-9 because the papers that indicated the use of one or more techniques did not explicitly mention the process of safety cases/assurance cases. However, this indicates that the method of safety cases/assurance cases is one of the most extensively used for audits that aim to ensure the safety aspect of the developed software.

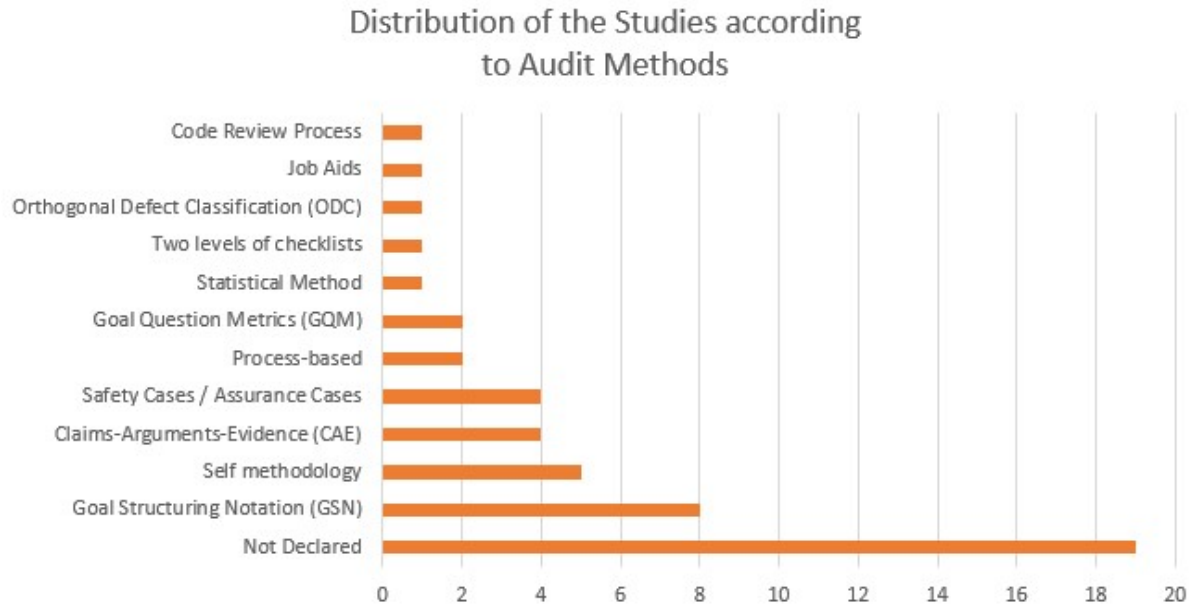


Figure 4-9 - Methods proposal for Assessment/Audit

The second approach is the Self methodology, with 10% of the papers. For these cases, the first identified method proposed was the Filter Model, where the certification process itself is a safety-critical [63]. The second method associates the web modeling and testing techniques, allowing them to ensure the functionality and reliability of web components and their applications [72].

The third method classified as self-methodology is the MEMS, which is used to measure middleware architectures by rating multiple quality attributes [86].

The fourth method classified as self-methodology is the AHAA, which means agile, hybrid assessment method for the automotive industry. According to Caffery,

Pikkarainen & Richardson [87], it supports the integration of agile practices using more traditional practices guided by a plan associated with SPI in safety-critical companies.

The fifth method proposed, according to Sagoo [88], is called the assurance approach. Based on systematically attempting to gain assurance in legacy Programmable Electronic Hardware (PEH) by investigating what evidence is available and forming an assurance argument, including the evidence generated. This approach essentially commences with identifying and assessing the available evidence against RTCA DO-254 and using the notion of safety nets and formal modeling to gain credible assurance evidence.

The papers selected allowed to identify eleven different methods where at least two of them, the CAE framework and the GQM approach, were associated with the GSN language. The code review process method is already predicted by DO-178C, for example, to fulfill one of its objectives and part of the expected process. The Job Aid is an FAA guideline set of worksheets used to audit a project in a series of four SOI audits. Also, the use of Two levels of checklists is a similar strategy to the job aid, created to fulfill all RTCA DO-178C standard objectives.

The Statistical Method is used by Dodd & Habli [61], which defines a method associated with the GQM approach. It collects and analyzes live data from aerospace software projects to assess a project's readiness for an SOI audit by comparing the project's data against historical data collected from past projects, which successfully passed the SOI audits. Finally, the Orthogonal Defect Classification (ODC) identifies a limited number of defect types and the relevant trigger (such as testing, analysis methods) for each defect analyzed. The results can be used for statistical quality control and in-process monitoring and reliability assessment [71].

Another method that is also associated is the Process-based method. According to Hatcliff *et al.* [89], the most common approach for a standard to specify criteria for processes or process characteristics used to support the creation and evaluation of a system. The idea being that while specific safety-related attributes may vary across systems, the processes identified and addressed can be relatively uniform. Standards and associated certification regimes that follow this approach are often termed process-based. Additionally, according to Rushby [48], the Standards-based approach is a method where the applicant is recommended or required to follow specific guidelines and standards; it is also similar to safety cases.

The last method to be discussed is the Assurance Cases, which includes the Safety Cases, which explains that Figure 4-9 represented these two techniques into only one audit method. According to GSN Standards [69], the assurance case is a central and stated argument supporting a set of evidence. These data indicate that a system, service, or organization can operate as intended for a specific application in a stable environment “In practice, an assurance case may have a particular focus. For example, a safety case intent to demonstrate that a given system is acceptably safe in a given context, while a security case intent to justify the security properties of a system.”.

Besides, it was identified that some papers that provided any audit method usually also referenced some standards based on the applicability (e.g., aerospace, automotive, medical). Figure 4-10 shows the distribution of the standards per paper. Usually, the papers refer to more than one regulation due to the different applicability. However, most of the mentioned standards were the DO-178B (26%) followed by DO-254 (16%) and DO-178C (14%), which are associated with software and hardware developments for aerospace and have well-defined certification authorities to evaluate the development prior certify an aircraft. Further, only seven papers (18%) did not indicate the standard's applicability because one was related to systematic review literature, which means a lack of an audit method was proposed. The other five papers were related to audit methods that were proposed independently of a specific standard.

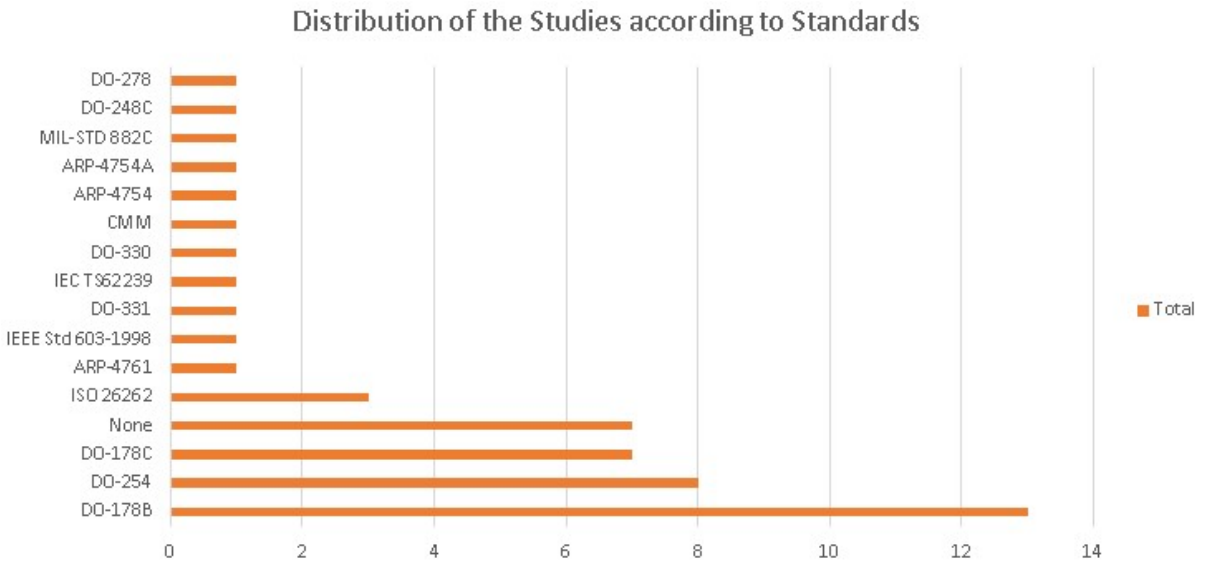


Figure 4-10 - Distribution of the Studies according to Standards [2], [3], [96]–[99], [13], [23], [90]–[95]

Figure 4-11 presents an association between the considerations of an audit method and the standards in the selected papers. To better comprehend the graphic, it was considered papers that discussed software approval by certification authorities subjects and ways to achieve compliance with the standard objectives. However, they do not explicitly indicated an audit method, and it was classified as “Not Declared” during data extraction for audit method classification. On the other hand, papers that discussed safety-critical software development and ways of obtaining assurance, but did not specify a standard, were considered “None” for data extraction related to the standard.

These classifications allowed the researcher to identify whether the audit’s method use would be mandatory to ensure compliance with the regulatory objectives and, if so, which would be the most used according to the literature. However, another aspect that it was able to assess after data extraction was whether the academia discussed any audit method without necessarily relating it to one or more specific standards. Another relevant finding is that most of the selected papers discuss developments related to the aeronautical industry and its related standards when discussing safety-critical software. It is possible to explain that the aeronautical world has well-established regulatory agencies and certification requirements that must be satisfied with airborne software. This explains why Figure 4-11 presents more methods associated with aeronautical standards (e.g., DO-178B, DO-178C, DO-254, DO-248C).

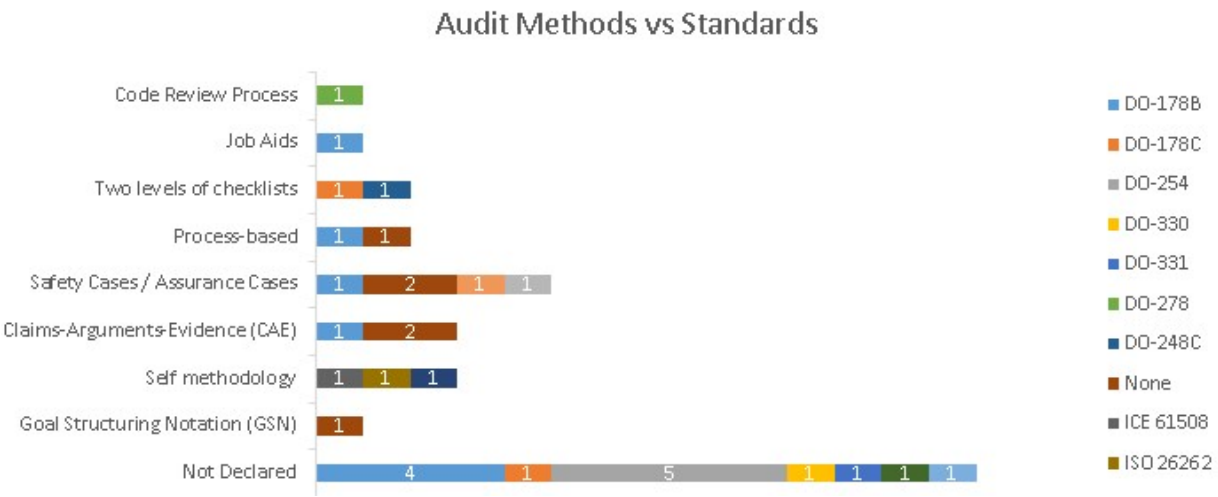


Figure 4-11 - Distribution of the Standards according to Audit Methods [1]–[3], [13], [90], [91], [99]

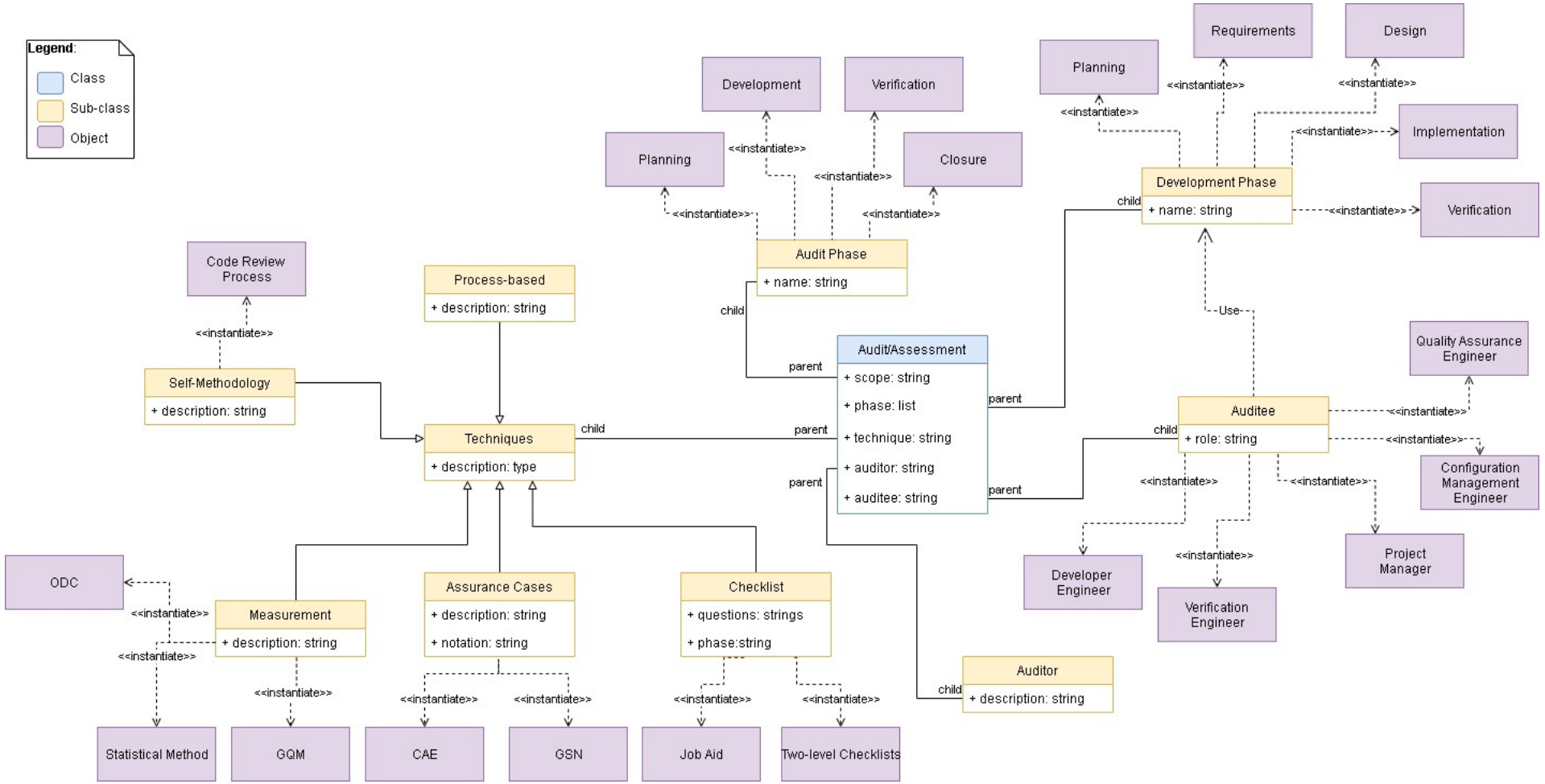


Figure 4-12 - Taxonomy for safety-critical software assessments

In Figure 4-12, the taxonomy is identified as a result of the research conducted for this SLR. During the data extraction step of the studies selected, we recognized five categories of Audit/Assessment: Techniques, Audit Phases, Development Phases, Auditor, and Auditee. A company or a software team can represent the auditee. The following roles usually participate in the audit, depending on the audit scope: developer engineer, verification engineer, project manager, configuration management engineer, and quality assurance engineer.

Five items are part of the development phase. The auditee acts as an actor since its process is the one who determines how these phases are detailed and gives a direction with the auditor should conduct the audit. The audit phases are based on the paper's author's expertise, but it is the most common audit division based on the software development definition. A total of 48% of the studies did not indicate a specific assessment method; however, the other 52% presented techniques categorized as measurement, assurance cases, and checklists.

The technique is a sub-class of Audit/Assessment and intends to present the techniques identified through the SLR conduct. In this case, it was sub-divided into the following sub-classes: checklists, assurance cases, measurement, self-methodology, and process-based. The measurement technique reported by 10% of the studies is the GQM, which was associated with the GSN technique in one of them. The other was associated with a self-methodology proposed. 60% of the studies used the assurance case technique. We identified that some papers used the CAE framework, others used the GSN notation, and sometimes both were discussed in the same study. Two papers use the checklist technique, one of them proposed by a regulation authority (Job Aid), and the other presented the Two-Level Checklists. Some papers proposed self-methodologies, such as the Code Review Process and process-based techniques, to ensure software safety-critical development.

4.3.2 Development methods were taken into account (RQ2)

The second research question’s objective - how the software development methods have been taken into consideration - is to identify whether any proposed audit method takes into account any method of software development or whether

they are independent of the defined life cycle. See Section 2.1.2 for more details about the software development model types.

It was identified that five (13%) papers were declaring any software development method (waterfall, v-model, agile, spiral, and incremental), which answers the question RQ2. From these five papers, 60% considered the V-Model method, and the other 40% at least the Waterfall method. The other thirty-four (87%) papers selected for this SLR, do not have references about a software development method. This lack of reference may indicate that this information does not impact the safety-critical software audit method established. The safety-critical software audit method is flexible enough to be adapted to any software development method used.

Figure 4-13 indicates that there is no relationship between the audit methods and development methods identified in this research. 79% of papers have an audit method disclosed but do not have any development method related. Also, 21% of the papers have declared a development method but did not have any citation to an audit method.

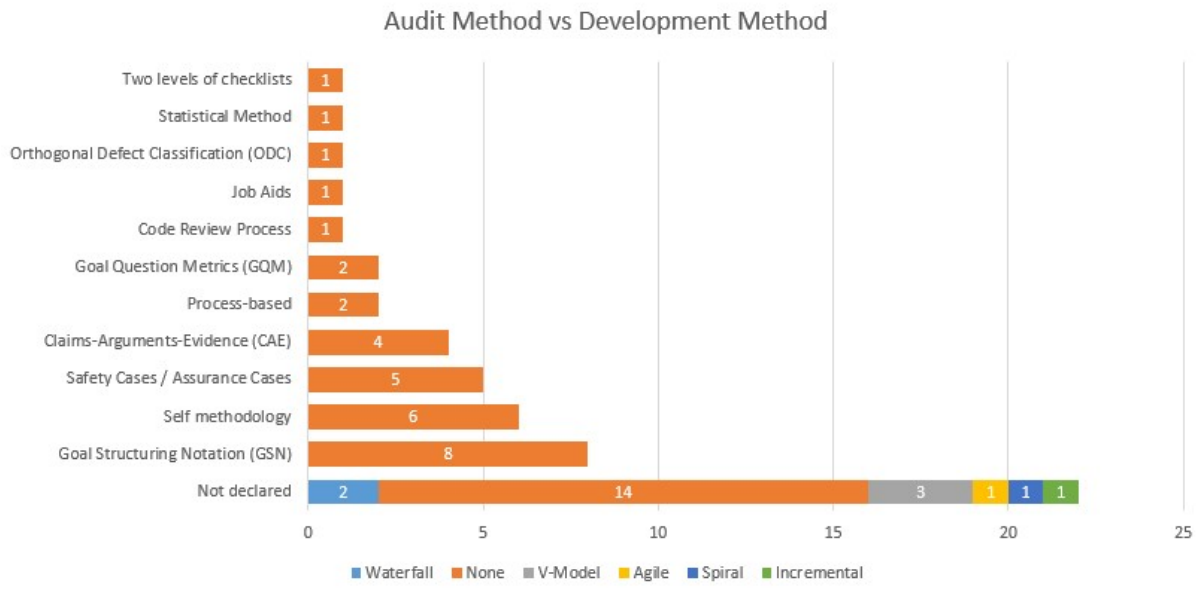


Figure 4-13 - Relationship between Audit Methods and Development Methods

4.3.3 Pros and Cons (RQ3)

The simple definition of pros and cons would be, in this case: something that has advantages or disadvantages based on the necessity or applicability of the audit

method. In this SLR, we found nine (9) different methods that could be implemented to conduct an audit, but it is essential to notice that they have different applicabilities. The description of each method's pros and cons is difficult because 66.7% of them are related to safety assessment and are pretty similar in their definition process, which means that they use analytic and statistic data to obtain the result. The other 33.3% are methods that can be easily applied to software audits and based on checklists and questions created to evaluate a defined objective. The answers could be yes, no, or not applicable.

As summarized in Table 4-6, for methods classified as self-methodology (Filter Model, MEMS, AHAA, among others), it was not performed an analysis of pros and cons. These papers identified did not provide enough information to establish a systematic comparative analysis among them, and there is not a relevant number of papers about these methods. Job Aid is a method primarily used in the aerospace industry. It provides guidance based on RTCA DO-178B, specifically for safety-critical software. FAA created the Job Aid to allow their representatives to have a guide to conduct the certification authorities' audits. However, this guidance is no longer available on the FAA website, which means that this method is restricted to be used and cannot be used to propose a new audit method.

The Two levels checklist is a method similar to Job Aid. However, it contains a guide through a paper that took into consideration the RTCA DO-178C. The first level is related to the software level applicable and dedicated to verifying the compliance with each applicable objective based on its description. The second level proposed is related to check compliance with the activities related to each applicable objective at the first level. However, it necessary to fill two checklists to assure full compliance with the applicable standard.

The Statistical Method collects and analyses live data from software projects to assess its readiness for a SOI audit based on the project's history. In the paper, the authors defined 15 metrics and used the GQM approach to refine and encode these metrics. However, to apply this technique, it is necessary to collect the data from software development and maintain data from past successful projects.

The most used method in the papers selected by this SLR is the GSN language. Because it allows determining goals and identifies its evidence, evaluating if the goal defined has been fulfilled or not, most of the papers this method for safety assessment and associated with CAE framework and GQM approach. However, it

was not possible to identify examples of applying this method in a safety-critical software audit.

The process-based method, as discussed by Hatcliff *et al.* [89] it is a standard that defines criteria to identify system hazards, a link between software requirements and system hazards, requirements traceability, and verification techniques against safety requirements to generate and evaluate a system. Besides, these standards treat the compliance subject to confirm the existence and completeness of the process-related artifacts. This method defines the minimum requirements necessary for a product to be certifiably safe. However, academia and industry agree the software assurance case must go further than demonstrate compliance to process-based criteria and must focus on the product itself.

The Assurance Cases/Safety Cases, according to Hawkins *et al.* [66] it is extensively used in the safety domain, and the most common term is assurance case. The assurance case term is used to generalize this method and to allow incorporate other attributes. Besides, the assurance arguments enable the software developer to improve the understanding and demonstrate the adequate assurance of software development. Nevertheless, the assurance cases/safety cases are related to the software's safety aspects, which may weaken assurance in other aspects of the software product if not correctly used.

Table 4-6 - Summary of pros and cons (RQ3)

Method	Pros	Cons
Job Aid	- Focused on the aerospace industry. - Questions for each software phase	- No longer available to be used as a basis for audit.
Two Levels Checklist	- Focused on DO-178C and DO-278A objectives and software levels. - It ensures the objectives and related activities.	- Necessary to fulfill two checklists to ensure compliance.
Statistical Method	- Project's data to assess readiness. - Successful project data to compare and perform the analyses.	- Not possible to use this method in an ongoing project. - Not clear how to get a statistical database.
GSN/GQM/CAE	- Goals to identify evidence to ensure compliance.	- Used for Safety Assessment, but not identified in examples for safety-critical software audits.
Process-Based	- Criteria for identifying system hazards - Compliance based on completeness of process-related.	- Focused on process-based criteria. - Users tend to define only a minimum requirement set.
Assurance Cases	- It allows developers to clearly and	- For the papers selected it is focused

Method	Pros	Cons
/ Safety Cases	systematically communicate why it is believed that there is sufficient assurance in the software [66].	only on safety aspects (from the point of view of security).

Based on the detailed methods described in the selected papers, Table 4-7. presents the comparison among the standards of the number of objectives that must be achieved based on the software failure condition. By analyzing the numbers, it is possible to verify that there are five additional objectives between RTCA DO-178B and RTCA DO-178C. Once the RTCA DO-278A was created based on RTCA DO-178C, but to attend the necessity of software in not airborne systems (ground equipment), which produced a slight difference between the number of objectives.

Table 4-7 - Comparison of the number of objectives among standards

Failure Condition	Software / Assurance Level	DO-178B	DO-178C	DO-278A
Catastrophic	A / AL1	66	71	71 + 14 (COTS)
Hazardous	B / AL2	65	69	71 + 14 (COTS)
Major	C / AL3	57	62	69 + 14 (COTS)
Minor	D / AL4	28	24	46 + 14 (COTS)
No Safety Effect	E / AL5	None	None	26 + 14 (COTS)

Considering this analysis, Table 4-8 presents an additional analysis correlating the methods and whether it is possible to adapt them to other standards based on the number of objectives of each standard and the similarity among them. Additionally, the RTCA DO-248C was not considered because it is a standard created to provide frequently asked questions, discussion papers, and rationales.

Although it was not considered the safety cases/assurance cases in the analysis of Table 4-8, according to Denney, Pai & Habli [100], the use of this type of method not only achieves the applicable standards objectives and software's correctness. It goes beyond allowing software safety when correctly used and considers all necessary aspects of the software development

Table 4-8 - Audit methods and applicability to some standards

Method	DO-178B	DO-178C	DO-278A
Code Review Process	By considering the similarity of the seven objectives, the additional two objectives were explained on RTCA DO-178C; are possible to adapt	By considering the similarity of the nine objectives, it is possible to adapt this method to be used in a RTCA DO-178C development.	Created based on RTCA DO-278 and specifically for the Coding Process, which encompasses nine objectives. The author also

Method	DO-178B	DO-178C	DO-278A
	to use with RTCA DO-178B.		proposes the use of a tool to conduct code inspection.
Job Aid	Created based on RTCA DO-178B, it allows evaluating all the 66 objectives and ensuring full compliance.	Considering the similarity of the objectives and considering there are five additional objectives, it is possible to adapt this method. However, it will not be possible to ensure full compliance once it would be necessary to create further questions.	Considering the similarity of the objectives and considering there are twenty-one additional objectives, it is possible to adopt this method. However, it will not be possible to ensure full compliance once necessary to create further questions.
Two Levels Checklist	By considering the similarity of the objectives and the difference of 5 additional objectives, this method fully complies with this standard.	Created based on RTCA DO-178C and RTCA DO-278A, it allows evaluating all the 71 objectives and ensure full compliance.	Created based on RTCA DO-178C and RTCA DO-278A, it allows evaluating all the 85 objectives and ensure full compliance.

4.4 RSL Final Considerations

The purpose of the systematic literature review was to investigate the existing auditing methods proposed in the literature, verify whether or not to consider the types of software development methods, and understand the pros and cons of each approach. This review aims not to propose a new method or even to indicate which one is the most appropriate because choosing one method depends on its application, but rather to list the practices found in the literature.

Safety-critical software audit. This systematic literature review identified at least eleven different methods to conduct an audit, depending on the defined scope. Mainly because two types of audits identified were: one related to safety assessment and another to comply with standards, as well, it was not explained in detail how to use each of them. Instead, the papers provided a brief overview of the audit methods or techniques to ensure compliance and a reference for the related papers. The audit method most identified in the papers selected in this systematic literature review are those related to safety assessment, as this topic is widely discussed in academia and even by the industry, having seen the number of papers we can find in academic databases. Also, the necessity of ensuring software development through software quality assurance activities is a topic that has been growing in recent years, as it is

possible to see in Figure 4-6, given the complexity of the software being increasingly increasing with the complexity of the systems.

Relevance of audit subject. The purpose of this thesis was not to assess which is the best method. However, it became apparent the increase in the number of papers related to this subject over the years. One of the reasons that could explain this increase is the evolution of safety-critical software complexity today and the necessity for regulatory agencies to ensure the minimum safety and security required. For the papers that discussed this subject, it was noticed that they were more related to software auditing aimed at the certification process. They are also more focused on seeking compliance with the objectives of development processes,

Figure 4-9 provides one of these indications, where nineteen papers selected discussed the audit subject but not specified the audit method. The focus was more on evaluating the final product as a whole and on its safety level of confidence, even if it was an evaluation in stages. One additional relevant information is the nationality of the papers' authors; they are concentrated in the USA and European countries, as presented in Figure 4-7. The two most known regulatory agencies (FAA and EASA) are justified by the two most known regulatory agencies and used as a reference by the other agencies. Another relevant piece of information identified was that almost all papers do not discuss the assessment phases in detail. In the papers, we did not identify a recommended division to split the assessment and evaluate the entire software.

