A Requirements Specification Method: An Experience Report in Aerospace

Carlos Renato dos Santos and Johnny Cardoso Marques

Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil
{carlosrenato,johnny}@ita.br

Abstract. Aerospace projects need well-written requirement documents. However, it is challenging to transform an abstract thought that has insufficiently detailed specifications into a requirement, especially without a defined process and a supporting tool. **Objective:** Apply a methodology in a real aerospace project to overcome the expensive requirements of engineering tools; This study relates a practical experience to creating a methodology to generate and track the requirements using Excel. However, due to the confidential information, some characteristics of the project must be omitted. **Method:** The method consists of taking the standard ISO/IEC/IEEE 29148 as a reference and using Excel formulas to semi-automate the requirements writing. **Results:** From unstructured requirements written document, where the linguistic construction used violates the ISO/IEC/IEEE 29148. A requirements document was recreated to track the aerospace project needs and to respect the characteristics of a well-written requirement. **Conclusions:** Our practical approach contributes to the implementation of requirements mentality without using expensive engineering tools and reducing technical requirements debt.

Keywords: Requirements · Technical debt · Requirements Engineering · Standards · Requirements Engineering Tools.

1 Introduction

Requirements engineering (RE) is a crucial process for the production of quality systems, even more in the context of complex and critical systems [1]. Every project starts with one idea that contains a glimpse of what the intended project should do. However, this idea usually changes throughout the lifecycle. In an inadequate scenario, the initial idea and these changes are not transparent (documented) throughout the lifecycle [6][7]. Consequently, the final product can present features that are different from those desired by the client.

This practical aerospace approach happens in a small development team where communication tends to be passed informally. Informal communication during development may be convenient in the short term. However, in the long term, these changes can be costly and challenging to deal with [6][4], particularly in the test stage, where all the tests are designed to confirm requirements

Proceedings of the 26th Workshop on Requirements Engineering (WER23), August 15-17, 2023, Porto Alegre, RS, Brazil.
DOI: 10.29327/1298356.26-20

implementation. Consequently, it increases the chance of generating technical debt, which is the distance between the optimal specification and the real system implementation, under domain assumptions and constraints.

The aerospace project demands a strong requirements document to minimize these sudden decisions. However, the discipline of writing requirements cannot be resumed just by "writing" them. Thus, the proposed method must address distinct development perspectives. Consequently, the technical requirement debt is reduced. The literature has addressed the various problems in the Requirements Specification, which may involve incomplete, incorrect, ambiguous, conflicting, or inconsistent requirements [5].

ISO/IEC/IEEE 29148 [3] is a standard that addresses requirements engineering and defines requirements engineering as concerned with discovering, eliciting, developing, analyzing, verifying, validating, communicating, documenting, and managing requirements. These activities, described by standard 29148, aim to deliver a requirements document that envelops the project.

Surveys revealed that one-third of the projects were never completed, and half succeeded only partially. The reason for this failure is the lack of or poor requirements document [6][9]. Each written requirement has certain characteristics that should be considered. The requirements should be singular, feasible, complete, consistent, unambiguous, verifiable, and traceable. Although achieving all of these characteristics is unrealistic, the objective is to fulfill most of them. Requirements are difficult to be considered complete. However, requirements can be sufficient [7].

Most requirement engineering problems occur because of the lack of a well-defined process, lack of client presence during the process, and poor requirements capture [6]. Lenarduzzi and Fucci [4] defined three types of technical requirement debt:

- **Incomplete users' needs** that is a technical debt due to ignoring the client or interested parts need; Can be caused by the laziest of requirements engineering to keep contact with the client, or the client intentionally does not track the project.
- **Requirement smells** that is when the linguistics construction used violates, for example, the ISO/IEC/IEEE 29148 [3] standard.
- **Mismatch implementation** is when is founded an incompatibility between the client's goal and design implementation. This type of debt can also be incurred when the requirements problem, framed in the requirements document, changes while the implementation does not change accordingly (Traceability).

This work intends to present how these three technical requirement debts were addressed in a practical aerospace project by applying the proposed method in a previous written requirements document. Create a documentation template that integrates communication and evolves with the development flow [10]. Thus, creating design "memory" and allowing future revisions of the decisions taken over time [8]. Consequently, sharing an implied understanding among involved

parties and reducing the need for explicit communication. Furthermore, these actions aim to lower the risk of misunderstanding.

The rest of this work is organized as follows: Section 2 gives the overview of the proposed method. Section 3 describes a real application of the proposed method. Section 4 expresses the conclusion and the next steps of the research.

