Requirements debt: causes, consequences, and mitigating practices

Viviane Duarte Bonfim
Community University of Region from Chapecó
Chapecó, Brazil
Federal University of Santa Catarina
Florianópolis, Brazil
viviane.duarte@posgrad.ufsc.br

Fabiane Barreto Vavassori Benitti
Federal University of Santa Catarina
Florianópolis, Brazil
fabiane.benitti@ufsc.br

Abstract—Background: Agile development was an important initiative that changed the traditional way of developing software into an agile development process. Action is more important than the process in the agile world, and requirements and documentation do more harm than good. However, when developing requirement engineering activities inadequately, they motivate the emergence of problems that directly affect the development, which can incur requirements debt. Aim: This study investigates the causes of requirements debt that incur requirements debt and actions that can minimize and or avoid them inside the context of agile software development. Method: To fulfill the objective, we performed qualitative research, supported by data analysis suggested by Grounded Theory, with 19 subjects in 13 agile organizations at a national and international level in different segments. Results: At the end of this study, we proposed a theoretical model containing the requirements debt causes and their effects and practices that might mitigate them, and the relation between these three factors.

Keywords - Agile Requirements Engineering; Agile Software development; Requirements Debt.

I. INTRODUCTION

The current software development scenario is characterized by the broad adoption of agile methodologies [1] because the Agile Development Software (ADS) is, more and more, gaining space due to its crescent popularity and the possibility of quick and continuous deliveries [2]. It speeds up the development and adapting to the changes along the developing process. These changes suggest a flexible approach to software development, including Requirements Engineering (RE) [3].

Traditionally, the RE activities are developed separately from the development and design process and are documented in specific artifacts, promoting a formalization during this process [4]. In agile Requirements Engineering, the requirements are defined gradually along with the interaction between stakeholders and the developing team, without meeting the same formalization and, therefore, are not always adequately documented [4] in contrast to what the RE recommends [5], promoting a lack of standardization in the activities that comprise it [6].

The absence of a good requirements process may cause the RE conduction steps to fail, generating consequences such as misunderstood, omitted, ill-defined, and poorly specified requirements, including technical debts. Cunningham originally proposed in 1992 the approach as a metaphor for the term Technical Debt (TD), referring to the coding practices intending to help developers monitor the immature software code. This metaphor is related to software failure, generally motivated by development shortcuts or to the commitments made by developers to meet an urgent demand, convenient in the short term. With time, this concept evolved to other development stages, manifesting itself at the Requirements Engineering stage, also known as Requirements Debt[1] [8]. The requirements debt corresponds to failures in the requirements specification, characterizing the distance between the desired specification of requirements and the actual implementation of these requirements in the system [9]. A study performed by Ernest [10] initially revealed the requirements debt concept frequently adopted by researchers that address technical debt. These debts are still poorly understood by organizations, thus hindering the perception of their causes and their consequences. Hence, it becomes complex to prevent and treat them [11].

This research investigates the causes of the generated requirements debt, their consequences, and their mitigation actions. This study aims to allow agile organizations to understand better the scenario involving requirements debt and, in this way, mitigate them, improving their practices, aiming to prevent these debts and reduce the cost of their payment [12].

This article presents in section II the related works and, in section III, the research method adopted. Section IV presents the proposed theoretical model. Section V describes the results that supported the research. Section VI presents the threats to the validity of the research. Section VII addresses the conclusions, limitations of the study, and suggestions for future work.

DOI: 10.18293/SEKE2022-114

[1] The term initially proposed by [10] is characterized by technical debt in requirements, but recent studies address it as requirements debt [13], which this article adopts.

II. RELATED WORK

In the literature, several works involve research to investigate the conceptualization of technical debts in general, such as the work prepared by Freire et al. [12]. The researchers used data from the InsighTD project, which includes a set of surveys aimed at studying technical debts in Software Engineering, to investigate preventive actions that, if adopted, can curb the occurrence of technical debts and the difficulties in adopting these actions. The study proposed by Ramač et al. [14] demonstrates the understanding of the TD concept, together with data characterizing the causes and effects of TD in the information technology (IT) industry located in Serbia, obtained from 93 professionals.

