Requirements for a Software Audit Model in Safety-Critical Domains

Talita Marques Ruiz Slavov[1][0000−0003−3103−4919], Johnny Cardoso Marques[2][0000−0002−1551−435X], and Luiz Eduardo Galvão Martins[1][0000−0002−7266−5840]

[1] University Federal of São Paulo, São José dos Campos SP, Brazil
talitaruiz@gmail.com, legmartins@unifesp.br
https://www.unifesp.br/campus/sjc/ppgit
[2] Aeronautics Institute of Technology, São José dos Campos SP, Brazil
johnny@ita.br

Abstract. This paper presents the requirements for a more flexible and adaptable Software Audit Model and its associated adaptable checklist to conduct a software audit. This flexibility allows applying the SAM independently of the software development life cycle chosen. The initial proposed SAM will enable us to evaluate, integrate, and adapt the Stages of Involvement in different software development cycles based on the events necessary for aircraft certification with safety-critical software.

Keywords: Requirements elicitation · Software Audit Model · Safety-Critical Software

1 Introduction

The technological advances and the constant concern to guarantee the safety of people obliges the aeronautical industry to use complex and critical software for different embedded systems increasingly. As part of the process of evaluating and approving embedded software, Order 8110.49 Chg 1[2] has guidance for assessing the software in stages (planning, development, verification, and final). Based on the characteristics of each development and type of life cycle, the audits performed using standards like Order 8110.49 Chg 1[2] and DO-178C[1] need to be customized. Regardless of the software development life cycle used and the percentage of artifacts achieved in each phase, minimizing the costs with reviews and optimizing the anticipation of any issues, we intend to propose a Software Audit Model (SAM) flexible enough to be tailored to several audit types,

Westfall[3] states that a requirement is a software s capability that provides value to or is needed by a stakeholder. The requirement defines the "what" of a software product. Also, Pressman[4] describes a software requirement specification (SRS) as a document created when a detailed description of all aspects of the software must be built must be specified before the project is to commence. Based on Westfall[3] and Pressman[4] assertions, we are going to present the requirements necessary to develop our SAM.

2 Background and Related Work

2.1 Definitions

Audit: An independent examination of the software life cycle processes and their outputs to confirm required attributes[1].

Certification process: A process that establishes communication, understanding, and agreements between the applicant and certification authority[1].

Embedded Software Development: It is typically split into five processes: Requirements, Design, Implementation, Verification, and Validation[5].

Software Audit Model (SAM): It is a methodology that allows audit any software development based on the software scope, allowing the use of a customized checklist.

2.2 Related work

Some recent works discuss the details of the audit processes for safety-critical software, like Dodd & Habli[7], which proposed a statistical method for supporting software certification audits. Steele & Knight[8] proposed the Filter Model method, in which a certification process is characterized as a safety-critical system where a system that should be rejected may be incorrectly certified, and this is considered an accident. Ruiz et al. [9], and Schwierz & Forsberg[10] conducted their assurance cases proposing an audit method that uses the Goal Structuring Notation (GSN) method. Habli & Kelly[11] associated the GSN method with Goal Question Metrics (GQM) method. Silva & Vieira[12] defined a framework and a process to map the system assessment issues (empirical data) to their root-causes, and to act upon those root-causes. In his article, Fulton [14] indicates the application of Job Aid[13][6].

3 Software Audit Model (SAM)

Based on Fig. 1, this article intends to present the main requirements for a safety-critical SAM, which will generate a flexible model to be adapted to any software development cycle (V-Cycle, Waterfall, Incremental, Spiral) by selecting questions from its database. It also intends to allow the applicant to utilize the audit process with any standard once the questions are related to a software development process instead of a specific standard.

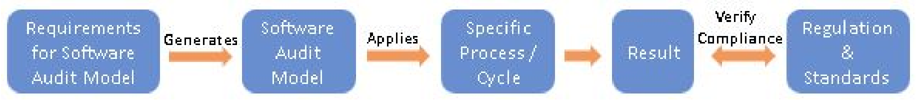

Fig. 1. SAM overview

3.1 Requirements for Software Audit Model (SAM)

As stated by Westfall[3], a requirement is used to determine the functionality that must be built into the software. In this paper, we aim to identify the main requirements of an audit process to develop a new process that can be adaptable. For the proposed SAM, the requirements elicited for the audit process are shown in Table 1 and Section 7. Also, the requirements were created based on the authors' several years of experience in software assessment, development, and aeronautical standards, as well as on papers selected for a systematic literature review[6] performed.