2 Method Definition

Our proposed method is adequate when the input of the development flow lacks information (superficial description) or when the team is not familiar with requirements engineering, and both scenarios are common in the aerospace environment where the method was applied. Generally, every time a late client has access to the project, new ideas emerge. Depending on the meeting formality, this information can only be verbalized without further documentation [10].

Inside this situation, the problem of minimizing the technical debt gap has emerged. An action to track all these modifications must be adopted. However, the agility of informal processes cannot be sacrificed. As mentioned earlier, one way to reduce debt is to have an assertive requirements document. Thus, during development, the requirements are updated to meet the aerospace project mission. Figure 1 shows the proposed flow for mitigating incomplete user needs, mismatch implementation, and requirement smells.

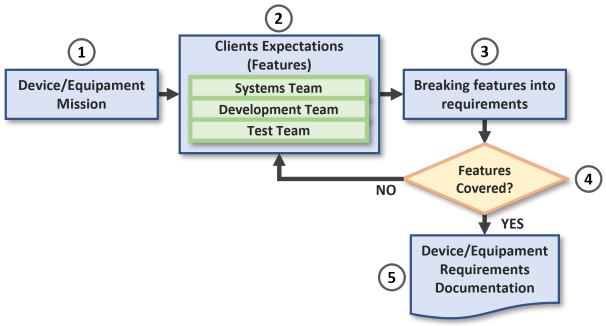

Fig. 1. Requirements update during the development flow.

First, each stage presented in Figure 1 is described. Stage 1 is when the idea of an aerospace device or equipment is born. This is the major abstraction level. The most important client states it, normally the system team, and can be expressed, for example, as "As a major client, I want a satellite control unit (SCU) that provides energy, regulates the temperature, and be controllable from the earth".

In stage 2, a meeting with all teams involved throughout the development will raise the features desired in the SCU. It is important to consider what I want,

what are my tools to develop, and what are my test capabilities. These three actions are represented by systems, development, and test teams. In essence, all chains of development shall be present during feature elicitation, thus avoiding future mishaps.

In stage 3, the features described by the three teams are converted into requirements, which are tied to the features; thus, there is the possibility of tracking the requirement back to the features that generate it. Consequently, requirements pass through a requirement analysis that aims to mitigate the requirement smells debt. Stage 3 is expanded and is detailed in Figure 2.

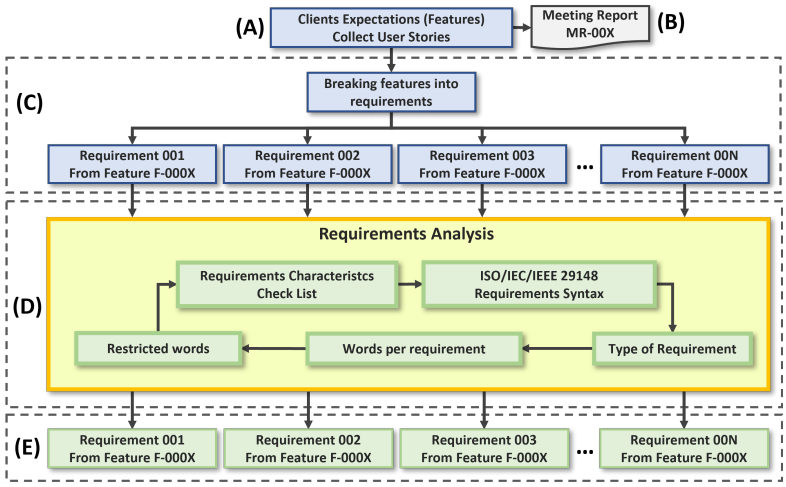

Fig. 2. Detailed requirements smells mitigation.

Figure 2 illustrates the core of this method. In (A), clients express their expectations, and the requirements engineer collects them as notes. The output is the generation of a meeting report (B). The meeting report generation is crucial to track back from the requirements where the initial idea emerged and who was responsible for that idea.

In (C) the features are transformed into requirements. However, to mitigate the technical debt of requirement smells, there is a requirement analysis in (D), where it is checked if the written requirement syntax complies with standard ISO/IEC/IEEE 29148, the number of words of each requirement is in accordance with the specified, no restricted words are used to describe the requirement [2] such "and/or", "easy", "necessary", "rapid", "user-friendly", etc. Furthermore, if the written requirement has the desired characteristics [2] [3] such as singularity, completeness, identifiable, etc. Finally, we identify the type of requirement as functional, nonfunctional, interface, configuration, etc.