Relationship with certification aspects. The systematic literature review indicated that certification aspects are a subject discussed mainly in the past two decades. Sometimes it associates the issue with standard, and at other moments discusses only the necessity to comply with certification objectives. This information leads us to conclude that a certification process can be addressed without associate an assurance method. On the other hand, those discussions that relate a method to achieving a common objective are more explicit. Figure 4-8 shows an indication of this, which allows us to see the relationship between what we classify as audit methods versus certification applicability.

Relationship with standards. At least 50% of the selected articles (see Figure 4-11) brought discussions about aeronautical standards and processes; a possible explanation for this is that this industry has well-defined and established certification authorities to evaluate the development prior certify an aircraft.

Additionally, these standards guide the development of safety-critical software to achieve a set of objectives that have been established, making it easier to carry out the compliance assessment of what has been developed (see Figure 4-13).

Relationship with software development methods. The purpose of verifying the connection of the proposed audit methods to the development methods used for software development is to confirm whether there is any relevant impact on the audit method that should be taken into account when proposing a new method or adapting an existing one. Despite that, the papers discussing some software development models, such as Waterfall, V-Model, Spiral, Incremental, none of them presented a clear association with how to perform the software audit that is under development. As most of the audit methods presented evaluate software development in phases (e.g., development, verification), the software development model does not significantly impact the strategy of conducting this activity. No paper has indicated a relationship between the two methods, which leads us to believe that the software audit methods found in this SLR can be adapted to any critical software development.

Past works related to SLR. The search for studies occurred in 2019 and used the IEEE Xplore, Science Direct, Springer Link, and ACM Digital databases. Even using different search strings from the one proposed in this systematic literature review, we did not find an SLR specifically conducted to verify audit methods. 66.7% of the papers are related to safety assessment. This data indicates that papers usually related to safety-critical software having “assessment” as a keyword increases the chance of talking about a safety analysis.

Future works and next steps. We suggest writing papers that discuss and propose audit methods used by the industry and their relationship with the standards identified in this systematic literature review. In this way, it would be possible to share how this audit method helps evaluate the compliance data and tailor them to similar standards.

The results of this systematic literature review (step 10 of Figure 4-1) should provide an overview of the existing literature about the audit method for safety-critical software. However, it was observed that performing an audit intended to obtain aircraft certification is quite a new subject and is not yet widely discussed. This information encourages further research into this area, as this subject is increasing

its relevance in the industry, not just in aerospace but also in automotive, medical, and other areas.

Chapter 5

SOFTWARE AUDIT MODEL

This chapter intends to present the proposed software audit model and details each of its phases and activities, including an initial questions database that accomplishes RTCA DO-178C and IEC 62304 standards. It also provides a set of recommendations on creating questions for standards to be used with the software audit model.

5.1 Requirements for software audit model

It is necessary to define the functional requirements to create a software audit process as a first step. According to Westfall [101], it elicits requirements to determine the functionality or capabilities that must be built into the software product to enable users or other stakeholders to accomplish their tasks, thereby satisfying the business requirements. Before detailing the proposed software audit model, the main requirements for this audit model are identified to allow a clear understanding of what is expected and what is needed to develop it. Appendix A presents the elicited requirements for the software audit model.

The requirements were classified as high, medium, and low based on their priority in the defined relevance for the software development model and the author's several years of experience. The foremost requirements define what is expected based on a software process development and their related development phases (Planning, Requirements, Design, Implementation, Verification, Validation,

Configuration Control, Quality Assurance, Tool Qualification, and Closure). All other elicited requirements detail what is expected from each phase to be assessed.

5.2 Software audit model

As depicted in Figure 5-1, guidelines for the audit preparation, receive documentation, evaluate documentation preliminarily, perform the audit, and follow-up are detailed with recommendations on how to execute each phase. The inspiration for this process was the Job Aid, which also split the audit process into these four main phases. However, the Job Aid predicted 4 specific stages of involvement (SOI), and the SAM does not predict it. It gives the auditor the freedom to evaluate specific activities as per their defined scope.

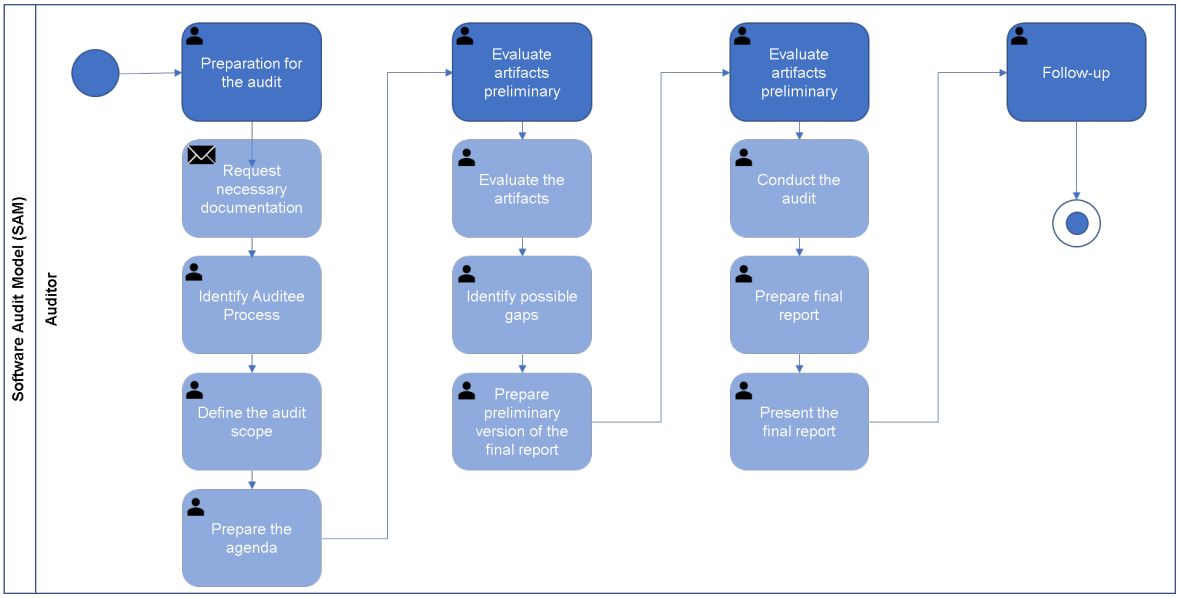


Figure 5-1 – Main phases of the software audit model inspired in the Job Aid

Another essential characteristic is that the entire proposed process can be tailored based on the software project's characteristics or auditor necessities. It depends on the audit type intended to execute and if it is a company independent auditor or an internal auditor, also known as software quality assurance auditor.

5.2.1 Roles and responsibilities

The following roles and responsibilities are involved in the execution of a software audit model:

Auditor: it is the person responsible for prepare, execute, and perform the follow-up of an audit. He must know the software development process that will be assessed to identify the gaps and, eventually, the issues against the standard or related requirements. It is a person who must have communication, analytical thinking, and observer skills, assess body language, have experience in various developments and processes to have diversified knowledge when conducting and analyzing audit findings.

Auditee: it is the person or company that develops the safety-critical software that will be assessed. It is responsible for providing the necessary information related to the software life-cycle data and answer the questions when needed. A group of people can represent it from the following teams: a software project manager, a software development team, a software verification team, a software configuration management team, and a software quality assurance team.

Software project manager: it is the person responsible for managing the entire software development from the schedule, resources, budget, and discuss commercial issues when necessary.

Software development team: it is the team responsible for developing the software itself, writing requirements, and implementing them on source code based on the software development process established.

Software verification team: it is the team responsible for verifying the software against the defined requirements and applicable standards.

Software configuration management team: it is responsible for the configuration control by creating and maintaining software development and verification environments.

Software quality assurance team: it is responsible for assuring that the software was developed following the applicable software process and standards.

5.2.2 Preparation for the audit

Before executing the audit, it is necessary to prepare for it by knowing the process under audit, the applicable standards, request the necessary documentation, understand the scope of the assessment, define the relevant questions, and prepare the agenda.

Request necessary documentation

It does not matter what the defined scope to be audited is; the first step is to require the necessary documentation for the auditee and its related development status. If it is the first contact with the auditee, it is necessary to request all the planning artifacts because it defines the safety-critical software development process. For the phases related to development and verification, sometimes it is impossible to receive the artifacts in advance due to confidentiality issues. However, at least the artifacts' status is necessary to know to establish their maturity and availability for the audit.

Identify the auditee process

Before executing an audit, it is necessary to understand the process used to develop safety-critical software. The practical way to do this is through the software plans or equivalent document responsible for establishing the development process. These plans or documents shall contain:

- The chosen software development model (waterfall, v-model, spiral, incremental, or any other).
- The organizational structure and the involved roles in the software processes.
- The process phases defined and detail each of them, including how to execute.
- An indication of applicable standards and how to accomplish them.

After the auditor is familiar with the software plans documentation, he can assess these documents to guide his audit and verify if compliance with the applicable standards is achieved or if there is any gap that must be solved.

Based on the process and documents available, it is essential to define the audit strategy used to conduct the assurance activity. An audit can be conducted in different ways, depending on the information the auditor has available and the intent of the activity. Below are exemplified some audit lines.

Desktop review: It can be conducted remotely and only requires the auditee to provide the auditor's necessary documentation to be inspected. This type of audit allows the auditor to evaluate the artifacts by himself, and if any question is raised, it can be clarified through e-mail or any other communication tool established by the companies after the auditors' inspection.

Evidence demonstration: Due to its nature, it requires the auditee to present the artifacts and related evidence to allow the auditor to evaluate compliance with the applicable standards. The evidence demonstration is also necessary because the artifacts that must be evaluated require a walkthrough among them and require different teams to attend the audit to clarify any eventual questions generated in the sample selected for the audit. There are two ways to perform this audit meeting:

- **On-site**: It requires the auditor to go to the auditee facilities to evaluate the artifacts required to fulfill the audit intents.
- **Remotely**: It can be conducted using some conference tool chosen by the auditee or by the auditor, but requires more attention of the auditor to evaluate the artifacts and answer the selected questions. In addition, in this modality, it is necessary to verify if the auditee has any restriction in presenting the audit data through a conference call due to its intellectual property restrictions.

Define the audit scope

Once the auditor understands the auditee software development process and the applicable references (e.g., standards, contracts, regulations), it is necessary to define the audit scope. The audits can be performed for each safety-critical software phase (planning, requirements, design, implementation, verification, closure) or grouped based on the necessity of the assessment requested. Usually, the planning

audit is the first to be conducted since it is also used to know and understand the process utilized to develop the software.

Commonly the audits are grouped into planning, development (requirements, design, implementation), verification, and closure, but this is only one of the possibilities. The safety-critical software development schedule advance and budget are data that helps to determine the scope of the audit because it allows the auditor to understand what is already done or not. Besides, by exchanging information with the auditee, it is possible to understand its safety-critical software development status and thus begin negotiating the audit scope.

It is essential to consider that the budget can influence safety-critical software development due to resource allocation to perform the development, influencing the schedule's progress. It is a relevant aspect to perform the audit, especially if it is an evidence demonstration conducted on-site and requires the auditor to plan expenses for flight tickets, accommodation, and travel to the place where the audit intends to be conducted is usually the auditee's facilities. Due to this, the auditor and auditee may discuss the possibility of cluster some safety-critical software development activities to be evaluated in only one audit event.

If it is a long-term development, it is possible to split it into how many audits are considered necessary, based on the negotiation with the auditee. However, if it is a short-term development, it is possible to perform fewer assessments by merging the development and verification phases. In both cases, it is crucial to identify which are the required questions to be used in the assessment because the clearer and more complete the questions are to address concerns of the applicable standards, the easier it becomes to assess whether the development was done as expected and it complies with the standard.

Define the audit questions

Once the audit’s scope has been defined, it is necessary to create the questions to be answered during its conduction based on the standards applicable for the safety-critical software development and the additional documents necessary to achieve compliance. Section 5.3 provides the recommendations for question creation and the main concerns that must be considered.

At this moment, the auditor can identify the necessity of creating additional questions based on the defined scope or even perform some tailorings in the questions based on the software project's characteristics.

Prepare the agenda

The audit agenda is an essential document because it: establishes the audit duration in business days, topics that the auditor needs to evaluate each day, and the teams involved in each topic (e.g., development team, verification team, quality assurance team, configuration management team). In some cases, it is possible to indicate the period used for each subject; this information helps to control the audit during its execution. The audit may be conducted in a pre-defined number of business days based on its defined scope, but this time is directly related to the number of topics selected to evaluate during the audit and how detailed the auditor wants to inspect each topic.

The agenda topics must be organized in a similar sequence of the safety-critical software process, thus allowing conduct the audit in a coherent sequence and verify the relationship between artifacts when necessary. For audits that do not involve certification authorities, the auditee must be consulted to indicate its availability or not for the intended period for the audit. Once that period is agreed upon, the agenda must be prepared and sent to the auditee's consent, minor adjustments can be requested, but the important is to be clear for both sides what the auditor is requesting to be assessed.

In the cases where a certification authority conducts the audit, it is necessary to check their availability before consult the auditee's availability. Once the certification authority indicates the period that they are available to perform the audit, it is necessary to negotiate with the auditee to reach a suitable date for everyone involved.

5.2.3 Evaluate documentation preliminary

After request and receive the necessary documentation for the audit, it is needed to perform a preliminary assessment to get familiar with the process to be audited and search for possible gaps and issues that shall be confirmed or clarified and during the audit execution. In addition, it is necessary to understand the auditee company’s rules regarding presenting restricted documents/data.

Evaluate the artifacts

If it is a planning or closure audit, the preliminary artifacts must be assessed to verify if all the required artifacts were provided based on the defined process for the safety-critical software or if it is missing an artifact necessary for the audit conduction. In case it is missing, the artifact must be requested unless it is considered confidential. The auditee must provide an alternative means to present the artifact (e.g., present the artifact through a conference meeting to check the specifically necessary information).

If the audit is related to all other phases (requirements, design, coding, verification), it is impossible to access the artifacts in advance. In this case, it is essential to understand the artifacts' status and if the auditee already has any consideration that can impact part of the audit scope and must be negotiated.

The rationale to not provide these artifacts preliminarily is that part of the information provided in them is considered the company's intellectual property and can not be shared outside the companies facilities. It can also be classified as confidential, thus not being possible to be accessed even in the companies’ facilities. Due to the company's intellectual property, it is necessary to verify and understand if the auditors must sign a Non-Disclosure Agreement (NDA).

Identify possible gaps to be checked

While evaluating the artifacts or their status preliminarily, it is possible to identify gaps in the process execution or issues against the applicable standards. In this case, it is recommended:

- Request a clarification for the auditee for a better understanding, and if the issue remains, took note in the preliminary final report to check the information during the audit execution itself, where it is possible to have a clear view of the generation of the artifacts.
- Only took note in the preliminary final report and, during the audit execution, seeks for evidence that can solve the issue or confirm it.

Prepare preliminary version of a final report

It is highly recommended to prepare a preliminary version of the final report with the known information, like the auditee company's name, company's location, date, period, agenda, desktop, remote or in-loco, the topics to be assessed. Prepared a preliminary version of the report is recommended because, during the audit meeting, the auditor may not have enough time to prepare and fulfill the report if all the necessary subjects must be handled during the audit.

If an issue is identified during the preliminary assessment, it can be recorded in this report to be confirmed during audit execution. Include the objectives of the standard that must be checked the compliance, if there is any. This version of the final report helps the auditor record the audit execution adequately and detail the necessary information without spending too much time during the execution, thus focusing on the audit itself.

5.2.4 Perform the audit

After preparation, obtaining the necessary information, and performing a preliminary assessment, it is necessary to conduct the audit and its related documentation and activities.

Conduct the audit

The audit must be conducted as planned based on the agreed agenda and in preparation for conduct it, including the related questions. Some aspects must be considered to its execution, like:

The number of people participating in the audit meeting. The auditee must ensure that all the software teams that took part in the phases under evaluation are available to answer any question. However, it is not recommended to let all of them participate in all audit moments because too many people in the audit room can cause distractions and even misunderstandings in the audit.

Define a focal point for each team. The auditee must define the focal points for each software team that intends to participate in the audit. This strategy helps to define only one person to answer the auditor's questions avoiding divergent answers to the same question. The other software team members can participate in the meeting, but only as listeners.

Conference room having adequate space and tools. The auditee must provide a conference room that adequately accommodates all persons involved in the audit, at least the one who is key to this event. Also, audiovisual resources are needed for the projection of the samples selected for the audit, allowing all participants to see what is being evaluated, and eventually call for some software team member that is not available in the room.

During the audit execution itself, the evidence must be recorded to auxiliary the compliance demonstration at the end of the development for each question answered. If a question is answered as "No", an issue must be opened related to it to perform the follow-up until its resolution. The customized checklist must drive the audit, but it is not limited to it. If there is any additional concern during the audit, the auditor must seek evidence to clarify this concern, and if there is not, an issue must be recorded. Either way, a note must be added to the final report to clarify the scope of the audit.

At the end of the audit, the auditor must summarize the issues identified and classify them according to their severity. For this software audit model proposed, the classification of the issue is:

- **Finding**: when an issue is raised against the applicable regulation or any other compliance document applicable to the safety-critical software development under evaluation.

- **Action**: when an issue is raised against the defined auditee's software process.
- **Observations**: when the issue is an improvement and recommendation based on the auditor’s experience and evaluated during the audit.
- **Questions**: when the auditor raises questions, it was impossible to clarify during the audit execution. Its answer may lead to an issue.

Prepare the final report

It is highly recommended to prepare the final report throughout the audit, making more precise and accurate records of everything that happened and was discussed and agreed upon in the audit period. The issues identified during the audit are the most relevant information in the report, and it is necessary to be recorded not only the identified issue but in which context it was identified and against what part of the standard or the safety-critical software process.

Present the final report

At the end of the audit, the last slot of the agenda is the final meeting presentation, where a summary of the report is presented with the main concerns, the issues identified, the auditee must treat that during the follow-up period, and any other relevant information considered by the auditor. It is crucial to perform a clear presentation and ensure that all the relevant people involved in the audit and present at the final presentation understand all the auditor's issues.

5.2.5 Follow-up

A deadline must be set for receiving the action plans for issues that have been identified and confirmed in the audit execution. It is suggested to provide a deadline of fifteen business days to allow the auditee to prepare the teams and reunite the necessary information to write an action plan, but this can vary based on the safety-critical software schedule and the company’s target. The action plan provided must

address the issue's primary concerns and provided a strategy to fix it. If it was identified as not an issue during the investigation, the related evidence must be provided with the action plan for the auditor's evaluation.

There is not a specific number of follow-up cycles to be performed, and this varies based on the number of issues, their related severity, and the effort to implement the solution. The import is to perform regular meetings to discuss the action plans and perform the necessary agreements until there are no more issues. For each cycle, the auditee must provide the necessary evidence for those issues indicated as solved to give the auditor confidence that the issue was solved as expected.

5.3 Recommendation on questions creation

The audit questions shall be created based on the applicable standard to be audited so that it is possible to verify full compliance with the chosen standard by answering all checklist questions. The first step to creating the questions is reading and understanding how the standard is written, for example, if it is “objective-oriented”, “technique-oriented”, or another method.

Another relevant aspect of being considered is if the standard is tailored for the project. In this case, only the applicable sections/objectives could be part of the questions creation process, driving the questions to consider the characteristics of the project.

5.3.1 Objective-oriented

As an “objective-oriented” standard, the RTCA DO-178C was chosen to be used as a sample. The objectives tables have references for sections that are responsible for providing details and explanations of what is expected and what must be done in each phase or activity of the process that is handled by the standard. It also indicates what the data item is expected as output and, when applicable, the

software level applicability of each objective is. Some standards can provide additional information regarding the relationship with the system aspects and software considerations.

For this type of standard, it is recommended to have at least one question for each objective, and sometimes it is necessary more than one question to verify the full compliance with the objective based on the standard description for each objective and its referenced sections. It is also possible to relate more than one objective for one question. Usually, this additional objective is related to configuration management activities executed along the software development process.

5.3.2 Technique-oriented

As a “technique-oriented” standard, the IEC 62304 was chosen to be used as a sample. The compliance must be sought directly with what is described in the standard sections, and for IEC 62304, the appropriate software level is indicated in each section. The sections are illustrated with what is expected to achieve compliance in the software and the necessary details to perform the software development.

For this standard, it is recommended to have at least one question for each section, but it is widespread to have more than one question to verify the full compliance with the section description. It is also possible to relate more than one section for one question. Usually, this additional section is related to configuration management activities executed along the software development process.

5.3.3 General recommendations

As a general recommendation for questions creation, the following aspects shall be taken into consideration:

- It must be written with clear statements, using the keywords of what is intended to seek compliance;
- It must be written in such a way that a specific characteristic is checked;
- Based on audit characteristics, it must be written considering that sample artifacts will be checked;
- Questions created for other standards can be reused when it handles a similar concern for both standards;
- More than one question can be used to verify compliance with the same objective, depending on its complexity.
- If the related certification authorities have additional concerns related to the selected standards, it can be considered during the creation of the questions.

The auditor's expertise and several years of experience can be used to improve the quality of the questions and how they can be used to conduct an assessment. Additionally, some additional questions may be created to complement the existent, based on the software development characteristics, mainly if any selected standards' tailoring was used.

Based on the first and second methods explained in previous sections of this chapter, questions were proposed for this software audit model to exemplify how it can be used. For the objective-oriented, it was selected RTCA DO-178C standard and proposed 123 questions evaluated and assessed by software specialists in this standard through a SAME. A similar SAME process occurred for the IEC 62304, which was the selected standard for the technique-oriented method and had a total of 84 questions proposed.

Chapter 6

SOFTWARE AUDIT MODEL EVALUATION

This chapter intends to present how the software audit model evaluation (SAME) was planned, conducted during this research and the evaluation of its results.

After the software audit model has been specified and defined with the necessary details to conduct an audit in a safety-critical software according to the auditor's necessity and the project characteristics, a set of questions were proposed based on each standard characteristic, additional known regulation authority's concerns, and the author's several years of experience.

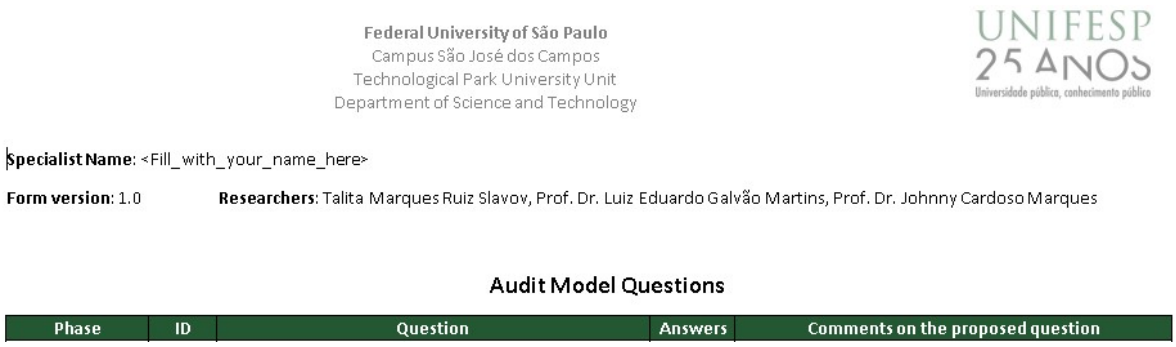
While the questions were under creation, two software audit model evaluations were elaborated to be conducted with software specialists in each of the selected standards. As a first step, the Research Ethics Committee proposal and the Statement of Free and Informed Consent template were sent to the UNIFESP Research Ethics Committee, with the necessary details to conduct the evaluation. After their approval, the SAME form was elaborated for each selected standard.

6.1 RTCA DO-178C evaluation

Based on the software audit model described in Section5.2, a set of questions was generated to attend the RTCA DO-178C from the aeronautical area. The

questions were created to verify the compliance with all the 71 development objectives defined in this standard, which means a safety-critical software level A where the most critical failure is considered catastrophic, including additional questions related to the tool qualification process.

The questions proposed to evaluate the RTCA DO-178C took one week to be created and additional two weeks to be considered mature enough to be evaluated by the selected software specialists through a SAME. The form presented in Figure 6-1 was created to allow the software specialists to give feedback and improvement suggestions. Besides the 85 initial proposed questions in the form, the software specialists also received the Statement of Free and Informed Consent to be fulfilled if they accepted to participate in the SAME.


Federal University of São Paulo
Campus São José dos Campos
Technological Park University Unit
Department of Science and Technology

UNIFESP 25 ANOS
Universidade pública, conhecimento público

Specialist Name: <Fill_with_your_name_here>

Form version: 1.0 **Researchers**: Talita Marques Ruiz Slavov, Prof. Dr. Luiz Eduardo Galvão Martins, Prof. Dr. Johnny Cardoso Marques

Audit Model Questions

Phase	ID	Question	Answers	Comments on the proposed question

Figure 6-1 – RTCA DO-178C SAME form

The first cycle of the SAME was conducted with eight software specialists who have extensive experience with software implementation and certification using RTCA DO-178C and its related supplements to evaluate the proposed set of questions and verify if it was correct and complete to be used in an audit event and consequently to verify the full compliance of the safety-critical software evaluated. However, only four of them answered all cycles of this SAME. Table 6-1 summarizes their profile considered to answer this evaluation.

Table 6-1 – RTCA DO-178C specialists' profile

Specialist	Age	Years of experience	Function
Specialist 1	45+	16 – 20 years	Product Development Engineer
Specialist 2	45+	16 – 20 years	Product Development

Specialist	Age	Years of experience	Function
			Engineer
Specialist 3	41 – 45	16 – 20 years	Design Assurance Engineer
Specialist 4	35 – 40	10 – 15 years	Core Compliance Expert for Development Assurance

The first evaluation cycle took about 21 days to be completed. Four software specialists returned the answers for this cycle with an average of 30 feedback, which represents 35.4% of the questions evaluated. After this first evaluation cycle, there were missing questions about specific concerns like previously developed software, change impact analysis, and certification aspects. These topics are not explicitly detailed in the RTCA DO-178C objectives but are essential topics to be discussed in the audit.

Based on the feedback, a total of 33 questions were added to the proposed questionnaire for RTCA DO-178C. The feedback was related to specific concerns like previously developed software, change impact analysis, and certification aspects related to the entire software development, not to each specific phase. Another aspect identified was that some questions could clarify specific matters and, in some cases, be split into more than one question to focus on each of the concerns covered in the question.

The fact of split questions also justifies the number of added questions, indicating that it was not exactly a lack of questions to address the concerns of the RTCA DO-178C standard, but a matter of organizing the content of the questions being asked more appropriately and directly and cover concerns that are not explicit in the RTCA DO-178C standard objectives.

A second cycle was necessary to be carried out to evaluate the 33 additional questions and the improvement performed in the others. It took about 15 days to be concluded. The same four specialists that evaluated the first cycle participated in the second one. As a result of this second cycle, it had an average of 16 feedbacks, which represents 13% of the questions evaluated, and it could be considered minor questions, once most of them were related to questions completeness and repeatability.

After evaluating the feedback received, 5 questions were removed from the final version of the questionnaire, since according to the software specialists, when

breaking it into more questions, some of them dealt with the same aspect, but with slightly different wording.

It was possible to notice that the several years of experience in auditing safety-critical systems brought different views and perspectives of the questions that were initially proposed based exclusively on the standard. Due to this auditing experience, many questions were evaluated not only for the standard but also with a view of what is practiced by the companies that use it for developing safety-critical software.

Table 6-2 – Sample of question analysis – Specialist 1 and 2

Question	Feedback Specialist 1	Feedback Specialist 2
Is the software architecture decomposing each module into software components and defining the data and control the necessary flow between them?	Does the software architecture decompose modules into software components, and does it define data and control flow between them?	#1. Although common, such architecture framework is not prescribed in DO-178C criteria, which defines and widely uses the concept of “components” only. Considering the research justification for a “flexible audit model” and the objective of such model being applicable to any embedded software, I recommend to remove the portion of this question asking for such prescription (modules composed by components) and replace “modules” by “components” in all other applicable questions and in the “Audit Model Requirements” listed after this checklist table. #2. The remaining of this question (about data and control flow) seems to be redundant with q. D.07 and both may be encompassed in just one question.

Another relevant consideration is that two of the software specialists also know the RTCA DO-178C, but in recent years works auditing the system aspects (SAE ARP-4754 A and RTCA DO-297), and this leads them to evaluate with different perspectives from the other two software specialists, a point of view more driven to system concerns, which is also essential when developing safety-critical software. Table 6-2 presents the comments performed for the same question and compares the similarity between the feedbacks in the first cycle of review. It is vital to notice that the comments were transcript as they were provided.

The other two software specialists also know the system area, but they are focused on audit safety-critical software developments in the latest years, which brought similar feedback in almost all questions evaluated by them. This similarity makes it easier to address the feedback and improve the questions to be mature enough to be used in the case study. Table 6-3 presents the comments performed for the same question and compares the similarity between the feedbacks in the first cycle of review. It is essential to notice that the comments were transcript as they were provided.