However, few studies have investigated the requirements debt, such as the work developed by Lenarduzzi and Fucci [13]. The research carried out by Lenarduzzi and Fucci [13] presented a definition of requirements debt (ReD) that includes the debt incurred during the identification, formalization, and implementation of requirements. Lenarduzzi and Fucci [13] proposed three types of requirements debt: Type 0: Incomplete user needs; ReD Type 1: Requirement smells; ReD Type 2: Incompatible implementation. The authors suggested concepts and strategies for detecting, quantifying, and reimbursing each type.

The research developed by [13] is the only literature found that explores requirements debt and was peer-reviewed. However, they do not examine the evidence that causes the requirements debt, their effects, and strategies to prevent them. In this sense, no studies aim to investigate and identify the causes of requirements debt and their consequences and the practices that can mitigate these debts, specifically in the context of agile development, evidencing the need to carry out studies in this area.

III. RESEARCH METHOD

For the development of this research, an empirical study was carried out, following a qualitative approach guided by the Grounded Theory (GT) data analysis techniques supported by Charmaz's perspective [15]. Fig. 1 displays the flow of development of the research.

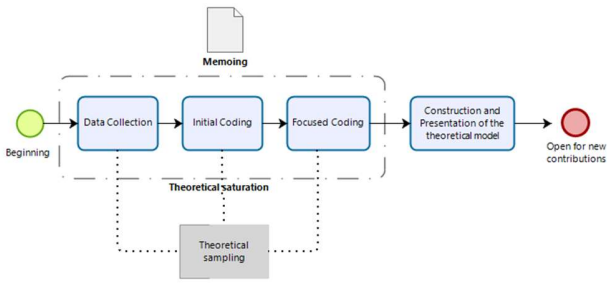


Figure 1. The process carried out during the research

The choice for GT is justified by its acceptance in the area of Software Engineering [16], as it facilitates the investigation of social and human aspects [17], and even being used to support data analysis, it results in consistent and valuable explanations about the phenomenon found [18], motivating the choice for this approach.

A. Research Question

This research aims to identify the causes that generate requirements debt and their effects, aiming at the activities of Requirements Engineering in agile organizations and actions that can mitigate the causes of these debts.

In this way, after the conclusion of the GT, it is possible to answer the following research questions:

RQ 1. What are the causes of requirements debt found in agile organizations?

RQ 1.1 What are the consequences of requirements debt?

RQ 2. What practices do agile organizations employ that can minimize requirements debt?

B. Data Collection

The analyzed data comes from the reports of 19 professionals in 13 agile organizations from 5 different countries (Brazil, France, Portugal, United Arab Emirates, and Belgium).

The research participants were selected from the researchers' contact networks or by indications from the companies' employees. The participants were contacted via e-mail, considering different profiles of employees, such as area of activity and the segment of companies, demonstrating through this diversity that theoretical sampling was achieved in the study, as shown in Table I. After the company and employees agreed to participate in the research, both filled out the consent form[2], and data collection took place.

It is relevant to mention that the participants did not need to be familiar with the term technical debt because if this aspect occurred during the interview, the researcher had the means to contextualize it to the participants.

The two instruments used for data collection were online interviews and a questionnaire. The chosen interview is semi-structured [19], as it has a previously defined script to help the interviewer in its conduction, yet, it is also supported by open questions [20]. The questionnaire[3] preparation considered the questions adopted in the interview. It was available through a form, and a single participant answered it. The participant belonged to a company located in Portugal. Such was the chosen process due to its availability and preference.

The interviews and questionnaire addressed aspects related to Requirements Engineering and Technical Debt in agile organizations and were recorded, with the participants' permission, followed by their transcripts, totaling 171 pages and

[2] Free and Clarified Consent https://forms.gle/uYFaoptYoAopKeuX6.