The forbidden words and the number of words per requirement are automatized inside Excel; Conditional formatting is used to indicate when the written

requirement does not contain the word "shall", thus, whenever a conditional parameter is violated, the cell changes its color (Figure 3). To indicate the restricted words a VBA script is used, highlighting in red the restricted words in the sentence. The usage of "and" in the first requirement raises a flag, where according to ECSS-E-ST-10-06C [2] "and" and "or" are restricted words (Figure 3).

Requeriment Description	# Words
The satellite Control Unit (SCU) need to control the temperature in several points of the satellite and the SCU need heat or cool internal modules.	25
The Satellite Control Unit shall shall monitor temperatures	8
When temperature > 100°C in Module X the SCU shall turn off Module X power.	15

Fig. 3. Requirements Excel data validation.

The requirement characteristics (Figure 4) and standard 29148 requirement syntax (Figure 5) are manually checklist-based, where the requirement engineer must verify if the written requirement satisfies the parameters. The output of requirement analysis (D) is a requirement formatted according to the desired characteristics (E).

		Requirements Characteristics Checklist						
Requeriment Description		Necessary	Appropriate	Complete	Singular	Feasible	Verifiable	Unambiguous
The Satellite Control Unit shall ...	...	X	X	X	X	X	X	X
When temperature > 100°C ...		X	X	X	X	X	X	X

Fig. 4. Requirements characteristics checklist.

		Requirements Synthax Checklist				
Requeriment Description		Condition	Subject	Action	Object	Constraint
The Satellite Control Unit shall ...	...		The SCU	shall monitor	temperature	
When temperature > 100°C ...		temperature > 100°C ...	the SCU	shall turn off	Module X power	

Fig. 5. Requirements syntax checklist.

The Excel spreadsheet is available for download at https://bit.ly/Requirement-Spreadsheet-WER2023. In the tab "About" there are explanations about the spreadsheet's fields.

As a result of the requirement analysis, all the processed data can be assessed by all personnel involved (Figure 1 (4)). Requirements engineering is primordially a political negotiation; with this in mind, a consensus must be pursued. Once consensus has been reached, the written requirements are ready to be inserted into the requirements document (Figure 1 (5)). Consequently, incomplete users' needs, requirement smells, and mismatching implementation are eliminated or reduced to an acceptable level.

3 Method Application

In this section, the proposed method is applied to a real requirements document where the requirements were written without proper knowledge, resulting in sentences that do not fulfill the characteristics of the requirements. The following sentences emerged from the previous attempt at written requirements for an aerospace device:

- "The software must carry out the reconnect if there is a loss of connection between the computer and the controller during the operation of the executable in operation";
- "The software must provide the user with the recording of voltage values during battery charges during the process of operating the executable in the application";
- "The software must provide tools to facilitate and speed up the creation of new projects"; and
- "The software must provide an easy-to-understand graphical interface for creating new projects".

These "requirements" are vague, hard/impossible to test, with several restricted words, excessively long to interpret, etc. These requirements have been applied to the proposed tool, and flags such as restricted words and a limited number of words have been raised.

The actions to solve these awkward "requirements" are to reduce the number of words and eliminate the restricted words. To exemplify, two of the presented "requirements" were passed through the method, and the final accepted requirement is shown in Figure 6. Notably, the Feature ID remains the same even though the Requirement ID is incremented, while the requirement is granulated into small requirements.

Restricted words were removed, each sentence containing the modal verb "shall", and the number of words used in each requirement was acceptable. Subsequently, a requirement syntax checklist can be performed. Figure 7 shows how it was implemented in the spreadsheet.

Finally, for the requirement to be accepted as valid, the characteristics checklist is applied. The analysis of requirement characteristics will demand attention. The characteristics checked will confirm the requirements are verifiable, unambiguous, and feasible. Complete, singular, and appropriate can be evaluated when the requirements evolve; thus, these requirements are analyzed against other discovered requirements.

Feat. ID	Req. ID	Version		Requeriment Description	# Words
F-0001	R-1-0-0	1.0		The software must carry out the reconnect if there is a loss of connection between the computer and the controller during the operation of the executable in operation.	28
F-0001	R-1-0-0	1.0		The Software shall monitor the connection every 20ms.	8
F-0001	R-1-1-0	1.0	...	When the connection is lost the software shall try new connection.	11
F-0001	R-1-1-1	1.0		When 10 unsuccessful attempts the software shall abort the reconnection.	10
F-0005	R-12-0-0	1.0		The software must provide the user with the recording of voltage values during battery charges during the process of operating the executable in the application.	25
F-0005	R-12-0-0	1.0		During the battery charges the software shall record the battery voltage formatted according the ref. 123.	16

Fig. 6. Rewritten requirements assisted by the tool.

