The Journal of Systems & Software 194 (2022) 111483

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Identification and measurement of Requirements Technical Debt in software development: A systematic literature review☆

Ana Melo [a], Roberta Fagundes [a], Valentina Lenarduzzi [b,*], Wylliams Barbosa Santos [a]

[a] University of Pernambuco, Recife, Brazil
[b] University of Oulu, Oulu, Finland

ARTICLE INFO

Article history:
Received 18 May 2021
Received in revised form 30 June 2022
Accepted 8 August 2022
Available online 12 August 2022

Keywords:
Technical debt
Requirements Technical Debt
Identification
Measurement
Systematic literature review

ABSTRACT

Context: Requirements Technical Debt are related to the distance between the ideal value of the specification and the actual implementation of the system, which are consequences of strategic decisions for immediate gains, or unintended changes in context. To ensure the evolution of the software, it is necessary to to manage TD. Identification and measurement are the first two stages of the management process; however, they are poorly explored in academic research in requirements engineering.
Objective: We aimed to investigating which evidence helps to strengthen the TD requirements management process, including identification and measurement.
Method: We conducted a Systematic Literature Review through manual and automatic searches considering 7499 studies from 2010 to 2020, and including 66 primary studies.
Results: We identified some causes related to Technical Debt requirements, existing strategies to help in the identification and measurement, and metrics to support the measurement stage.
Conclusion: The studies on Requirements Technical Debt are still preliminary, especially regarding management software. Yet, however, the interpersonal aspects that prove difficult in the implementation of such activities are not sufficiently addressed. Finally, the provision of metrics to help measure technical debt is part of the contribution of this search, providing insights into the application in its requirements context.

1. Introduction

In software development, even in well-planned projects, some challenges can negatively impact the final delivery, such as pressure from the customer to complete the software before the deadline, limited resources, or pressure from the market itself (Rios et al., 2018a). In this scenario, the development team needs to use alternative solutions to accomplish the tasks in the short term, often without considering the possibility of negatively impacting the software in the long term (da Silva Maldonado et al., 2017). In this context, quality issues may be identified in the project during or after its implementation. Thus, tasks that have been compromised must be improved at some point during the project, failure to do so may result in a phenomenon known as Technical Debt (TD) (Cunningham, 1992).

TD refers to problems caused when software development tasks are pending or inefficiently executed (Kruchten et al., 2012). While these actions may provide benefits in the short term, such as increased productivity, they also pose risks to the project and hinder its development (Guo et al., 2016; Rios et al., 2018b). Thus, TD includes items usually controlled in a software project, such as unimplemented features. Moreover, it covers less visible aspects such as code smells and outdated documentation (Brown et al., 2010).

Initially, the focus of TD was on coding activities (Cunningham, 1992), but as the research developed, the concept was extended to the other phases of software and software development, for example, in requirements engineering (Li et al., 2015a). According to Ernst (2012), an inadequate elicitation or analysis of requirements causes errors that increase the incidence of TD in software projects. In this setting, the technical debt of requirements may occur intentionally, such as when a conscious decision is made not to be rigorous in the elicitation process, or unintentionally, when the requirements engineers are inexperienced and may not have the needed skills to perform technical and long-term procedures (Rios et al., 2019).

Despite the importance of requirements engineering at the software development, there is still no consensus whether Requirements Technical Debt should be considered as a type of

☆ Editor: Aldeida Aleti.
* Corresponding author at: University of Oulu, Oulu, Finland.
E-mail addresses: accm@ecomp.poli.br (A. Melo), roberta.fagundes@upe.br (R. Fagundes), valentina.lenarduzzi@oulu.fi (V. Lenarduzzi), wbs@upe.br (W.B. Santos).

https://doi.org/10.1016/j.jss.2022.111483

TD; moreover, the literature lacks a definition (Alves et al., 2014; Lenarduzzi and Fucci, 2019). Although Ernst first mentioned the definition of Requirements Technical Debt, Wang and Huang (2020) consider it a brief description that does not provide enough information to conceptualize it, which could involve, for example, multiple reasons or causes to induce it.

Regardless of how TD occurs, it is necessary to keep it managed to ensure the software evolution and quality, in order to avoid late discovery of its amplitude and consequently, costs that cause the incidence of interest for correction (Alves et al., 2018). Identification and measurement are the first two steps in the management process (Li et al., 2014). They are essential activities to know what type of TD exists, where it is located, and how to estimate its impact on the software (Alves et al., 2016). However, they are considered the phases in which there is greater difficulty in achieving (Besker et al., 2018) and, that practitioners realize that most of the time spent managing technical debt is lost in understanding and measuring it (Besker et al., 2019).

The measurement step is essential to estimate the costs, interest, effort required, and the TD impact on the software. However, scientific evidence shows that professionals lack knowledge on how to calculate the interest of TD, that is, the extra effort required that will be spent in the future if the TD is not paid at the time of its identification (de Melo et al., 2021). In this sense, presenting evidence and metrics that help calculate TD interest assertively can allow software development organizations to organize their refactoring activities based on accurate estimates, avoiding the accumulation of TD (Lenarduzzi et al., 2021).

Although professionals and researchers have been giving much attention to TD in recent years to (Rios et al., 2018a; Gama et al., 2019), in the requirements engineering area, the process management, especially in the identification and measurement of TD, is still a gap to be explored in academic research (Alves et al., 2018). Furthermore, Yli-Huumo et al. (2016) believe that there are no adequate tools or resources to manage a Requirements Technical Debt. In this way, sufficient evidence that helps to meet the information needs that the present study aims at was not identified, especially regarding the use of metrics that guide assertive decisions about the reimbursement of the TD of requirements.

In this context, the current work presents a Systematic Literature Review (SLR) to identify and give an overview of the state of art related to TD management in software requirements, specifically about the stages of identification and measurement of this type of TD. In the end, as main contributions, evidence and studies are presented that show the leading causes attributed to the emergence of the technical debt of requirements; strategies that contribute to its identification, as well as supporting metrics in the measurement stage; finally, identify gaps and opportunities for the development of new research, encouraging other researchers to continue research in the area.

In addition to this section, the rest of this work is structured as follows: Section 2 presents the background; Section 3 contains the methodology used to perform the SLR; Sections 4 and 5 report and discuss the results obtained; Section 6 exposes the threats to the validity of this research; and finally, Section 7 contains the conclusions and future studies.

2. Background

The purpose of this section is to introduce concepts that underpin this very research, as well of the related works.

2.1. Technical debt

According to Seaman and Guo (2011), TD is defined as immature or incomplete artifacts present in the software development life cycle, causing higher costs and low quality. The creation of these artifacts can accelerate development in the short term. However, low quality tends to generate expenses in the long run due to maintenance efforts used for corrections. According to McConnell (2008), technical debt can be categorized into two types:

- **Unintentional TD**, which occurs involuntarily and non-strategically, is often caused by poorly planned activities because of inexperienced professionals or changes in the environment;
- **Intentional TD**, is deliberate and strategically motivated when professionals make decisions to achieve short-term benefits resulting from shortcuts, alternative solutions, and unexecuted tasks.

Additionally, according to Rios et al. (2018a), TD may be present in different activities and phases of the software development life cycle. With this, these same authors present a set containing 15 different types of identified technical debts. Table 1 presents each type and their respective definition.

2.2. Requirements Technical Debt

Requirements engineering is one of the areas of software engineering that aims at including the usage and analysis of techniques and activities to obtain, specify, and document a set of requirements that attend the needs of stakeholders with a high-quality product (Vazquez and Simões, 2016). The requirements describe the software, its behavior, functionality, constraints, and all other attributes. However, this is a tricky step that stakeholders (analysts, users, developers) often do not pay enough attention to (Wiegers and Beatty, 2013).

According to Van Vliet et al. (2008), stages and tasks of requirements engineering when performed inadequately can cause problems that affect the development of the software, such as low-quality elicitation, incomplete or outdated requirements along with other existing problems, are real examples of technical debt. For Ernst (2012) and Brown et al. (2010), the TD of requirements is related to the distance between the ideal value of the specification of requirements and the actual implementation of the system, which is a consequence of strategic decisions for immediate gains, or unintended changes in context, which lead to future costs.

This type of technical debt is part of the the exchanges made regarding which requirements the development team needs to implement or how they should be implemented and according to Abad and Ruhe (2015), can be defined as trade-offs during the specification of requirements. Given what has been said, the requirements that are partially implemented, not satisfactorily specified, poorly prioritized or developed without considering their dependencies and relationships, represent errors that increase the incidence of TD, leading to increased interest and effort needed for the correction if this type of technical debt is not identified and managed in a timely manner (Ernst, 2012; Abad and Ruhe, 2015).

Recently, the work of Lenarduzzi and Fucci (2019) defined the TD of requirements in three types:

- **Type 0: Incomplete Users' needs**

Represents TD that result from neglecting the needs of stakeholders or a specific group of stakeholders. For example, in the case of consumer-centric systems such as mobile applications, TD

Table 1
Types of technical debt.