[3] Questionnaire: https://forms.gle/nRoMnSGBn1ijszQ8A.

approximately ninety thousand words between interviews and questionnaire.

It is noteworthy that the data collection and analysis ended only when the research reached theoretical saturation (when no new data emerged to add to the study [15]).

TABLE I. PROFESSIONALS PARTICIPATING IN THE RESEARCH

Participant profile			
Identification	**Role**	**Experience**	**Occupation area**
PA1	Project Manager	Experienced	Software or Internet Provider Management
PA2	Requirements Engineer	Experienced	Software or Internet Provider Management
PB1	Project Manager	Beginner	Solutions for Logistics and transportation
PB2	Project Manager	Beginner	Solutions for Logistics and transportation
PC1	Scrum Master	Experienced	Solutions for Logistics and transportation
PC2	Business Analyst	Beginner	Solutions for Logistics and transportation
PD1	Process Manager	Experienced	Hotel, restaurants and gas stations management
PD2	Project Manager	Experienced	Hotel, restaurants and gas stations management
PD3	Project Manager	Experienced	Hotel, restaurants and gas stations management
PE1	Requirements Engineer	Experienced	Digital transformation
PE2	Business Analyst	Experienced	Digital transformation
PE3	Requirements Engineer	Beginner	Digital transformation
PF1	Project Manager	Experienced	Solutions for Ministry of Health
PF2	Requirements Engineer	Experienced	Solutions for Ministry of Health
PG	Product Manager	Experienced	Technology solutions for the automotive sector
PH	Software Manager	Experienced	Uninformed
PI	Agile Coach	Experienced	Financial Institutions Products
PJ	Software Engineer	Experienced	Technology for the public sector
PK	Technical Leader	Experienced	Solutions for Airline
PL	Technical Leader	Experienced	Solutions for Public Sector
PM	CEO	Experienced	Solutions for Information security

C. *Data Analysis*

In the data analysis, coding techniques were used based on the Grounded Theory (GT) approach, supported by Charmaz [16], consisting of two steps for data coding: initial and focused. The GT assisted in the coding and data interpretation, and the MaxQda tool (https://www.maxqda.com/) supported performing the data analysis.

1) *Initial Coding*

Initial coding is the first step of data analysis following the constructivist approach [21], in which data are carefully examined, with line-by-line or segmental analysis of all transcribed material resulting from data collection [15]. With each interview or questionnaire, the transcription of the new data was coded and constantly compared with existing data. Table II presents segments giving rise to the different codes created, as reported by the research participants. We presented just a few examples of coded segments to support and represent the initial coding process.

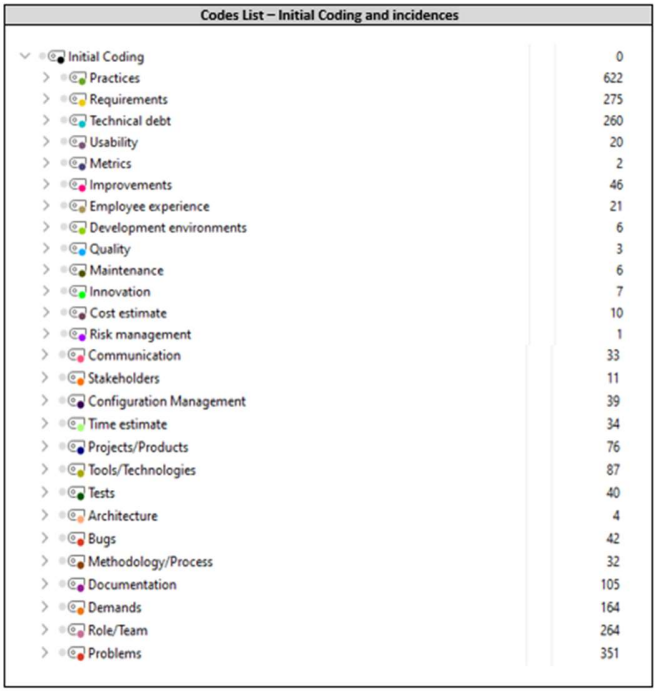
TABLE II. EXAMPLES OF PARTICIPANT REPORTS