3.2 Development of the Software Audit Model (SAM)

The proposed SAM is focused on safety-critical software and includes questions to conduct the audits to verify compliance with applicable standards. These questions are selected based on the development stage and specific characteristics of the audited process to confirm whether the development was carried out following the processes. The suggested SAM aims to guarantee compliance with the DO-178C. However, the proposed questions were not created to verify individual compliance with each objective, but rather the phase as a whole. Additionally, besides the questions itself, a classification regarding the stage to which they belong (Planning, Development, Verification, or Closure), and the answer options (Yes, No, Not Applicable) are necessary to allow the customization.

Once the SAM is developed and detailed, using a question database, it will be possible to use it by customizing a checklist to carry out and conduct an audit. A specific audit can be performed at any phase of software development; thus, the software development cycle that best fits this approach would be the Waterfall as it foresees the end of one development phase to the beginning of another. When discussing a development cycle, it is possible to put together more than one development phase (requirements, design, and coding). In an audit or when discussing Spiral or Incremental development cycles, it is possible to evaluate the deployment of a function from its requirement until its test.

Table 1. Requirements for the SAM

ID	Description	Classification
REQ-1.1	The SAM shall evaluate any software processes, independently of the software life-cycle used.	High
REQ-1.3	The audit model shall evaluate if the Requirements Process identify the software requirements from the system requirements.	High

After generating a customized checklist and executing the audit, it is possible to analyze recorded responses and related evidence, understand whether the process was followed or if there were issues regarding compliance. It makes it possible to check which objectives of the applicable standard were achieved by the project development. In our research, this compliance will be verified against the DO-178C. By selecting as a sample the Software Requirements Process defined for our SAM (see Section 7), it is possible to present as a sample the following questions created to evaluate the software compliance, as indicated on Table 2. It is essential to notice that this is a sample of our question database that has 91 questions split into the SAM (See Table 1) defined phases.

The intent of this proposed SAM is not only to verify that compliance with the process established. But to understand whether it was possible to obtain compliance with regulations and standards, in this way, the certification authorities, for example, can be sure that the software is safe enough to be used as expected. Furthermore, by proposing a SAM that is flexible, adaptable and has questions focused on the software development process, regardless of regulations or standards, it allows not only the DO-178C to be applied but also other safety-critical software standards.

Table 2. Software Audit Model - Sample of proposed questions

ID	Question	Options
R.02	Are the system requirements correctly refined by the software high-level requirements?	Y / N / NA
R.04	Are the software high-level requirements defining the software behavior and external interfaces?	Y / N

4 Discussion

Before proposing a SAM that is flexible and adaptable to many software development cycles, we performed a SLR to understand what has been discussed in the literature regarding assessing a safety-critical software and which methods they are proposing. Although some identified purposes are used for the safety-critical software assessment, still, the papers consulted are not explicit on whether there is an established methodology or if only the existent methods have been adapted for their purpose.

Considering the academic literature consulted, the concept of a SAM that is flexible enough to be adapted to the aeronautical world and allow conduct supplier audits more effectively was recognized. In this way, planning the software audit's scope will no longer be dependent on reaching a specific milestone. But instead, a minimum amount of software scope already developed, allowing the auditor to customize the checklist that will guide his/her audit based on this information. Thus, the main requirements of this SAM were elicited to identify which functionalities would be necessary for the proposed SAM to be flexible and adaptable to any software development.

As presented in Table 1, the requirement of REQ-1.1 is the main requirement since it defines the foremost functionality that allows the evaluation of any software process independent of the software life-cycle used. The requirements REQ-1.2 up to REQ-1.21 defines the phases of a software development cycle that shall be assessed, including their main concerns. These requirements are essential because they will drive the initial set of questions creation. The requirements REQ-1.22 up to REQ-1.26 are related to SAM usability, like determining the necessity of flow-down the requirements into a question database, allowing the auditor to customize the questions for the checklist and the acceptable answers for each defined question and the necessity to provide the necessary evidence to answer a question.