Type	Definition
Design	Refers to TD discovered by analyzing the source code and identifying violations of principles of good object-oriented design.
Code	Refers to problems found in the source code (violating best practices or coding rules) that negatively affect its readability and make it difficult to maintain.
Architecture	Refers to problems found in the product architecture, which affect the architecture requirements. Generally, this TD is the result of initial solutions below the ideal, compromising internal aspects of quality.
Test	Refers to problems found in testing activities that affect their quality.
Documentation	Refers to the problems found in software project documentation.
Defect	Refers to known defects, usually identified by test activities or the user. The development team agrees to correct them, but due to competing priorities and limited resources, they will be delayed.
Infrastructure	Refers to infrastructure problems that, if present in the software organization, delay or hinder development activities. Such TD negatively affects the team's ability to produce a quality product.
Requirements	Refers the distance between the optimal requirements specification and the actual system implementation.
People	Refers to people issues that, if present in the software organization, can delay or hinder some development activities.
Build	Refers to issues that make the build task harder, and unnecessarily time consuming.
Process	Refers to inefficient processes, e.g. (the projected process may not be appropriate).
Automation	Refers to the work involved in the automation of functionality tests developed to support continuous integration and faster development cycles.
Usability	Refers to inappropriate usability decisions that will need to be adjusted later.
Service	Refers to inappropriate web services that lead to incompatibility between service features and application requirements.
Versioning	Refers to problems in source code versioning, such as unnecessary code forks.

arises when the needs of users expressed in feedback channels are forgotten. The authors also present that this type of TD of requirements can be quantified as the proportion between the user needs that have already been elicited and all possible necessities, including neglected ones. The principal one is the cost of obtaining all the user's remaining needs, and the interest is the cost associated with the risk of missing an important need. In other words, a requirements engineer must decide whether it is worth spending time identifying additional user needs, taking into account, for example, the current development stage of the software associated with implementing a requirement detected later.

- **Type 1: Requirement smells**

Represents the TD that arises when linguistic constructions may indicate a violation of ISO/IEC/IEEE 29148:2018, that relates to the quality of requirements. These smells also exist for other requirements documentation approaches, e.g. UML. If these smells are not removed, the requirement may be implemented incorrectly, making it difficult to reuse and evaluate. In this sense, the authors state that requirement smells also need to be reimbursed similarly to code smells. Hence, the principal can be quantified as the cost to correct them and the interest as the negative impact on the stages of software development in which they are associated with.

- **Type 2: Mismatch implementation**

Represents the TD incurred when developers implement a solution to a requirements problem. Thus, an incompatibility is identified between the stakeholders' objective framed during the specification of requirements and the actual implementation of the system. According to the authors, a way to identify this type of technical debt requirement can be based on approaches to traceability between the requirements specification and source code, such as RE-KOMBINE (Ernst, 2012) (will be presented in Section 4.3). Finally, this third type of Requirements Technical Debt is quantified as the cost of comparing the current software implementation with the set of possible changes (Principal), additionally the performance of the selected change (Interest).

2.3. Technical debt management

According to Li et al. (2015a), the management of TD is an important step to achieve good quality in the development and maintenance of the software, since most of the debts are often not managed. By Tom et al. (2013), it is necessary to define processes that can track these technical debts, so that later, decisions can be based on the identified problem. Also, recent research shows that knowing the existence of technical debt influences the behavior of the team, i.e. applying the best techniques of identification and measurement, for example, can significantly improve software development practices (Tonin, 2018).

The management process includes activities used to control and reduce the technical debt in a software project. In this circumstance, with the inclusion of different techniques, tools, and evidence, companies aim to reduce and prevent shortcuts and solutions that do not achieve the expected success (Li et al., 2015a). However, most of the TD items are inadequately managed, thereby further increasing the risk of high maintenance costs (Tonin et al., 2017). Therefore, it is appropriate to find the best ways to ensure that the TD achieves is properly managed, which will facilitate decision-making on future activities (Alves et al., 2018).

In the work of Li et al. (2014), a technical debt management method was proposed, containing five steps: identification, measurement, prioritization, repayment, and monitoring, which are described below.

1. **Identification:** the process of visualizing the TD, identifying its causes and other attributes present in software development that led to its existence. This activity is crucial for the proper management of TD;
2. **Measurement:** analyzes and quantifies the costs and efforts required to assist in decision making regarding technical debt reimbursement;
3. **Prioritization:** organize the payment of technical debts in order of importance, analyzing factors such as technical issues and financial implications;
4. **Repayment:** regarding the partial or total payment of the technical debt, avoiding postponing it if it could negatively affect the project;
5. **Monitoring:** validates whether the technical debt is being diluted, delayed, or continues to cause costs.

Table 2
Related works.

Work	Goal
Alves et al. (2016)	TD management strategies and TD taxonomy
Nascimento et al. (2018)	Investigate and conceptualize requirements smells
BenIdris (2020)	Analyze TD in empirical studies
Wang and Huang (2020)	Conceptualize requirements TD and find approaches to manage it
Lenarduzzi et al. (2021)	Identify TD prioritization tools, strategies, processes and factors
Our work	Analyze strategies and metrics to identify and measure TD of requirements

2.4. Related works

This subsection presents the works related to the objectives proposed in this study. They are listed in chronological order and can be identified in Table 2. Next, the details of each study are presented, and a comparative analysis of the differences concerning this study is made.

The work of Alves et al. (2016) was aimed at conducting a Systematic Mapping Study (SMS) to analyze which strategies have been proposed to help manage TD in software projects and analyze their main types. The search process was conducted automatically, recovering searches from 2010 to 2014, and in the end, 100 studies were considered. Among the results, they proposed an initial taxonomy of the TD types and a list of existing strategies for identification and management. Finally, a current state-of-the-art analysis identified gaps where new research efforts could be invested.

According to the authors, this study was the first step towards the creation and validation of taxonomy on the types of TD and the development of new technologies that help in its management. However, even though it is considered a relevant study in the area, it still not present evidence to help measure TD, particularly through the with the use of metrics, so that decisions about repayment can be based on this type of evidence, which is part of the objective of this work. Finally, the authors analyzed primary studies until December 2014, so the SLR presented in this study complements the results based on recent years.

In work proposed by Nascimento et al. (2018), an SMS was conducted to investigate evidence on the subject of requirements smells, thus helping in their understanding and assisting researchers in future studies. The search was performed automatically, recovering research papers with a publication date from January 2013 to March 2018, and at the end, 41 studies were considered. Among the results, it was identified that the concept had gained visibility in recent years and the development and existence of support by tools.

Thus, the authors' focus was on collecting evidence about requirement smells, which, if not corrected during project implementation, may influence the occurrence of requirements TD. The respective work , then becomes relevant in presenting information that can improve the requirements engineering process and other areas dependent on it. However, the authors' focus became conceptualizing this phenomenon rather then presenting strategies to address it.

In the work proposed by BenIdris (2020), an SMS was executed to identify and analyze TD in empirical studies published from 2014 to 2017. The search process was carried out in an automated way, and in the end, 100 studies were considered. Among the main results, the presentation of the most common indicators and evaluators to identify and evaluate TD and, the identification of tools and strategies that help to investigate and estimate. The work presents essential information for the study area. However, the authors have not addressed the results for the software industry and academia, and it does not present opportunities for further research. Furthermore, they did not have metrics that could be used to support their measurement.

The work of Wang and Huang (2020) contemplated the investigation of the current state of the TD of requirements, in order to be able to present a precise definition of this type of TD. To achieve this goal, they conducted an SMS, and a survey. Among the results, ten measurement techniques were identified, and suggestions from software professionals about the detection of this TD. Finally, they discovered that academia and industry deal with this TD differently and encouraged both sides to collaborate. It is believed that this work will support the aforementioned study above since part of the information is associated with that. Nonetheless, metrics that help measure the costs or efforts, related to Requirements Technical Debt were not analyzed.

Lastly, the paper of Lenarduzzi et al. (2021) investigated which approaches to prioritizing TD have been proposed in software engineering research and industry. In order to do so, they conducted an SLR, which at the end included 44 primary studies. Among the results, they observed that research on TD prioritization is preliminary and that there is no consensus on which factors are essential and how to measure them. Subsequently, they will propose a mind map that can help software professionals during TD prioritization.

The referred study is similar to the current proposal of this work because it aims to provide information that helps to assertively carry out a specific stage of the management process of TD. However, this work aims to contribute to the first two stages: identification and measurement, presenting various information for the context under study.

The works cited and the present article are related because they seek ways to understand and manage TD. But in contrast with the works mentioned, this study conducted two types of search (manual and automatic). Also, we use snowballing method as a complement. This generates more research sources to be considered. On the one hand, although, they have not addressed the causes for their emergence and metrics for measurement methods, this present work is similar to the study of Nascimento et al. (2018) and Wang and Huang (2020). The works of Alves et al. (2016) and BenIdris (2020) focused on TD in general, gathering evidence to help in its management. Finally, the paper of Lenarduzzi et al. (2021) analyzed a specific step (prioritization) of the TD management process but differed by not addressing evidence focused on its identification and measurement.

3. Systematic literature review

The SLR conducted in this work was based on the method proposed by Kitchenham and Charters (2007). According to the authors, a systematic approach is pre-defined using a protocol and procedures to identify and interpret the evidence available in Primary Studies (P), related to one or more research questions. About the objectives, SLRs try to connect primary studies in terms of their results and investigate whether these results are consistent. SLRs aim, therefore, to synthesize evidence, including considering its strength (Kitchenham et al., 2011). The process of this SLR is presented in Fig. 1.

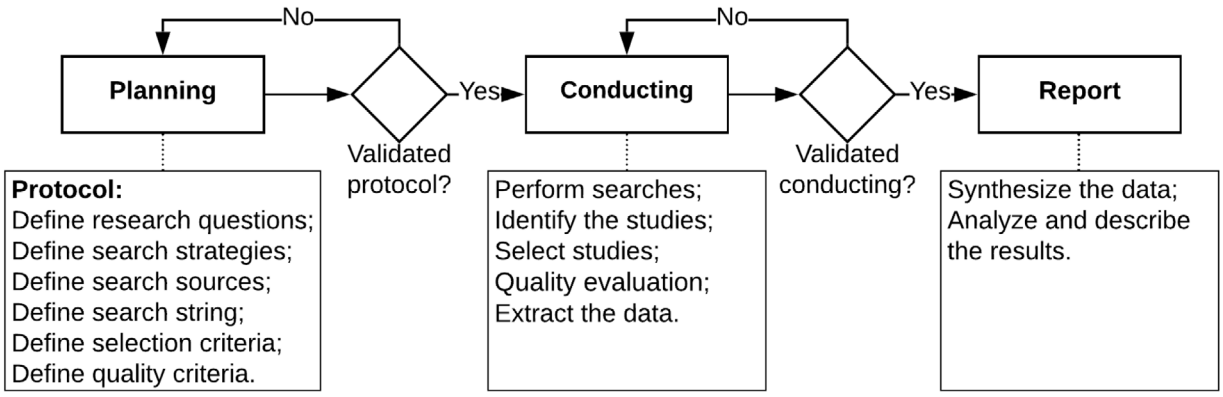

Fig. 1. Process of conducting systematic literature review.