Initial Coding	Participant Interview Segments	Participant
Requirements	*"This type of situation of requirements lacking generates a lack of adherence in an approved project and its development that we will have to return later to correct it, demand rework, generate DT, and create a complexity above what it should be".*	PC1
Technical debt	*"When business TDs happen, I believe they occur because of a lack of understanding or poor initial alignment, which was not mapped very well when the request arrived and was seen after a delivery that should have included it".*	PE1
Practices	*"We make references based on tags: e.g., workspace, documents, and notification. We are then able to connect everything, like a matrix".*	PJ
Problems	*"This process of talking to the customer to understand the requirements is not going well because it is tricky. Let's put it this way: this is a project thing. We don't have a person responsible for a feature (difficulty communicating with the customer)".*	PF1
Documentation	*"If we look at the software engineering content, this vision document is a requirement that will have functional and non-functional requirements. We call it a vision document. What for some people, I think would be that first, initial document, for us, it is inverted. Our vision document is already more detailed".*	PI

When a particular segment was selected, some related concept was verified. The new segment was associated with the existing code according to its similarity, proximity, and relationship to the created code. If it did not relate to an existing one, the analyzed segment resulted in a new code—all codes aimed at the software development process and related to Requirements Engineering and Technical Debt.

After several sessions of iteration and comparison between the data, at the end of the initial coding, 2599 coded segments were obtained, extracted from the interviews and questionnaire, and grouped into 27 codes, shown in Table III. Next to each code is its incidence (number of coded segments related to a given code).

TABLE III. CODES LIST



2) *Focused Coding*

According to the GT approach used, focused coding is the second stage of the coding process. In this step, the researchers analyzed the most relevant initial codes and organized them into subcategories and provisional categories that originated the final categories after refinement sessions. The grouping into categories and subcategories occurred according to identified similarities and differences [15], mapped and refined during comparative cycles in the data analysis, establishing connections between the categories and subcategories distributed in the following contexts: Technical Debt, Requirements Engineering, Agile Methodologies of development [22].

As a result, the researchers identified the main category of the research, represented by "Factors that impact Requirements Engineering" and its three final categories: "Causes that Generate Requirements Debt"; "Consequences that Characterize Requirements Debt"; and "Practices that can

reduce and/or address requirements debt." The categories have a set of subcategories that reflect the evidence of each of them. Tables IV, V, and VI show the subcategories. We chose this color model to match the theoretical model's subtitles in section IV. This evidence summarizes the reports verbalized by the participants during the data collection sessions and the interpretations made by the researchers and demonstrates perceptions conditioned to causes, consequences, and practices.

TABLE IV. "CAUSES THAT GENERATE REQUIREMENTS DEBTS" - CATEGORY, SUBCATEGORIES AND EVIDENCES

Category	Subcategories	Evidences
Causes that generate requirements debts	Failure to receive requirements - Requirements debts Elicitation	- Failing to clarify the demand initially received from the customer
	Failed to refine requirements requisitos_TD_Requirements Analysis	- Requirements absence
		- Delaying the development of some demands or requirements
	Inconsistency in requirements specification	- Developing without specifying the requirements (generates rework)
		- Requirements specification failure
	Not validating before development_TD_Validation	- Not validating before developing (it was not as expected)
	Absence of requirements monitoring TD_Requirements Management	- Having backlog too extensive that are difficult manage (implicates time)
		- Lack of requirements monitoring/traceability