Table 6-3 – Sample of question analysis – Specialist 3 and 4

Question	Feedback Specialist 3	Feedback Specialist 4
Are the software derived requirements traceable to their rationales?	Besides ensuring the derived requirements are traced to their rationales (here I would suggest to change from “traceable” to “traced”), it is very important to ensure the derived have been provided to the system process for its acceptance (concerning safety impacts). That should be addressed in this question or in a dedicated question.	Traceable and provided for safety review.

After two cycles of the SAME, 112 questions (See Appendix B) were generated to attend the following safety-critical software development phases: planning, requirements, design, implementation, verification, configuration management, quality assurance, tool qualification, and closure. The questions address the RTCA DO-178C standard objectives and include concerns related to guidelines provided in the standard for each area of interest.

Although this set of questions could be considered similar to a job aid from FAA, it differentiates in the fact that there are proposed based on the RTCA DO-178C instead in the certification authority requirements; that is, it is based on applicants and specialist’s knowledge and a defined software audit model. Another differential from job aid is that this set of questions can be combined according to the scope of the audit to be performed and the auditee's software development process, thus allowing greater flexibility for the auditor and improving its skills and ability to conduct the audit.

6.2 IEC 62304 evaluation

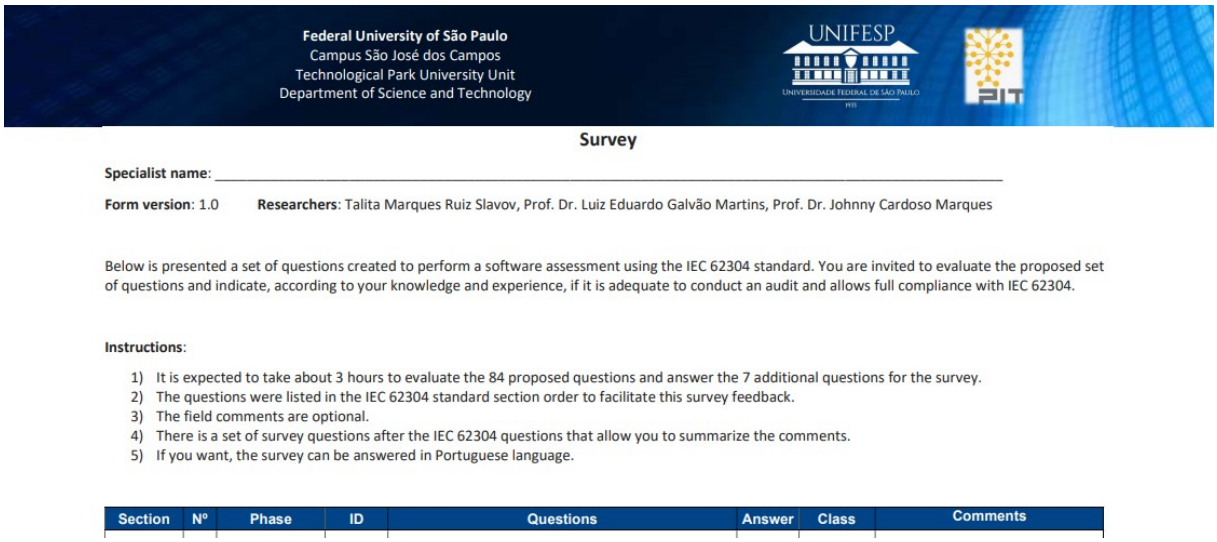
Once the questions were created and considered mature by the software specialists in the RTCA DO-178C, it was the moment to modify them and create questions for IEC 62304. Although these standards are intended for different areas, it was still possible to observe similarities between them during this process of adapting the questions. The questions were created to attend the five main sections of this standard (Section 5 through Section 9), considering the recommendations and information provided in all other document sections. It took one week to adapt the questions and another one week to be considered mature enough to be sent for the software specialists in IEC 62304.

From the 112 questions proposed for RTCA DO-178C, 33.9% of them were considered similar, and it was only necessary to adapt the term used in the standards or some aspect that was requested in one standard but not for the other. Although the other 74 questions were considered not applicable, it was necessary to create 46 new questions to address the concerns of the IEC 62304 standard. Table 6-4 presents a sample of similarity between the questions.

Table 6-4 – Sample of similar questions

Phase	DO-178C Question	IEC 62304 Question
Planning	Are software organizations (including quality assurance independence), organizational responsibilities, system lifecycle processes, and certification liaison process detailed?	Is the planning documentation, or equivalent, providing or referencing a document that provides title, naming or naming convention, purpose, the intended audience of the document, and procedures and responsibilities for development, review, approval, and modification of the development life cycle?
Requirement	Are the sampled software high-level requirements defining the software functional, performance, safety-related, and external interfaces?	Are the sampled software system requirements defining the software functions and capabilities, that is, performance, physical characteristics, computer environment, and capability for upgrades?
Design	Is the software architecture defined	Is the software architecture defined

Phase	DO-178C Question	IEC 62304 Question
	based on the software high-level requirements?	based on the software system requirements?
Implementation	Is the sampled software source code correctly implementing the software low-level requirements and software architecture?	Is the sampled software source code correctly implementing the software requirements and software architecture?
Verification	Are the sampled software tests reviewed to verify consistency and completeness?	Are the sampled software tests, related test procedure, and their acceptance criteria reviewed to verify consistency and correctness?
Configuration Management	Are the sampled baselines and traceability data generated as per Configuration Management Process?	Is the sampled change request traceability documented as expected, that is, following the sequence change request, problem report, and the approval of the change?



Federal University of São Paulo
Campus São José dos Campos
Technological Park University Unit
Department of Science and Technology

UNIFESP

Survey

Specialist name: _____

Form version: 1.0 **Researchers**: Talita Marques Ruiz Slavov, Prof. Dr. Luiz Eduardo Galvão Martins, Prof. Dr. Johnny Cardoso Marques

Below is presented a set of questions created to perform a software assessment using the IEC 62304 standard. You are invited to evaluate the proposed set of questions and indicate, according to your knowledge and experience, if it is adequate to conduct an audit and allows full compliance with IEC 62304.

Instructions:

1) It is expected to take about 3 hours to evaluate the 84 proposed questions and answer the 7 additional questions for the survey.
2) The questions were listed in the IEC 62304 standard section order to facilitate this survey feedback.
3) The field comments are optional.
4) There is a set of survey questions after the IEC 62304 questions that allow you to summarize the comments.
5) If you want, the survey can be answered in Portuguese language.

Section	Nº	Phase	ID	Questions	Answer	Class	Comments

Figure 6-2 – IEC 62304 SAME form

Some questions were tailored from RTCA DO-178C due to their similarity, and the others were created specifically for IEC 62304. Figure 6-2 presents the SAME form created when the questions were considered mature enough to be evaluated by software specialists. Initially, was proposed 84 questions that were provided in the form with 7 additional questions explicitly created for this SAME; as a way to obtain additional information about the interviewee. The software specialists also received

the Statement of Free and Informed Consent to be fulfilled if they accepted to participate in the SAME.

The SAME was conducted with twenty software specialists who have extensive experience with software implementation and assessment using the IEC 62304 to evaluate the proposed set of questions and verify if it was correct and complete to be used in an audit, in a similar process described for RTCA DO-178C. However, only three of them answered this evaluation. Table 6-5 summarizes their profile considered to answer this SAME.

Table 6-5 – IEC 62304 specialists' profile

Specialist	Age	Years of experience	Function
Specialist 1	35 – 40	10 – 14	Senior product development engineer and Sr. Software engineer
Specialist 2	45+	0 – 5	Design software supervisor and tester
Specialist 3	40 – 45	0 – 5	Quality Assurance Officer

The SAME took only one cycle of around 40 days to be completed. Three specialists returned the answers for this evaluation with 12 feedback, representing 14.3% of the questions evaluated. This percentage indicates that the questions were mature enough to be used in an audit. This low number of feedback occurred, mainly because the questions were proposed based on a tailoring process of the RTCA DO-178C, which brought all the necessary feedback to improve the audit process itself.

One feedback was related to one specific word, which was changed based on the IEC 62304 standard. The second was a general comment on how to answer the questions, but this is detailed in CHAPTER 5 as part of SAM's description. The third suggestion is the question's evidence; however, this level of detail is not the intention of this research once the expected evidence is intrinsically related to the audited development process. The other 9 feedbacks were related to split one question into additional question to make it concerns more precise and easy to answer during the audit execution (e.g., V.01, V.02, and V.03 – See Appendix C), and include additional information in the question, to ensure that the IEC 62304 standard concern would be completely covered during the software audit.

A total of 90 questions (See Appendix C) were generated to attend the following safety-critical software development phases for medical devices: software

development process, software maintenance process, software risk management process, software configuration management process, and software problem resolution process. The questions are intended to address each section's concerns, including the guidelines provided by each of them and other support sections of the standard.

Table 6-6 summarizes the similarities and differences between the RTCA DO-178C and IEC 62304 after the analysis to reuse the questions already created to RTCA DO-178C, where 38 questions were reused.

Table 6-6 – Similarities and differences between the standards

Similarities	Differences
- Requirement process. - Configuration management process. - Verification process.	- Risk management process. - Maintenance process.
- Although the problem resolution has a process exclusive in IEC 62304, they have similar concerns.	- Specific terms like SOUP (software of unknown provenance).
- Although the release has a process exclusive in IEC 62304, they have similar concerns.	- Software safety classification: IEC 62304 have 3 classes, RTCA DO-178 C have 4 development assurance level.

Chapter 7

CASE STUDY

This chapter intends to present how the case study was planned and conducted during this research.

Once the software audit model proposed was considered mature enough to be used and software specialists evaluated the questions, the next step of this research was to conduct a case study with aerospace safety-critical software. The software selected to conduct both instantiations of the case study to evaluate the suggested software audit model and its related questions (see Appendix B) is presented in detail in Section 7.2.

Figure 7-1 depicts the step-by-step process used to conduct this case study, starting with the case study protocol elaboration, which includes the necessary stages to conduct the case study itself, passing through the execution of two software audits, which encompasses the analysis of the results obtained. Finally, based on the results of the two audit's instantiations, they are discussed and compared to identify the positive aspects and points of improvements.

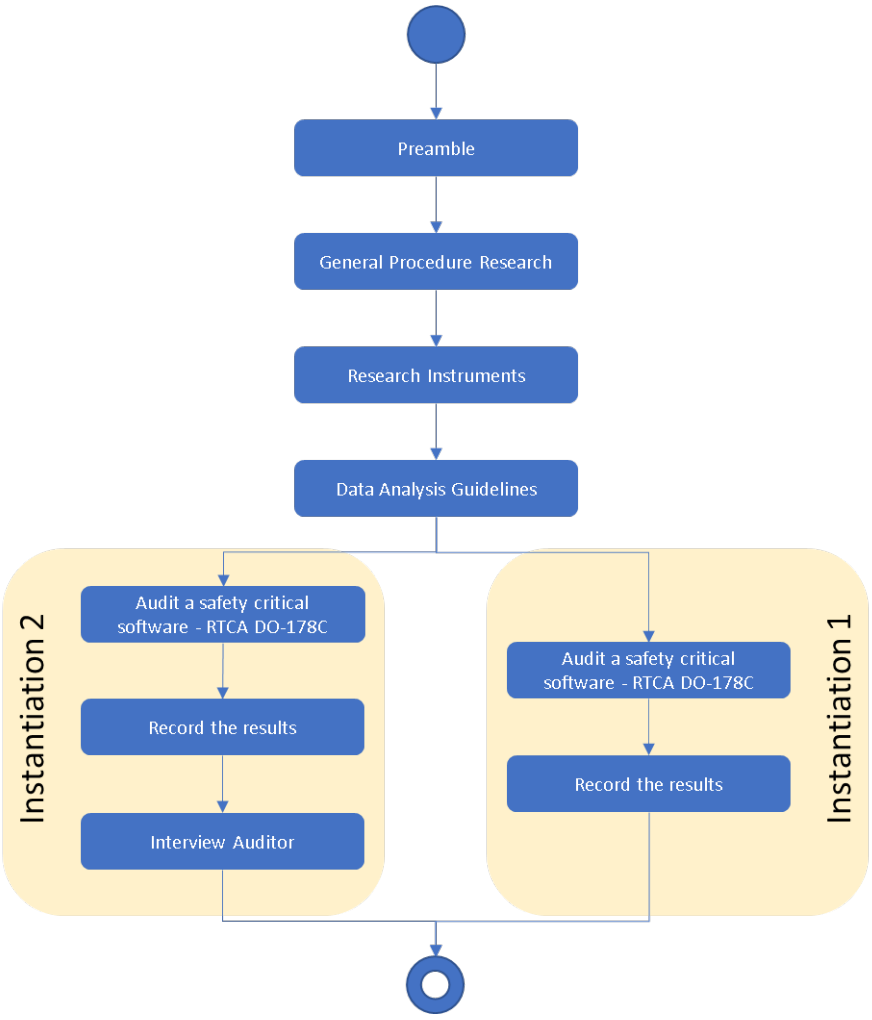


Figure 7-1 – Case Study workflow

7.1 Case study protocol

The case study protocol aimed to collect and analyze the data of the software audit model applicability. For this case study, it was chosen aerospace safety-critical software developed as DAL A - the highest level defined per RTCA DO-178C standard. The case study protocol was written establishing the tools that should be used: evaluate the documentation through the software audit records and an interview with the independent software auditor, allowing to obtain its feedbacks and impressions of the proposed SAM, to execute the two instantiations defined.

It was planned to collect the two software audit checklists' answers fulfilled by software auditors of an aerospace company to obtain the software audit records. Due to the company’s confidentiality, all the collected data was uncharacterized, and the

documents were stored using the uncharacterized version to handle the necessary data. Besides, as part of the proposed case study protocol, questions were elaborated to guide the interview activity so that, at the moment of performing this activity, the questions could help extract the most relevant data to be analyzed and used to improve the software audit model. Table 7-1 describes the questions used to conduct the interview.

Table 7-1 – Interview questions

ID	Question	Rationale
1	What is your experience in a software audit?	It allows understanding the experience and expertise of the auditor to execute this activity and give relevant feedback.
2	How did you define the audit scope?	Evaluate if the proposed process of “Preparation for the audit - Define the audit scope” is adherent and mature enough to be used.
3	How did you select the applicable questions?	Evaluate if the proposed process of “Preparation for the audit - Define the audit questions” is adherent and mature enough to be used.
4	How did you conduct the audit?	Evaluate if the proposed process of “Perform the audit - Conduct the audit” is adherent and mature enough to be used.
5	How did you answer the questions?	Evaluate if the proposed process of “Perform the audit - Conduct the audit” is adherent and mature enough to be used.
6	Was the intent of the questions clearly defined?	Evaluate if the proposed process of “Recommendation on questions creation” is mature enough to be used.
7	Were the questions easy to answers and/or provide evidence?	Evaluate if the proposed process of “Perform the audit - Conduct the audit” is adherent and mature enough to be used.
8	Were the questions helpful to your audit?	Evaluate if the proposed software audit model process as a whole.
9	What is your feedback for this software audit model proposed?	Identify the necessary improvements.

7.2 Instantiations of the case study protocol

After the case study protocol was considered ready to be used, it was requested permission from the aerospace company to use one of its safety-critical software developments to conduct the audit. It was authorized to perform the audit in one software classified as DAL A, the highest development level of RTCA DO-178C, and developed using the V-Model software development process. Due to software development characteristics and the companies process, it was necessary to adapt the SAM proposed to attend its process.

Some of the aspects adapted were (i) not use the questions created for the quality assurance activities, (ii) create additional question considering the software development process established by the company, (iii) create questions for other standards like SAE ARP-4754A and RTCA DO-331, (iv) update the terms of some questions to those used by the aerospace company, (v) adapt the final report meeting and follow-up process. However, this adaptation did not change the SAM's central purpose as described in this research, thus allowing the case study's execution

Two software audits were performed in the safety-critical software, one executed by the software audit model author and the other executed by an independent software auditor. Table 7-2 presents the auditor‘s profile, who executed the audit, including their actual function and the years of experience with auditing software.

Table 7-2 – Auditor's profile

Specialist	Age	Years of experience	Function
Software audit model author's	32	9.5	Design Assurance Process Specialist
Independent auditor	39	10	Design Assurance Process Specialist

Before executing the audit, it was necessary to define the scope by identifying the artifacts available and verifying which software development process phase it belongs to. It was used the procedure described in the "Preparation for the audit - Define the audit scope" phase (see Section 5.2.2) to select the sample of the audit,

allowing to evaluate the process described and verify the adherence of the process to an actual software audit, as well, identify any necessary improvement.

With the samples selected, it was necessary to choose the questions used to compose the checklist to be fulfilled during the audit. It is essential to highlight that in this stage lies one of the necessary tailorings for this case study's execution (see Table 7-3). Because it is in-house software development, the audit was performed by software quality assurance engineers, and consequently, the questions applicable to the Quality Assurance phase were not considered applicable for the audit scope. This step was used to evaluate the “Preparation for the audit - Define the audit questions” procedures (see Section 5.2.2).

Table 7-3 – Instantiation 1 and 2 – Questions tailoring

SAM Question ID	Tailoring	Explanation
V.07	The questions moved from the verification phase to the requirements phase.	SAM’s initial questions database was proposed to perform the audit in a supplier software, as an external auditor. However, the case study was executed with the software quality assurance team, which is considered part of the software life cycle data as per DO-178C. Besides, the aerospace company’s software audit process predicts evaluating the phase’s activities (development and review) together.
V.08	The questions moved from the verification phase to the design phase.	
V.09	The questions moved from the verification phase to the implementation phase.	

Once the sample was selected and the checklists ready to be used, the software auditors were ready to execute the audit, as defined in the “Perform the audit” (see Section 5.2.4). However, here is another point where the defined process needed to be tailored to the reality and processes already existing in the aerospace company. As it is in-house software development, there was no need to create an agenda to be agreed upon between the parties, and the audit could be performed in the desktop modality since the software auditors had access to all the artifacts that

needed to be evaluated. The software development team only needed to be available for any queries and clarifications that would be necessary.

Appendix D presents the customized checklist, with 46 questions used to carry out the safety-critical software audit and the auditor's answers. An additional table summarized all the 136 artifacts consulted during the safety-critical software audit and used to answer each of the selected questions.

Appendix E presents the customized checklist, with 43 questions used to carry out the safety-critical software audit and the auditor's answers. An additional table summarized all the 184 artifacts consulted during the safety-critical software audit and used to answer each of the selected questions.

As the next step of the audit execution, the follow-up of the issues was performed until all issues identified were closed, the process proposed on "Follow-up" phase (see Section 5.2.5) was followed as defined, and the software quality assurance engineers recorded the follow-up activities as requested by its processes, in the company's issue tool. The selected phases to be audited, related execution, and follow-up are described in Section 7.2.1 and Section 7.2.2, where the case study protocol instantiation is discussed in detail.

7.2.1 Instantiation 1

The first instantiation of the case study protocol was for a software audit executed by the SAM author, as this would allow identifying the points of improvement for the SAM being proposed and allow a better understanding and improve the perception of the process application as a whole.

For this instantiation, the selected portion of the safety-critical software was developed in compliance with RTCA DO-178C. Based on the aerospace company's processes and the available artifacts, the following phases were considered a basis to select the questions that should compose the software audit checklist: design, implementation, verification, and configuration management. Since RTCA DO-178C classifies Configuration Management as an integral process, any software audit that is executed for software development that has this standard as a means of compliance must include the questions related to this phase.

After the software audit scope was defined, 46 questions were evaluated and selected to be answered through the customized checklist. These questions were divided into 14 for design, 10 for implementation, 14 for verification, and 8 for configuration management. With the customized checklist ready to be used, the next step was to execute the software audit and answer each question.

Considering the necessity to perform tailoring in the audit execution due to the aerospace company's processes and the fact that it was used the desktop review modality, the software auditor evaluated each available artifact with her knowledge about the RTCA DO-178C and the software development process. At the same time, this initial evaluation took notes and the evidence recorded in the customized checklist.

After an initial analysis, the software auditor verified each question's request and searched evidence for each of them with the associated artifacts that had already been evaluated. If the evidence is found, the question was answered with "Y". Otherwise, either the software auditor did not consider some specific aspect and returned to assess the artifacts, or indeed an issue had been identified. In the latter case, the question was answered as "N", and an issue was recorded using the issues' tool and tracked in the related question.

Forthcoming the end of the audit execution, the software auditor ensured that selected questions were indeed fulfilled and had artifacts associated with them as evidence. One action against the aerospace company's established Configuration Management process and three observations were identified, all of them also associated with the Configuration Management process. Once all questions were completely satisfied, the software auditor prepared the final report, following the aerospace company's process, punctuating the issues found and recording them in the issues system.

Appendix D provides a complete extract of the tailored checklist, answers provided, and the list of artifacts consulted. Table 7-4 presents the Issue ID and the related SAM Question ID, indicating that all the issues were associated with the software configuration management process in this software audit.

Table 7-4 – Instantiation 1 – Issues raised

SAM Question ID	Issue ID	Explanation
CM.03	OBS-001	The audited safety-critical software is in a

SAM Question ID	Issue ID	Explanation
CM.04		mature state. Therefore all phases are well defined and executed, leading the issues to be concentrated in the configuration management process. CHAPTER 8 - discuss in detail each of the identified observations and the action.
CM.04	OBS-003	
CM.06	OBS-004	
CM.04	Action-001	

As a next step, the software auditor presented the final report for the software development team, thus explaining how each of the issues was identified, the compliance gaps with the established process or applicable standard, and formalizing the software audit results. The software development team could make questions to understand the issue and the observations and present additional evidence for the software auditor when necessary. At the end of this presentation meeting, a resolution date was agreed for the raised issues, and the follow-up phase was started.

After executing this first instantiation of the case study protocol and as the author of the SAM, the software auditor's perception is that the proposed process can be applied to real safety-critical software development projects and companies that develop safety-critical software. It was possible to verify that given the nature of the software project in which the case study was applied, tailorings were required in the questions and even minor updates on conducting the software audit. However, the SAM proposed already foreseen these tailorings, which gives the main characteristic of being flexible. CHAPTER 8 discusses the results and feedback in detail.

7.2.2 Instantiation 2

The second instantiation of the case study protocol was for a software audit executed by an independent software auditor. Therefore, it would be possible to obtain feedback from a person who had not participated in creating the SAM but would have the necessary knowledge to indicate the weaknesses and strengths of what is being proposed in this research. Besides, with SAM’s second execution in another software development, it is possible to assess its flexibility from the point of

view of the capacity to be applied to various safety-critical software development projects.

For this instantiation, the selected portion of the safety-critical software was developed in compliance with RTCA DO-178C, RTCA DO-331, and SAE ARP-4754A standards. It is essential to highlight that although this thesis is focused on a safety-critical software audit, the software selected to be audit should show compliance with other standards than RTCA DO-178C. However, the results presented in this thesis only considered the RTCA DO-178C portion.

Based on the fact that this research only suggested questions for the RTCA DO-178C, the procedures described in Section 5.3 were used to create the questions related to RTCA DO-331 and ARP-4754A, allowing the software auditor to evaluate the artifacts available for this case study instantiation fully. However, these questions are not be discussed as part of this research, once it is not the case study's focus.

Based on the aerospace company's processes and the available artifacts, the following phases were considered a basis to select the questions that should compose the software audit checklist: requirements, design, verification, and configuration management.

After the software audit scope was defined, a total of 43 questions for RTCA DO-178C, 1 question for RTCA DO-331, and 21 questions for SAE ARP-4754A were evaluated and selected to be answered through the customized checklist. These questions regarding RTCA DO-178C were divided into 11 for requirements, 13 for design, 13 for verification, and 6 for configuration management. With the customized checklist ready to be used, the next step was to execute the software audit and answer each question.

Considering the necessity to perform tailoring in the audit execution due to the aerospace company’s processes and the fact that it was used the desktop review modality, the independent software auditor evaluated each of the available artifacts with her knowledge about the RTCA DO-178C, RTCA DO-331, SAE ARP-4754A, and the software development process. At the same time, this initial evaluation took notes and the evidence recorded in the customized checklist.

After an initial analysis, the independent software auditor verified each question’s request and tried to answer each of them with the associated artifacts that had already been evaluated. If the evidence is found, the question was answered with "Y". Otherwise, either the independent software auditor did not consider some

specific aspect and returned to assess the artifacts, or indeed an issue had been identified. In the latter case, the question was answered as "N", and an issue was recorded using the issues' tool and tracked in the related question.

Forthcoming the end of the audit execution, the independent software auditor ensured that selected questions were indeed fulfilled and had artifacts associated with them as evidence. For this software audit execution, four observations were identified, which were distributed as 2 for the configuration management process, 1 for the verification process, and 1 for the requirement process. Once all questions were completely satisfied, the independent software auditor prepared the final report, following the aerospace company's process, punctuating the issues found and recording them in the issues system.

Appendix E provides a complete extract of the tailored checklist, answers provided, and the list of artifacts consulted. Table 7-5 presents the Issue ID and the related SAM Question ID.

Table 7-5 – Instantiation 2 – Issues raised

SAM Question ID	Issue ID	Explanation
CM.04	OBS-002	Although the audited safety-critical software is in a mature state, the development and verification activities are focused on more than one standard, which leads the software team to comply with them and has a complex process. CHAPTER 8 - discusses in detail each of the identified observations.
V.16	OBS-003	
R.01	OBS-004	
CM.08	OBS-005	

As a next step, the independent software auditor presented the final report for the software development team, thus explaining how each of the issues was identified, the compliance gaps with the established process or applicable standard, and formalizing the software audit results. The software development team was allowed to make questions to understand each of the issues and present additional evidence for the independent software auditor when available. At the end of this presentation meeting, the target date was agreed for each of the raised issues, and the follow-up phase was started.

7.2.3 Independent auditor interview

As the objective of the case study was to evaluate the applicability of the proposed SAM and verify its ability to be tailored in various types of software development, the second instantiation was necessary so that an independent auditor could use it and provide her impressions regarding the use, even indicating strengths and weaknesses, and assessing whether SAM has met its purpose.

The interview was conducted with the independent software auditor, which executed the second instantiation. This interview took about one hour to be completed to capture the independent author's perceptions about SAM and the suggested question database (see Appendix B) and any feedback. The complete interview questions' answers are described in Appendix F. CHAPTER 8 - discuss and analyze the feedback.

Chapter 8

RESULTS AND DISCUSSION

This chapter intends to present the proposed software audit model's results and discuss them based on the proposed SAM.

The evaluation process for the SAM’s suggested questions was presented and discussed in CHAPTER 6, as its result was essential for the two case studies’ instantiation planned for the SAM’s evaluation. Therefore, this section's focus is to discuss the result of the case study, its findings, and feedback received, in addition to presenting the improvements made in the proposed questions database.

The results and discussions described below focus on the case study’s instantiations and aimed to elucidate the research findings, understand the SAM’s maturity and its points of improvement, and answer some of the research questions established at the beginning of this research.

8.1 Identified issues in the instantiations

The case study consisted of two instantiations, the first of which resulted in 46 selected questions and obtaining a total of 3 observations and 1 action, and Table 8-1 describes these results. Due to the aerospace company's confidentiality, only a generic description of the issues encountered is presented in light of the question that helped identify them.

Table 8-1 – Instantiation 1 – Issues detail

SAM Question ID	Issue ID	Description
CM.03 CM.04	OBS-001	A review record artifact used to record the review of a modification performed indicated a link and its associated revision of an artifact under review. When evaluated by the software auditor, it did not present any information associated with the modification under evaluation.
CM.04	OBS-003	For the generation of baseline description, it is necessary to list some data. However, to avoid data consistency problems, it was decided to reference, through links, the software release document that already provides these data. However, the release document used as a reference was updated after releasing its associated baseline, leading the software auditor to question the data's validity. The issue was not considered an action, as the established process does not have any specific rules for generating baseline descriptions and can recover the revision of the release document through dates.
CM.06	OBS-004	When investigating how a given release document was generated, it was identified that the procedure for its generation did not describe how that artifact should be formalized and also what would be the process to invalidate its use.
CM.04	Action-001	A baseline was released for use and as part of a formal certification process. However, the review release document used to perform this release had a negative consideration, which indicated that the baseline could not be released for use. Also, after a conversation with the software development team during the audit, the review release document was modified to remove the negative consideration.

For OBS-001 and OBS-003, it was possible to verify that it was associated with the established software process's execution, allowing identifying improvements in the software process and even training software team developers. The guidance provided allowed the software auditor to consider various aspects of the configuration management process, thus assisting in conducting his audit and what types of evidence should be sought.

OBS-003 allowed identifying that the strategy of referencing a second artifact to identify specific information is good. However, it needs to be implemented, taking into account several aspects, as in these situations, it is necessary to indicate the

revision of the artifact that is being used as a reference because if it is modified in a future version, the process can identify with precision what information has been considered.