3.1. Planning

3.1.1. Research questions

This work's main objective is to provide evidence to assist identifying and measuring of TD requirements in software development. To understand this objective, the following Research Question (RQ) was defined: "How to assist in the identification and measurement of the Requirements Technical Debt in software development?". To answer this question, we have divided it into sub-questions:

RQ1: *What has caused the technical debt of requirements in software development?*

Identifying a TD is not only about understanding how and where it occurred — but also analyzing the causes that led to its occurrence. In addition, Rios et al. (2018a) reports that this is a topic (causes for TD insertion in projects) that remains little explored in academic research. The answers to this question will help you understand the causes of the emergence of TD requirements, aiding in their identification and prevention.

RQ2: *What strategies are proposed to help identify and measure the Requirements Technical Debt in software projects?*

As important as managing TD items in a project is implementing efficient and effective management strategies for such activities. Answering this question may help practitioners in the selection of strategies and tools already available. Based on the evidence, they will be able to analyze and adapt the strategies that best fit their needs, objectives, infrastructure, and other related factors.

RQ3: *What metrics are being used to assist in the process of measuring the Requirements Technical Debt?*

This question aims to identify a set of metrics that are considered valid and provide more accurate estimates when measuring a TD. In addition, it seeks to understand the main variables considered in this step, such as the principal and interest of the technical debt.

RQ4: *What difficulties are pointed out during the management of Requirements Technical Debt in software development?*

As mentioned earlier, the main goal of this work is to provide evidence to help identify and measure the TD of requirements. However, expected to identify and present background gaps and opportunities for the development of new research to encourage other researchers to investigate in this area. In this way, direct recent efforts that can offer support in identifying and measuring, as well as managing, technical debt as a whole. It should also be noted that some of these difficulties are related to the findings presented in the paper, such as the challenge of being able to measure debt.

Table 3
Sources and digital libraries used.

Manual search
Information and Software Technology (IST)
International Journal of Software Engineering and Knowledge Engineering (SEKE)
International Requirements Engineering Conference (RE)
International Workshop on Managing Technical Debt (MTD)
International Conference on Technical Debt (TechDebt)
Journal of Systems and Software (JSS)
Automatic Search
ACM Digital Library
IEEE Digital Library (IEEEXplore)
Science Direct
SCOPUS
SpringerLink

3.1.2. Sources and search string

The research for primary studies was initially conducted through manual and automated search of specialized and renowned scholarly scientific sources and digital libraries in Computer Science and the subjects related to the objective of this thesis. Note that some manual databases are usually indexed by the digital libraries used in the automated search. However, we chose to consider them ensure that all primary studies are analyzed in the respective databases would be analyzed. Table 3 lists the search sources used.

In searching for relevant results among digital libraries in the automatic search, a search string was formed based on two criteria: (i) higher number of results recovered from digital libraries and (ii) studies strongly related to the search theme. Considering the objective of this research, the search string is formed by three main keywords, that is, we are looking for studies that present evidence on technical debt in requirements engineering, or that report on the application of metrics during the measurement of a TD, so that, in the end, it becomes a support resource for conducting this stage of the Requirements Technical Debt management process.

We would like to highlight that our search string is based on the definitions of the secondary studies of Behutiye et al. (2017) related to TD; the study of Saha et al. (2012) on software requirements; finally, the study of Riaz et al. (2009) adapting the terms related to measurement and metrics. Hereby, the following search string was defined:

("technical debt" OR "technical debit" OR "design debt" OR "debt metaphor") ***AND*** *("requirement*" OR "requirement engineering" OR "software requirements" OR "user story" OR "measurement" OR "metrics" OR "measure" OR "measurement metrics")*

Table 4
Inclusion criteria.

I1: Studies published between 2010 and 2020
I2: Studies wrote in English
I3: Studies related to the identification and measurement of Requirements Technical Debt in software development

Table 5
Exclusion criteria.

E1: The study does not answer at least one research question
E2: The study is not accessible
E3: Secondary and tertiary studies
E4: The study is a copy or an older version of another study already considered
E5: Studies published before 2010

We used the asterisk character (*) in order to capture possible term variations such as plurals and verb conjugations.

3.1.3. Selection criteria

Inclusion (I) and Exclusion (E) criteria were defined to assist the primary studies' selection process. The criteria can be observed in the Tables 4 and 5.

3.1.4. Quality assessment criteria

The quality evaluation of the studies was performed to ensure that the final selection list included the most relevant to this work's objective. For this purpose, the Quality Criteria (Q) proposed by Dyba et al. (2007) were used, which are evaluated in the following quality guidelines:

Reporting: the quality of the logic of the objectives and the context of the study;
Credibility: the rigor of the research methods used to establish the validity of the data collection tools and analysis;
Rigor: evaluates the credibility of study methods to ensure that they are valid and meaningful;
Relevance: address the relevance of the study to the software industry and the research community.

In this process, all studies were read in their entirety, and a score was assigned at the end to each criterion listed in Table 6. The possible scores were:

[0] The study does not meet the quality criteria;
[1] The study fully meets the quality criteria.

It was also defined following Dyba et al. (2007), that if the primary study did not meet Q1, it would be excluded. Similarly, if Q2, together with Q3, were not completed, the study would be removed. Also, following the recommendations of Lima et al. (2019), a minimum score of 6 points was required for the study to be considered in the final list of this SLR, i.e. it had to achieve more than 50% of the criteria. For this study, the quality assessment was performed by three researchers, who assigned their respective scores to the primary studies, from which, in the end, the arithmetic mean was calculated, as detailed in Section 3.3.

3.2. Conducting

In the second stage of this SLR, the manual search was initially performed by the primary studies through access to the annals of the search sources. In the automatic search, the studies were identified by applying the search string to the digital libraries. The primary search resulted in 6508 primary studies. The inclusion and exclusion criteria were used, resulting in 250 selected studies. These had their titles, abstracts, and keywords analyzed. During the quality assessment process, the resulting studies were fully read to identify which ones met the quality criteria and satisfactorily answered the research questions, resulting in 61 studies.

In the sequence, to complement the evidence already identified and to guarantee the integral inclusion of studies related to this work's objective, the conduction of the snowballing method was included (Wohlin, 2014). At this point, a total of 991 references cited in the 61 primary studies included were analyzed. In the end, 25 studies were selected through the references and, after full reading and quality assessment, five studies were included, resulting in the final version of this SLR (66 studies), as detailed in Fig. 2.

One of the exclusion criteria in this study is to exclude primary studies that are a copy or older version of another study already included in the SLR. Thus, a total of 381 primary studies were excluded because they were duplicates. Specifically, after applying the snowballing method, a total of 213 studies (out of a total of 381) were excluded after the analysis of the references.

After the data extraction phase, the data were extracted, aiming to obtain the information needed to answer the research questions, and a spreadsheet was used for in process. Table 7 presents extracted data from the 66 studies.

The process of interpretation of the results was initiated from the extracted data, elaborating tables, graphs, and networks to present the identified information to answer the research questions. We would like to emphasize that this procedure was performed with the qualitative analysis tool Atlas.ti. The final list of analyzed studies is available in Appendix. Finally, to avoid research bias, the entire process and analysis of SLR were executed, discussed, and reviewed by all the authors of this work.

3.3. Verifiability and replicability

To allow replication and extension of our work by other researchers, we prepared a replication package[1] for this study with the complete results obtained.

4. Results

This section reports the evidence found in the systematic literature review. Sub-questions between the following subsections present the quantitative and qualitative results and their analysis. But initially, an overview of the 66 primary studies analyzed in this systematic review is provided.

4.1. Overview of primary studies

A total of 66 studies were published in the period from 2010 to 2020. The respective search period was considered because Ernst in 2012 offered the first definition of Requirements Technical Debt. In this sense, it became appropriate to analyze primary studies from the last ten years of research in the area. As shown in Fig. 3, only 19 studies were published by 2014, while almost 71% of the studies were published from 2015 on. With this, it is possible to identify that the number of publications and research in the area has increased in recent years. The year 2022 stood out with 12 publications in contrast to 2014 with only two, confirming results of other secondary studies published recently, as the year with fewer publications on the subject of TD (Becker et al., 2018; Lenarduzzi et al., 2021).

When considering the distribution of studies based on location and type of search, a relatively high percentage was identifies in studies (75% or 49 studies) published and attached on a digital basis. While only 25% of studies (17 studies) have been published

[1] http://bit.ly/StudiesPrimary

Table 6
Quality criteria.

Q1: Is the research related to the identification and measurement of TD requirements?	Reporting
Q2: Are the objectives clearly defined?	Reporting
Q3: Is there an adequate description of the context in which the research was carried out?	Reporting
Q4: Is the application domain clearly expressed?	Reporting
Q5: Was the research design appropriate to meet the research objectives?	Rigor
Q6: Was the data analysis sufficiently rigorous?	Rigor
Q7: Is the type of research conducted clearly expressed?	Rigor
Q8: Are the results clearly described?	Credibility
Q9: Is it possible to identify the place of publication of the research?	Credibility
Q10: Is the contribution clearly expressed?	Credibility
Q11: Does the research make it clear who contributes?	Relevance

Table 7
Information extracted from primary studies.