TABLE V. "CONSEQUENCES THAT CHARACTERIZE REQUIREMENTS DEBTS" - CATEGORY, SUBCATEGORIES AND EVIDENCES

Category	Subcategories	Evidences
Consequences that characterize requirements debts	Functional and non-functional requirements (FR/FN) Inconsistency Requirements Analysis TD	- Requirements inconsistency or absence (FR/NFR)
	FR/NFR Not met due to a bad specification _TD_RE	- Lack of requirements due to a bad specification
		- Missing features and only realized customer´s feedback post-delivery
	Requirements not met realized in the delivery_TD_Validation	- Requirements not met and realized post- delivery
	Not everything that was requested was delivered _TD_Requirements Management	- Not everything is developed due to backlog accumulation/excess over time
		- Not everything that was requested was delivered (lack of requirements traceability)
		- Requirements/demands are dropped due to scope change

TABLE VI. "PRACTICES THAT CAN REDUCE AND/OR ADDRESS REQUIREMENTS DEBTS" - CATEGORY, SUBCATEGORIES AND EVIDENCES

Category	Subcategories	Evidences
Practices that help reducing and/or prevent requirements debts	Reducing requirements debts	- Assessing impact of TD - usually pays debt in the next sprint
	Meeting requirements elicitation	- Recording all requirements, demands, and stories in the backlog
		- Building process flow or BDD to understand the demand
		-Performing requirements gathering through multidisciplinary workshops
	Helping requirements analysis	- Performing the requirements or demands refinement (sprint release)
		- Defining and validating what will be prioritized
		- Performing ceremonies (initials, sprint planning, sprint review)
		- Registering and classifying FR and NFR – using checklist for NFR
		- Following a requirements process
	Supporting the implementation of requirements specification	- Keeping and describing the requirements specification document clearly
		- Developing prototypes as part of the requirements specification document
		- Drawing up sequence and use case diagram and database model
	Implementing requirements validation	-Validating prototypes after requirements specification with customer
		-Performing experimentation, prototypes, MVP or customer's research
	Assisting in requirements management	- Managing the requirements, deliveries, backlog delays in progress
		- Performing requirements traceability

The focused coding, when completed, made it possible to build the theoretical model that underlies the entire study through the inductive/deductive process.

IV. THEORETICAL MODEL

The GT's coding theory supports the theoretical model presented in Fig. 2 and portrays the result of data analysis. During the elaboration of the theoretical model, it considered some findings that emerged from the data, supported by the analytical resources of the MaxQda tool.