For a software product that is planned to be formally released for approval and consequent use on an aircraft, information such as the list of open problems reports and implemented problems report is of utmost importance to know precisely which functions are enabled in the software product and thus know what is being approved. If the reference is updated to include or exclude any of this information, the software product may not be described correctly.

OBS-004 allowed the software auditor to identify an improvement to the software process established for the aerospace company, despite not generating an issue against the used standards, found a gap in the established software process. Depending on how the established software process is executed, it may generate an issue in the future when using an artifact that was not mature enough to be used or even using an artifact that was already obsolete or invalidated due to some wrong information. In this last example, depending on the type of incorrect information used, it is possible to bring more severe consequences for the software product, even making it unfeasible to be used in aircraft.

If a company has a process well-defined, there is standardization and harmonization between existing software projects, allowing the creation of the historical database, thus impoving the resource flexibility among various software projects.

Regarding Action-001, the issue was also associated with the configuration management process. However, the core was how the software process was executed, identifying software process definition issues and execution, and even raising the necessity of the software team's training. Also, updating the result of an artifact without following the established modification process may result in a software product that does not meet all the selected standard objectives.

Based on that, it was possible to verify that SAM helped the software auditor search for the evidence for the chosen standard and consider other aspects necessary to attend the aerospace company's software process. The SAM questions associated with the knowledge in conducting software audit allowed identifying issues and improvements for the software process and even for the software team that executes it (e.g., Training needs).

The second instantiation resulted in 43 selected questions and obtaining a total of 4 observations, and Table 8-2 describes these results. Due to the aerospace company's confidentiality, only a generic description of the issues encountered is presented in light of the question that helped identify them.

Table 8-2 – Instantiation 2 – Issues detail

SAM Question ID	Issue ID	Description
CM.04	OBS-002	A software campaign results document used to record the test results referenced a requirement baseline in a given revision. However, this revision of the requirement baseline does not exist.
V.16	OBS-003	An issue was found and registered adequately in the problem report during Test A's execution, and the same issue happened for Test B, but it was not appropriately registered. After an interview with the software development team, an errata was made for the problem report, performing analysis and indicating that the same issue of Test A occurred for Test B.
R.01	OBS-004	During some requirements review, it was identified that its result classified the requirements for a specific category other than the one initially defined. As this was an improvement process and did not predict this category of information for the requirement, no action was open.
CM.08	OBS-005	The document responsible for establishing a version of tools applicable to the software project has an inconsistency between two sections, in which one determined the use of this tool, but the other did not determine which version of this tool should be used.

OBS-002 is similar to what was found for instantiation 1 (see Table 8-1- OBS-002), which corroborates that when using SAM, it is possible to identify issues and the software process and harmonizes the assessment of software auditors, even in software projects that have different characteristics.

OBS-003 is associated with the verification process, more precisely because the issues identified during the test campaigns were registered as expected. When searching for evidence to answer the question, the software auditor was not only able to assess whether there was an associated problem report, but he went further and verified that the content of both the tests and the associated problem was synchronized and with the correct information.

This flexibility that SAM gave to the software auditor allowed him to use his knowledge to seek additional information and find the issue. Also, if this issue were not identified, the test would not be corrected, and an additional software test campaign would be necessary, increasing the cost and time to deliver a software product or the software product would not be tested appropriately, allowing any issue to be identified in the aircraft after its approval.

OBS-004 was identified in the requirements phase and can be considered minor because it was a problem in transcribing the category of some requirements from one tool to another. However, depending on which category classification, in one case more extreme, it could lead the software developer not to implement the requirement and consequently have a function partially implemented, which may cause problems for the aircraft during its use. This demonstrates the relevance of the questions considering the software process established for the development and making the software auditor's assessment more flexible during the audit execution.

Finally, the OBS-005 is associated with the configuration management process and is related to the version of tools selected for a software project. In this case, the impact was minor, as it is a corporate tool. However, for cases in which that the tools are specific to the software product, due to the nature of the development, the lack of this information, and even the use of a "wrong" version, it may lead to a not compilable source code or in some cases it can generate a different result than expected, causing problems for the software product.

This instantiation allowed to verify the flexibility of the SAM, as well as the maturity of the proposed questions, and still verify that different software auditors can find similar issues for the same question, evidencing the gain in using the base of proposed questions as well as the recommendations described in CHAPTER 4 for the execution of software audits of various software projects.

The flexibility that the SAM gave can be considered an improvement that the proposed SAM brought compared to the models and techniques identified during the research carried out (RQD4.1 – see Section 3.1). Besides, it was noted that this improvement was also a benefit for companies using SAM since the questions became drivers for the software auditor who could still use his knowledge and evidence provided in the software audit to obtain the answers he needed to fulfill the software audit (RQD4.2 – see Section 3.1).

8.2 Feedback and improvements

Based on the author of the SAM, who executed the first instantiation and in the independent auditor interview (see Appendix F), the conclusion is that the proposed SAM can be applied to real software development projects and to companies that develop safety-critical software. It was possible to verify that given the nature of the software project, where the case studies were applied, tailorings were required for some questions and even a little in conducting the software audit. However, this type of tailorings is already foreseen in the proposed SAM, which gives it the characteristic of being flexible.

The proposed questions proved to be efficient in assisting in the software audit conduction, mainly because they point aspects that helped the software auditor during the evaluation of each of the artifacts, sometimes even behaving as a reminder of what should be looked at, or even what type of evidence should be sought. These aspects were also presented and discussed in Section 7.2.1 and Section 8.1

The proposed process itself proved to be efficient in recording the discussions that occurred during the audit, indicating the aspects that were evaluated, objectives that were fulfilled. And even helped the software auditor to have a complete view at the end of the software development because it may allow the aerospace company to automate the tracking of which objectives of the standard have been met.

Another aspect of being considered is that since it was an in-house software development process, some questions from the suggested questions database required an improvement to consider the software audit's intent, which was to comply with the activities requested by Table A-9 from RTCA DO-178C. However, some of this feedback was still implemented on the questions database suggested in Appendix B. Table 8-3 presents the updated questions and related explanations.

Table 8-3 – SAM's question database improvements

SAM Question ID	Reason
R.06, R.07, and R.08	- R.06 was initially proposed to check the accuracy, consistency, completeness, and compatibility with the target. However, during the SAM instantiation for the aerospace company, it was noticed that the development assurance level is different for the target compatibility

SAM Question ID	Reason
	and completeness aspects. The question was split into three, and questions R.07 and R.08 were created. - R.06 has been rewritten so that its guidance became more evident to the software auditor of what aspects to consider. It was included the “and related algorithms”.
R.12, D.23, I.13, V.20	It has been created to cover the independence aspect.
D.05, D.06, and D.08	It has a similar explanation of the requirements phase's questions regarding the accuracy, consistency, completeness, and compatibility with the target aspects.
D.11	It has been rewritten to include the architecture and make guidance clearer to the software auditor on what aspects to consider.
D.12	It has been rewritten to include the word flow and make guidance clearer to the software auditor.
D.21, V.02, CM.02, and CM.07	It has been rewritten to make guidance clearer to the software auditor
I.03 and I.04	It has a similar explanation of the requirements phase questions regarding accuracy, consistency, and completeness.
I.10 and I.14	- It has been updated to refer to the Software Development process instead of the Configuration Management process. The Objective/DAL column has also been updated. - I.14 has been created to cover concerns related to source code and executable object code regarding the configuration management process
V.03	It has been rewritten so that its guidance is more evident to the software auditor on what aspects to consider. It was included the “including normal and robustness aspects”.
CM.08	It was created to cover concerns regarding the development and verification environment aspects.

Based on the feedback from Table 8-3, the final version of the proposed question database for SAM has 123 questions, representing 11 questions added, which have already been considered during the software audit of the two case studies instantiated.

In addition to questions improvements, other types of feedback that were received are associated with the questions and the process proposed in the SAM, are they:

- Questions related to software configuration management could be broken down into more questions to focus on specific aspects. However, these aspects are more dependent on how companies decide to establish their software configuration management process, and that is why SAM has not incorporated this feedback. However, it already has a specific section with recommendations on how to create additional questions.
- The questions related to the problem report were only proposed considering what is requested in RTCA DO-178C. However, the software auditors indicated that additional questions could also be created to cover the certification authorities' concerns and contractual aspects through their guidance. However, these concerns are determined based on the specific software development, and so it is not recommended to add these questions to the question database initially proposed for SAM.
- The questions could present the answer option N/A (Not Applicable), like this, depending on the instantiation of SAM, would allow justifying why a question was considered not applicable and even help the software auditor to justify which objectives were met and which were not necessary or applicable. This feedback can be implemented when SAM is instantiated for a company that develops safety-critical software. In an instantiation to perform an audit of software suppliers, it may not be necessary, as the audits are based on a specific scope and may not occur more than once for the same phase.

Finally, the independent software auditor was asked about the need to have the option “Partial” as an answer. According to his perception when performing his software audit, this option did not seem necessary since if the answer is partial, some criterion based on the standard was not followed, and an issue would be raised. There was also no negative feedback or difficulties described in instantiating the proposed SAM, demonstrating its maturity and adherence to this research's proposed objectives.

Based on the results presented above and the discussions made for each topic, it was possible to evaluate that the proposed SAM adds, in a positive way, the processes established among companies that develop or evaluate safety-critical

software. Because when instantiating a mature software audit process, the companies can anticipate issues of projects under development or those that need to be evaluated, which allows for a smoother certification process and with few or no problems identified by certification authorities. This conclusion can be used as part of the answer for the RQD4 (see Section 3.1).

Chapter 9

CONCLUSION

This chapter intends to present a wrap-up of the research, contributions, social insertion, and future research perspectives.

This research's main goal was to propose a safety-critical software audit model (SAM) that could be flexible to be tailored to any safety-critical software area allowing the software auditor to define the audit scope according to its necessity. The defined steps were met and recorded throughout this dissertation.

CHAPTER 2 described the entire literature review for this research, and it was through it, mainly through the Systematic Literature Review conduct and described in CHAPTER 4, that the RQD1 (see Section 4.3.1) and RQD2 (see Section 4.3.2) could be answered. The Systematic Literature Review identified several methods to conduct an audit, but most of them were related to safety assessment and not the safety-critical software itself. For those related to the RQD1 main objective, it was not possible to find a complete explanation on how to instantiate and even tailor the method to be used. Section 4.3 discusses in detail these findings and answers for RQD1, RQD2, and RQD3.

The two methods identified to conduct an audit on critical embedded software to certify the aircraft (RQD2) were one of them proposed by a regulation authority (Job Aid) that has been discontinued and is no longer available to be consulted and used by the companies. The other is the Two-Level Checklists, which explains how to use it but did not provide the questions database or the detailed process to allow it instantiation. This lack of additional information did not allow a complete comparison between the proposed SAM and the audit methods identified in the literature.

Among the audit models identified and that met the objective of this research and the RQD2, no audit model was found that would allow flexibility in the manner described in its established process. Still, in search of responding to the RQD3 question, Two-Level Checklists was the one that would have the potential to allow this flexibility, but as its process has not been fully described to indicate how it can be instantiated, it is not possible to assert that it has this capacity.

CHAPTER 3 describes the research methodology adopted for this research, assisting in searching for answers to RQD4, RQD4.1, and RQD4.2 questions. For this, CHAPTER 5 was written to describe the complete process for using the SAM, in addition to proposing an initial question database for RTCA DO-178C and IEC 62304, as this way, it was possible to evaluate the question database with software specialists for each of the chosen areas, as described in CHAPTER 6.

CHAPTER 7 describes the evaluation method chosen to evaluate the proposed SAM, and thus, through the execution of a case study with two instantiations, conduct two software audits and obtain its feedback. Some of the results identified are:

- It was verified that SAM allows companies to have a software audit process flexible enough to be applied to any software project.
- The questions that assist the auditor in his assessment give him the freedom to check several aspects.
- The proposed questions allow different auditors to conduct the audit and identify similar issues, demonstrated in the case study instantiations.
- It also provides a clearer view of which objectives applicable to the certification process are being met, classifying the issues found based on their severity, and controlling the issues until their closure.

Considering that some safety-critical software developments are required to undergo an approval process by certification authorities to be formally used, the answer for RDQ4 is that establishing a software audit process allows verifying compliance with certification requirements in advance, enabling the company that develops safety-critical software to anticipate and solve problems and ensure compliance with these requirements.

The SAM establishes a process that allows the companies to perform an audit and seek compliance with the defined standards and regulations. When

implementing SAM within the company, it is implementing a process that will help the company achieve the maturity required for a software product to be approved by certification authorities.

As an answer to the RQD4.1, the main difference, and improvement observed in SAM was its flexibility in tailoring a checklist for conducting a software audit and creating new questions to attend to the companies' necessity. It allows not only to conduct a software audit based on a defined standard but also to verify compliance with more than one applicable standard and the process established to develop the software product.

Another essential aspect to point out is regarding the carry-out audits on suppliers; this flexibility for elaborating the audit checklist allows the company to make intermediate evaluations on the software product under development and reduce the number of audits to be carried out. Because in some cases, according to the milestone and the project phase, it would be necessary to conduct the audit in two or more stages to ensure that a minimum percentage of artifacts from the same milestone was evaluated.

As discussed previously, the answer to RQD4.2 is the benefits listed below, which were identified by executing the two instantiations of the case study:

- Establish the minimum necessary to be evaluated by different auditors without taking away the freedom to look at other aspects that may be considered relevant.
- It allows an easy identification of which objectives were or were not met.
- It facilitates the elaboration of the final audit report.
- It allows easy tracking of the issues identified and their associated artifacts.
- By anticipating the gaps, it improves the software product's maturity, which leads to a fast and smooth evaluation process by certification authorities.
- It helps ensure the software product's safety once evaluated regarding accomplishing the established requirements and standards.
- It can be used not only by companies that develop safety-critical software but also by companies that provide consulting services specialized in safety-critical software auditing.

Finally, in 1995 Mankins [102] created the first definitions of the Technology Readiness Level (TRL), which nowadays go from TRL 1 where the basic principles are observed and reported up to TRL 9, where the innovation is already proved through a successful system and/or mission operations. After executing the two instantiations of the case study and the fact that the aerospace company instantiates the SAM with the necessary tailorings to its necessity and reality, it is possible to consider that it can be classified as TRL 7.

9.1 Contributions

This research's main contribution is to provide a safety-critical software audit process that is flexible enough to be tailored by any company that develops this safety-critical software. Besides, its entire utilization process is described through this research, helping companies understand the stages of conducting a software audit, starting in its preparation for its realization, through its conduction, and finally, monitoring until all issues that have been identified are addressed and closed.

In addition to the software audit process based on RTCA DO-178C and IEC 62304, there is also a recommendation section for creating questions, allowing the company to instantiate SAM by using the suggested questions and creating their questions to their needs and project characteristics.

Another contribution made is that through adaptations and the creation of new questions, SAM can be adapted for other types of product developments that require an audit process to be evaluated. Therefore, it can be adapted for systems, hardware, non-critical software. It is only necessary to create the appropriate questions for the instantiation and adapt some steps of the defined process according to the needs and level of rigor that the product development demands.

The most valuable contribution of SAM is the ability to allow the software audit to be performed in a group of activities chosen by the auditor, instead of specific phases pre-determined by other software audit models identified in the literature, which can also help the company to save money and evaluate the development in early stages, identifying the issues in advance and solve them.

9.2 Social Insertion

Some companies that develop or use safety-critical software produce aircraft, medical devices, railway applications, and automotive systems. They are considered medium and large size. These companies have more than one safety-critical system to deal with, which means that to develop their products, it is necessary to comply with specific standards and regulations and even have its products approved or certified by various certification authorities, establishing their certification requirements.

Another type of company that can benefit from the proposed SAM is those that do not develop software but are specialized in providing consulting services for safety-critical software auditing that need to be developed according to a standard and regulations. These companies can be small and medium-sized and usually provide services to the companies mentioned previously, but not limited.

Also, certification authorities may be another target audience for the application of SAM because, depending on their level of rigor established, to carry out the approval of safety-critical software, system, or hardware, it is necessary to conduct one or more audits throughout product development to obtain all the necessary evidence that the product developed complies with standards and regulations.

Small companies can also benefit from SAM because, with an already established process, it is enough to instantiate what was proposed according to the company's characteristics and thus guarantee a process capable of identifying issues in advance, allowing for savings avoiding possible rework.

The social contributions of this research lie in the possibility that these companies that use safety-critical software can apply the proposed SAM to assess their internal developments or even their suppliers, thus allowing an increase in the quality of the final software product that is delivered and the anticipation of issues as the software is under development.

Its innovation lies in the fact that SAM proved to be flexible enough to be used by different companies, in various software applications, in addition to providing a complete process regarding its use, that is, how to execute each of the associated phases of an audit.

9.3 Future Perspectives

Some of the points of a perspective of continuity of research that has been identified are:

- Provide recommendations on how to adapt the proposed audit process for use in projects developed based on Assurance Cases;
- Provide recommendations on addressing certification authorities' concerns based on what has been defined as applicable for the project;
- Provide recommendations on adapting the proposed software audit model to other types of development other than software, like tool qualification, system, hardware, and even on safety assessment;
- Adapt the SAM to standards like RTCA DO-330, RTCA DO-331, IEC 62279, IEC 61508, IEC 61511, ARP-4754 A, RTCA DO-254, RTCA DO-297, and others used to guide the development of a safety-critical product;
- Extend the bibliographic review considering standards similar to SAE ARP-4754 and SAE ARP-4751 to other critical contexts (medical devices and railway applications);
- Adapt the SAM to conduct the audit in railway application, automotive systems, nuclear systems, space systems;
- Improve the SAM to consider safety aspects, including cybersecurity, which is one of the biggest challenges in safety-critical systems.
- Develop software that automates the proposed SAM and can help conduct audits, extract reports data, and help on data analysis.

REFERENCES

[1] SC-205, *RTCA DO-178C -- Software Considerations in Airborne Systems and Equipment Certification*, 3rd ed. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[2] SC-205, *RTCA DO-330, Software Tool Qualification Considerations*. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[3] SC-205, *RTCA DO-331, Model-Based Development and Verification Supplement to DO-178C and DO-278A*. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[4] SC-205, *RTCA DO-332 - Object Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A*. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[5] SC-205, *RTCA DO-333 - Formal Methods Supplement to DO-178C and DO-278A*. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[6] J. C. Marques and L. A. V. Dias, “Mitigação de Riscos de Certificação Civil em um Processo de Certificação de Aeronave Militar: Uma Abordagem para Software Embarcado,” in *XII-SIGE-Simpósio*, 2010.

[7] FAA, “Advisory Circular 20-115D,” 2017. [Online]. Available: https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/1032046.

[8] ISO/TC 210, *ISO/IEC 62304 - Medical device software — Software life-cycle processes*, vol. 3. British Standard, 2006.

[9] J. Marques and S. Yelisetty, “An Analysis of Software Requirements Specification Characteristics In Regulated Environments,” *Int. J. Softw. Eng. Appl.*, vol. 10, no. 6, pp. 1–15, 2019, doi: 10.5121/ijsea.2019.10601.

[10] L. Peter, L. Hajek, P. Maresova, M. Augustynek, and M. Penhaker, “Medical Devices: Regulation, Risk Classification, and Open Innovation,” *J. Open Innov. Technol. Mark. Complex.*, vol. 6, no. 2, p. 42, Jun. 2020, doi: 10.3390/joitmc6020042.

[11] I. Levels, I. Versus, and S. Integrity, “IEC 62279 - Railway applications -- Communications, signalling and processing systems -- Software for railway control and protection systems,” *Int. Electrotech. Comm.*, 2002, [Online]. Available: https://webstore.iec.ch/publication/6748#additionalinfo.

[12] M. Heinrich *et al.*, “Security Requirements Engineering in Safety-Critical Railway Signalling Networks,” *Secur. Commun. Networks*, vol. 2019, pp. 1–14, Jul. 2019, doi: 10.1155/2019/8348925.

[13] SC-167, *RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification*. 1992.

[14] FAA, “ORDER 8110.49 Chg 1 - Software Approval Guidelines,” 2011. [Online]. Available: https://www.faa.gov/documentLibrary/media/Order/Order_8110.49_Chg_1.pdf.

[15] Food And Drugs Administration, “Classify Your Medical Device,” 2020. https://www.fda.gov/medical-devices/overview-device-regulation/classify-your-medical-device (accessed Feb. 22, 2021).

[16] Food And Drugs Administration, “Device Classification Panels.” https://www.fda.gov/medical-devices/classify-your-medical-device/device-

classification-panels (accessed Feb. 22, 2021).
[17] Food And Drugs Administration, “Premarket Approval (PMA).” https://www.fda.gov/medical-devices/premarket-submissions/premarket-approval-pma (accessed Feb. 22, 2021).
[18] P. Johnston and R. Harris, “The Boeing 737 MAX Saga: Lessons for Software Organizations,” *Softw. Qual. Prof.*, vol. 21, no. 3, pp. 5–12, 2019, [Online]. Available: www.asq.org.
[19] Joint Authorities Technical Review, “Boeing 737 MAX Flight Control System,” 2019. [Online]. Available: https://www.faa.gov/news/media/attachments/Final_JATR_Submittal_to_FAA_Oct_2019.pdf.
[20] L. E. G. Martins and T. Gorschek, “Requirements engineering for safety-critical systems: A systematic literature review,” *Inf. Softw. Technol.*, vol. 75, pp. 71–89, 2016, doi: 10.1016/j.infsof.2016.04.002.
[21] K. Fowler, Ed., *Mission-critical and safety-critical systems handbook : design and development for embedded applications*. Elsevier Inc, 2010.
[22] B. Gallina, I. Sljivo, and O. Jaradat, “Towards a Safety-Oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification,” in *2012 35th Annual IEEE Software Engineering Workshop*, Oct. 2012, pp. 148–157, doi: 10.1109/SEW.2012.22.
[23] I. 22/SC 32, *ISO 26262 - Road vehicles — Functional safety*. ISO, 2011.
[24] CENELEC, *EN 50126-1 - Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 1: Generic RAMS Process*. IEEE, 2017.
[25] A. N. Srivastava and J. Schumann, “Software health management: A necessity for safety critical systems,” *Innov. Syst. Softw. Eng.*, vol. 9, no. 4, pp. 219–233, 2013, doi: 10.1007/s11334-013-0212-0.
[26] L. E. G. Martins and T. Gorschek, “Requirements Engineering for Safety-Critical Systems: Overview and Challenges,” *Focus: Reliability Engineering*, vol. 34, no. 4, IEEE, pp. 49–57, 2017.
[27] E. P. Enoiu, A. Cauevic, D. Sundmark, and P. Pettersson, “A Controlled Experiment in Testing of Safety-Critical Embedded Software,” *Proc. - 2016 IEEE Int. Conf. Softw. Testing, Verif. Validation, ICST 2016*, pp. 1–11, 2016, doi: 10.1109/ICST.2016.15.
[28] V. Hartonas-Garmhausen, S. Campos, A. Cimatti, E. Clarke, and F. Giunchiglia, “Verification of a safety-critical railway interlocking system with real-time constraints,” in *Digest of Papers - 28th Annual International Symposium on Fault-Tolerant Computing, FTCS 1998*, 2000, vol. 2000-Janua, pp. 53–64, doi: 10.1016/S0167-6423(99)00016-7.
[29] H. D. Benington, “Production of large computer programs.,” *IEEE Comput. Soc. Press. Proc. 9th Int. Conf. Softw. Eng.*, vol. 35, no. Issue 3, pp. 299–310, 1983, doi: https://doi.org/10.1109/MAHC.1983.10102.
[30] W. W. Royce, “Managing the development of large software systems: concepts and techniques.,” *ICSE ’87 Proc. 9th Int. Conf. Softw. Eng.*, pp. 328–338, 1987.
[31] J. Marques and A. M. da Cunha, “Tailoring Traditional Software Life Cycles to Ensure Compliance of RTCA DO-178C and DO-331 with Model-Driven Design,” in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, Sep. 2018, pp. 1–8, doi: 10.1109/DASC.2018.8569351.
[32] N. B. Ruparelia, “Software development lifecycle models,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 3, pp. 8–13, 2010, doi:

10.1145/1764810.1764814.

[33] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 8th Editio. United States: New York: McGraw-Hill Education, 2015.

[34] B. W. Boehm, "A spiral model of software development and enhancement," *Computer (Long. Beach. Calif).*, vol. 21, no. 5, pp. 61–72, 1988, doi: https://doi.org/10.1109/2.59.

[35] I. Sommerville, *Software Engineering*, 10th Editi. England: Pearson, 2015.

[36] M. Voelter, A. van Deursen, B. Kolb, and S. Eberle, "Using C language extensions for developing embedded software: a case study," in *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA 2015*, 2015, vol. 25-30-Oct-, pp. 655–674, doi: 10.1145/2814270.2814276.

[37] E. A. Lee, "What's ahead for embedded software?," *Computer (Long. Beach. Calif).*, vol. 33, no. 7, pp. 18–26, Sep. 2000, doi: 10.1109/2.868693.

[38] P. Liggesmeyer and M. Trapp, "Trends in Embedded Software Engineering," *Focus 1*, pp. 19–25, 2009.

[39] M. Shen, W. Yang, G. Rong, and D. Shao, *Applying Agile Methods to Embedded Software Development : A Systematic Review permission to use and the databases are listed in the Table I the systematic review protocol and shows some related software) OR (XP AND software) OR (Lean AND software) "*. IEEE, 2012.

[40] M. Veerabahu, "5 Differences between Embedded Software Engineer and Software Developer," *Linked in*, 2014. https://www.linkedin.com/pulse/5-differences-between-embedded-maharajan (accessed Jul. 02, 2020).

[41] R. J. Kusters, R. van Solingen, and J. J. M. Trienekens, "Strategies for the identification and specification of embedded software quality," in *STEP '99. Proceedings Ninth International Workshop Software Technology and Engineering Practice*, 1999, pp. 33–39, doi: 10.1109/STEP.1999.798477.

[42] R. Fulton, "Assuring certifiability of outsourced software development - A DER's perspective," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, 2005, vol. 2, pp. 1–5, doi: 10.1109/DASC.2005.1563470.

[43] M. Elliott, R. Dawson, and J. Edwards, "Towards real process improvement from internal auditing — A case study," *Softw. Qual. J.*, pp. 53–64, 2006, doi: 10.1007/s11219-006-6001-3.

[44] I. 176/SC 2 Q. Systems, *ISO 9001:2000 Quality management systems — Requirements*. ISO, 2000.

[45] W. Oman, "A case study in SQA audits," *Softw. Qual. J.*, vol. 27, pp. 13–27, 1993, doi: https://doi.org/10.1007/BF00417424.

[46] M. Ortega and T. Rojas, "Construction of a Systemic Quality Model for Evaluating a Software Product," *Softw. Qual. J.*, pp. 219–242, 2003, doi: https://doi.org/10.1023/A:1025166710988.

[47] W. G. Tuohey, "Benefits and Effective Application of Software Engineering Standards," *Softw. Qual. J.*, vol. 10, no. 1, pp. 47–68, 2002, doi: 10.1023/A:1015772816632.

[48] J. Rushby, "New Challenges In Certification For Aircraft Software," in *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*, 2011, doi: https://doi.org/10.1145/2038642.2038675.

[49] J. A. Jiménez, J. A. M. Merodio, and L. F. Sanz, "Checklists for compliance to DO-178C and DO-278A standards," *Comput. Stand. Interfaces*, vol. 52, pp. 41–50, 2017, doi: https://doi.org/10.1016/j.csi.2017.01.006.

[50] M. Graydon, “Retrospectively Documenting SAFEGUARD’s Possession of the Overarching Properties,” in *Proceedings - 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume, DSN-S 2019*, 2019, pp. 27–28, doi: 10.1109/DSN-S.2019.00019.

[51] C. M. Holloway, “Understanding the Overarching Properties,” 2018.

[52] M. Bernhart, S. Reiterer, K. Matt, A. Mauczka, and T. Grechenig, “A Task-Based Code Review Process and Tool to Comply with the DO-278/ED-109 Standard for Air Traffic Managment Software Development: An Industrial Case Study,” in *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering*, Nov. 2011, pp. 182–187, doi: 10.1109/HASE.2011.54.

[53] J. Marques and A. M. da Cunha, “ARES: An Agile Requirements Specification Process for Regulated Environments,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 29, no. 10, pp. 1403–1438, Oct. 2019, doi: 10.1142/S021819401950044X.

[54] J. Münch, O. Armbrust, M. Kowalczyk, and S. Martín, *Software Process Definition and Management*, 1st Editio. Springer-Verlag Berlin Heidelberg, 2012.

[55] J. C. Marques, S. M. H. Yelisetty, A. M. Da Cunha, and L. A. V. Dias, “CARD-RM: A Reference Model for Airborne Software,” in *2013 10th International Conference on Information Technology: New Generations*, Apr. 2013, pp. 273–279, doi: 10.1109/ITNG.2013.44.