Information	Research question
Title	Overview
Author	Overview
Year of publication	Overview
Place of publication	Overview
Research method	Overview
Cause attributed to the emergence of requirements TD	RQ1
Proposed strategy to identify the requirements TD	RQ2
Proposed strategy to measure the TD of requirements	RQ2
Metric used to measure the TD of requirements	RQ3
Difficulty reported when managing the requirements TD	RQ4

Table 8
Authors with greater representation among primary studies.

Number of primary studies	Author's name
10	Carolyn Seaman
	Rodrigo O. Spínola
8	Manoel Mendonça
7	Antonio Martini
6	Jan Bosch
5	Yuepu Guo
	Terese Besker
4	Forrest Shull
	Alexander
	Chatzigeorgiou
	Nicolli Rios
3	Valentina Lenarduzzi
	Paris Avgeriou
	Nico Zazworka

in journals and manual databases. This analysis can be better visualized previously in Fig. 2.

One of the goals of this work was to analyze studies that addressed the TD of requirements, as well as studies that examined the process of measuring and providing metrics to examine the possibility of using and, and if necessary, adapting these proposed metrics in the context of requirements. It was identified that 48 studies (approximately 72%), were related and provided evidence focused on TD of requirements. At the same time, 18 studies (approximately 28%) were associated with analyzing the process of measuring a technical debt and offering metrics.

Soon after, it was identified that the 66 primary studies selected were written by 130 different authors, showing a broad interest in this subject. However, it was found that only 13 researchers were involved in at least three articles each. In the following sequence, the respective authors are presented in Table 8 in order of representativeness.

Finally, the analysis of P was performed on the research method applied, which was based on the classification presented in the work of Molléri et al. (2019). Regarding the primary studies that applied more than one research method, we prioritized the method they reported as the primary method used to conduct the work. For example, one P conducted a case study, and in a complementary way, interviews were conducted; however, the case study was considered the primary method. Thus, as shown in Fig. 4, the *case study* stood out in a total of 26 publications, following *archival research* (10), which investigates the data through documental analysis and reports, for example. *Thematic analysis* (9), *survey* (7), *design science research* and *interview* (each with four publications), *empirical study* (3), *experiment* (2), and finally *action research* (1).

4.2. RQ1: What has caused the technical debt of requirements in software development?

Second Li et al. (2015b), one of the variables that that help identify a TD, are the provoked that caused its emergence. However, Rios et al. (2018a) reports that this is a topic (causes for TD insertion in projects) that remains little explored in academic research. In this sense, the aim of this question was to determine the main causes of the emergence of TD of requirements, to facilitate its identification and to present the main indicators of its occurrence.

After the analysis, 33 causes (codes) were identified and, in the sequence, the level of grounded (quantity of citations in P) and density (amount of association with other codes, which can be observed in the networks) for each one was verified. For space reasons, the 15 causes of greater representativity considering these two variables are presented in Table 9. The other causes are reported throughout this subsection and, are detailed online at the following link.[2]

In the sequence, as explained above, the analysis of the results was supported by the qualitative tool Atlas.ti. This tool was idealized by Muhr (1991), based on Grounded Theory for its development. Among its many features is the possibility to build states of the art, multimedia analysis of videos, statistical treatment of data, analysis of surveys and, database coding. Because of that, many researchers from different areas have used Atlas.ti in their research.

Among the analysis options that the tool offers, there is the possibility to create networks, which would graphically represent the relationship between the identified codes, and connect those

2 http://bit.ly/DetailingRQ1

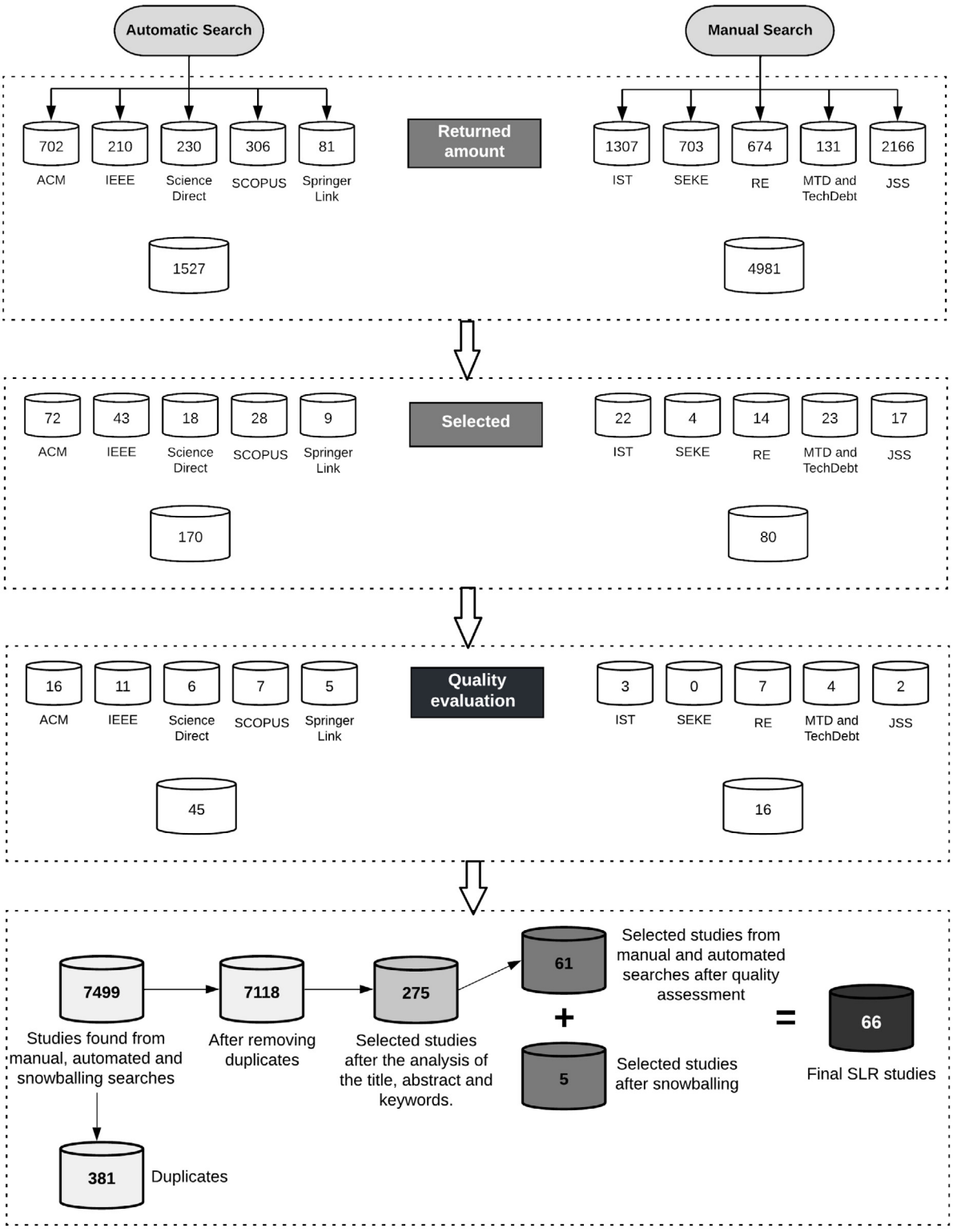

Fig. 2. Process of conducting systematic literature review.

that can cause or influence the existence of others (Figs. 5, 6, 7, and 8). The information illustrated by rectangles with black lines represents the evidence of greater representativeness considering its ground plan and density. Regarding the rectangles with the gray stripes represent the knowledge about the lower representativeness of the P is represented. Thus, four networks were created for this issue, and the 33 causes are illustrated below.

The process of building the networks was based on the evidence provided by the primary studies. For example, in Fig. 5, among the P's, it was found that poorly planned requirements elicitation interviews were related to the lack of a script for their creation, so these two causes were linked. The same process was applied to the other evidence, ultimately resulting in four networks.

Table 9
Leading causes attributed to the emergence of Requirements Technical Debt.

Cause	Grounded	Density	Amount	Primary studies
Low level of detail in requirements documentation	10	3	13	P74, P91, P154, P183, P192, P209,P214, P229, P233, P275
Ambiguous requirements	7	3	10	P101, P125, P134, P135, P247,P249, P263
Non-definition of non-functional requirements	9	1	10	P74, P81, P91, P196, P209, P214,P216, P233, P273
Vague and incomplete requirements	7	2	9	P125, P133, P154, P165, P196,P209, P263
Lack of communication with the customer	6	2	8	P9, P91, P117, P154, P214, P229
Shortcuts and alternative solutions	4	3	7	P11, P13, P111, P117
Schedule pressure	5	1	6	P81, P101, P154, P161, P183
Clients do not reflect what they want	4	2	6	P101, P132, P167, P247
Lack of experience	5	1	6	P8, P167, P183, P229, P249
Inadequate prioritization of requirements	4	1	5	P25, P74, P91, P132
Inadequate writing and grammar	3	1	4	P133, P225, P263
Poorly planned interviews	1	3	4	P167
Lack of a script	1	2	3	P167
Inaccurate or complex requirements	1	1	2	P274
Inadequate elicitation	1	1	2	P25

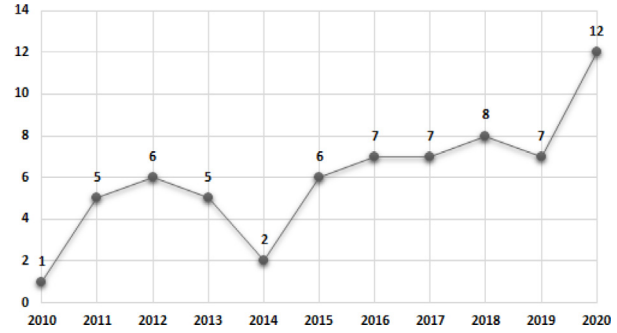

Fig. 3. Publications evolution in the last 10 years.