		Requirements Synthax Checklist				
Requeriment Description		Condition	Subject	Action	Object	Constraint
The software must carry out the reconnect if there is a loss of connection between the computer and the controller during the operation of the executable in operation.						
The Software shall monitor the connection every 20ms.			The software	shall monitor	connection	20 ms
When the connection is lost the software shall try new connection.		connection	the software	shall try	new connection	
When 10 unsuccessful attempts the software shall abort the reconnection.	...	10 unseccessfull attempts	The software	shall abort	recommendation	
The software must provide the user with the recording of voltage values during battery charges during the process of operating the executable in the application.						
During the battery charge the software shall record the battery voltage formatted according the ref. 123.		battery charge	the software	shall record	battery voltage	ref. 123

Fig. 7. Requirement syntax checklist.

4 Conclusion

The use of a semi-automatized requirements writing tool helps to write a strong requirements document and to exercise the foundation of requirements engineering. Furthermore, the use of a tool that analyzes the requirement syntax reduces the misconception of writing the requirements that express "how" instead of "what".

As mentioned by J. Vilela et al. [10], the usage of a requirement tool using Excel demonstrates beneficial in contrast with no use of a specific requirement tool. Moreover, it is a reaction to the use of expensive requirement tool licenses. The requirement debts, particularly the requirement smells, were minimized owing to the syntax analysis. The incomplete users' needs and mismatch implementation were mitigated by reinforcing teams' participation and feedback, as presented in Figure 1.

However, the use of Excel as a requirement tool has shown some future problems, such as the limitation of the number of requirements. It appears to be useful for small projects when only a few hundred requirements can describe the entire project. Large projects require specific tools. In conclusion, using Excel as

an alternative worked well in writing and analyzing requirements, aiding people with no experience in writing the proper requirements. As demonstrated by the previous attempt, in which the written requirement had no structure, the tool guided to a better-written requirement.

As a future work, a study to create a tool that uses natural language processing (NLP) to further analyze the requirement syntax together with a database to accommodate more requirements has been started.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Almendra, C., Silva, C., Martins, L.E.G., Marques, J.: How assurance case development and requirements engineering interplay: a study with practitioners. Requirements Engineering **27**, 273–292 (2022)
2. ECSS-E-ST-10-06C: ECSS-E-ST-10-06C – Technical requirements specification. ESA, edn. (2009), https://ecss.nl/standard/ecss-e-st-10-06c-technical-requirements-specification/
3. ISO/IEC/IEEE-29148: ISO/IEC/IEEE 29148-2018: Systems and software engineering - Life cycle processes - Requirements engineering. IEEE, edn. (2018), https://ieeexplore.ieee.org/servlet/opac?punumber=8559684
4. Lenarduzzi, V., Fucci, D.: Towards a holistic definition of requirements debt. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–5 (2019). https://doi.org/10.1109/ESEM.2019.8870159
5. Marques, J.C.: Uma análise das características de especificação de requisitos de software em normas de ambientes regulados. In: Anais do WER19 - Workshop em Engenharia de Requisitos (2019)
6. de Melo, A.C.C., Fagundes, R., Lima, J.V.V., Alencar, F., Santos, W.: Identificaçao e mensuraçao da dıvida técnica de requisitos: um survey na indústria de software. In: Anais do WER21 - Workshop em Engenharia de Requisitos (2021)
7. do Prado Leite, J.C.S.: Requirements: The never-ending story. In: Anais do WER22 - Workshop em Engenharia de Requisitos (2022)
8. dos Santos, C.R., Marques, J.C.: Asp: An aerospace specification process for hardware logic. In: Anais do WER22 - Workshop em Engenharia de Requisitos (2022)
9. Rodrigues da Silva, A., Olsina, L.: Special issue on requirements engineering, practice and research. Applied Sciences **12**(23) (2022). https://doi.org/10.3390/app122312197, https://www.mdpi.com/2076-3417/12/23/12197
10. Vilela, J., Castro, J., Martins, L.E.G., Gorschek, T., Almendra, C.: Requirements communication in safety-critical systems. In: Anais do WER19 - Workshop em Engenharia de Requisitos (2019)