[56] SC-205, *RTCA DO-278A, Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems*. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[57] TC 65/SC 65A, *IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems*, 2.0. 2010.

[58] T. 65/SC 65A, *IEC 61511 - Functional safety - Safety instrumented systems for the process industry sector*, 1.0. IEC, 2020.

[59] J. Rushby, “Just-in-Time Certification ∗,” in *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, 2007, no. Iceccs, doi: 10.1109/ICECCS.2007.26.

[60] *Order 8110.49 chg 1 - software approval guidelines*. FAA - Federal Aviation Administration, 2011.

[61] I. Dodd and I. Habli, “Safety certification of airborne software: An empirical study,” *Reliab. Eng. Syst. Saf.*, vol. 98, no. 1, pp. 7–23, 2012, doi: https://doi.org/10.1016/j.ress.2011.09.007.

[62] J. Cheng, R. Metoyer, and J. Cleland-huang, “How Do Practitioners Perceive Assurance Cases in Safety-Critical Software Systems ?,” in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, 2018, pp. 57–60, doi: https://doi.org/10.1145/3195836.3195838.

[63] P. Steele and J. Knight, “Analysis of Critical Systems Certification,” in *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*, Jan. 2014, pp. 129–136, doi: 10.1109/HASE.2014.26.

[64] S. Linling, Z. Wenjin, and T. Kelly, “Do safety cases have a role in aircraft certification?,” *Procedia Eng.*, vol. 17, pp. 358–368, 2011, doi: https://doi.org/10.1016/j.proeng.2011.10.041.

[65] B. R. Poreddy and S. Corns, “Arguing Security of Generic Avionic Mission Control Computer System (MCC) using Assurance Cases,” *Procedia Comput. Sci.*, vol. 6, pp. 499–504, 2011, doi: https://doi.org/10.1016/j.procs.2011.08.092.

[66] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, “Assurance cases and prescriptive software safety certification: A comparative study,” *Saf. Sci.*, vol. 59, pp. 55–71, 2013, doi: https://doi.org/10.1016/j.ssci.2013.04.007.

[67] A. Ruiz, X. Larrucea, and H. Espinoza, “A Tool Suite for Assurance Cases and Evidences: Avionics Experiences,” EuroSPI 20., vol. 543, Springer, Cham, 2015, pp. 63–71.

[68] A. Schwierz and H. Forsberg, “Assurance Case to Structure COTS Hardware Component Assurance for Safety-Critical Avionics,” in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, 2018, pp. 1–10, doi: https://doi.org/10.1109/DASC.2018.8569774.

[69] ACWG, *Goal Structuring Notation Community Standard*. The Assurance Case Working Group, 2018.

[70] I. Habli and T. Kelly, “A Model-Driven Approach to Assuring Process Reliability,” in *2008 19th International Symposium on Software Reliability Engineering (ISSRE)*, 2008, pp. 7–16, doi: https://doi.org/10.1109/ISSRE.2008.19.

[71] N. Silva and M. Vieira, “Towards Making Safety-Critical Systems Safer: Learning from Mistakes,” in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, 2014, pp. 162–167, doi: https://doi.org/10.1109/ISSREW.2014.97.

[72] L. Davila-Nicanor and P. Mejia-Alvarez, “Reliability improvement of web-based software applications,” in *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.*, 2004, pp. 180–188, doi: 10.1109/QSIC.2004.1357959.

[73] Y. Liu, Y. Li, and L. Zhang, “Control mechanisms across a buyer-supplier relationship quality matrix,” *J. Bus. Res.*, vol. 63, no. 1, pp. 3–12, 2010, doi: 10.1016/j.jbusres.2009.01.005.

[74] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, “Systematic Review in Software Engineering,” 2005.

[75] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Technical Report EBSE-2007-01, 2007.

[76] L. E. G. Martins and T. Gorschek, “Requirements engineering for safety-critical systems: A systematic literature review,” *Inf. Softw. Technol.*, vol. 75, pp. 71–89, 2016, doi: https://doi.org/10.1016/j.infsof.2016.04.002.

[77] M. V Zelkowitz and D. Wallace, “Experimental validation in software engineering,” *Inf. Softw. Technol.*, vol. 39, no. 11, pp. 735–743, 1997, doi: https://doi.org/10.1016/S0950-5849(97)00025-6.

[78] M. Unterkalmsteiner, T. Gorschek, A. K. M. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, “Evaluation and Measurement of Software Process Improvement -- A Systematic Literature Review,” *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 398–424, 2012, doi: https://doi.org/10.1109/TSE.2011.26.

[79] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009, doi: 10.1007/s10664-008-9102-8.

[80] J. Vilela, J. Castro, L. E. G. Martins, and T. Gorschek, “Integration between requirements engineering and safety analysis: A systematic literature review,” *J. Syst. Softw.*, vol. 125, pp. 68–92, Mar. 2017, doi: 10.1016/j.jss.2016.11.031.

[81] S. Tiwari and A. Gupta, “A systematic literature review of use case specifications research,” *Inf. Softw. Technol.*, vol. 67, pp. 128–158, Nov. 2015,

doi: 10.1016/j.infsof.2015.06.004.

[82] J. C. Marques, “Processo de Reuso de Software Aeronáutico: Uma Proposta para a EMBRAER,” Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil, 2005.

[83] J. C. Marques and A. M. da Cunha, “Reference Method for Airborne Software Requirements,” in *32nd IEEE/AIAA Digital Avionics Systems Conference (DASC)*, 2013, pp. 1–29, doi: https://doi.org/10.1109/DASC.2013.6719712.

[84] M. J. R. Lemes, F. O. Altoé, A. J. V. Domiciano, and A. J. Carbonari, “Software certification in airborne systems: process and challenges,” in *1st Latin-American Symposium on Dependable Computing*, 2003.

[85] A. Kornecki and J. Zalewski, “Software certification for safety-critical systems: A status report,” *Proc. Int. Multiconference Comput. Sci. Inf. Technol. IMCSIT 2008*, vol. 3, pp. 665–672, 2008, doi: 10.1109/IMCSIT.2008.4747314.

[86] Y. Liu, K. Foster, T. Nguyen, and J. W. Keung, “Quality Assessment of Mission Critical Middleware System Using MEMS,” in *2009 Ninth International Conference on Quality Software*, Aug. 2009, pp. 259–268, doi: 10.1109/QSIC.2009.41.

[87] F. M. Caffery, M. Pikkarainen, and I. Richardson, “AHAA -- Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs,” *2008 ACM/IEEE 30th Int. Conf. Softw. Eng.*, pp. 551–560, 2008, doi: https://doi.org/10.1145/1368088.1368164.

[88] J. S. Sagoo, “An approach for the assurance of legacy systems based on programmable electronic hardware,” *11th Int. Conf. Syst. Saf. Cyber-Security (SSCS 2016)*, 2016, doi: https://doi.org/10.1049/cp.2016.0858.

[89] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones, “Certifiably safe software-dependent systems: challenges and directions,” in *Proceedings of the on Future of Software Engineering - FOSE 2014*, 2014, pp. 182–200, doi: 10.1145/2593882.2593895.

[90] SC-190, *RTCA DO-278, Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance*. RTCA -- Radio Technical Commission for Aeronautics, 2002.

[91] SC-205, *RTCA DO-248C, Supporting Information for DO-178C and DO-278A*. RTCA -- Radio Technical Commission for Aeronautics, 2011.

[92] F. SAFT, *MIL-STD-882C, MILITARY STANDARD: SYSTEM SAFETY PROGRAM REQUIREMENTS*. Department of Defense of USA, 1993.

[93] S-18 and WG-63, *ARP-4754 A -- Guidelines for Development of Civil Aircraft and Systems*. SAE International, 2012.

[94] S-18, *ARP4754 - Certification Considerations for Highly-Integrated Or Complex Aircraft Systems*. SAE International, 1996.

[95] S-18, *ARP4761 - GUIDELINES AND METHODS FOR CONDUCTING THE SAFETY ASSESSMENT PROCESS ON CIVIL AIRBORNE SYSTEMS AND EQUIPMENT*. SAE International, 1996.

[96] SEI, *SEI Software Capability Maturity Model (CMM)*. Software Engineering Institute, 1997.

[97] IEC, *IEC TS 62239-1:2012 - Process management for avionics - Management plan - Part 1: Preparation and maintenance of an electronic components management plan*, 1.0. IEC, 2012.

[98] PE/NPE - Nuclear Power Engineering, *603-1998 - IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations*. IEEE SA, 1998.

[99] SC-180, *RTCA DO-254, Design Assurance Guidance for Airborne Electronic Hardware*. 2000.
[100] E. Denney, G. Pai, and I. Habli, “Perspectives on software safety case development for unmanned aircraft,” *Proc. Int. Conf. Dependable Syst. Networks*, 2012, doi: 10.1109/DSN.2012.6263939.
[101] L. Westfall, *The Certified Software Quality Engineer Handbook*. 2008.
[102] J. C. Mankins, “TECHNOLOGY READINESS LEVELS - A White Paper,” 2004.

Appendix A

SOFTWARE AUDIT MODEL – REQUIREMENTS

Table A-1 – List of SAM requirements

ID	Description	Priority
REQ-1.1	The audit model shall evaluate any software processes independently of the software lifecycle model used.	High
REQ-1.2	The audit model shall evaluate if the Planning Process identifies and documents the characteristics of the following processes described in software plans: ✓ The Requirements Process must include how they are reviewed and traceable to system requirements, identifying each evidence and criteria used. ✓ The Design Process must describe modules, components, inputs, and outputs of each architecture element. ✓ The Implementation process must include the definition of the programming language(s) used. ✓ The Verification Process must include the coverage of requirements, proving the correct implementation of them, including the acceptance criteria and expected results. ✓ The Validation Process must ensure that Requirements, Designs, and Implementations are correct. ✓ The Configuration Control Process must include the capture, flow, and management of problems reports found during development; and ✓ The tools used in the Requirements, Design, Implementation, Verification, Validation, and Configuration Control Processes.	High
REQ-1.3	The audit model shall evaluate if the Requirements Process identifies the software requirements from the system requirements.	High
REQ-1.4	The audit model shall evaluate if the Software Requirements identify software's behavior and external interfaces.	High
REQ-1.5	The audit model shall evaluate if the Software Requirements contain the	High

ID	Description	Priority
	behavioral descriptions or Software Product (SP) properties comprised of Performance aspects; Functionalities; External interfaces, and Limits, ranges, and data type.	
REQ-1.6	The audit model shall evaluate if the Design Process uses the outputs of the Requirements Process to develop software architecture, including modules and allocation of functions expressed by software requirements.	High
REQ-1.7	The audit model shall evaluate if the Design Process decomposes each module into software components and defines data and controls the necessary flow between them.	High
REQ-1.8	The audit model shall evaluate if the Implementation Process uses the outputs of the Design Process to generate the source and executable code.	High
REQ-1.9	The audit model shall evaluate if the Verification Process describes the tests required to confirm that the implementation has been performed correctly according to Requirements and Design.	High
REQ-1.10	The audit model shall evaluate if the Verification Process is applied in three different scopes to ensure coverage of the following types of test: Unit, Integration, and Functional.	High
REQ-1.11	The audit model shall evaluate if the Unit Testing confirms that the Design Process was correctly implemented by each component (unit) of the Software Product (SP).	High
REQ-1.12	The audit model shall evaluate if the Integration Testing confirms the correct data and control flow between the modules and components.	High
REQ-1.13	The audit model shall evaluate if the Functional Testing confirms that Software Requirements were correctly implemented in the Software Product (SP).	High
REQ-1.14	The audit model shall evaluate if the Validation Process confirms the accuracy and correctness of each output produced by all processes already executed, including The Requirements Review; The Design Review; The Implementation Review, and The Verification Review.	High
REQ-1.15	The audit model shall evaluate if the Requirements Review ensures that all requirements captured from the system requirements are correctly refined by the Software Requirements.	High
REQ-1.16	The audit model shall evaluate if the Design Review guarantees the integrity of the modules created.	High

ID	Description	Priority
REQ-1.17	The audit model shall evaluate if the Design Review ensures that: software requirements are correctly allocated into modules, data and control flows between the components are correct, and internal logic details are correctly refined from software requirements.	High
REQ-1.18	The audit model shall evaluate if the Implementation Review ensures that the source-code is correctly produced.	High
REQ-1.19	The audit model shall evaluate if the Implementation Review guarantees the accuracy of the executable code produced, ensuring that all components are appropriately linked and constructed.	High
REQ-1.20	The audit model shall evaluate if the Verification Review ensures that test cases, procedures, and results, as well as review, inspection, and analysis, confirm that the software is implemented as specified by the Requirements, Design, and Implementation.	High
REQ-1.21	The audit model shall evaluate if the Configuration Control Process identifies and control all artifacts generated during software development, including problem reports.	High
REQ-1.22	The audit model shall refine the evaluations required by REQ1-1 to REQ1-22 into a set of questions.	High
REQ-1.23	The audit model shall store the set of questions in a database.	Medium
REQ-1.24	The audit model shall allow the Auditor to indicate the phases under audit to customize the checklist with the applicable questions.	Medium
REQ-1.25	The defined questions shall present three options of answers: yes, no, not applicable.	Low
REQ-1.26	The audit model shall require evidence for each question.	Medium

.

Appendix B

SOFTWARE AUDIT MODEL – QUESTIONS FOR RTCA DO-178C

Table B-1 – Question for RTCA DO-178C

Nº	Phase	ID	Questions	Answer	Objective / DAL
1	Planning	P.01	Are the software plans and standards created, reviewed, and approved as defined in the Software Process established?	Y/N	A-1 Obj. 1, 2, 5, 6, 7 A, B, C, D
2	Planning	P.02	Are software organizations (including quality assurance independence), organizational responsibilities, system lifecycle processes, and certification liaison process detailed?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D
3	Planning	P.03	Are the additional considerations, e.g., previously developed software, option-selectable software, option-selectable software, user-modifiable software, COTS, field-loadable software, multiple-version dissimilar, service history, or others that may be presented in Certification Authorities regulations or guidance (FCARs, CRIs, IP, ACs, AMCs, etc.) described, when applicable?	Y/N/NA	A-1 Obj. 4 A10 Obj. 1, 2 A, B, C, D
4	Planning	P.04	Is the change impact analysis performed describing the necessary	Y/N/NA	A-1 Obj. 4

Nº	Phase	ID	Questions	Answer	Objective / DAL
			details of the applicable modification, including the description of proposed changes, the identification of activities impacted by the changes, and which credits from previous development will be sought?		A, B, C, D
5	Planning	P.05	Is the software level defined for the software, including for each partitioned component?	Y/N	A-1 Obj. 1 A, B, C, D
6	Planning	P.06	Is the Software Development Process described to include inputs, outputs, and transition criteria?	Y/N	A-1 Obj. 2 A, B, C
7	Planning	P.07	Is the Software Development Process described to include software overview, system overview, hardware platform (processor, redundancy)?	Y/N	A-1 Obj. 2 A, B, C
8	Planning	P.08	Is the Requirements Process described to include the high-level requirements definition, bi-directional traceability to system requirements, and requirement hierarchy?	Y/N	A-1 Obj. 1, 2, 3, 4 A-2 Obj. 1 A, B, C, D
9	Planning	P.09	Is the derived requirements process, for high-level and low-level requirements, described to include the requirements definition, rationale definition, and bi-directional traceability to its rationale?	Y/N	A-1 Obj. 1, 2, 3, 4 A-2 Obj. 2, 5 A, B, C, D
10	Planning	P.10	Is the Design Process described to include the low-level requirements definition, bi-directional traceability to high-level requirements, and requirement hierarchy?	Y/N	A-1 Obj. 1, 2, 3, 4 A-2 Obj. 4 A, B, C, D
11	Planning	P.11	Is the Design Process described to include the operating system used?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D

Nº	Phase	ID	Questions	Answer	Objective / DAL
12	Planning	P.12	Is the Implementation Process described to include the definition of the programming languages, applicable standards, naming conventions, and conditions and constraints?	Y/N	A-1 Obj. 1, 2, 3, 4 A2 Obj. 6 A, B, C, D
13	Planning	P.13	Is the Implementation Process described to include the definition of building the software and how to integrate it on the target platform?	Y/N	A-1 Obj. 1, 2, 3, 4 A2 Obj. 7 A, B, C, D
14	Planning	P.14	Is the Implementation Process described to include other relevant aspects of the software like memory mapping, linking, and loading data?	Y/N	A-1 Obj. 1, 2, 3, 4 A2 Obj. 7 A, B, C, D
15	Planning	P.15	Are the sampled software source code and executable object code in accordance with the defined configuration management process?	Y/N	A-1 Obj. 1, 2, 3, 4 A-8 Obj. 1 A, B, C, D
16	Planning	P.16	Is the Verification Process described to include the review of requirements, design, implementation data, and verification data?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D
17	Planning	P.17	Is the Verification Process described to include requirements coverage, proving their correct implementation, testing techniques (unit, integration, and functional), acceptance criteria, expected results, and an alternative method (review, analysis, inspection) when the test is not possible?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D
18	Planning	P.18	Is the Configuration Control Process described to include the artifacts identification, control category, baselines, traceability and capture,	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D

Nº	Phase	ID	Questions	Answer	Objective / DAL
			workflow (including PR classification, OPR criteria) and management of problem reports, archive, release, retrieval, and software load control?		
19	Planning	P.19	Is the Quality Assurance Process described to include the reviews, sampling, audits, issues identification and tracking, and necessary independence?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D
20	Planning	P.20	Is the Tool Qualification Process described for the qualifiable tools, including the necessity of a Tool Qualification Plan for tools with TQL other than 5?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D
21	Planning	P.21	Are the tools used in the requirements, design, implementation, verification, validation, quality assurance, and configuration control process described indicating which must be qualified with its applicable TQL and related justification?	Y/N	A-1 Obj. 1, 2, 3, 4 A, B, C, D
22	Planning	P.22	Is there a previously qualified tool? If yes, the tool qualification strategy as well as its associated data provided?	Y/N/NA	A-1 Obj. 1, 2, 3, 4 A, B, C, D
23	Planning	P.23	Are the software development and verification environment defined?	Y/N	A-1 Obj. 3 A-8 Obj. 6 A, B, C, D
24	Planning	P.24	Is the software verification process providing guidelines on how to perform the regression analysis? If it is used, were provided the necessary justifications?	Y/N/NA	A-1 Obj. 1, 2, 3, 4 A, B, C, D
25	Planning	P.25	Are the standards (requirements,	Y/N/NA	A-1

Nº	Phase	ID	Questions	Answer	Objective / DAL
			design, and source-code) developed as expected?		Obj. 5 A, B, C
26	Planning	P.26	Are the plans and standards consistent among them?	Y/N	A-1 Obj. 7 A, B, C, D
27	Requirements	R.01	Are the sampled software high-level requirements traceable to system requirements and back to software (bi-directional traceability) high-level requirements?	Y/N	A-3 Obj. 6 A, B, C, D
28	Requirements	R.02	Are the sampled software high-level requirements correctly satisfying the system requirements?	Y/N	A-3 Obj. 1 A, B, C, D
29	Requirements	R.03	Are the sampled software derived requirements traced to their rationales?	Y/N	A-2 Obj. 2 A, B, C, D
30	Requirements	R.04	Are the sampled derived requirements provided to the system process to evaluate safety impacts	Y/N	A-2 Obj. 2 A, B, C, D
31	Requirements	R.05	Are the sampled software high-level requirements defining the software functional, performance, safety-related, and external interfaces?	Y/N	A-2 Obj. 1 A, B, C, D
32	Requirements	R.06	Are sampled software high-level requirements, and related algorithms, reviewed to verify accuracy, and consistency?	Y/N	A-3 Obj. 2, 3, 7 A, B, C, D
33	Requirements	R.07	Are the sampled software high-level requirements reviewed to verify the compatibility with the target?	Y/N	A-3 Obj. 3 A, B
34	Requirements	R.08	Are the sampled software high-level requirements reviewed to verify completeness?	Y/N	A-3 Obj. 1, 2 A, B
35	Requirements	R.09	Do sampled software high-level requirements were reviewed to	Y/N	A-3 Obj. 5

Nº	Phase	ID	Questions	Answer	Objective / DAL
			ensure conformance to the established Software Requirements Standards?		A, B, C
36	Requirements	R.10	Are the issues identified for the sampled software high-level requirements treated during the review cycle? If it was not possible to treat, do the issues were recorded as problem reports and treated as per Configuration Management Process?	Y/N	A-8 Obj. 3 A, B, C, D
37	Requirements	R.11	Are the sampled software high-level requirements kept and maintained in accordance with the defined configuration management process?	Y/N	A-8 Obj. 1 A, B, C, D
38	Requirements	R.12	Are the sampled software high-level requirements reviewed with the required independence?	Y/N	A-3 Obj. 1, 2, 7 A, B
39	Design	D.01	Have the sampled software low-level requirements bi-directional traceability to software high-level requirements?	Y/N	A-4 Obj. 6 A, B, C, D
40	Design	D.02	Are the sampled high-level requirements correctly satisfied by the software low-level requirements?	Y/N/NA	A-4 Obj. 1 A, B, C
41	Design	D.03	Are the sampled software derived low-level requirements traced to their rationales?	Y/N	A-2 Obj. 5 A, B, C
42	Design	D.04	Are the sampled software derived low-level requirements provided to the system process to evaluate safety impacts?	Y/N/NA	A-2 Obj. 5 A, B, C
43	Design	D.05	Are the sampled software low-level requirements, and related algorithms, reviewed to verify accuracy, consistency?	Y/N	A-4 Obj. 2, 3, 7 A, B, C, D

Nº	Phase	ID	Questions	Answer	Objective / DAL
44	Design	D.06	Are the sampled software low-level requirements reviewed to verify the compatibility with the target?	Y/N	A-4 Obj. 3 A, B
45	Design	D.07	Are the sampled software low-level requirements reviewed to verify completeness?	Y/N	A-4 Obj. 1, 2 A, B, C
46	Design	D.08	Is the software architecture defined, including components (including the data and control flow between them) and allocation of the functions, data structure definition of a PDI, inputs, and outputs of each element of the architecture and how memory is allocated to components?	Y/N	A-2 Obj. 3 A-4 Obj. 7, 8, 9 A, B, C
47	Design	D.09	Is the software architecture defined based on the software high-level requirements?	Y/N	A-4 Obj. 8 A, B, C
48	Design	D.10	Does the software architecture was reviewed to ensure the decomposition of the modules into software components?	Y/N	A-4 Obj. 8 A, B, C
49	Design	D.11	Are the sampled software low-level requirements and software architecture correctly defining the allocation of the components based on software high-level requirements?	Y/N	A-4 Obj. 3 A, B
50	Design	D.12	Are the sampled software low-level requirements and software architecture correctly defining the correct data flow and control flow between the components?	Y/N	A-4 Obj. 3, 10 A, B
51	Design	D.13	Are the sampled software low-level requirements and software architecture correctly detailing the internal logic from software	Y/N	A-4 Obj. 3, 10 A, B

Nº	Phase	ID	Questions	Answer	Objective / DAL
			components?		
52	Design	D.14	Are the sampled software low-level requirements sufficiently detailed to allow implementation without further design decisions at the software level?	Y/N	A-4 Obj. 3, 10 A, B
53	Design	D.15	When designing partitioning protection are related considerations being considered?	Y/N/NA	A-4 Obj. 13 A, B, C, D
54	Design	D.16	Is the software architecture reviewed to verify consistency and completeness, as well as, compatibility with the target?	Y/N	A-4 Obj. 9, 10 A, B, C
55	Design	D.17	Is the Design Process executed adherent to Software Development Plan?	Y/N	A-4 Obj. 4, 11 A, B
56	Design	D.18	Is the software design architecture defining the strategy for control and data flow monitoring and responses to failure conditions?	Y/N	A-4 Obj. 7 A, B, C
57	Design	D.19	Is the software design partitioning integrity confirmed?	Y/N	A-4 Obj. 13 A, B, C, D
58	Design	D.20	Does the software design was reviewed to ensure conformance to the Software Design Standard?	Y/N	A-4 Obj. 5, 12 A, B, C
59	Design	D.21	Are the issues identified for the sampled software design handled during the review cycle? If it was not possible, were the issues recorded as problem reports and handled as per Configuration Management Process?	Y/N	A-8 Obj. 3 A, B, C, D
60	Design	D.22	Are the software design samples kept and maintained in accordance	Y/N	A-8 Obj. 1 A, B, C, D

Nº	Phase	ID	Questions	Answer	Objective / DAL
			with the defined configuration management process?		
61	Design	D.23	Are the sampled software design reviewed with the required independence?	Y/N	A-4 Obj. 1, 2, 7, 8, 9, 13 A, B
62	Implementation	I.01	Does the sampled software source code have bi-directional tracing to software low-level requirements?	Y/N	A-5 Obj. 5 A, B, C
63	Implementation	I.02	Is the sampled software source code correctly implementing the software low-level requirements and software architecture?	Y/N	A-5 Obj. 1, 2 A, B, C
64	Implementation	I.03	Is the sampled software source code reviewed to verify consistency and, accuracy?	Y/N	A-5 Obj. 6, 7 A, B, C
65	Implementation	I.04	Is the sampled software source code reviewed to verify completeness?	Y/N	A-5 Obj. 1, 6 A, B, C
66	Implementation	I.05	Is the software build performed correctly and following the applicable set of instructions?	Y/N	A-8 Obj. 5 A, B, C, D
67	Implementation	I.06	Is the software build examined for compiling, linking and loading data, and memory map?	Y/N	A-2 Obj. 7 A, B, C, D
68	Implementation	I.07	Is the software load generated correctly and following the applicable set of instructions?	Y/N	A-2 Obj. 7 A, B, C, D
69	Implementation	I.08	Is the sampled software source code conforming to Software Code Standard established?	Y/N	A-5 Obj. 4 A, B, C
70	Implementation	I.09	Are the issues identified for the sampled software implementation recorded as problem reports and treated as per Configuration	Y/N	A-8 Obj. 3 A, B, C, D

Nº	Phase	ID	Questions	Answer	Objective / DAL
			Management Process?		
71	Implementation	I.10	Are the sampled software source code and executable object code in accordance with the defined Software Development process?	Y/N	A-5 Obj. 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D
72	Implementation	I.11	Are the sampled parameter data items reviewed to verify correctness and completeness?	Y/N/NA	A-5 Obj. 8 A, B, C, D
73	Implementation	I.12	Are all elements from the sampled parameter data item verified?	Y/N/NA	A-5 Obj. 9 A, B, C
74	Implementation	I.13	Are the sampled software implementation reviewed with the required independence?	Y/N	A-5 Obj. 1, 2, 6, 8, 9 A, B
75	Implementation	I.14	Are the sampled software source code and executable object code kept and maintained in accordance with the defined configuration management process?	Y/N	A-8 Obj. 1, 2, 3, 4 A, B, C, D
76	Verification	V.01	Are the sampled SW unit test, SW-SW tests, and SW-HW integration tests created in accordance with the established process?	Y/N	A-1 Obj. 1 A, B, C
77	Verification	V.02	Are the sampled software tests traceability data correctly generated based on the Software Process specification?	Y/N	A-6 Obj. 1, 3 A-7 Obj. 3, 4 A, B, C, D
78	Verification	V.03	Are the sampled software tests reviewed to verify consistency and completeness, including normal and robustness aspects?	Y/N	A-7 Obj. 1 A, B, C
79	Verification	V.04	Are the sampled software test fully covering the software high-level requirements and/or software low-	Y/N	A-6 Obj. 2, 4 A-7

Nº	Phase	ID	Questions	Answer	Objective / DAL
			level requirements it traces to?		Obj. 3, 4 A, B, C, D
80	Verification	V.05	Are the sampled software tests and their related results ensuring the correct implementation of software high-level requirements and/or software low-level requirements it traces to?	Y/N	A-6 Obj. 1, 3 A7 Obj. 3,4 A, B, C, D
81	Verification	V.06	Are the sampled software unit tests verifying the correct implementation of each software component (unit) of the software product?	Y/N	A-7 Obj. 1 A, B, C
82	Verification	V.07	Are the sampled software integration tests verifying the correct data and control flow between the components?	Y/N	A-7 Obj. 1 A, B, C
83	Verification	V.08	Are the sampled software high-level requirements verifiable (considering the observability of a variable in the abstraction level specified)?	Y/N	A-3 Obj. 4 A, B, C
84	Verification	V.09	Are the software low-level requirement verifiable (considering the observability of a variable in the abstraction level specified)?	Y/N	A-4 Obj. 4 A, B
85	Verification	V.10	Is the sampled software source code verifiable and testable (considering the observability of a variable in the abstraction level specified)?	Y/N	A-5 Obj. 3 A, B
86	Verification	V.11	Are the sampled software verification campaigns planned and executed as per the Software Verification Process and stored as per Configuration Management Process?	Y/N	A-7 Obj. 1 A-8 Obj. 2 A, B, C
87	Verification	V.12	Are the sampled software verification campaign results reviewed to verify	Y/N	A-7 Obj. 2 A, B, C