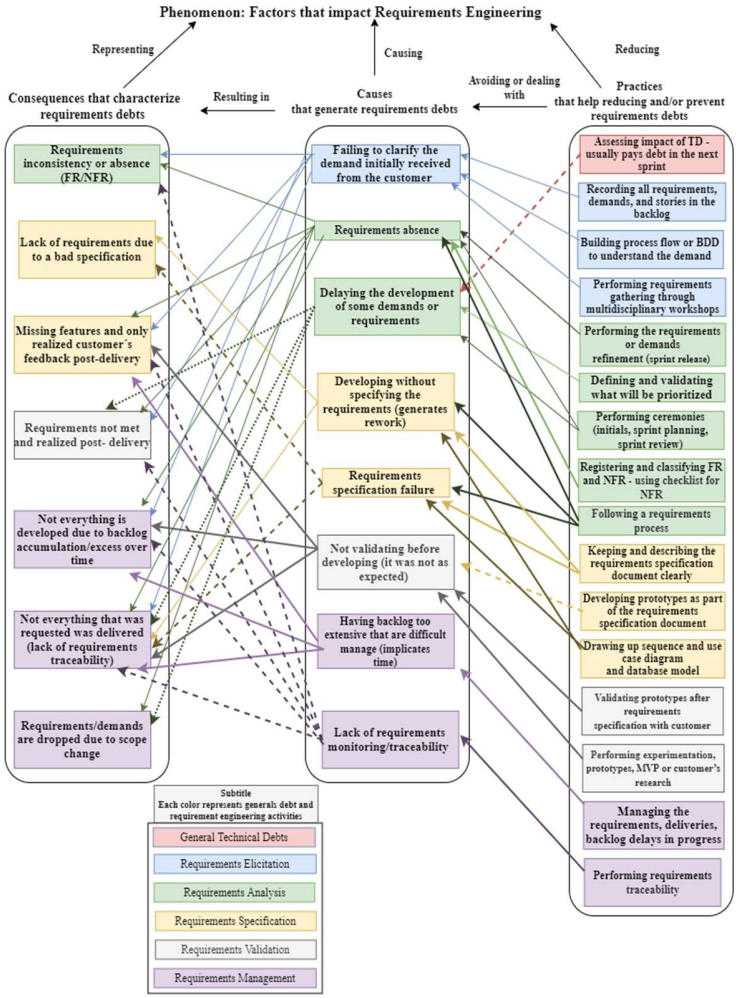


Figure 2. Proposed Theoretical Model

Three factors (categories) that impact Requirements Engineering characterize the model. The categories appear in three columns: Consequences highlighting the requirements debt themselves in the first column. In the central column are the causes for requirements debt, and in the third column, the practices that can minimize the causes and, consequently, the requirements debts.

The arrows employed define the relationship between these factors, indicating the causes that incur requirements debt and which practices can minimize or avoid them and consequently the debts. The option for the different styles and colors of the arrows intends to facilitate the understanding and legibility of the theoretical model.

A. *Data Validation*

The research data validation occurred with the participants by completing a form[4] after elaborating the theoretical model. The issues involved in the form are associated with the three factors that impact Requirements Engineering: Causes of requirements debt, their consequences, and the practices that can mitigate these debts. Of the 19 research participants, 10 participated in data validation[5], as some had left the companies they worked for at the time of data collection, and the other participants did not complete the validation form.

V. RESULTS

According to the model proposed in section IV, we identified eight leading causes that generate requirements debt, seven consequences that characterize the emergence of debts, and 16 possibilities of practices that can prevent and or treat these causes and consequently curb requirements debt.

After concluding the research, we believe that the theoretical model answered the research questions, establishing a connection between the categories, subcategories, and evidence resulting from the data analysis, unifying them, allowing a better understanding and identification of the causes that incur requirements debt and alternatives to minimize them.

The results obtained can help companies understand the causes of debt requirements, their effects, and what actions they can take to mitigate such debts. In the following sections, the participants' reports exemplify at least one cause, one consequence, and one practice due to space restrictions.

A. *RQ 1. What are the causes of requirements debts found in agile organizations?*

Among the observed causes for requirement technical debt, the most reported cause was "Absence of requirements", evidenced by 13 participants. According to the mentions, the "absence of requirements" is a cause of requirements debt, perceived after delivery. Such debts point out that during the requirements analysis stage, the teams should refine a particular request to prevent identifying the absences only after delivery, as reported by PC1. According to the interviewees, the lack of understanding or initial alignment may reflex this cause.

"The absence of requirements causes a lack of adherence to the developed software. When this occurs, it is necessary to return to the requirements process to correct them, update them, and the following phases of the development process. This absence generates rework and additional cost, making the software development process more complex"[6]. – PC1, Scrum Master.

B. *RQ 1.1. What are the consequences of Requirements Debt?*

We considered the consequence most frequently mentioned by the participants. The main one identified: "They do not develop everything – Accumulation or excess of the backlog over time," with 10 citations. The consequence mentioned makes management difficult, compromising the requested requirements, and often, no practice is adopted to facilitate monitoring. According to research participants, this accumulation occurs due to changes in priorities, as highlighted by PJ, since, with the backlog growing disproportionately, some requirements may be disregarded and never developed.

"We did not respond to everything that was requested. Many things that stay in the backlog are due to other requests, and they are priorities related to those in the Backlog". - PJ, Software Engineer.