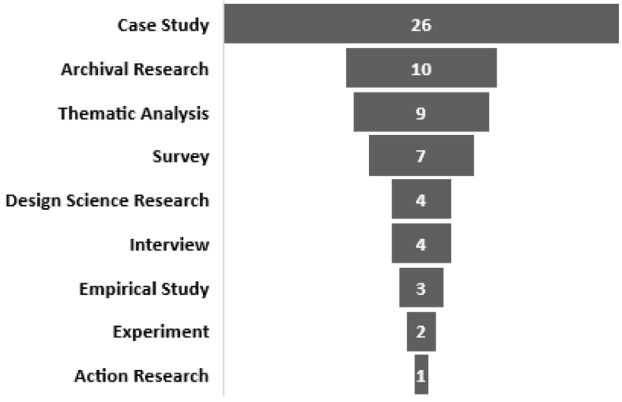

Fig. 4. Applied research method in primary studies.

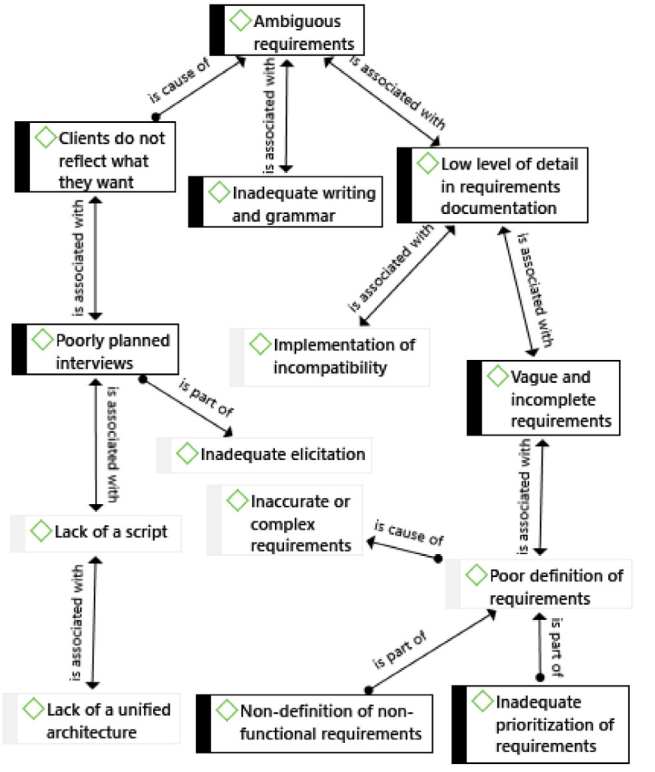

Fig. 5. Association of the causes of the emergence of the TD requirements (Network 1).

As illustrated in Fig. 5, the first network presents 14 causes associated with the emergence of the TD of requirements. It becomes possible to notice, for example, that poorly planned interviews are part of an inadequate elicitation, and they are also associated with the lack of a script. This often causes clients not to reflect what they want, causing ambiguous requirements. Furthermore, it is observed that the absence of details in the documentation may lead to vague and incomplete requirements, resulting from their low definition and prioritization, as in the case of prioritizing requirements that do not offer greater value to the client.

The second network, presented in Fig. 6, is related to the causes attributed to the emergence and identification of intentional technical debt requirement. With this, it is possible to see that it can be caused when professionals and software teams consciously choose to take shortcuts and alternative solutions involving activities related to software requirements. This cause can be directly influenced by schedule pressure and pressure from the client itself, which is considered by Spínola et al. (2013) the root cause of most TD.

The causes associated with the emergence and identification of unintentional Requirements Technical Debt were associated and presented in Fig. 7. By this means, it is possible to see that this TD can be caused by inexperienced professionals and insufficient amount of budget and human resources available in software projects.

Also, unintentional Requirements Technical Debt may be caused by the difficulty of predicting change impacts, i.e., predicting possible future updates or changes in requirements. This cause is associated with existing conflicts between stakeholders, as well as, the volatility of the requirements, i.e., the changes and updates that will occur throughout the project, since in the initial

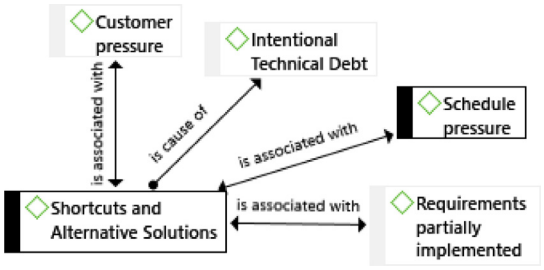

Fig. 6. Association of the causes of the emergence of the intentional requirements TD (Network 2).

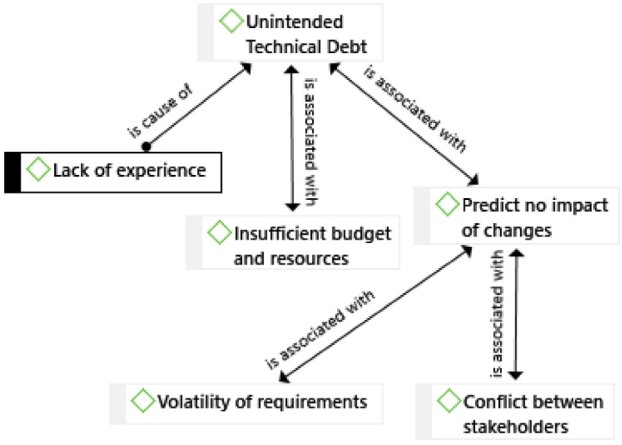

Fig. 7. Association of the causes of the emergence of the unintentional requirements TD (Network 3).

phase the requirements are not precisely defined and specifications are usually not known until the system is implemented.

Finally, the latest causes associated with the emergence of the Requirements Technical Debt are presented in Fig. 8. For example, it is possible to see that the non-validation of requirements is related to the absence of a review of the client's requirements. Sometimes, not enough attention is given to a detailed review of the requirements specification regarding their quality and domain-specific content.

In addition, the non-validation of requirements is mostly originated by a lack of communication with the customer. Thus, this cause can generate outdated requirements, i.e. they refer to cases where they were developed at an appropriate level of quality (in the first versions of the system). Subsequently, the specifications are not updated with new requirements or changes to those already existing.

4.3. RQ2: What strategies are proposed to help identify and measure the Requirements Technical Debt in software projects?

This question objective was to identify and present the strategies that already exist to assist in the identification and measurement of the Requirements Technical Debt. In total, 16 strategies were identified and shown in Table 10, considering representativeness among the primary studies. In the sequence, each strategy is detailed in its applicability.

For this question, it became possible to analyze that part of the strategies, besides identifying and measuring technical debt, it also helped in the other management process steps. Thus, the strategies were separated for further clarification.

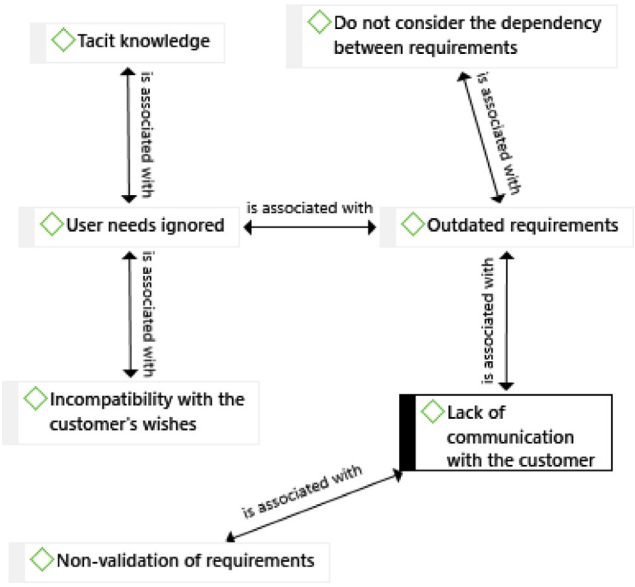

Fig. 8. Association of the causes of the emergence of the TD requirements (Network 4).

4.3.1. Identification and measurement

Customer review: the work's success with quality requirements consists of involving stakeholders progressively, developing lists of sustainable requirements, and recording existing pending issues. Therefore, reviewing the requirements with the client, including the development team, should be considered essential for all management. Companies can adjust the product and specified requirements based on customer feedback to identify TDs more efficiently;

Face-to-face communication: in general, communication about requirements is hierarchical and based on e-mails and documents. However it is recommended that software professionals communicate directly with their peers and stakeholders. Face-to-face communication is efficient in the exchange of information between different stakeholders, helping to identify TDs. Therefore, face-to-face communication should be considered as essential;

Peer review: benefits in reviewing requirements are highlighted in the literature, especially on defect identification and TDs. Among the forms of review, there is peer review. It consists of the analyst conducting the interview with the client and recording the audio of the dialogue. The audio is reviewed by another analyst (reviewer), who writes down ambiguities and lists the questions he would have asked if he had been the analyst. The questions are used for clarification in future interactions with the client;

Simple cost–benefit analysis: a list is created with TD items, where each one represents a task that has been left undone but is at risk of causing future problems. It involves the creation of a hierarchy of criteria (quantitative and qualitative criteria, objective and subjective, which are relevant for the decision), the assignment of weights and scales, and finally a series of comparisons between the items, indicating which should be paid first. Part of these criteria would be the definition of principal, interest, and probability of interest. In the end, for example, a company may decide to approach 75% of the high interest-bearing TDs, 25% of the medium interest-bearing TDs, and defer

Table 10
Strategies used to identify and measure Requirements Technical Debt.