Nº	Phase	ID	Questions	Answer	Objective / DAL
			the correctness and completeness?		
88	Verification	V.13	Are the sampled software requirements coverage achieved as expected using the methods defined in the Software Verification Process? And when present, the lack of coverage is justified?	Y/N	A-7 Obj. 3, 4 A, B, C
89	Verification	V.14	Have the sample tests achieved the necessary structural coverage (per software DAL)? When necessary, is the lack of coverage justified?	Y/N	A-7 Obj. 5, 6, 7, 8 A, B, C
90	Verification	V.15	Are the sampled software analyses used to complement the test coverage correctly performed and reviewed?	Y/N	A-7 Obj. 5, 6, 7, 8 A, B, C
91	Verification	V.16	Are the issues identified during sampled software verification campaigns recorded as problem reports?	Y/N	A-8 Obj. 3 A, B, C, D
92	Verification	V.17	Are the sampled software tests and related activities controlled in accordance with the defined configuration management process?	Y/N	A-8 Obj. 1 A, B, C, D
93	Verification	V.18	Are the sampled software analysis (memory, timing, source to object code) performed and reviewed?	Y/N	A-5 Obj. 6 A, B, C
94	Verification	V.19	Is the test environment representative, and is it set up correctly?	Y/N	A-6 Obj. 5 A, B, C, D
95	Verification	V.20	Are the sampled software verification executed with the required independence?	Y/N	A-6 Obj. 3, 4 A-7 Obj. 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B

Nº	Phase	ID	Questions	Answer	Objective / DAL
96	Configuration Management	CM.01	Are the sampled artifacts identified as per the Software Configuration Management Process?	Y/N	A-8 Obj. 1 A, B, C, D
97	Configuration Management	CM.02	Are the software life cycle data controlled according to their respective configuration management control categories established in the Configuration Management Process?	Y/N	A-8 Obj. 1, 2, 3, 4 A, B, C, D
98	Configuration Management	CM.03	Are the sampled artifacts stored and maintained as per Configuration Management Process?	Y/N	A-8 Obj. 1 A, B, C, D
99	Configuration Management	CM.04	Are the sampled baselines and traceability data generated as per Configuration Management Process?	Y/N	A-8 Obj. 2 A, B, C, D
100	Configuration Management	CM.05	Are the sampled issues identified recorded and tracked as per Configuration Management Process?	Y/N	A-8 Obj. 3 A, B, C, D
101	Configuration Management	CM.06	Are the archive, retrieval, and release executed as per Configuration Management Process?	Y/N	A-8 Obj. 4 A, B, C, D
102	Configuration Management	CM.07	Is the software load control, including part numbering, media identification, and intermixability, executed as per Configuration Management Process?	Y/N	A-8 Obj. 5 A, B, C, D
103	Configuration Management	CM.08	Is the software development and verification environment controlled according to Software Development Process approved?	Y/N	A-8 Obj. 6 A, B, C, D
104	Quality Assurance	QA.01	Is the assurance obtained that the plans and standards comply with the applicable regulation?	Y/N	A-9 Obj. 1 A, B, C, D
105	Quality Assurance	QA.02	Is the assurance obtained that the software product was developed and verified in compliance with the plans	Y/N	A-9 Obj. 2, 3, 4 A, B, C, D

Nº	Phase	ID	Questions	Answer	Objective / DAL
			and standards approved?		
106	Quality Assurance	QA.03	Is the software conformity review performed for the software released?	Y/N/NA	A-9 Obj. 5 A, B, C, D
107	Quality Assurance	QA.04	Are the QA artifacts generated following the Quality Assurance Process established?	Y/N	A-1 Obj. 1 A, B, C, D
108	Quality Assurance	QA.05	Are the non-conformities identified recorded and tracked as per Quality Assurance Process?	Y/N	A-8 Obj. 3 A, B, C, D
109	Quality Assurance	QA.06	Are the tool conformity reviews performed for the tools already qualified?	Y/N	A-9 Obj. 5 A, B, C, D
110	Tool Qualification	TQ.01	Are plans and qualification process established for the qualified tools?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification Considerations
111	Tool Qualification	TQ.02	Is the tool operational requirements document created and reviewed as defined in the tool qualification process?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification Considerations
112	Tool Qualification	TQ.03	Is the tool operational installation report created and reviewed as per the Tool Qualification Process?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification Considerations
113	Tool Qualification	TQ.04	Is the tool operational verification document created and reviewed as per the Tool Qualification Process?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification Considerations
114	Tool Qualification	TQ.05	Are the tool verification campaigns planned, executed, and stored as per Tool Qualification Process?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification

Nº	Phase	ID	Questions	Answer	Objective / DAL
					Considerations
115	Tool Qualification	TQ.06	Are the tool verification campaign results reviewed to verify the correctness?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification Considerations
116	Tool Qualification	TQ.07	Are the tool closure artifacts generated for the tool already qualified?	Y/N/NA	See RTCA DO-330 – Software Tool Qualification Considerations
117	Closure	C.01	Are the closure artifacts created and reviewed as per the Software Process established?	Y/N	A-1 Obj. 1 A, B, C, D
118	Closure	C.02	Are all the software life cycle data recorded in the software artifacts?	Y/N	A-10 Obj. 3 A, B, C, D
119	Closure	C.03	Are the process deviations recorded in the artifacts?	Y/N	A-10 Obj. 3 A, B, C, D
120	Closure	C.04	Are the open problem reports recorded and justified in the artifacts?	Y/N	A-10 Obj. 3 A, B, C, D
121	Closure	C.05	Is the compliance data summarized in the artifacts?	Y/N	A-10 Obj. 3 A, B, C, D
122	Closure	C.06	Is the compliance statement provided in the closure artifacts?	Y/N	A-10 Obj. 3 A, B, C, D
123	Closure	C.07	Are the closure artifacts compliant with the applicable regulations?	Y/N	A-10 Obj. 3 A, B, C, D

Appendix C

SOFTWARE AUDIT MODEL – QUESTIONS FOR IEC 62304

Table C- 1 – Question for IEC 62304

Nº	Phase	ID	Questions	Answer	Class
1	Planning	P.01	Is the software safety classification defined for the software, including for each partitioned component?	Y/N	A, B, C
2	Planning	P.02	Are system software processes, deliverable of the activities and tasks, traceability between system requirements, software requirements, software system test, and risk control measures, software configuration control and change management, and software problem resolution detailed?	Y/N	A, B, C
3	Planning	P.03	Is the Software Development Process described to include inputs, outputs, scope, complexity, and software safety classifications?	Y/N	A, B, C
4	Planning	P.04	Is the Software Development Process described to include processes for the software system and the deliverables of the activities and tasks?	Y/N	A, B, C
5	Planning	P.05	Is the Requirements Process described to include the references for system requirements, software system	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
			requirements definition, software requirements definition, traceability to software system requirements, and requirement hierarchy?		
6	Planning	P.06	Are the software plan(s) and standards created, updated, and reviewed, and approved as defined in the Software Process established?	Y/N	A, B, C
7	Planning	P.07	Is the software development plan(s) referencing procedures for coordinating the validation activities necessary for software development and design?	Y/N	A, B, C
8	Planning	P.08	Are the software development plan(s) described to include standards, methods, and associated tools?	Y/N	C
9	Planning	P.09	Is the Verification Process described to include how to integrate the software items and perform the testing during integration?	Y/N	B, C
10	Planning	P.10	Is the Verification Process described to include deliverables that require verification, verification tasks, milestones to perform the verification, acceptance criteria for the deliverables verification?	Y/N	A, B, C
11	Planning	P.11	Are the software development plan(s) described to include software risk management regarding the activities and tasks needed?	Y/N	A, B, C
12	Planning	P.12	Is the documentation planning providing or referencing a document that provides title, naming or naming convention, purpose, the intended audience of the document, and procedures and responsibilities for	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
			development, review, approval, and modification of the development life cycle?		
14	Planning	P.13	Is the Configuration Control Process described to include the classes, types categories of items to controlled, configuration items identification, control category, the organization responsible for these activities, relationship with other organizations, and management of problem resolution process?	Y/N	A, B, C
14	Planning	P.14	Is the software development plan(s) describing the tools, items, or settings, used to develop the medical device software, such as compiler/assembler versions, make files, batch files, and specific environment settings?	Y/N	B, C
15	Planning	P.15	Is the Configuration Control Process described to include the configuration items under-documented configuration management before they are verified?	Y/N	B, C
16	Requirements	R.01	Are the sampled software system requirements traceable to system requirements?	Y/N	A, B, C
17	Requirements	R.02	Are the sampled software system requirements correctly satisfying the system requirements?	Y/N	A, B, C
18	Requirements	R.03	Are the sampled software system requirements defining the software functions and capabilities, that is, performance, physical characteristics, computer environment, and capability for upgrades?	Y/N	A, B, C
19	Requirements	R.04	Are the sampled software system requirements defining the software	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
			inputs and outputs, such as data characteristics, ranges, limits, and defaults?		
20	Requirements	R.05	Are the sampled software system requirements defining the software external interfaces, security, usability, data definition and database, installation and acceptance, operation and maintenance, user documentation, user maintenance, and regulatory requirements?	Y/N	A, B, C
21	Requirements	R.06	Are the sampled software system requirements defining software-driven alarms, warnings, operator messages?	Y/N	A, B, C
22	Requirements	R.07	Are the sample software system requirements, including risk control measures implemented in software for hardware failures and potential software defects?	Y/N	B, C
23	Requirements	R.08	Is the risk analysis re-evaluated after requirements were established and updated?	Y/N	A, B, C
24	Requirements	R.09	Are the software requirements, including the system requirements, updated as a result of the software analysis activity?	Y/N	A, B, C
25	Requirements	R.10	Are sampled software requirements reviewed to verify implementation, consistency, and completeness, as well as it is testable, uniquely identifiable, and traceable?	Y/N	A, B, C
26	Design	D.01	Is the software architecture defined based on the software system requirements?	Y/N	B, C
27	Design	D.02	Does the software architecture was developed to ensure the	Y/N	B, C

Nº	Phase	ID	Questions	Answer	Class
			decomposition of the modules into software components and interfaces with external components?		
28	Design	D.03	If a software item is identified as SOUP, are the functional and performance requirements for the SOUP item described?	Y/N	B, C
29	Design	D.04	If a software item is identified as SOUP, are the system hardware and software necessary to support the proper operation (processor type and speed, memory type and size, communication, and display software requirements.) of the SOUP item described?	Y/N	B, C
30	Design	D.05	Is the segregation between software items that is essential to risk control identified and stated how to ensure that the segregation is effective?	Y/N	C
31	Design	D.06	Is the software architecture reviewed to verify the implementation of related requirements including those relating to risk control, the interfaces between software items and between them and hardware, as well as, if supports the proper operation of any SOUP items?	Y/N	B, C
32	Design	D.07	Is the software detailed design developed and documented in detail for each software unit of the software item?	Y/N	C
33	Design	D.08	Is the software detailed design developed and documented in detail for any interfaces between the software unit and external components (hardware or software), as well as any interfaces between software units?	Y/N	C

Nº	Phase	ID	Questions	Answer	Class
34	Design	D.09	Is the software detailed design verified and documented regarding the architecture implementation and if it is free from contradiction?	Y/N	C
35	Implementation	I.01	Is the sampled software source code correctly implementing the software requirements and software architecture?	Y/N	A, B, C
36	Verification	V.01	Are the sampled SW unit test created per the established process?	Y/N	B, C
37	Verification	V.02	Are the sampled SW integration tests created per the established process?	Y/N	B, C
38	Verification	V.03	Are the sampled SW system tests created per the established process?	Y/N	B, C
39	Design	D.10	If a software item is identified as SOUP, are the system hardware and software necessary to support the proper operation (processor type and speed, memory type and size, communication, and display software requirements.) of the SOUP item described?	Y/N	B, C
40	Verification	V.04	Are the sampled software tests, related test procedure, and their acceptance criteria reviewed to verify consistency and correctness?	Y/N	B, C
41	Planning	P.16	Is the Verification Process described to include strategies, methods, procedures, and acceptance criteria for verifying each software unit?	Y/N	B, C
42	Verification	V.05	Are the sampled software unit tests verifying the correct implementation of each software component (unit) of the software item per the acceptance criteria?	Y/N	B, C

Nº	Phase	ID	Questions	Answer	Class
43	Verification	V.06	Are the sampled software unit additional acceptance criteria considering the proper event sequence, data and control flow, planned resource allocation, fault handling (error definition, isolation, and recovery), initialization of variables, self-diagnostics, memory management, and memory overflows, and boundary conditions?	Y/N	C
44	Verification	V.07	Are the sampled software unit and software integration verification results documented as expected?	Y/N	B, C
45	Verification	V.08	Are the sampled software integration tests verifying the software units integration into software items and/or software systems, as well as, hardware items, software items, and support for manual operations?	Y/N	B, C
46	Verification	V.09	Are the sampled software integration tests addressing whether the integrated software items perform as intended, that is, the required functionality of the software, implementation of risk control measures, specified timing, other behavior, the functioning of interfaces, and testing under abnormal conditions?	Y/N	B, C
47	Verification	V.10	Are the sampled regression tests executed appropriately in such a way that demonstrates that defects have not been introduced?	Y/N	B, C
48	Verification	V.11	Are the sampled software system verification tests executed as per the Software Verification Process and	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
			stored, including reference to test case procedures, the results (pass/fail), version of the software tested, relevant HW and SW test configuration, test tools, date tested, and tester identification?		
49	Verification	V.12	Are the sampled software integration verification tests executed as per the Software Verification Process and stored, including the results (pass/fail), records necessary to be repeated, and tester identification?	Y/N	B, C
50	Verification	V.13	Are the sampled software verification tests executed as per the Software Verification Process and stored including the results (pass/fail), records necessary to be repeated, and tester identification?	Y/N	B, C
51	Verification	V.14	Are the anomalies identified during sampled software verification recorded per the problem resolution process?	Y/N	B, C
52	Verification	V.15	Are the sampled software system test fully covering the software requirements, including input stimuli, expected outcomes, pass/fail criteria, procedures, and the adequacy of verification strategy and test procedures?	Y/N	A, B, C
53	Verification	V.16	When changes were performed during software tests execution, the software tests were modified, updated, or created additional tests due to modification and re-executed to ensure that no unintended side effects were introduced?	Y/N	B, C
54	Verification	V.17	When changes were performed during	Y/N	B, C

Nº	Phase	ID	Questions	Answer	Class
			software tests execution, the risk management activities have been conducted?		
55	Release	RL.01	Is the software verification activities complete and its results evaluated before the software release?	Y/N	B, C
56	Release	RL.02	Are the anomalies identified during the development process identified, documented, and evaluated to ensure that nonacceptable risks are imposed on a software product?	Y/N	B, C
57	Configuration Management	CM.01	Are the archive and retrieval executed as per the Development Process?	Y/N	A, B, C
58	Configuration Management	CM.02	Is the retention policy/procedure adequately defined?	Y/N	A, B, C
59	Configuration Management	CM.03	Is the software release reliable, that is, its related procedure includes the following information: replication, media labeling, packaging, protection, storage, and delivery?	Y/N	A, B, C
60	Planning	P.17	Are the software maintenance plan describing the necessary maintenance procedures, criteria for determining whether a problem, use of the software risk management and software problem resolution processes, and use of the software configuration management process for managing modifications to the existing system?	Y/N	A, B, C
61	Maintenance	M.01	Are the sampled maintenance execution monitoring and documenting the received feedbacks, and evaluating if this feedbacks have any impact on safety?	Y/N	A, B, C
62	Maintenance	M.02	Are the problems identified during maintenance recorded per the problem	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
			resolution process?		
63	Maintenance	M.03	Are the feedbacks that results in changes in the software system analyzed to identify its effect on the organization, release, and interfaces?	Y/N	B, C
64	Maintenance	M.04	Are the feedbacks that results in change requests evaluated and approved?	Y/N	A, B, C
65	Maintenance	M.05	Are the regulation authorities and users notified of the approved change requests, their impacts on a software product, the reason for these changes, and how to obtain a new version of the software product?	Y/N	A, B, C
66	Maintenance	M.06	Are the software development or maintenance process used to implement the approved changes?	Y/N	A, B, C
67	Risk Management	RM.01	Are the risk analysis conducted to identify the software items that can contribute to a hazardous situation, and for the ones classified as hazardous, it details the classification reason?	Y/N	B, C
68	Risk Management	RM.02	For the SOUP items, is the list of anomalies evaluated to identify any relevant item that could lead to a hazard situation?	Y/N	B, C
69	Risk Management	RM.03	Is the risk management file identifying the potential causes of the software item contributing to a hazardous situation, as well as the sequence of events that lead to it?	Y/N	B, C
70	Risk Management	RM.04	Are the risk control measures defined and documented for the hazardous situations identified in the risk management file?	Y/N	B, C

Nº	Phase	ID	Questions	Answer	Class
71	Risk Management	RM.05	Are the risk control measures implemented in the software item specified in the requirements, with a new assessment of the software class, and following the development process defined?	Y/N	B, C
72	Risk Management	RM.06	Are the implementation of the risk control measures verified and documented as established in the verification process and risk management process?	Y/N	B, C
73	Risk Management	RM.07	Is the risk control measures traceability documented as expected, that is, following the sequence hazardous situation, software item, software cause, risk control measure, and them to its verification?	Y/N	B, C
74	Risk Management	RM.08	Are the changes in the software product (including SOUP) analyzed to determine the introduction of potential hazard situations or the necessity of risk control measures?	Y/N	A, B, C
75	Risk Management	RM.09	Are the changes in the software, including changes in the SOUP, analyzed to verify if it interferes with existing risk control measures?	Y/N	B, C
76	Configuration Management	CM.04	Is the Configuration Control Process described to include the SOUP configuration items used with at least title, manufacturer, and unique SOUP designator (version, release date, patch number, or upgrade designation)?	Y/N	A, B, C
77	Configuration Management	CM.05	Are the sampled configuration items identified as per the Software Configuration Management Process?	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
78	Configuration Management	CM.06	Are the configuration items changed only through approved change requests?	Y/N	A, B, C
79	Configuration Management	CM.07	Are the sampled changes implemented per the change requested approved, and the necessary activities repeated including the software class assessment?	Y/N	A, B, C
80	Configuration Management	CM.08	Are the implementation of the sampled change request verified and documented as established in the verification and configuration management processes?	Y/N	A, B, C
81	Configuration Management	CM.09	Is the sampled change request traceability documented as expected, that is, following the sequence change request, problem report, and the approval of the change?	Y/N	A, B, C
82	Configuration Management	CM.10	Are the sampled change request data recorded and tracked as per Configuration Management Process?	Y/N	A, B, C
83	Problem Resolution	PR.01	Are the sampled problem reports fulfilled with the type (corrective, preventive, or adaptive to a new environment), scope (size of the change, number of devices affected, resources involved, time to change), and criticality (performance, safety, or security)?	Y/N	A, B, C
84	Problem Resolution	PR.02	Are the sampled problem reports have documented in the investigation to identify the causes, relevance to safety, related change requests for the actions needed to correct the problem? If no action was performed, a consistent rationale was adequately	Y/N	A, B, C

Nº	Phase	ID	Questions	Answer	Class
			documented?		
85	Problem Resolution	PR.03	Are the sampled problem reports advised for the relevant stakeholders?	Y/N	A, B, C
86	Problem Resolution	PR.04	Are the sampled change requests followed per Problem Resolution Process?	Y/N	A, B, C
87	Problem Resolution	PR.05	Are the sampled artifacts related to the change control process stored and maintained as per Configuration Management and Problem Resolution Processes?	Y/N	A, B, C
88	Problem Resolution	PR.06	Are the sampled problem reports analyzed to detect trends?	Y/N	A, B, C
89	Problem Resolution	PR.07	Are the sampled problem reports verified to ensure their correct resolution, adverse trends reversed, appropriately implemented, and check the existence of additional problems?	Y/N	A, B, C
90	Problem Resolution	PR.08	Are the test documentation including information such as test results and related data, anomalies found, relevant hardware and software test configurations, and relevant test tools?	Y/N	A, B, C

Appendix D

INSTANTIATION 1 – TAILORED CHECKLISTS AND ANSWERS

Below is presented Table D-1, which contains the tailored checklist fulfilled during the software audit executed in Section 7.2.1. The answers were modified only to remove confidential data; except for these modifications, the checklist is the exact one used in the software audit.

Table D-1 – Tailored checklist for Instantiation 1

Nº	ID	Phase	Question	Option	√	Evidence / Comments
1	D.01	Design	Have the sampled software low-level requirements bi-directional traceability to software high-level requirements?	Y/N	**Y**	The bi-directional traceability to software high-level requirements was performed as expected and defined. **See ref.:** 5, 7, 56, 72, 77, 85
2	D.02	Design	Are the sampled high-level requirements correctly satisfied by the software low-level requirements?	Y/N	**Y**	The sample low-level requirements correctly satisfy the high-level requirements through implementation and checked by the review process.

Nº	ID	Phase	Question	Option	√	Evidence / Comments
						The CR-XYZ01 has been opened to treat an issue related to it. **See ref.:** 5, 7, 56, 57, 58, 59, 60, 72, 73, 74, 75, 77, 78, 79, 80, 85, 86, 87, 88
3	D.05	Design	Are the sampled software low-level requirements, and related algorithms, reviewed to verify accuracy, consistency?	Y/N	**Y**	All the review was conducted as expected. **See ref.:** 5, 7, 57, 58, 59, 60, 73, 74, 75, 78, 79, 80, 86, 87, 88
4	D.06	Design	Are the sampled software low-level requirements reviewed to verify the compatibility with the target?	Y/N	**Y**	All the review was conducted as expected considering the compatibility with the target. The CR-XYZ02 has been opened to treat an issue related to it. **See ref.:** 5, 7, 57, 58, 59, 60, 73, 74, 75, 78, 79, 80, 86, 87, 88
5	V.08	Verification	Are the software low-level requirements and architecture verifiable (considering the observability of a variable in the abstraction level specified)?	Y/N	**Y**	The architecture was not part of the audit, since no changes were performed in the PREFIXDOC001. Regarding the low-level requirements, see the following evidence. **See ref.**: 5, 7, 57, 58, 59, 60, 73, 74, 75, 78, 79, 80, 86, 87, 88

Nº	ID	Phase	Question	Option	√	Evidence / Comments
6	D.07	Design	Are the sampled software low-level requirements reviewed to verify completeness?	Y/N	**Y**	The only sample selected related to TOOL2 Model FUNCTION3 Review Completeness is related in the references below, as well as its related review checklist. **See ref.**: 35, 86, 87, 88
7	D.11	Design	Are the sampled software low-level requirements and software architecture correctly defining the allocation of the components based on software high-level requirements?	Y/N	**Y**	The architecture was not part of the audit, since no changes were performed in the PREFIXDOC001. All samples selected were reviewed for the correctness of the model components. **See ref.**: 5, 7, 57, 58, 59, 60, 73, 74, 75, 78, 79, 80, 86, 87, 88
8	D.12	Design	Are the sampled software low-level requirements and software architecture correctly defining the correct data flow and control flow between the components?	Y/N	**Y**	The analysis of data flow and control flow between the components has been performed. **See ref.**: 154, 155, 156
9	D.13	Design	Are the sampled software low-level requirements and software architecture correctly detailing the internal logic from software components?	Y/N	**Y**	The architecture was not part of the audit, since no changes were performed in the PREFIXDOC001. For the low-level requirements, this analysis is performed with the review checklists. **See ref.**: 5, 7, 57, 58, 59, 60, 73, 74, 75, 78, 79, 80, 86, 87, 88

Nº	ID	Phase	Question	Option	√	Evidence / Comments
10	D.14	Design	Are the sampled software low-level requirements sufficiently detailed to allow implementation without further design decisions at the software level?	Y/N	**Y**	According to design implementation and related review, there were no further design decisions. **See ref.**: 5, 7, 56, 57, 58, 59, 60, 72, 73, 77, 78, 79, 80, 85, 86, 87, 88
11	D.20	Design	Does the software design was reviewed to ensure conformance to the Software Design Standard?	Y/N	**Y**	For the low-level requirements, this analysis is performed with the review checklists. **See ref.**: 5, 7, 57, 58, 59, 60, 73, 78, 79, 80, 86, 87, 88
12	D.21	Design	Are the issues identified for the sampled software design handled during the review cycle? If it was not possible, were the issues recorded as problem reports and handled as per Configuration Management Process?	Y/N	**Y**	The issues opened were treated as per the Configuration Management Process defined. **See ref.**: 5, 7, 57, 58, 59, 60, 73, 78, 86
13	D.22	Design	Are the software design samples kept and maintained in accordance with the defined configuration management process?	Y/N	**Y**	According to design implementation and related review, there were no further design decisions. **See ref.**: 5, 7, 55, 56, 57, 58, 59, 60, 71, 72, 73, 76, 77, 78, 79, 80, 85, 86, 87, 88
14	D.23	Design	Are the sampled software design reviewed with the required independence?	Y/N	**Y**	The required independence has been respected as expected. **See ref.**: 5, 7, 57, 58, 59, 60, 73, 78, 79, 80, 86, 87, 88
15	I.01	Implementation	Does the sampled software source code have bi-directional tracing to software low-level requirements?	Y/N	**Y**	This question has been checked in the selected source code sample and related review checklists. **See ref.**: 32, 34, 36, 37, 50, 51, 52, 53, 54, 61, 62, 63,

Nº	ID	Phase	Question	Option	√	Evidence / Comments
						64, 65, 66, 67, 68, 69, 81, 82, 83, 84, 89, 90, 91, 93, 94, 95, 96, 97
16	I.02	Implementation	Is the sampled software source code correctly implementing the software low-level requirements and software architecture?	Y/N	**Y**	This question has been checked in the selected source code sample and related review checklists. For the issues identified, a CR has been opened to treat the issue. **See ref.**: 32, 34, 36, 37, 50, 51, 52, 53, 54, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 81, 82, 83, 84, 89, 90, 91, 93, 94, 95, 96, 97
17	I.03	Implementation	Is the sampled software source code reviewed to verify consistency and accuracy?	Y/N	**Y**	This question has been checked in the selected source code sample and related review checklists. **See ref.**: 32, 34, 36, 37, 51, 52, 53, 65, 66, 67, 68, 82, 90, 91, 94, 95, 96
18	V.09	Verification	Is the sampled software source code verifiable and testable (considering the observability of a variable in the abstraction level specified)?	Y/N	**Y**	This question has been checked in the selected source code sample and related review checklists. **See ref.**: 32, 34, 36, 37, 51, 65, 66, 82, 90, 94
19	I.04	Implementation	Is the sampled software source code reviewed to verify completeness?	Y/N	**Y**	For the TOOL5 rules, a CR for this type of review was not selected as part of the sample. The other completeness aspects are covered through requirement tests and related source code coverage. **See ref.**: 120, 122, 128, 132, 139

Nº	ID	Phase	Question	Option	√	Evidence / Comments
20	I.08	Implementation	Is the sampled software source code conforming to Software Code Standard established?	Y/N	**Y**	This question has been checked in the selected source code sample and related review checklists. **See ref.**: 32, 34, 36, 37, 50, 51, 52, 53, 54, 61, 62, 63, 64, 65, 66, 67, 68, 69, 81, 82, 83, 84, 89, 90, 91, 93, 94, 95, 96, 97
21	I.09	Implementation	Are the issues identified for the sampled software implementation recorded as problem reports and treated as per Configuration Management Process?	Y/N	**Y**	The issues opened were treated as per the Configuration Management Process defined. **See ref.**: 90
22	I.10	Implementation	Are the sampled software source code and executable object code in accordance with the defined Software Development Process?	Y/N	**Y**	The sample selected did not present the executable object code. For the source code, the software development process has been followed as expected. **See ref.**: 32, 34, 36, 37, 50, 51, 52, 53, 54, 61, 62, 63, 64, 65, 66, 67, 68, 69, 81, 82, 83, 84, 89, 90, 91, 93, 94, 95, 96, 97
23	I.13	Implementation	Are the sampled software implementation reviewed with the required independence?	Y/N	**Y**	The required independence has been respected as expected. **See ref.**: 32, 34, 36, 37, 51, 52, 53, 65, 66, 67, 68, 82, 83, 84, 90, 94, 95, 96
24	I.14	Implementation	Are the sampled software source code and executable object code kept and maintained in accordance with the	Y/N	**Y**	The sample selected did not present the executable object code. The source code has been kept and maintained as expected.