C. *RQ 2. What practices do agile organizations employ that can minimize requirements debt?*

Among the practices that can mitigate the requirements debt, we present the most expressive for the study, as reported by the participants: "Managing the Requirements, deliveries, delays, backlog, and progress," revealed by 15 participants. As highlighted in the interviews, the practice of "Managing the Requirements, deliveries, delays, backlog, and progress" allows controlling and monitoring the progress of requirements through their status, such as if they are already met, if they need development, and if they are late. It is also possible to track what is in the backlog, how long a particular demand or requirement remains there, and the requirements awaiting development. Some companies mentioned that to manage their requirements, they use specific tools because it allows them to explore the visualization of the status of each functionality/requirement through representative graphics, as mentioned by PJ.

"An Agile Board monitors all the steps: What is in the backlog, what is under analysis, awaiting development, in development, awaiting review, in review, awaiting test, in test, ready. Each of the requirements is in one of these blocks. In addition, we have the Burndown where we verify and follow up the deliveries concerning the sprint time". Agile Board integrates with Redmine. - PJ, Software Engineer.

VI. THREATS TO VALIDITY

There were mechanisms adopted in this study to mitigate some threats, highlighting some points described below. Focusing on construction validity, during the development of the study, we sought to explore the data collection instruments (interview and questionnaire) with participants from different

[4]Example of data validation: https://www.survio.com/survey/d/P7H8V2F7G2T4F7A4E

[5] Data Validation Results: https://drive.google.com/file/d/14m_cvUfuto8hrWJ1AD5_o0bFs_Xj7kd8/view?usp=sharing

[6] It is noteworthy that in the GT, transcripts must occur in full according to the participant´s speech. However, to provide better compression, there were some adaptations without changing context due to the article´s language.

countries and segments to absorb a significant set of data until data saturation occurred. When a participant did not understand a term or question in the interviews, the researcher was available to clarify. The same happened for questions related to filling the questionnaire out.

Considering the internal validity, the researcher's interpretation during the data coding was possibly not as faithful to the portrayed data from the interviews and questionnaire as it should be. This interpretation could reflect on the study´s results, even when the researchers reanalyzed the data when any doubts would arise. To minimize possible limitations, the research participants validated the results of the data analysis. However, of the nineteen research participants, ten participated in the validation, which can also characterize a threat to validity. A systematic study is underway to mitigate the threat regarding data validation. Said study will support validation and potentialize the obtained in this research.

VII. CONCLUSIONS AND FUTURE WORK

This article presented a study on the state of practice regarding the Requirements Debt involving 19 participants from agile organizations in different segments located in 5 countries. The research contemplated a qualitative approach supported by Grounded Theory data analysis techniques.

This study made it possible to observe that organizations do not have mechanisms to identify and recognize their requirements debts, and as a consequence, they do not manage such debts, making their mitigation difficult. To help organizations minimize these weaknesses, the theoretical model we propose synthesizes the results of this research. The theoretical model demonstrates the relationship between the identified categories and evidence, allowing agile organizations to recognize which causes generate requirements debts and their incurred debts for a better understanding and ways to minimize them by adopting the recommended practices.

In future work, we intend to provide a library of practices through a platform, suggesting a set of practices that help agile organizations minimize or mitigate requirements debt, regardless of the adopted process. We also mean to provide ways for companies to recognize the causes of requirements debt and their already existing debts to facilitate the implementation of practices.

REFERENCES

[1] N. Rios, M. Mendonça and C. Seaman. “Causes and effects of the presence of technical debt in agile software projects”. Twenty-fifth Americas Conference on Information Systems, AMCIS, 2019.

[2] K. Elghariani and N. Kama. “Review on agile requirements engineering challenges”. 3rd International Conference on Computer and Information Sciences (ICCOINS), August. 2016. DOI: 10.1109/ICCOINS.2016.7783267.

[3] E. Schön, D. Winter, M. J. Escalona and J. Thomaschewski. “Key challenges in agile requirements engineering”. International Conference on Agile Software Development: Agile Processes in Software Engineering and Extreme Programming, pp. 37-51, April 2017.