A sample of questions for the checklist's database of the proposed SAM is presented; the proposed database initially contains 91 items distributed among the software development phases. For this article, the selected sample is related precisely to the requirements phase, once it is one of the essential aspects of the software development process. Also, it allows us to demonstrate that the proposed SAM not only assesses whether compliance with the software development process or if there is a requirement created. It is also concerned with elicited specific characteristics of the software, if there is the necessary information to flow-down the requirement for the next phase, and induces the auditor to evaluate several aspects of the software, still in its definition phase.

5 Conclusion

The objective of this paper is to define the main requirements (See Table 1) to create our SAM. Considering the value of eliciting requirements before any product development, we used the same thinking to specify our requirements. We decided to use this technique to identify the main requirements, ensuring a robust development with a higher chance of being accurate and avoiding rework. We have increased the ability to understand the SAM as a whole and identify which points should be improved before beginning its development, allowing us to perform the necessary adjustments and adaptations by eliciting requirements to create the SAM. Our proposed SAM is more flexible than those currently presented in the academic literature. It is adaptable to more than one type of standard, regardless of the software development life cycle used.

Additionally, eliciting requirements must not only be used to understand the functionality of the software or a system. Still, it can also be employed to the elaboration of different processes, such as, for example, the SAM for safety-critical software proposed in this paper. Such a technique helped us identify the "functionalities" needed to obtain the differential of this SAM. The next step of our research involves continuing to validate the proposed questions with software specialists who have considerable experience in aeronautical software assessment and development. At the end of this validation, a software audit will be performed in a safety-critical software developed using a V-cycle Process. Finally, we also plan to analyze the obtained result by comparing with Waterfall and Incremental process as a way to validate the proposed SAM in different software life cycles.

6 Acknowledgements

This work was partially funded by São Paulo Research Foundation (FAPESP) under the grant no. 2019/09396-0.

7 Appendix - SAM Database

The SAM requirements list and questions database are available at `https://drive.google.com/open?id=1xlDelWknyATDWaZzKo1mxckyhVLiMIv6`.

References

1. SC-205: DO-178C - Software Considerations in Airborne Systems and Equipment Certification. 3rd edn. RTCA - Radio Technical Commission for Aeronautics, (2011)
2. Federal Aviation Administration: Software Approval Guidelines, (2011)
3. Westfall, L.: The certified software quality engineer handbook. 1st edn. American Society for Quality, Milwaukee (2010)
4. Pressman, R. S.: Software engineering : a practitioner's approach. 7th edn. McGraw-Hill, New York (2010)
5. Marques, J. C., Cunha, A. M.: Tailoring Traditional Software Life Cycles to Ensure Compliance of RTCA DO-178C and DO-331 with Model-Driven Design. 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), 1–8 (2018)
6. Slavov, T. M. R., Marques, J. C., Martins, L. E. G.: Safety-Critical Software Audit: A Systematic Literature Review. Journal **Article Not Published**
7. Dodd, I., Habli, I.: Safety certification of airborne software: An empirical study. **Reliability Engineering & System Safety**, 7–23 (2012)
8. Steele, P., Knight, J.: Analysis of Critical Systems Certification. Journal **2014 IEEE 15th International Symposium on High-Assurance Systems Engineering**, 129–136 (2014)
9. Ruiz, A., Larrucea, X., Espinoza, H.: A Tool Suite for Assurance Cases and Evidences: Avionics Experiences. EuroSPI 2015: Systems, Software and Services Process Improvement edn. Springer, Cham, (2015)
10. Schwierz, A., Forsberg, H.: Assurance Case to Structure COTS Hardware Component Assurance for Safety-Critical Avionics. Journal **2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)**, 1–10 (2018)
11. Habli, I., Kelly, T.: A Model-Driven Approach to Assuring Process Reliability. Journal **2008 19th International Symposium on Software Reliability Engineering (ISSRE)**, 7–16 (2008)
12. Silva, N., Vieira, M.: Towards Making Safety-Critical Systems Safer: Learning from Mistakes. Journal **2014 IEEE International Symposium on Software Reliability Engineering Workshops**(5), 162–167 (2014)
13. Federal Aviation Administration: Conducting Software Review Prior to Certification, (2004)
14. Fulton, R.: Assuring certifiability of outsourced software development - a DER'S perspective. 24th Digital Avionics Systems Conference **24th Digital Avionics Systems Conference**, 5pp (2005)