Nº	ID	Phase	Question	Option	√	Evidence / Comments
			defined configuration management process?			**See ref.**: 32, 34, 36, 37, 50, 51, 52, 53, 54, 61, 62, 63, 64, 65, 66, 67, 68, 69, 81, 82, 83, 84, 89, 90, 93, 94, 95, 96, 97
25	V.01	Verification	Are the sampled SW unit tests, SW-SW tests, and SW-HW integration tests created in accordance with the established process?	Y/N	**Y**	The TESTs were created following the defined process. **See ref.**: 98, 103, 108, 112, 114
26	V.02	Verification	Are the sampled software tests traceability data correctly generated based on the Software Process specification?	Y/N	**Y**	The traceability data has been correctly generated as per the software process specification. **See ref.**: 54, 69, 97, 101, 102, 106, 107, 110, 111, 113, 115, 116, 118, 119, 120
27	V.03	Verification	Are the sampled software tests reviewed to verify consistency and completeness, including normal and robustness aspects?	Y/N	**Y**	For all selected samples, the review checklist has been fulfilled. For the issues identified, a CR has been opened to treat the issue. **See ref.**: 100, 105, 109, 117, 121, 127, 131, 138, 142
28	V.04	Verification	Are the sampled software test fully covering the software high-level requirements and/or software low-level requirements it traces to?	Y/N	**Y**	The selected sample covers the software high-level requirements, as well, the low-level requirements they trace to. In addition, the review checklist also checked this data. **See ref.**: 98, 100, 103, 105, 108, 109, 112, 114, 117, 121

Nº	ID	Phase	Question	Option	√	Evidence / Comments
29	V.05	Verification	Are the sampled software tests and their related results ensuring the correct implementation of software high-level requirements and/or software low-level requirements it traces to?	Y/N	**Y**	The software tests and related results ensure the correct implementation of the software high-level requirements and software low-level requirements. **See ref.**: 98, 103, 108, 112, 114, 123, 126, 130, 135, 136, 137, 141
30	V.06	Verification	Are the sampled software unit tests verifying the correct implementation of each software component (unit) of the software product?	Y/N	**Y**	The sampled software unit tests verify the correct implementation of each software component. **See ref.**: 103, 108, 112, 114, 123, 126, 135, 137, 141
31	V.07	Verification	Are the sampled software integration tests verifying the correct data and control flow between the components?	Y/N	**Y**	The sampled software integration tests verify the correct implementation of each software component. **See ref.**: 98, 123, 126, 135, 137, 141
32	V.11	Verification	Are the sampled software verification campaigns planned and executed as per the Software Verification Process and stored as per Configuration Management Process?	Y/N	**Y**	The sampled verification campaigns were planned and stored as per the defined process. **See ref.**: 122, 129, 132, 139
33	V.12	Verification	Are the sampled software verification campaign results reviewed to verify the correctness and completeness?	Y/N	**Y**	The sampled software verification campaign results were reviewed as expected. **See ref.**: 127, 131, 138, 142
34	V.13	Verification	Are the sampled software requirements coverage achieved as expected using the methods defined in	Y/N	**Y**	The software requirements coverage is performed through the TESTs review as per the defined process.

Nº	ID	Phase	Question	Option	√	Evidence / Comments
			the Software Verification Process? And when present, the lack of coverage is justified?			**See ref.**: 99, 100, 104, 105, 109, 117, 121
35	V.16	Verification	Are the issues identified during sampled software verification campaigns recorded as problem reports?	Y/N	**Y**	The issues opened were treated as per the Configuration Management Process defined. **See ref.**: 127, 131
36	V.17	Verification	Are the sampled software tests and related activities controlled in accordance with the defined configuration management process?	Y/N	**Y**	All the verification artifacts were generated as per the defined process. **See ref.**: 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142
37	V.19	Verification	Is the test environment representative, and is it set up correctly?	Y/N	**Y**	During the TR review, the environment has been checked and no issues were identified. **See ref.**: 124, 125, 129,133, 134, 140
38	V.20	Verification	Are the sampled software verification executed with the required independence?	Y/N	**Y**	The reviews and test execution were conduct with the requested independence. **See ref.**: 100, 105, 109, 117, 121, 122, 127, 129, 131, 132, 133, 134, 138, 139, 140, 142
39	CM.01	Configuration Management	Are the sampled artifacts identified as per the Software Configuration	Y/N	**Y**	All artifacts selected as sample for this audit, have followed the identification rules as per the software

Nº	ID	Phase	Question	Option	√	Evidence / Comments
			Management Process?			configuration management process. **See ref.**: See all artifacts listed in Table D-2
40	CM.02	Configuration Management	Are the software life cycle data controlled according to their respective configuration management control categories established in the Configuration Management Process?	Y/N	**Y**	All artifacts selected as sample for this audit, have followed the configuration data control as per the software configuration management process. **See ref.**: See all artifacts listed in Table D-2
41	CM.03	Configuration Management	Are the sampled artifacts stored and maintained as per Configuration Management Process?	Y/N	**Y**	All artifacts selected as sample for this audit, have been stored and maintained as per the software configuration management process. **See ref.**: See all artifacts listed in Table D-2 **See issue:** OBS-001
42	CM.04	Configuration Management	Are the sampled baselines and traceability data generated as per Configuration Management Process?	Y/N	**N**	The baselines and traceabilities were generated per the configuration management process, however, some issues were identified in their content. **See ref.**: 4, 5, 6, 7, 8, 9, 10, 11, 12, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153. **See issue:** Action-001, OBS-001, OBS-003
43	CM.05	Configuration Management	Are the sampled issues identified recorded and tracked as per Configuration Management Process?	Y/N	**Y**	All the sampled change reports followed the configuration control process established. **See ref.**: 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48

Nº	ID	Phase	Question	Option	√	Evidence / Comments
44	CM.06	Configuration Management	Are the archive, retrieval, and release executed as per Configuration Management Process?	Y/N	**Y**	The release has been performed as per the defined process; however, an observation has been raised due to a lack of process. **See ref**.: 157, 158, 159 **See issue**: OBS-004
45	CM.07	Configuration Management	Is the software load control, including part numbering, media identification, and intermixability, executed as per Configuration Management Process?	Y/N	**Y**	All the procedures were followed as expected. **See ref.**: 157, 158, 159
46	CM.08	Configuration Management	Is the software development and verification environment controlled according to Software Development Process approved?	Y/N	**Y**	The software development and verification environment are controlled as per the defined process. **See ref.**: 124, 125, 129, 133, 134, 140

Below is presented Table D-2, which contains the list of artifacts consulted during the software audit executed in Section 7.2.1. Their identification and revisions have been modified due to companies confidentiality.

Table D-2 – Instantiation 1 – artifacts audited

Nº	Comment ID	Artifact	Revision
1	4	PREFIX_BL_PROJECT_BAS1_001_LOAD_X_Y_Z_3	N/A
2	5	PREFIX_BL_PROJECT_BAS1_002_LOAD_X_Y_Z_1	N/A
3	6	PREFIX_BL_PROJECT_BAS1_003_LOAD_X_Y_Z_4	N/A
4	7	PREFIX_BL_PROJECT_BAS1_004_LOAD_X_Y_Z_2	N/A
5	8	PREFIX_BL_PROJECT_TEST_001	N/A
6	9	PREFIX_BL_PROJECT_TEST_002	N/A
7	10	PREFIX_BL_PROJECT_TEST_003	N/A
8	11	PREFIX_BL_PROJECT_TEST_004	N/A
9	12	PREFIX_BL_PROJECT_TEST_005	N/A
10	31	CR-XYZ12	N/A
11	32	CR-XYZ06	N/A
12	33	CR-XYZ13	N/A
13	34	CR-XYZ08	N/A
14	35	CR-XYZ03	N/A
15	36	CR-XYZ09	N/A
16	37	CR-XYZ10	N/A
17	38	CR-XYZ14	N/A
18	39	CR-XYZ15	N/A
19	40	CR-XYZ16	N/A
20	41	CR-XYZ17	N/A
21	42	CR-XYZ18	N/A
22	43	CR-XYZ19	N/A
23	44	CR-XYZ20	N/A
24	45	CR-XYZ21	N/A
25	46	CR-XYZ22	N/A
26	47	CR-XYZ23	N/A
27	48	CR-XYZ24	N/A
28	50	SUB-FUNCTION2_ACID2.c	[0020]
29	51	PREFIXCKL_FUNCTION4_SUB-FUNCTION2_CKL-A.xls	[0021]
30	52	SUB-FUNCTION2_ACID2_TOOL1Report.txt	[0022]
31	53	SUB-FUNCTION2_ACID2_Dataflow.pdf SUB-FUNCTION2_ACID2_Standard_Checking.pdf	[0023]

Nº	Comment ID	Artifact	Revision
32	54	CodeTrace.log, Trace_Report.html, folder_list.txt, generateTraceabilityMatrix.bat	[0024]
33	56	FUNCTION1.rtf	[0001]
34	55	CR-XYZ04	N/A
35	57	PREFIXCKL_SUB-FUNCTION1_CKL-B.xls	[0005]
36	58	SUB-FUNCTION1_ACID1_TOOL1Report.txt, SUB-FUNCTION1_ACID2_TOOL1Report.txt, SUB-FUNCTION1_ACID3_TOOL1Report.txt	[0006]
37	59	SUB-FUNCTION1_TOOL2.htm	[0007]
38	60	SUB-FUNCTION1_par_Parameter.html, SUB-FUNCTION1_TOOL3.pdf	[0014]
39	61	SUB-FUNCTION1_ACID1.c	[0025]
40	62	SUB-FUNCTION1_ACID2.c	[0025]
41	63	SUB-FUNCTION1_ACID3.c	[0025]
42	64	SUB-FUNCTION1_def.h	[0025]
43	65	PREFIXCKL_CustomHeader_CKL-A.xls	[0026]
44	66	PREFIXCKL_FUNCTION4_SUB-FUNCTION1_CKL-A.xls	[0027]
45	67	SUB-FUNCTION1_ACID1_ConstantCheckerReport.txt, SUB-FUNCTION1_ACID2_ConstantCheckerReport.txt, SUB-FUNCTION1_ACID3_ConstantCheckerReport.txt	[0028]
46	68	SUB-FUNCTION1_ACID1_Standard_Checking.pdf, SUB-FUNCTION1_ACID1_Dataflow.pdf, SUB-FUNCTION1_ACID2_Dataflow.pdf, SUB-FUNCTION1_ACID2_Standard_Checking.pdf, SUB-FUNCTION1_ACID3_Dataflow.pdf, SUB-FUNCTION1_ACID3_Standard_Checking.pdf	[0029]
47	69	CodeTrace.log, Trace_Report.html, folder_list.txt	[0030]
48	70	PROGRAM_BOX2.ld	[0042]
49	71	CR-XYZ05	N/A
50	72	FUNCTION2.rtf	[0002]
51	73	PREFIXCKL_FUNCTION2_CKL-B.xls	[0008]
52	74	FUNCTION2_ACID1_TOOL1Report.txt, FUNCTION2_ACID2_TOOL1Report.txt, FUNCTION2_ACID3_ TOOL1Report.txt	[0009]
53	75	FUNCTION2_TOOL2.htm, FUNCTION2_TOOL3.pdf	[0010]
54	76	CR-XYZ06	N/A
55	77	FUNCTION1.rtf	[0003]

Nº	Comment ID	Artifact	Revision
56	78	PREFIXCKL_SUB-FUNCTION1_CKL-B.xls	[0011]
57	79	SUB-FUNCTION1_ACID1_TOOL1Report.txt, SUB-FUNCTION1_ACID2_TOOL1Report.txt, SUB-FUNCTION1_ACID3_TOOL1Report.txt	[0012]
58	80	SUB-FUNCTION1_TOOL3.pdf, SUB-FUNCTION1_par_Parameter.html, SUB-FUNCTION1_TOOL2.htm	[0012]
59	81	SUB-FUNCTION1_def.h, SUB-FUNCTION1_ACID2.c	[0031]
60	82	PREFIXCKL_ FUNCTION4_ SUB-FUNCTION1_CKL-A.xls	[0032]
61	83	SUB-FUNCTION1_ACID2_ TOOL1Report.txt	[0033]
62	84	SUB-FUNCTION1_ACID2_Dataflow.pdf, SUB-FUNCTION1_ACID2_Standard_Checking.pdf	[0034]
63	85	FUNCTION3.rtf	[0004]
64	86	PREFIXCKL_MASTER_ FUNCTION3 _CKL-B.xls	[0013]
65	87	SignalAssessment.xls	[0015]
66	88	TOOL4 - Issues.xlsx	[0016]
67	89	FUNCTION5_CiaAppFUNCTION5.c	[0035]
68	90	PREFIXCKL_ FUNCTION5_CiaAppFUNCTION5_CKL-A.xls	[0036]
69	91	FUNCTION5_CiaAppFUNCTION5_Dataflow.pdf, FUNCTION5_CiaAppFUNCTION5_Standard_Checking.pdf	[0037]
70	93	CR-XYZ11	N/A
71	94	SUB-FUNCTION3 _ACID1.c, SUB-FUNCTION3 _ACID2.c	[0038]
72	95	PREFIXCKL_ FUNCTION4_ SUB-FUNCTION3_CKL-A.xls	[0039]
73	96	SUB-FUNCTION3_ACID1_TOOL1Report.txt, SUB-FUNCTION3_ACID2_TOOL1Report.txt, SUB-FUNCTION3 _ACID3_TOOL1Report.txt	[0040]
74	97	SUB-FUNCTION3 _ACID1_Dataflow.pdf, SUB-FUNCTION3 _ACID1_Standard_Checking.pdf, SUB-FUNCTION3 _ACID2_Dataflow.pdf, SUB-FUNCTION3 _ACID2_Standard_Checking.pdf	[0041]
75	98	TEST-1164.xlsx, TV_TEST-1164-TC-05.xlsx, TV_TEST-1164-TC-13.xlsx	[0048]
76	99	UDC_Report_TEST-1164_SUB-FUNCTION4.txt	[0080]
77	100	PREFIXCKL_TEST-1164_CKL-A.xlsx	[0064]
78	101	TC_to_REQ.html, TC_to_REQ.log,	[0053]

Nº	Comment ID	Artifact	Revision
		TC_to_REQ_Unfound_TESTs_Files_Log.txt, TC_to_REQ_Unfound_TESTs_Verification_List_Log.txt	
79	102	Verification_List.xlsx	[0054]
80	103	TEST-1242.xlsx, TV_TEST-1242-TC-10.xlsx, TV_TEST-1242-TC-13.xlsx, TV_TEST-1242-TC-14.xlsx	[0049]
81	104	UDC_Report_TEST-1242_SUB-FUNCTION5.txt	[0081]
82	105	PREFIXCKL_TEST-1242_CKL-A.xlsx	[0065]
83	106	TC_to_REQ.html, TC_to_REQ.log, TC_to_REQ_Unfound_TESTs_Files_Log.txt, TC_to_REQ_Unfound_TESTs_Verification_List_Log.txt	[0055]
84	107	Verification_List.xlsx	[0056]
85	108	TEST-1103.xlsx, TV_TEST-1103-TC-01.xlsx, TV_TEST-1103-TC-02.xlsx, TV_TEST-1103-TC-03.xlsx	[0050]
86	109	PREFIXCKL_TEST-1103_CKL-A.xlsx	[0066]
87	110	TC_to_REQ.html, TC_to_REQ.log, TC_to_REQ_Unfound_TESTs_Files_Log.txt, TC_to_REQ_Unfound_TESTs_Verification_List_Log.txt	[0057]
88	111	Verification_List.xlsx	[0058]
89	112	TEST-1075.xlsx	[0051]
90	113	Verification_List.xlsx	[0059]
91	114	TEST-1224.xlsx, Inspec_TEST-1224-TC-01.xlsx	[0052]
92	115	TC_to_REQ.html, TC_to_REQ.log, TC_to_REQ_Unfound_TESTs_Files_Log.txt, TC_to_REQ_Unfound_TESTs_Verification_List_Log.txt	[0060]
93	116	Verification_List.xlsx	[0061]
94	117	PREFIXCKL_TEST-1224_CKL-A.xlsx	[0067]
95	118	Verification_List.xlsx	[0062]
96	119	TC_to_REQ.html, TC_to_REQ.log, TC_to_REQ_Unfound_TESTs_Files_Log.txt, TC_to_REQ_Unfound_TESTs_Verification_List_Log.txt, JustificationFile.xlsx	[0063]
97	120	TESTToTestCampaign.xlsx	[0043]
98	121	PREFIXCKL_CompletenessReview_CKL-A.xlsx	[0067]
99	122	PREFIXCAMP_CODE1HL.doc	[0044]
100	123	TestResults_PTG.xls	[0072]
101	124	PREFIXCKL_CODE1HL_PTG_TR.xlsx	[0082]
102	125	PREFIXCKL_CODE1HL_SWTB_TR.xlsx	[0083]

Nº	Comment ID	Artifact	Revision
103	126	PREFIXTestResults_CODE1HL_LOAD_X_Y_Z_3_002 (PTG1, PTG2, PTG3, and SWTB)	[0073]
104	127	PREFIXCKL_CAMP_CODE1HL.xlsx	[0068]
105	128	PREFIXCAMP_CODE1LL.doc	[0045]
106	129	PREFIXCKL_CODE1LL_SWTB_TR.xlsx	[0079]
107	130	PREFIXTestResults_CODE1LL_LOAD_X_Y_Z_4_001 (SWTB)	[0074]
108	131	PREFIXCKL_CAMP_CODE1LL.xlsx	[0069]
109	132	PREFIXCAMP_MODELHL.doc	[0046]
110	133	PREFIXCKL_MODELHL_PTG_TR.xlsx	[0084]
111	134	PREFIXCKL_MODELHL_SWTB_TR.xlsx	[0085]
112	135	TestResults_PTG.xls	[0075]
113	136	PREFIXTestResults_MODELHL_LOAD_X_Y_Z_3_001 (SWTB)	[0076]
114	137	Inspec_TEST-1229-TC-01.xlsx, Inspec_TEST-1229-TC-02.xlsx	[0077]
115	138	PREFIXCKL_CAMP_MODELHL.xlsx	[0070]
116	139	PREFIXCAMP_MODELLL.doc	[0047]
117	140	PREFIXCKL_MODELLL_PTG_TR.xlsx	[0086]
118	141	TestResults_PTG.xls	[0078]
119	142	PREFIXCKL_CAMP_MODELLL.xlsx	[0071]
120	143	BAS_INT-CKL_ PREFIX_BL_PROJECT_BAS1_001.xlsx	[0087]
121	144	BAS_INT-CKL _ PREFIX_BL_PROJECT_BAS1_002.xlsx	[0088]
122	145	BAS_INT-CKL _ PREFIX_BL_PROJECT_BAS1_003.xlsx	[0089]
123	146	PREFIXDOC002-LOAD_RN.xlsx	[0090]
124	147	BAS_INT-CKL _ PREFIX_BL_PROJECT_BAS1_004.xlsx	[0091]
125	148	PREFIXDOC003-LOAD_ RN.xlsx	[0092]
126	149	BAS_INT-CKL_PREFIX_BL_PROJECT_TEST_001.xlsx	[0093]
127	150	BAS_INT-CKL_PREFIX_BL_PROJECT_TEST_002.xlsx	[0094]
128	151	BAS_INT-CKL_PREFIX_BL_PROJECT_TEST_003.xlsx	[0095]
129	152	BAS_INT-CKL_PREFIX_BL_PROJECT_TEST_004.xlsx	[0096]
130	153	BAS_INT-CKL_PREFIX_BL_PROJECT_TEST_005.xlsx	[0097]
131	154	DataControlCouplingAnalysisReport.doc	[0017]
132	155	CODE1_CouplingAnalysis.xlsx	[0018]
133	156	MBD_CouplingAnalysis.xlsx	[0019]
134	157	PREFIXDOC002	[0098]
135	158	PREFIX_PROJECT_022-BaselineProcedures.doc	[0099]
136	159	PREFIXDOC004 - Annex X - Build Instructions.doc	[0100]

Appendix E

INSTANTIATION 2 – TAILORED CHECKLISTS AND ANSWERS

Below is presented Table E-1, which contains the tailored checklist fulfilled during the software audit executed in Section 7.2.2. The answers were modified only to remove confidential data; except for these modifications, the checklist is the exact one used in the software audit.

Table E- 1 – Tailored checklist for Instantiation 2

Nº	ID	Phase	Question	Options	√	Comments / Evidence
1	R.01	Requirements	Are the sampled software high-level requirements traceable to system requirements and back to software (bi-directional traceability) high-level requirements?	Y/N	**Y**	The bi-directional traceability to parent system requirements was performed as expected and defined. **See issue**: [OBS-004] **See ref**: 31, 32, 33, 34, 35, 36, 37
2	R.02	Requirements	Are the sampled software high-level requirements correctly satisfying the	Y/N	**Y**	High-Level requirements satisfy parent system requirements. **See ref**: 31, 32, 33, 34, 35, 36, 37

Nº	ID	Phase	Question	Options	√	Comments / Evidence
			system requirements?			
3	R.03	Requirements	Are the sampled software derived requirements traced to their rationales?	Y/N	**Y**	Although the sampled High-Level requirements were not classified as derived, they were correctly traced to applicable rationale. **See ref**: 34, 35, 36, 37, 38
4	R.05	Requirements	Are the sampled software high-level requirements defining the software functional, performance, safety-related, and external interfaces?	Y/N	**Y**	The sampled High-Level requirements were correctly defined. **See ref**: 34, 35, 36, 37
5	R.06	Requirements	Are the sampled software high-level requirements, and related algorithms, reviewed to verify accuracy and consistency?	Y/N	**Y**	The review was conducted as expected considering the algorithm aspects, accuracy, and consistency. **See ref**: 37
6	R.07	Requirements	Are the sampled software high-level requirements reviewed to verify the compatibility with the target?	Y/N	**Y**	The review was conducted as expected considering compatibility with the target aspects. **See ref**: 37
7	V.07	Requirements	Are the sampled software high-level requirements verifiable (considering the observability of a variable in the abstraction level specified)?	Y/N	**Y**	The sampled requirements were verifiable. This aspect is evaluated during the review. **See ref**: 34, 35, 36, 37
8	R.08	Requirements	Are the sampled software high-level requirements reviewed to verify completeness?	Y/N	**Y**	The updated sections were reviewed for completeness and the baseline description contains the statement of the completeness for the module. **See ref**: 37, 13

Nº	ID	Phase	Question	Options	√	Comments / Evidence
9	R.09	Requirements	Do the sampled software high-level requirements were reviewed to ensure conformance to the established Software Requirements Standards?	Y/N	**Y**	The review was conducted as expected considering conformance to the applicable standard. **See ref**: 37
10	R.11	Requirements	Are the sampled software high-level requirements kept and maintained in accordance with the defined configuration management process?	Y/N	**Y**	The sampled High-Level Requirements were implemented in the branch. Process for baseline creation due to the branch was agreed and documented by CR-XYZ25 (link to CR). **See ref**: 34, 35, 36
11	R.12	Requirements	Are the sampled software high-level requirements reviewed with the required independence?	Y/N	**Y**	The review was conducted with independence. **See ref**: 37
12	D.01	Design	Have the sampled software low-level requirements bi-directional traceability to software high-level requirements?	Y/N	**Y**	The bi-directional traceability to High-Level Requirements was performed as expected and defined. **See ref**: 40, 41, 42, 43
13	D.02	Design	Are the sampled high-level requirements correctly satisfied by the software low-level requirements?	Y/N	**Y**	Design Model satisfies High-Level Requirements. **See ref**: 39, 40, 41, 42, 43
14	D.05	Design	Are the sampled software low-level requirements, and related algorithms, reviewed to verify accuracy, consistency?	Y/N	**Y**	The review was conducted as expected considering the algorithm aspects, accuracy, and consistency. **See ref**: 41, 42, 43
15	D.06	Design	Are the sampled software low-level requirements reviewed to verify the compatibility with the target?	Y/N	**Y**	The review was conducted as expected considering compatibility with the target aspects. **See ref**: 41, 42, 43

Nº	ID	Phase	Question	Options	√	Comments / Evidence
16	V.08	Design	Are the software low-level requirements and architecture verifiable (considering the observability of a variable in the abstraction level specified)?	Y/N	**Y**	According to PREFIXDOC005, section 4.2.2 the TOOL3 models do not define the software architecture. The sampled requirements were verifiable. This aspect is evaluated during the review. **See ref**: 39, 40, 41, 42, 43
17	D.07	Design	Are the sampled software low-level requirements reviewed to verify completeness?	Y/N	**Y**	The review was conducted as expected considering the completeness aspects. **See ref**: 41, 42, 43
18	D.11	Design	Are the sampled software low-level requirements and software architecture correctly defining the allocation of the components based on software high-level requirements?	Y/N	**Y**	According to PREFIXDOC005, section 4.2.2 the TOOL3 models do not define the software architecture. For the Design Model, this analysis is performed with the review checklists. **See ref**: 41, 42, 43
19	D.12	Design	Are the sampled software low-level requirements and software architecture correctly defining the correct data flow and control flow between the components?	Y/N	**Y**	According to PREFIXDOC005, section 4.2.2 the TOOL3 models do not define the software architecture. For the Design Model, this analysis is performed with the review checklists. **See ref**: 41, 42, 43
20	D.13	Design	Are the sampled software low-level requirements and software architecture correctly detailing the internal logic from software components?	Y/N	**Y**	According to PREFIXDOC005, section 4.2.2 the TOOL3 models do not define the software architecture. For the Design Model, this analysis is performed with the review checklists. **See ref**: 41, 42, 43

Nº	ID	Phase	Question	Options	√	Comments / Evidence
21	D.14	Design	Are the sampled software low-level requirements sufficiently detailed to allow implementation without further design decisions at the software level?	Y/N	**Y**	According to PREFIXDOC005, section 4.2.2 the TOOL3 models do not define the software architecture. For the Design Model, this analysis is performed with the review checklists. **See ref**: 41, 42, 43
22	D.20	Design	Does the software design was reviewed to ensure conformance to the Software Design Standard?	Y/N	**Y**	The review was conducted as expected considering conformance to the applicable standard. **See ref**: 41, 42, 43
23	D.22	Design	Are the software design samples kept and maintained in accordance with the defined configuration management process?	Y/N	**Y**	The artifacts were correctly kept in accordance with the applicable documentation. **See ref**: 39, 40, 41, 42, 43
24	D.23	Design	Are the sampled software design reviewed with the required independence?	Y/N	**Y**	The review was conducted with independence. **See ref**: 41, 42, 43
25	V.01	Verification	Are the sampled SW unit tests, SW-SW tests, and SW-HW integration tests created in accordance with the established process?	Y/N	**Y**	According to PREFIXDOC005, section 4.2.2.3 only Integration Tests are applicable. The software tests were created according to the process established in PREFIXDOC005. **See ref**: 14, 15, 44, 45, 46, 48, 50, 51, 52, 55, 58, 61, 64, 66, 68, 82, 86, 90, 94, 98, 102, 106, 110, 114, 118, 124, 129, 133, 137, 141, 145, 149, 153, 157, 161, 165, 169, 173, 177, 182
26	V.02	Verification	Are the sampled software tests traceability data correctly generated based on the Software Process specification?	Y/N	**Y**	The bi-directional traceability to requirements was correctly generated. **See ref**: 14, 15, 16, 17, 44, 45, 46, 48, 50, 51, 52, 55, 58, 61, 64, 66, 68, 82, 86, 90, 94, 98, 102, 106, 110, 114, 118, 124, 129, 133, 137, 141, 145, 149, 153, 157, 161, 165, 169, 173, 177, 182

Nº	ID	Phase	Question	Options	√	Comments / Evidence
27	V.03	Verification	Are the sampled software tests reviewed to verify consistency and completeness, including normal and robustness aspects?	Y/N	**Y**	The review was conducted as expected considering consistency, completeness, and normal/robustness aspects. **See ref**: 47, 49, 53, 54, 56, 57, 59, 60, 62, 63, 65, 67, 69, 83, 87, 91, 95, 99, 103, 107, 111, 115, 119, 120, 125, 126, 130, 134, 138, 142, 146, 150, 154, 158, 162, 166, 170, 174, 178
28	V.04	Verification	Are the sampled software test fully covering the software high-level requirements and/or software low-level requirements it traces to?	Y/N	**Y**	The sampled tests cover the selected requirements. **See ref**: 14, 15, 16, 17, 44, 45, 46, 48, 50, 51, 52, 55, 58, 61, 64, 66, 68, 82, 86, 90, 94, 98, 102, 106, 110, 114, 118, 124, 129, 133, 137, 141, 145, 149, 153, 157, 161, 165, 169, 173, 177, 182
29	V.05	Verification	Are the sampled software tests and their related results ensuring the correct implementation of software high-level requirements and/or software low-level requirements it traces to?	Y/N	**Y**	The sampled tests and related results ensure the correct implementation of the requirements. **See ref**: 14, 15, 16, 17, 18, 44, 45, 46, 48, 50, 51, 52, 55, 58, 61, 64, 66, 68, 72, 74, 76, 78, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 121, 124, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 182, 184, 185, 186, 187, 188, 190, 192, 194, 196
30	V.07	Verification	Are the sampled software integration tests verifying the correct data and control flow between the components?	Y/N	**Y**	The sampled tests verify the data and control flow between components. **See ref**: 44, 45, 46, 48, 50, 51, 52, 55, 58, 61, 64, 66, 68, 72, 74, 76, 78, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 121, 124, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155,

Nº	ID	Phase	Question	Options	√	Comments / Evidence
						157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 184, 185, 186, 187, 188, 190, 192, 194, 196
31	V.11	Verification	Are the sampled software verification campaigns planned and executed as per the Software Verification Process and stored as per Configuration Management Process?	Y/N	**Y**	The verification campaigns were executed according to the process and witnessed by PPQA Engineer. **See ref**: 14, 15, 70, 80
32	V.12	Verification	Are the sampled software verification campaign results reviewed to verify the correctness and completeness?	Y/N	**Y**	The review was conducted as expected considering the correctness and completeness of the results. **See ref**: 73, 75, 77, 79, 85, 89, 93, 97, 101, 105, 109, 113, 117, 122, 128, 132, 136, 140, 144, 148, 152, 156, 160, 164, 168, 172, 176, 180, 183, 189, 191, 193, 195, 197
33	V.13	Verification	Are the sampled software requirements coverage achieved as expected using the methods defined in the Software Verification Process? And when present, the lack of coverage is justified?	Y/N	**Y**	The coverage of the requirements is achieved with the TESTs execution and review activities. **See ref**: 14, 15, 16, 17, 18, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 72, 73, 74, 75, 76, 77, 78, 79, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 184, 185, 186, 187, 188, 190,

Nº	ID	Phase	Question	Options	√	Comments / Evidence
						192, 194, 196, 183, 189, 191, 193, 195, 197
34	V.16	Verification	Are the issues identified during sampled software verification campaigns recorded as problem reports?	Y/N	**Y**	The failed results were captured in CRs. See Observation. **See issue**: OBS-003 **See ref**: 185, 186, 187, 194, 196, 197
35	V.17	Verification	Are the sampled software tests and related activities controlled in accordance with the defined configuration management process?	Y/N	**Y**	All verification results were stored according to the CM procedures. **See ref**: 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 72, 73, 74, 75, 76, 77, 78, 79, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197.
36	V.19	Verification	Is the test environment representative, and is it set up correctly?	Y/N	**Y**	The environment is presented in the Verification Procedures document and is evaluated by the PPQA Engineer during its execution. **See ref**: 14, 15, 70, 80.