[4] E. Bjarnason, K. Wnuk and B. Regnell. “A case study on benefits and side-effects of agile practices in large-scale requirements engineering.” AREW '11: Proceedings of the 1st Workshop on Agile Requirements Engineering, pp. 1-5, July 2011. DOI: https://doi.org/1.1145/2068783.2068786

[6] H. F. Soares, N. S. R. Alves, T. S. Mendes, M. Mendonça and R. O. Spinola. “Investigating the link between user stories and documentation debt on software projects”. International Conference on Information Technology - New Generations, April 2015. DOI: 10.1109/ITNG.2015.68.

[7] W. Cunningham. “The WyCash portfolio management system”. Proc. OOPSLA, October 1992. DOI: https://dl.acm.org/doi/pdf/10.1145/157710.157715

[8] P. Avgeriou, P. Kruchten, I. Ozkaya and C. Seaman. “Managing Technical Debt in Software Engineering”. Dagstuhl Seminar 16162, January 2016. DOI: 10.4230 / DagRep.6.4.110.

[9] Y. Guo, R. O. Spínola and C. Seaman. “Exploring the costs of technical debt management – a case study”. Empirical Software Engineering, vol. 21. ed. 1. pp. 159-182, 2016. DOI: 10.1007/s10664-014-9351-7.

[10] N. A Ernst. “On the role of requirements in understanding and managing technical debt”. Third International Workshop on Managing Technical Debt (MTD), June 2012. DOI: 10.1109/MTD.2012.6226002.

[11] W. N. Behutiye, P. Rodríguez, M. Oivo and A. Tosun. “Analyzing the concept of technical debt in the context of agile software development: A systematic literature review”. Information and Software Technology, vol. 82, pp. 139-158, February 2017. DOI: https://doi.org/10.1016/j.infsof.2016.10.004.

[12] S. Freire, N. Rios, M. Mendonça, D. Falessi, C. Seaman, C. Izurieta and R. O. Spínola. “Actions and impediments for technical debt prevention: results from a global family of industrial surveys”. SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, pp. 1548–1555, March 2020. DOI: https://doi.org/10.1145/3341105.3373912.

[13] V. Lenarduzzi and D. Fucci. “Towards a holistic definition of requirements debt”. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), vol 1, pp. 1-5, September 2019. DOI: 10.1109/ESEM.2019.8870159.

[14] R. Ramač, V. Mandić, N. Taušan, N. Rios, M. G. Mendonça, C. Seaman and R. Oliveira Spinola. “Common causes and effects of technical debt in Serbian IT: InsighTD survey replication”. 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), August 2020. DOI: 10.1109/SEAA51224.2020.00065.

[15] K. Charmaz. “Constructing Grounded Theory: A Practical Guide through Qualitative Analysis”. Sage Publications, 2006.

[16] K. Madampe, R. Hoda, J. Grundy and P. Singh. “Towards understanding technical responses to requirements changes in agile teams”. IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW), pp 153-156, June 2020. DOI: https://doi.org/10.1145/3387940.3392229.

[17] R. Hoda and J. Noble. “Becoming agile: A grounded theory of agile transitions in practice”. IEEE/ACM 39th International Conference on Software Engineering (ICSE), May, 2017. DOI: 10.1109/ICSE.2017.21.

[18] R Hoda. “Decoding Grounded Theory for Software Engineering”. IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), May 2021. DOI: 10.1109/ICSE-Companion52605.2021.00139.

[19] W. C. Adams. “Handbook of practical program evaluation: Conducting Semi-structured interviews”. 3. Ed, Chapter Sixteen, pp. 365-376, 2010. ISBN: 978-0-47052247-9.

[20] J. Melegati, A. Goldman, F. Kon and X. Wang. “A model of requirements engineering in software startups”. Information and Software Technology, vol. 109, pp. 92-107, 2019. DOI: https://doi.org/10.1016/j.infsof.2019.02.001.

[21] K. Charmaz. “Constructing Grounded Theory. 2nd., 2014.

[22] P. Bourque, R. E Fairley. “Swebook. Guide to the Software Engineering”. Version 3.0. IEEE Computer Society, 2014.