Strategy	Grounded	Density	Amount	Primary studies
Manual management	5	4	9	P72, P110, P125, P177, P196
Automated management	5	2	7	P61, P125, P177, P196
Customer review	4	2	6	P117, P125, P126, P178
Tools and softwares	3	2	5	P61, P77, P177
Merge manual and automatic management	2	2	4	P76, P134
Documentation template	3	1	4	P72, P79, P110
Face-to-face communication	2	1	3	P132, P178
Peer review	2	1	3	P135, P178
Simple cost–benefit analysis	2	0	2	P23, P49
Analytical hierarchy process	2	0	2	P13, P49
Quantification approach	2	0	2	P15, P192
Approach of the nearest neighbor	2	0	2	P4, P99
Cause and effect diagram	2	0	2	P183, P229
Preventive actions	2	0	2	P203, P275
RE-KOMBINE	1	1	2	P25
Backlog	1	0	1	P76
Payment map	1	0	1	P208
Identification through ISO/IEC/IEEE 29148:2018	1	0	1	P263

those with low interest. This strategy is also used to prioritize TD (Lenarduzzi et al., 2021).

Quantification approach: used to quantify TD and technical interest, but for this, it would be necessary to answer questions about investments, such as (1) How big is my TD? (2) How much interest am I paying for the TD? (3) Is the debt growing? and how fast? (4) What will be the consequence of keeping this TD for future maintenance? Finally, the Requirements Technical Debt can be quantified considering the ratio between the user's needs that are already elicited and all possible user needs, including neglected ones;

Approach of the nearest neighbor: this approach leverages the experience gained in previously resolved TDs and relies on the effort required to correct these currently identified debts. The intuition is that the average time it takes to convert a TD of requirements is similar to the correction of previous debts in the project;

Cause and effect diagram: it is used to organize the causes that led to a TDs incidence, helping to identify it quickly, taking into account that the data are previously described;

Preventive actions: This strategy is not necessarily related to identifying and measuring the TD requirements, but it was thought essential to present it. Primary studies report that preventing the occurrence of a TD can be cheaper than its payment. Preventive actions support professionals and software teams in applying acceptable practices that minimize their occurrence. The preventive actions related to TD requirements are: controlling and negotiating the software requirements; well-defined requirements; good communication between stakeholders; well-defined scope statement; requirements change tracking, and customer commitment.

Payment map: software professionals can consult this map to guide their decisions about eliminating TD in their projects. As a guide, the map it can inform a set of practices in response to the need for TD payment. Furthermore, can be used as a communication device to support teams to effectively communicate existing TDs to managers and better decide on the payment of the TD requirements, for example.

Identification through ISO/IEC/IEEE 29148:2018: describes the quality of requirements from specification to other requirements engineering activities in the software development life cycle. It provides details for creating consistent textual requirements, including characteristics and attributes, and language criteria of the requirements. From the moment that these quality standards are violated, requirements smells and technical debt may arise. Analyzing this document according to the specified projects and documented requirements helps to identify inconsistencies, low levels of quality, and violations, and, consequently, identify TD.

Documentation template: it is a model that is filled in the document TD and contains various data, especially concerning the measurement. The TD documentation template proposed by Seaman and Guo (2011) is shown in Table 11.

4.3.2. Management

Manual management: it refers to the process performed manually by software professionals. It would be to identify and measure the TD without the use of tools or software;

Automated management: the automated management uses software and automated resources to identify and measure the TD. When selecting one of these resources, the main question is related to the number of false positives that return: how many TDs are analyzed in more detail and are not necessarily true. When these automated resources produce many false positives, it only distracts the location from the actual technical debts. After reviewing the primary studies, a single tool (RE-KOMBINE) was identified to manage TD in requirements (Ernst, 2012). It is based on the use of objective models used to and follows the software's evolution, identifying changes in requirements and understanding how they impact the current implementation of the software;

Merge manual and automatic management: combining manual management with tool and software analysis is a practical and effective way to identify TDs in industrial projects;

Analytical hierarchy process: this process assigns weights and scales to different criteria used to measure the TD. Then, a series of pairwise comparisons are made between the items to obtain a prioritized TD classification. Based on this process, the items at the top of the list must be treated first. In other words, this strategy focuses on those TD items that have a potentially severe impact on the project regarding the total amount of interest that the project needs to pay. This strategy is not necessarily ideal, but it can decrease the project's risk level and keep the TD under control;

Backlog: often, part of the requirements documentation needs to be updated. However, there are more urgent tasks that need attention. In this case, write down the pending task in a TD list (similar to a daily task list), so that you do not lose sight of the correction that needs to be taken in the future.

Table 11
Technical debt documentation template.

ID	Technical debt identification number
Date	Technical debt identification date
Responsible	Person who identified the TD
Location	Description of where the debt item is
Description	Justification of why that item needs to be considered
Estimated Principal	Work required to pay off the TD
Estimated Interest Amount	Extra effort needed in the future if the TD item is not paid
Estimated Interest Probability	Probability of extra work needed, if the TD is not paid off in the future

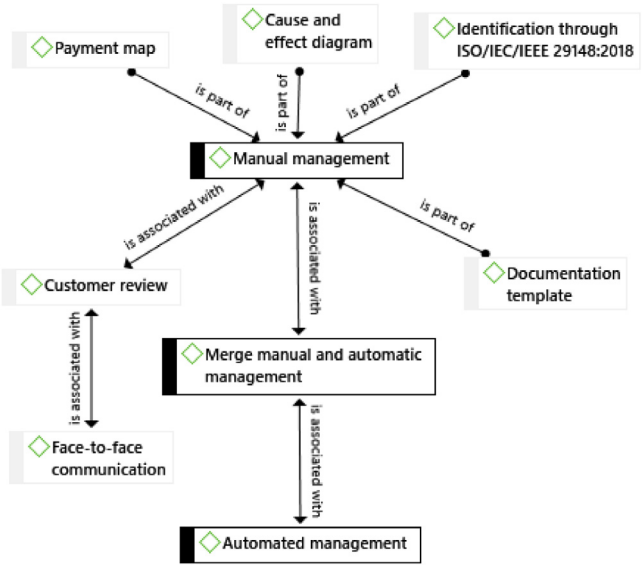

Fig. 9. Association of strategies for identification and measurement of the TD requirements.

Finally, among the 16 strategies, it was analyzed that 9 of them were related, as presented in Fig. 9. It is possible to notice, for example, that the template of documentation can be used during manual management. This type of management is associated with the review process with the client, which is strongly recommended to be performed in person. However, the manual management can also be related to the use of the payment map, the organization of the information and causes of a TD in the cause and effect diagram, and the analysis of the ISO/IEC/IEEE 29148:2018 standards. Additionally, it is strongly recommended to use both management types to ensure an effective TD reduction process.

4.4. RQ3: What metrics are being used to assist in the process of measuring the Requirements Technical Debt?

This question aimed to identify metrics that could help software professionals measure data related to the repayment of Requirements Technical Debt. Initially, to provide a better understanding of the measurement stage of a TD, part of the primary studies presented definitions regarding the variables associated with technical debt that need to be calculated and measured, which are defined in detail in the sequence.

Principal: refers to the effort required to complete a task that has been left unattended. A task is a representation of a technical debt that is at risk of causing future problems if it is not repaid. The principal is calculated according to the number of technical debts that must be corrected in the software, the hours to fix each one, and the labor cost.

Interest: it is the penalty (in terms of more significant effort and lower productivity) that will have to be paid in the future due to the non-correction of technical debts at the present moment of identification. It refers to an estimate of the amount of extra work required to maintain the quality of the software if there is an unpaid technical debt.

Interest Probability: it refers to the likelihood that if the technical debt is not repaid, it will make other works more expensive over some time. The probability of interest is time-sensitive.

Subsequently, among the metric studies, some elements were identified that help measure: (i) the principal of TD; (ii) the interest on TD; (iii) the decision to reimburse TD at the time of identification; (iv) the decision to reimburse TD at the time of identification or at some point in the future; finally (v) the uncertainty about measuring TD. The purpose of this evidence is to support the decision to reimburse TD requirements. Although it is not yet possible to present an accurate value in practice for each variable, these metrics are useful in understanding the factors involved in measuring technical debt, specifically about the likelihood of adaptation and use in the context of software requirements.

4.4.1. Quantifying the principal

Subsequently, the recommendations of Curtis et al. (2012a) (P258), the primary studies P33 and P177 presented a metric to calculate the Principal of the TD, being this a function of three main variables: (i) the number of TD items that should be reimbursed; (ii) the time needed to correct each item; and (iii) the cost to fix each TD item. The following metric is presented in sequence.

$$
\begin{aligned}
&\text{Principal} = \\
&((\sum high - severity\ TD) \times (percentage\ to\ be\ fixed) \times \\
&(average\ hours\ needed\ to\ fix) \times (\$\ per\ hour)) + \\
&((\sum medium - severity\ TD) \times (percentage\ to\ be\ fixed) \times \\
&(average\ hours\ needed\ to\ fix) \times (\$\ per\ hour)) + \\
&((\sum low - severity\ TD) \times (percentage\ to\ be\ fixed) \times \\
&(average\ hours\ needed\ to\ fix) \times (\$\ per\ hour))
\end{aligned}
$$

Following the context of requirements, the number of TD items can be measured through a detailed analysis and review of the requirements specification documentation. However, given certain factors, such as the budget constraints, software companies are rarely able to correct all the TDs projects. Therefore, each debt must be weighted according to its severity level, such as low, medium, and high, to determine the percentage of TDs that will be reimbursed for each level. The severity level refers to the impact that the TD may cause on the project. For example, it analyzes whether the debt is in a strategic area of the software, which in turn negatively impacts the client's business. In other words, a high TD does not necessarily become the most complex debt that will demand more effort to correct. But it may be related to the most critical functionalities or areas in the software.

Soon after, the time to correct a TD includes the time to analyze the debt, understand and determine its correction, assess

the potential side effects, implement and test the correction, and the time to release the correction in operations (Curtis et al., 2012b). Finally, it is worth noting that this metric's variables can be adjusted to reflect better the company's experience and objectives, team, or specific project.