Nº	ID	Phase	Question	Options	√	Comments / Evidence
37	V.20	Verification	Are the sampled software verification executed with the required independence?	Y/N	**Y**	The execution was performed with independence. **See ref**: 72, 73, 74, 75, 76, 77, 78, 79, 84, 85, 88, 89, 92, 93, 96, 97, 100, 101, 104, 105, 108, 109, 112, 113, 116, 117, 121, 122, 127, 128, 131, 132, 135, 136, 139, 140, 143, 144, 147, 148, 151, 152, 155, 156, 159, 160, 163, 164, 167, 168, 171, 172, 175, 176, 179, 180, 181, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197
38	CM.01	Configuration Management	Are the sampled artifacts identified as per the Software Configuration Management Process?	Y/N	**Y**	All artifacts selected as sample for this audit, have followed the identification rules as per the software configuration management process. **See ref.**: See all artifacts listed in Table E-2.
39	CM.02	Configuration Management	Are the software life cycle data controlled according to their respective configuration management control categories established in the Configuration Management Process?	Y/N	**Y**	All artifacts selected as sample for this audit, have followed the configuration data control as per the software configuration management process. **See ref.**: See all artifacts listed in Table E-2.
40	CM.03	Configuration Management	Are the sampled artifacts stored and maintained as per Configuration Management Process?	Y/N	**Y**	All artifacts selected as sample for this audit, have been stored and maintained as per the software configuration management process. **See ref.**: See all artifacts listed in Table E-2.
41	CM.04	Configuration Management	Are the sampled baselines and traceability data generated as per Configuration Management Process?	Y/N	**Y**	The baselines and trace abilities were correctly generated. **See ref**: 5, 6, 7 and 8

Nº	ID	Phase	Question	Options	√	Comments / Evidence
42	CM.05	Configuration Management	Are the sampled issues identified recorded and tracked as per Configuration Management Process?	Y/N	**Y**	The CRs were executed correctly. **See ref**: 19, 20, 21, 22, 23, 24, 25, 26, 27, 28
43	CM.08	Configuration Management	Is the software development and verification environment controlled according to Software Development Process approved?	Y/N	**Y**	The environment is presented in the Verification Procedures document and is evaluated by the PPQA Engineer during its execution. **See issue**: [OBS-005] **See ref**: 14, 15, 70, 80

Below is presented Table E-2, which contains the list of artifacts consulted during the software audit executed in Section 7.2.2. Their identification and revisions have been modified due to companies confidentiality.

Table E-2 –Instantiation 2 – artifacts audited

Nº	Comment ID	Artifact	Revision
1	5	2020_08_21_BASXp4a	[0154]
2	6	PREFIXDOC007_RevD	[0155]
3	7	PREFIXDOC006_RevAQ	[0156]
4	8	PREFIXDOC006_RevAP__PREFIXDOC007_RevC	[0157]
5	13	BDR - FUNCTION1 BAS Xp1a	[0003]
6	14	PREFIXDOC006	AP, AQ
7	15	PREFIXDOC007	C, D
8	16	PREFIXDOC008	AQ
9	17	PREFIXDOC009	C
10	18	PREFIXDOC010	E
11	19	CR-XYZ25	N/A
12	20	CR-XYZ26	N/A
13	21	CR-XYZ27	N/A
14	22	CR-XYZ28	N/A
15	23	CR-XYZ29	N/A
16	24	CR-XYZ30	N/A
17	25	CR-XYZ31	N/A
18	26	CR-XYZ32	N/A
19	27	CR-XYZ11	N/A
20	28	CR-XYZ33	N/A
21	31	SYSTEM1-SYSREQSW-N1-7200	[0001]
22	32	SYSTEM1-SYSREQSW-N1-7201	[0001]
23	33	SYSTEM1-SYSREQSW-N1-7202	[0001]
24	34	FUNCTION1-SYSREQSW-N2-9702	[0001]
25	35	FUNCTION1-SYSREQSW-N2-9703	[0001]
26	36	FUNCTION1-SYSREQSW-N2-9835	[0001]
27	37	FUNCTION1-SYSREQSW-N2_Table.xlsx	[0002]
28	38	SYSTEM1-RAT-NEW1	[0001]
29	40	SUB-FUNCTION3.mdl	[0004]
30	39	SUB-FUNCTION3_par.m	[0004]
31	41	SUB-FUNCTION3_checklist_ACID1.xlsx	[0005]
32	42	SUB-FUNCTION3_checklist_ACID2.xlsx	[0005]

Nº	Comment ID	Artifact	Revision
33	43	SUB-FUNCTION3_checklist_ACID3.xlsx	[0006]
34	44	TEST-1242.xlsx	[0007]
35	45	TV_TEST-1242-TC-10.xlsx	[0008]
36	46	TV_SUB-FUNCTION5_MainFnc_1.xls	[0009]
37	47	TV_SUB-FUNCTION5_MainFnc_1_CKL.xlsx	[0010]
38	48	TV_SUB-FUNCTION5_MainFnc_3.xls	[0011]
39	49	TV_SUB-FUNCTION5_MainFnc_3_CKL.xlsx	[0046]
40	50	TEST-1245/TEST-1245.xlsx	[0012]
41	51	TV_TEST-1245-TC-21.xlsx	[0013]
42	52	TV_SUB-FUNCTION6.xls	[0014]
43	53	TV_SUB-FUNCTION6_ACID2_CKL.xlsx	[0047]
44	54	TV_SUB-FUNCTION6_CKL.xlsx	[0047]
45	55	TV_SUB-FUNCTION2_3.xls	[0015]
46	56	TV_SUB-FUNCTION2_3_ACID2_CKL.xlsx	[0048]
47	57	TV_SUB-FUNCTION2_3_CKL.xlsx	[0048]
48	58	TV_SUB-FUNCTION2_1.xls	[0016]
49	59	TV_SUB-FUNCTION2_1_ACID2_CKL.xlsx	[0049]
50	60	TV_SUB-FUNCTION2_1_CKL.xlsx	[0049]
51	61	TV_SUB-FUNCTION2_4.xls	[0017]
52	62	TV_SUB-FUNCTION2_4_ACID2_CKL.xlsx	[0050]
53	63	TV_SUB-FUNCTION2_4_CKL.xlsx	[0050]
54	64	TV_SUB-FUNCTION2_5.xls	[0018]
55	65	TV_SUB-FUNCTION2_5_CKL.xlsx	[0051]
56	66	TV_SUB-FUNCTION7_1.xls	[0019]
57	67	TV_SUB-FUNCTION7_1_CKL.xlsx	[0052]
58	68	TV_SUB-FUNCTION8_4.xls	[0020]
59	69	TV_SUB-FUNCTION8_4_CKL.xlsx	[0053]
60	70	PREFIX_PROJECT2_SW_QA_WITNESS_011	[0115]
61	72	TV_TG_SUB-FUNCTION6_final.PDF	[0078]
62	73	TV_TG_SUB-FUNCTION6_final_CKL.xlsx	[0117]
63	74	TV_TG _SUB-FUNCTION2_1_final.pdf	[0079]
64	75	TV_TG _SUB-FUNCTION2_1_final_CKL.xlsx	[0118]
65	76	TV_TG _SUB-FUNCTION2_3_final.pdf	[0080]
66	77	TV_TG _SUB-FUNCTION2_3_final_CKL.xlsx	[0119]
67	78	TV_TG _SUB-FUNCTION5_3 _final.pdf	[0081]
68	79	TV_TG _SUB-FUNCTION5_3 _final_CKL.xlsx	[0120]
69	80	PREFIX_PROJECT2_SW_QA_WITNESS_012	[0116]

Nº	Comment ID	Artifact	Revision
70	82	TV_SUB-FUNCTION7_1.xls	[0021]
71	83	TV_SUB-FUNCTION7_1_CKL.xlsx	[0054]
72	84	TV_TG _SUB-FUNCTION7_1_final.pdf	[0082]
73	85	TV_TG _SUB-FUNCTION7_1_final_CKL.xlsx	[0121]
74	86	TV_SUB-FUNCTION8_4.xls	[0022]
75	87	TV_SUB-FUNCTION8_4_CKL.xlsx	[0055]
76	88	TV_TG _SUB-FUNCTION8_4_final.pdf	[0083]
77	89	TV_TG _SUB-FUNCTION8_4_final_CKL.xlsx	[0122]
78	90	TV_SUB-FUNCTION9.xls	[0023]
79	91	TV_SUB-FUNCTION9_CKL.xlsx	[0056]
80	92	TV_TG_SUB-FUNCTION9_final.pdf	[0084]
81	93	TV_TG_SUB-FUNCTION9_final_CKL.xlsx	[0124]
82	94	TV_SUB-FUNCTION10.xls	[0024]
83	95	TV_SUB-FUNCTION10_CKL.xlsx	[0057]
84	96	TV_TG_SUB-FUNCTION10_final.pdf	[0085]
85	97	TV_TG_SUB-FUNCTION10_final_CKL.xlsx	[0126]
86	98	TV_SUB-FUNCTION11.xls	[0025]
87	99	TV_SUB-FUNCTION11_CKL.xlsx	[0058]
88	100	TV_TG_SUB-FUNCTION11_final.pdf	[0086]
89	101	TV_TG_SUB-FUNCTION11_final_CKL.xlsx	[0127]
90	102	TV_SUB-FUNCTION12.xls	[0026]
91	103	TV_SUB-FUNCTION12_CKL.xlsx	[0059]
92	104	TV_TG_SUB-FUNCTION12_final.pdf	[0087]
93	105	TV_TG_SUB-FUNCTION12_final_CKL.xlsx	[0128]
94	106	TV_SUB-FUNCTION13.xls	[0027]
95	107	TV_SUB-FUNCTION13_CKL.xlsx	[0060]
96	108	TV_TG_SUB-FUNCTION13_final.pdf	[0088]
97	109	TV_TG_SUB-FUNCTION13_final_CKL.xlsx	[0129]
98	110	TV_SUB-FUNCTION14.xls	[0028]
99	111	TV_SUB-FUNCTION14_CKL.xlsx	[0061]
100	112	TV_TG_SUB-FUNCTION14_final.pdf	[0089]
101	113	TV_TG_SUB-FUNCTION14_final_CKL.xlsx	[0130]
102	114	TV_SUB-FUNCTION15_2.xls	[0029]
103	115	TV_SUB-FUNCTION15_2_CKL.xlsx	[0062]
104	116	TV_TG_SUB-FUNCTION15_2_final.pdf	[0090]
105	117	TV_TG_SUB-FUNCTION15_2_final_CKL.xlsx	[0131]
106	118	TV_SUB-FUNCTION16.xls	[0030]

Nº	Comment ID	Artifact	Revision
107	119	TV_SUB-FUNCTION16_ACID2_CKL.xlsx	[0063]
108	120	TV_SUB-FUNCTION16_CKL.xlsx	[0063]
109	121	TV_TG_SUB-FUNCTION16_final.pdf	[0091]
110	122	TV_TG_SUB-FUNCTION16_final_CKL.xlsx	[0132]
111	124	TV_SUB-FUNCTION17.xls	[0031]
112	125	TV_SUB-FUNCTION17_ACID2_CKL.xlsx	[0064]
113	126	TV_SUB-FUNCTION17_CKL.xlsx	[0064]
114	127	TV_TG_SUB-FUNCTION17_final.pdf	[0092]
115	128	TV_TG_SUB-FUNCTION17_final_CKL.xlsx	[0133]
116	129	TV_SUB-FUNCTION18.xls	[0032]
117	130	TV_SUB-FUNCTION18_CKL.xlsx	[0065]
118	131	TV_TG_SUB-FUNCTION18_final.pdf	[0093]
119	132	TV_TG_SUB-FUNCTION18_final_CKL.xlsx	[0134]
120	133	TV_SUB-FUNCTION19_Q.xls	[0033]
121	134	TV_SUB-FUNCTION19_CKL.xlsx	[0066]
122	135	TV_TG_SUB-FUNCTION19_Q _final.pdf	[0094]
123	136	TV_TG_SUB-FUNCTION19_Q _final_CKL.xlsx	[0135]
124	137	TV_SUB-FUNCTION20.xls	[0034]
125	138	TV_SUB-FUNCTION20_CKL.xlsx	[0067]
126	139	TV_TG_SUB-FUNCTION20_final.pdf	[0095]
127	140	TV_TG_SUB-FUNCTION20_final_CKL.xlsx	[0136]
128	141	TV_SUB-FUNCTION21_1.xls	[0035]
129	142	TV_SUB-FUNCTION21_1_CKL.xlsx	[0068]
130	143	TV_TG_SUB-FUNCTION21_1_final.pdf	[0096]
131	144	TV_TG_SUB-FUNCTION21_1_final_CKL.xlsx	[0137]
132	145	TV_SUB-FUNCTION22.xls	[0036]
133	146	TV_SUB-FUNCTION22_CKL.xlsx	[0069]
134	147	TV_TG_SUB-FUNCTION22_final.pdf	[0097]
135	148	TV_TG_SUB-FUNCTION22_final_CKL.xlsx	[0138]
136	149	TV_SUB-FUNCTION23.xls	[0037]
137	150	TV_SUB-FUNCTION23_CKL.xlsx	[0070]
138	151	TV_TG_SUB-FUNCTION23_final.pdf	[0098]
139	152	TV_TG_SUB-FUNCTION23_final_CKL.xlsx	[0139]
140	153	TV_SUB-FUNCTION24.xls	[0038]
141	154	TV_SUB-FUNCTION24_CKL.xlsx	[0071]
142	155	TV_TG_SUB-FUNCTION24_final.pdf	[0099]
143	156	TV_TG_SUB-FUNCTION24_final_CKL.xlsx	[0140]

Nº	Comment ID	Artifact	Revision
144	157	TV_SUB-FUNCTION25_3.xls	[0039]
145	158	TV_SUB-FUNCTION25_3_CKL.xlsx	[0072]
146	159	TV_TG_SUB-FUNCTION25_3_final.pdf	[0100]
147	160	TV_TG_SUB-FUNCTION25_3_final_CKL.xlsx	[0141]
148	161	TV_SUB-FUNCTION26.xls	[0040]
149	162	TV_SUB-FUNCTION26_CKL.xlsx	[0073]
150	163	TV_TG_SUB-FUNCTION26_final.pdf	[0101]
151	164	TV_TG_SUB-FUNCTION26_final_CKL.xlsx	[0142]
152	165	TV_SUB-FUNCTION27_BOX2.xls	[0041]
153	166	TV_SUB-FUNCTION27_BOX2_CKL.xlsx	[0074]
154	167	TV_TG_SUB-FUNCTION27_BOX2_final.pdf	[0102]
155	168	TV_TG_SUB-FUNCTION27_BOX2_final_CKL.xlsx	[0143]
156	169	TV_SUB-FUNCTION28_2.xls	[0042]
157	170	TV_SUB-FUNCTION28_2_CKL.xlsx	[0075]
158	171	TV_TG_SUB-FUNCTION28_2_final.pdf	[0102]
159	172	TV_TG_SUB-FUNCTION28_2_final_CKL.xlsx	[0144]
160	173	TV_SUB-FUNCTION29_L270.xls	[0043]
161	174	TV_SUB-FUNCTION29_L270_CKL.xlsx	[0076]
162	175	TV_TG_SUB-FUNCTION29_L270_final.pdf	[0103]
163	176	TV_TG_SUB-FUNCTION29_L270_final_CKL.xlsx	[0145]
164	177	TV_SUB-FUNCTION30_3To5.xls	[0044]
165	178	TV_SUB-FUNCTION30_3To5_CKL.xlsx	[0077]
166	179	TV_TG_SUB-FUNCTION30_3To5_final.pdf	[0104]
167	180	TV_TG_SUB-FUNCTION30_3To5_final_CKL.xlsx	[0146]
168	181	PREFIXTestResults_MODELHL_LOAD_X_Y_Z_4_001	[0105]
169	182	PREFIXCAMP_CODE1LL_ LOAD _ X_Y_Z_3_001 (PREFIXCAMP_CODE1HL)	[0045]
170	183	PREFIXCKL_CAMP_CODE1HL	[0147]
171	184	20200717-142209.7z	[0106]
172	185	Log_ACID2_white	[0107]
173	186	Log_ACID2_black	[0108]
174	187	TEST-1245-TCs-01-62__ACID2	[0109]
175	188	TV_TG_ SUB-FUNCTION31_final.pdf	[0110]
176	189	TV_TG_ SUB-FUNCTION31_final_CKL.xlsx	[0148]
177	190	TV_TG_SUB-FUNCTION32_final.pdf	[0111]
178	191	TV_TG_SUB-FUNCTION32_final_CKL.xlsx	[0149]
179	192	TV_TG_SUB-FUNCTION2_3_final.pdf	[0112]

Nº	Comment ID	Artifact	Revision
180	193	TV_TG_SUB-FUNCTION2_3_final_CKL.xlsx	[0150]
181	194	TV_TG_SUB-FUNCTION33_final.pdf	[0113]
182	195	TV_TG_SUB-FUNCTION33_final_CKL.xlsx	[0151]
183	196	TV_TG_SUB-FUNCTION34_final.pdf	[0114]
184	197	TV_TG_SUB-FUNCTION34_final_CKL.xlsx	[0152]

Appendix F

INDEPENDENT AUDITOR INTERVIEW

What is your experience in a software audit?

9.5 years of experience in safety-critical software auditing to directly assess software development (Table A-9) and its flow-down from the system, including an audit of the part of the system used as part of the software (ARP 4754 A).

7 years of experience in software audits developed by suppliers, aiming to evaluate compliance with the applicable standards. They are focusing not only on RTCA DO-178C but also on SAE ARP-4754 and RTCA DO-254.

How did you define the audit scope?

The audit scope was defined by evaluating the software life cycle data delivered and, based on that, the activities performed and the objectives of the standard that should be me and what was foreseen in the plans and standards defined for the project.

How did you select the applicable questions?

First, I checked each baseline's purpose and the expected content of each of them, as defined in the plans and related software process. Based on this, I pre-selected the phases in the checklist, and then, as it was an audit based on modifications, I evaluated each of the questions individually to see if they were associated with the content of the baseline that needed to be evaluated and also if

the objective applied to the project. The questions associated with CM did not pass this assessment, as all were applicable.

How did you conduct the audit?

Once the samples and questions were selected, I went through the artifacts and made the necessary evaluations. As there were many artifacts, I used the strategy of evaluating all the artifacts by making notes and observations, correlating the artifact with the objectives. Then I went back to the questions to answer each one based on the notes and observations. In the end, I checked which questions had not yet been answered and searched for the evidence for them or identified their respective objectives were not being met.

I did it because I found it easier to look at an artifact and then answer than to go from each question and look for artifacts.

Given the characteristics of the audited project that complies with RTCA DO-178C and SAE ARP-4754 simultaneously, I identified the need to create additional questions related to SAE ARP-4754, to ensure that the scope of the project was covered entirely.

How did you answer the questions?

In addition to what was answered on how to conduct the audit, another strategy that I used was to do a bidirectional trace between the artifacts and the questions. So that I was able to identify what were the objectives covered in the audit and which was the complete life-cycle data audited. I did not feel the need for a partial answer, but the N/A answer can be interesting, as the way it was audited, the non-applicable questions were not considered, but perhaps they could be considered to be justified as to their non-applicability, and so it would be possible to guarantee the completion of the audit at the end of the project.

Was the intent of the questions clearly defined?

I understand that, yes, the questions did not have guidance to answer, but as their objective was to guide what could be checked by the auditor, I understand that it

was adequately giving the auditor the flexibility to seek the necessary evidence as per the characteristics of the audited project. If additional guidance were given for each of the questions, this would result in the auditor not being as flexible and could still be limited to a specific software project.

Were the questions easy to answers and/or provide evidence?

Recording the evidence was very easy because compared to the audit model that we have formally in the company, it was just answering a Yes or No, without writing a whole associated text to indicate what was seen and how the questions already did that. As for the difficulty of answering the questions, perhaps in the part of configuration management, the questions were a little bit more complicated, as they are described in a more general way and encompass several aspects that should be taken into account to be answered as Yes. Perhaps it could be broken down into more questions for the software project being audited. By thinking about supplier auditing, the questions may be appropriate, but perhaps it is worth expanding the questions associated with the problem report, taking into account the standards and the certification authorities' guidance.

Were the questions helpful to your audit?

Yes! Even thinking about the team's point of view, we have a minimum rule of what should be looked at in the audit. We know that QAs have different profiles and look at various aspects. This helped ensure the minimum necessary audit quality and made it easier for the team to evidence everything that was audited and ensure that the applicable objectives were covered as expected, even facilitating the team to carry out the software project's conformity review.

It also helped to remember aspects that should be considered during the audit to guarantee the objectives. See what was answered about revisiting the samples in the question about answer the questions.

What is your feedback for this software audit model proposed?

I thought that the way to respond was very good. Its main differential is to visualize in a straightforward way what objectives were answered, identifying which aspects were answered. It removed the need to read much text to understand what was evaluated. It allowed us to establish a standard of record the audits, making it even easier to implement automatisms in the software projects we work on, to make more meaningful analyzes: coverage of the standard objectives, objectives that present more issues/difficulties, which were the most common issues of a software project, which questions give more problems, improve corporate processes, recurrence of issues. The process becomes more independent of people, guaranteeing the company's knowledge base through an established process, guaranteeing the perpetuation within the company.

The checklist helps even novice Quality Assurance Engineers to have adequate guidance of what should be seen and evaluated

Negative feedback: traceability has become very dull and dense in how we implemented it in the company.

Appendix G

PAPERS SELECTED FOR THE SLR

The SLR references are available in Table G-1.

Table G- 1 – Papers selected for the SLR

ID	Citations	Authors	Title
[1]	6	A. B. Bujoka, S. T. MacMahona, P. Granta, D. Whelanb, W. J. Rickardc, and F. McCafferya	Approach to the development of a Unified Framework for Safety Critical Software Development
[2]	36	I. Dodd and I. Habli	Safety certification of airborne software: An empirical study
[3]	16	W. Youn and B. Yi	Software and hardware certification of safety-critical avionic systems: A comparison study
[4]	1	M. García-Valls, J. Escribano-Barreno and J. García-Muñoz	An extensible collaborative framework for monitoring software quality in critical systems
[5]	49	L. E. G. Martins and T. Gorschek	Requirements engineering for safety-critical systems: A systematic literature review
[6]	72	R. Hawkins, I. Habli, T. Kelly and J. McDermid	Assurance cases and prescriptive software safety certification: A comparative study
[7]	20	P. Ayrault, T. Hardin and F. Pessaux	Development Life-cycle of Critical Software Under FoCaL
[8]	0	A. Kornecki and J. Zalewski	Hardware Certification for Safety-Critical Real-Time Systems
[9]	18	T. Tasić and U. Grottker	An overview of guidance documents for software in metrological applications
[10]	6	J. A. Jiménez, J. A. M. Merodio and L. F. Sanz	Checklists for compliance to DO-178C and DO-278A standards
[11]	4	B. R. Poreddy and S. Corns	Arguing Security of Generic Avionic Mission Control Computer System (MCC) using Assurance Cases
[12]	11	S. Linling, Z. Wenjin and T. Kelly	Do safety cases have a role in aircraft certification?

ID	Citations	Authors	Title
[13]	10	S. A. Vilkomir, J. P. Bowen and A. K. Ghose	Formalization and assessment of regulatory requirements for safety-critical software
[14]	63	A. Kornecki and J. Zalewski	Certification of software for real-time safety-critical systems: state of the art
[15]	5	G. K. Hanssen, G. Wedzinga and M. Stuip	An Assessment of Avionics Software Development Practice: Justifications for an Agile Development Process
[16]	5	A. Ruiz, X. Larrucea and H. Espinoza	A Tool Suite for Assurance Cases and Evidences: Avionics Experiences
[17]	8	N. Silva and M. Vieira	Towards Making Safety-Critical Systems Safer: Learning from Mistakes
[18]	11	L. Davila-Nicanor and P. Mejia-Alvarez	Reliability improvement of Web-based software applications
[19]	0	R. Fulton	Assuring certifiability of outsourced software development - a DER'S perspective
[20]	12	K.A. Eastaughffe, A. Cant, and M.A. Ozols	A framework for assessing standards for safety critical computer-based systems
[21]	20	A. Kornecki and J. Zalewski	Software certification for safety-critical systems: A status report
[22]	2	Y. Liu, K. Foster, T. Nguyen, and J. W. Keung	Quality Assessment of Mission Critical Middleware System Using MEMS
[23]	8	M. Bernhart, S. Reiterer, K. Matt, A. Mauczka and T. Grechenig	A Task-Based Code Review Process and Tool to Comply with the DO-278/ED-109 Standard for Air Traffic Management Software Development: An Industrial Case Study
[24]	0	A. Schwierz and H. Forsberg	Assurance Case to Structure COTS Hardware Component Assurance for Safety-Critical Avionics
[25]	10	C. Areias, J. C. Cunha, D. Iacono, and F. Rossi	Towards Certification of Automotive Software
[26]	2	A. Schwierz and H. Forsberg	Design assurance evaluation of microcontrollers for safety critical avionics
[27]	8	P. Steele and J. Knight	Analysis of Critical Systems Certification
[28]	2	J. Marques and A. M. Cunha	Tailoring Traditional Software Life Cycles to Ensure Compliance of RTCA DO-178C and DO-331 with Model-Driven Design
[29]	25	I. Habli and T. Kelly	A Model-Driven Approach to Assuring Process Reliability
[30]	4	F. Yan	Comparison of means of compliance for onboard software certification
[31]	81	J. Rushby	New challenges in certification for aircraft software
[32]	0	J .S. Sagoo	An approach for the assurance of legacy systems based on programmable electronic hardware
[33]	0	M. Graydon	Retrospectively Documenting SAFEGUARD's Possession of the Overarching Properties

ID	Citations	Authors	Title
[34]	38	J. Rushby	Just-in-Time Certification
[35]	5	M. Lange and T. Dewey	Achieving Quality and Traceability in FPGA/ASIC Flows for DO-254 Aviation Projects
[36]	14	J. Lewis and L. Rierson	Certification concerns with integrated modular avionics (IMA) projects
[37]	51	F. McCaffery, M. Pikkarainen, and I. Richardson	Ahaa --agile, hybrid assessment method for automotive, safety critical smes
[38]	6	J. Chen, M. Goodrum, R. Metoyer, and J. Cleland-Huang	How do practitioners perceive assurance cases in safety-critical software systems?
[39]	69	J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones	Certifiably safe software-dependent systems: challenges and directions