4.4.2. Quantifying the interest

As previously presented, TDs interest is the extra costs that will be spent on maintenance due to quality problems that will arise. In this sense, studies P15 and P43, state that the interest is the difference in maintenance effort between a certain level of quality and the ideal level. To estimate the Maintenance Effort (ME), the following metric is used:

$$ME = \frac{MF \times RV}{QF}$$

The metric above shows that the ME is a function of:

Maintenance Fraction (MF): represents the amount of maintenance effort that will be spent on an annual basis, measured as a percentage of changes involving updating requirements (added, modified, or deleted) annually due to maintenance;

Rebuild Value (RV): is an estimate of the effort (person-months) that needs to be spent on rebuilding software, i.e. correct the existing TDs, determined by the metric:

$$RV = SS \times TF$$

System Size (SS) represents the total size of software measured in lines of code. Alternatively, SS can be measured using functional size (i.e. function points). Technology Factor (TF) represents the language's productivity factor, providing conversion to the effort (i.e., person-month).

Quality Factor (QF): it is a factor used to explain the level of quality of the software. It is assumed that the higher the quality level, the less effort is spent on maintenance. This statement is justified by previous research, which illustrates that making changes in software with superior quality is more efficient (Chidamber et al., 1998). The QF is determined from the following metric:

$$QF = 2^{((QualityLevel-3)\div 2)}$$

According to primary studies, the above metric is a simplified model to consider the quality level (1 to 5 stars). For this purpose, the metric provides the following factors: 0.5, 0.7, 1.0, 1.4, 2.0, based on the work of Bijlsma (2010).

4.4.3. "If" decision

In study P96, a metric was presented to calculate whether it was worth paying back the TD when it was identified by relating the principal to interest. This metric came from the concept of Seaman et al. (2012), which states that the TD should be paid if the principal is lower than the total interest.

$$\frac{CPrincipal}{TInterest} = result$$

CPrincipal is the current principal to be paid, and TInterest is the total interest in the software life cycle. The total interest usually cannot be calculated unless the software life cycle is known, so TInterest is generally considered the interest calculated at a chosen point in the future. In other words, stakeholders need to understand if the repayment cost (principal) is lower than the total interest paid from now until the end of the software life cycle. From the analysis of the result, the following decisions can be considered:

- If the result is less than 1, it means that it is worth paying TD in the present, i.e. the costs will be lower than those which require to be refunded in the future;
- If the result is equal to 1, it means that there is no great loss in putting off the payment of TD. Postponing the reimbursement will not accumulate in great costs;
- If the result is greater than 1, it means that it does not recommend paying TD at the moment it was identified, that is, the costs of reimbursement in the future will be lower.

Finally, the authors pinpoints that choosing a point in the future before the final life cycle of the software is a safe choice. In fact, in the worst case, it is taken into account that the principal will cover only part of the interest. Furthermore, they point out that CPrincipal and TInterest are not described in terms of total costs in dollars, but in a set of factors associated with the reimbursement process of TD requirements, for example. It is assumed that each element may be related to a positive or negative value that may increase or decrease over time.

4.4.4. "When" decision

In primary study P96, a metric was presented to calculate the best time to repay the technical debt, whether in the current period in which it was identified or at a specific point in the future. For example, "should we reimburse now, or can we wait six months?" According to the authors, to answer this question, stakeholders need to know whether repaying at a chosen point in the future (F) is more or less convenient than repaying now. The metric follows in the sequence.

$$\frac{FPrincipal}{(TInterest - FInterest)} - \frac{CPrincipal}{TInterest} = result$$

The metric calculates the ratio between the principal in F and the remaining interest calculated as the total interest (TInterest) minus the interest paid in F. Soon after, it subtracts the proportion calculated about the same balance on the repayment in the current situation. From the analysis of the result, one can consider the following decisions:

- If the first term is greater than the second, the result will be a negative number, which means that it will be less convenient to pay TD later since the gain from repaying the debt about the remaining interest will be smaller than now;
- If the result is low enough (close to 0), it means that repayment is not urgent, since it does not bring many benefits now compared to making it in the future.

4.4.5. Uncertainty of a measurement

According to Curtis et al. (2012a), there is no exact measurement for the TD, since the calculations are based only on structural failures that the organization needs to fix. In this sense, not all organizations reimburse the TDs based on appropriate techniques and metrics. Moreover, small changes in the variables related to a TD can cause large changes in its measurement, thus revealing the final estimates' sensitivity.

Based on this context, the primary study P43, states that calculations involving a TD may suffer systematic errors, caused, for example, by the low measurements of the tools. As a result, the authors define Uncertainty and Error, reporting that random errors or uncertainties in the measurement of a TD are frequent and refer to the delta that exists between the expected value of a measurement and its actual measurement. These errors can overestimate or underestimate the expected value of a measurement.

The primary study P43 used as a basis the operations and theories proposed by Taylor (1997), which states that the correct way to express the result of a measurement is to produce the

Table 12
Difficulties identified in the Requirements Technical Debt management process.

Difficulty	Grounded	Density	Amount	Primary studies
Manage TD efficiently	1	9	10	P38
Lack of access to tools	6	1	7	P76, P117, P154, P158, P178
Understanding that TD is a problem	1	4	5	P117
Measure TD	3	1	4	P4, P117, P146
Allocate more time in eliciting requirements	2	2	4	P25, P178
Engage the team in the TD management process	2	1	3	P49, P117
Balance the benefits of repaying TD with the costs of this process	2	1	3	P8, P132
Manage unintentional TD	2	1	3	P9, P38
Team morale	2	1	3	P119, P228
Adapt the team to the TD management process	1	2	3	P111
Automated management	1	2	3	P61
Urgent management of TD	1	2	3	P24
Customer collaboration in this process	1	1	2	P74
Culture or personal feelings	1	1	2	P50
Manage older TDs	1	1	2	P38
Conflicting goals	1	1	2	P50
Organizational restrictions	1	1	2	P50
Tradition	1	1	2	P50

best estimate of the greatness and the interval within which you are sure the greatness resides. In this sense, the authors of the primary study adapted to the context of TD, presenting the following metric:

$$measured\ value\ of\ TD_{principal} = (TD_{principal})_{best} \pm \delta_{TD}$$

This metric represents the best estimate of a TD, with a margin of error or uncertainty about the principal TD (δ_{TD}). The estimate is between $(TD_{principal})_{best} - \delta_{TD}$ and $(TD_{principal})_{best} + \delta_{TD}$. Following the recommendations of Taylor (1997), for convenience, uncertainty is always defined as positive, so that $(TD_{principal})_{best} + \delta_{TD}$ is always the most likely value of the greatness measured and $(TD_{principal})_{best} - \delta_{TD}$ is the least likely.

Finally, the authors of the primary study conclude that these calculations become appropriate in the measurement process, but that they still need to be validated in the domain of TD, mainly involving the different types of TD, for example, requirements. Furthermore, they reinforce the importance of understanding that the propagation of uncertainty is a critical factor that needs to be investigated to improve the decision-making process about which TD items to refactor.

4.5. RQ4: What difficulties are pointed out during the management of Requirements Technical Debt in software development?

This question's objective was to identify the main difficulties pointed out by the studies in the process of managing the TD requirements. It is intended to present opportunities for new research, future work, and development of new technologies, and assist software professionals in the difficulties reported, shown in Table 12 in order of representativeness. Soon after, two networks were created for this issue in order to relate and associate the 18 difficulties identified.

The first network, illustrated in Fig. 10, presents nine difficulties reported during the management of TDs requirements. It becomes possible to notice, for example, that there is a difficulty in managing the TD efficiently, which causes an increase in maintenance costs at a rate that will eventually exceed the delivery value to the client. This difficulty is associated with the TDs measurement because according to some reports, predicting the debt correction effort is often a more challenging task than predicting the effort to develop the software (Hassouna and Tahvildari, 2010). In addition, it is also related to the difficulty of managing old TD, as the cost of patching increases according to the time it remains in the software in most cases.

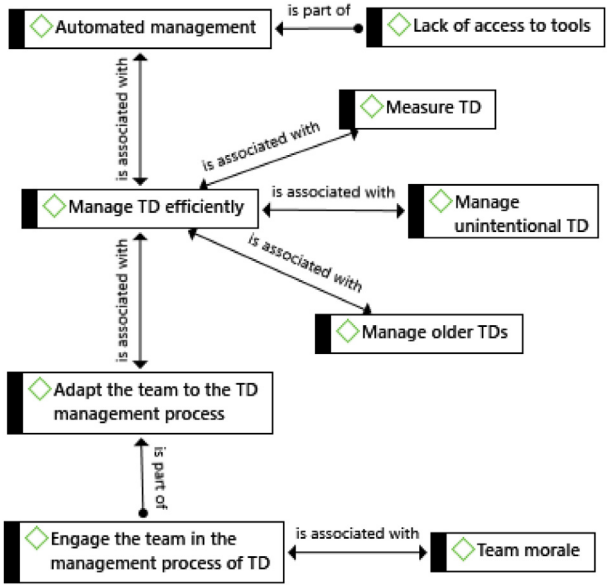

Fig. 10. Difficulties reported in requirements TD management (Network 1).

The following, is still, about efficiently managing the technical debt of requirements efficiently. Report that unintentional TD is much more problematic to manage than intentional TD. Also, performing automated management along with the lack of tools in the context of requirements become difficulties that compromise the quality of correction of the TD, because many projects do not have access to automated tools, apart from inadequate infrastructure that makes teams reluctant to take on correction tasks.

Finally, there is difficulty in adapting the team to the management process of TD, because during the adaptation time the productivity usually drops, since the project or company has to go through a period of learning and education. This difficulty is associated with engaging the team in the management process because instead of only a few people documenting the TD requirements in the backlog, the collaboration of the other members would be necessary. This may be related to the morale of the team because if the TD is not corrected, it may hurt professionals' motivation. Professionals mention that being criticized for introducing a TD can be a bad feeling. Moreover, of improvements

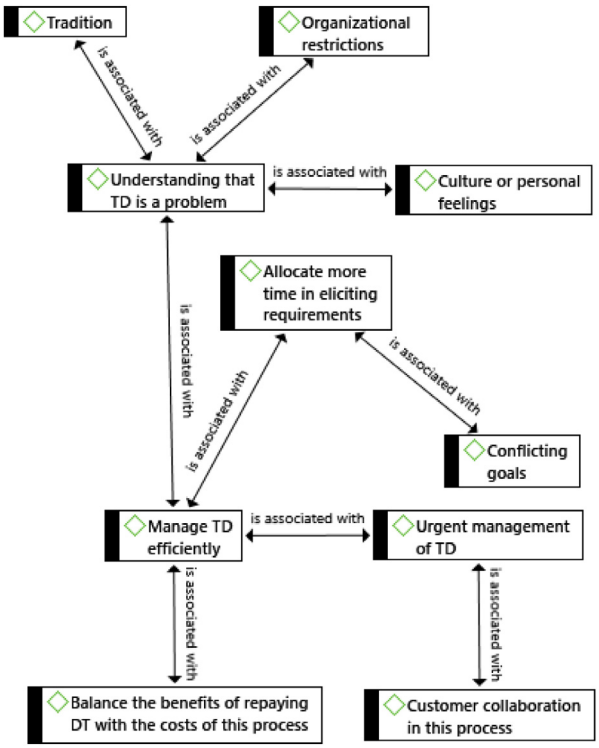

Fig. 11. Difficulties reported in requirements TD management (Network 2).

through technical debt management and to convince managers why it is necessary to do so.

In the second network created for this issue, which is presented in Fig. 11, it is possible to identify that the difficulty in managing the TD of requirements efficiently continues to relate to other codes, such as the urgency of the client's correction. This is associated with a lack of collaboration and communication which are essential elements in requirements engineering. This becomes a challenge to be able to validate requirements that have not been detailed in a satisfactory way, for example.

Soon after, there is the difficulty in balancing the benefits of managing the TD with the costs associated with this process, as well as the efforts spent on replanning the requirements, along with other implementation demands that the development teams have. Furthermore, there is difficulty allocating more time and effort to be paid during the elicitation of the requirements, often caused by conflicting goals, i.e., different objectives to be achieved in that period.

In the sequence, there is the need to understand that the technical debt of requirements if not managed correctly, can become a critical problem to the software project and be the root cause of other issues that arise. This difficulty may be related to the culture of the team or to personal feelings, i.e., employees may prefer to always or never reduce technical debt, depending on their feelings and skills regardless of actual interest. It may also be associated with organizational constraints, i.e., an action is only taken if the organization orders it.

Finally, there is the difficulty of tradition, that is, a particular practice is not changed or is not included in the daily project routine because it deviates from the standard way that the team performs the tasks.

5. Discussion

This systematic literature review aimed to investigate the current state of management research, specifically the identification and measurement of the TD requirements present in software development. To this end, 66 primary studies were analyzed to provide an overview of what has been discussed in the area by analyzing four research questions.

This section presents a summary of the main discussions of the results that indicate their implications for software development industry professionals and researchers.

5.1. Implications for professionals

The results have essential discussions for software development professionals, particularly those seeking guidance, strategies, tools, and information in the literature that can help in certain situations they face in their projects. The results of this SLR imply the following discussions for software development professionals:

(1) When 33 causes that cause the TD of requirements (RQ1) are presented, professionals are invited to a self-analysis about which of these inadequate actions exist in their projects. Furthermore, it assists in the identification of already existing TDs through these listed causes and in the verification of what can be improved to avoid the appearance of this debt;

(2) Among the recorded causes, the low level of detail in requirements documentation is highlighted, which is the cause of the emergence of vague, incomplete and ambiguous requirements. Based on this analysis, it is recommended that industry and professionals pay more attention and effort to the process of specifying and documenting requirements, and provide more explicit follow-up and detailed review when implementing these activities;

(3) It should be noted that part of the causes attributed to the emergence of the TD of requirements is associated with the client and stakeholders' collaboration. It is up to the industry professionals to tighten their communication during the life cycle of the requirement, ensuring its integrity according to what the client wants, avoiding incompatibilities and excessive updates throughout the software development;

(4) Most of the strategies that already help identify and measure the TD requirements (RQ2) are related to manual management. These are strategies that use documentation templates, face-to-face communication, and a payment map of TD, which becomes a challenge to implement in the industry, taking into account that most processes are automated due to agility and implementation time. It is up to the professionals to analyze and adapt the strategies that best fit their needs, infrastructure, and other related factors;

(5) Among the strategies identified, it is noticeable that two of them (Simple cost–benefit analysis and Analytical hierarchy process) can help professionals continue managing the TD requirements. Such strategies can help not only with measurement by assigning weights and scales to the various criteria used in estimates, but also with prioritizing the order of payment of identified items. This happens when making comparisons between the TD items, concentrating the payment on those that have a profound potential impact on the project, for example;

(6) Knowing the causes for the emergence of TD requirements can support software development teams in defining actions that could have been taken to hamper these items of technical debt. Evidence in this context is presented in the strategy of preventive actions reported in RQ2. This information will help professionals use best practices that will help prevent the emergence of this type of TD, and consequently avoid future costs and maintenance efforts.

(7) There is a lack of knowledge on the part of the industry on how to calculate the principal and mainly the interest of TD, which leads to increased costs and decreased quality of the

software, which will possibly impact the clients. The metrics presented (RQ3) become a strategy that allows professionals to support these estimates, avoiding the accumulation of TD caused by inaccurate measurements;

(8) It is possible to notice that there is a lack of practical information in TD requirements is short. To solve this problem and, the software industry can benefit from these contributions, evaluations, tools, and strategies coming from academia, it is necessary that software professionals play an active role in empirical studies, authorize, participate and provide the required data for academic research;

(9) Some of the difficulties reported (RQ4) are associated with measuring the TD of requirements, managing it efficiently, and balancing the benefits with this process's costs. By analyzing this work, professionals will find evidence to help them solve these problems or to help them execute activities through strategies and metrics.

(10) It became possible to analyze that merging the use of the strategies presented with the help of metrics can become an effective practice when measuring a TD of requirements. Four of the strategy, namely, simple cost–benefit analysis; documentation template; quantification approach; analytical hierarchy process, to be applied in the management of a TD, it is necessary to calculate, among its variables, the principal and the interest. For these measurements, metrics were presented throughout this work, thus increasing the assertiveness of the estimates and results when using them.

5.2. Implications for researchers

The results present an active investigation area, but it still requires a deeper analysis, especially to validate the evidence presented in real cases. To guide future research, the results of this SLR imply the following discussions for researchers:

(1) There is an absence of tools and software that can be used in the context of TD requirements. Researchers are encouraged to develop such automated resources that support the process of managing this type of specific TD, especially on its measurement, adding the metrics identified to provide more accurate estimates;

(2) Empirical studies become necessary to provide practical evidence for the application of the proposed metrics in the software industry. Part of the metrics are only mentioned in primary studies but not investigated in real cases. This research is necessary to refine and adapt the metrics in the context of the requirements, so that they can be integrated into the working environment of the companies;

(3) With the exception of RQ3, the other questions presented networks that would represent the relationships between the identified codes (information). According to the reports in the studies, these networks aimed to associate principles that cause or influence the existence of others. As a proposal for future research, it is recommended to confirm, in practice, whether these presented relationships are associated with real projects, or to what extent a cause for the emergence of the TD requirements can influence the existence of another cause, for example;

(4) Most investigations focus on technical aspects of the technical debt management process, but little attention is given to social and interpersonal issues. This is proven in RQ4 when part of the difficulties is related to professionals' culture and personal feelings, team morale, tradition, and organizational constraints. Research on these aspects becomes necessary to help overcome these problems and strengthen the process of managing TD requirements, proposing solutions that go beyond technical aspects;

(5) It becomes necessary, new research to identify technical debt from the analysis of quality standards, norms, and other information that an ISO provides, especially in the context of requirements. Analyze to what level software companies adhere to these quality standards, in which of them major violations occur, and consequently the occurrence of TD as well as the development of a strategy to facilitate the adoption and use of these ISO in everyday business. All these new researches will support the construction of consistent and quality requirements.

(6) Difficulties reported include business resistance to manage TD, engagement of the team in this process, and understanding that requirements debt can become a critical problem if not managed. This proves the lack of research to strengthen collaboration between academia and industry. There is a need for training so that professionals can understand the concept of TD, its consequences and how to differentiate it it from other problems in their projects. Based on this understanding, companies acceptance of managing the problem can be fostered as they understand the real benefits.

(7) Among all the strategies presented in RQ2, only one of them provides a payment map. That is, there is an absence of manuals, guides, and mainly evidence briefings that gather the main information, evidence, strategies, and tools on technical debt, either in a general context or in its specific types, such as requirements. The information presented in this template would facilitate the acceptance and use of industry professionals. Considering the schedule pressure and time to market, for example, providing information that is compiled, illustrated, and quickly understood, would facilitate the integration into the day-to-day of projects.

6. Threats to validity

This section addresses the possible threats that affect the SLR results, and in the sequence, the actions taken to minimize these biases are presented. To analyze these threats, we adopted the guideline proposed by Petersen et al. (2015) (i.e. descriptive validity, theoretical validity, interpretive validity, and repeatability).

Descriptive validity In order to mitigate this threat, the entire selection process and, in particular, the data extraction protocol have been validated by all the authors verifying whether the schema was properly formulated according with the defined research goal and the relative research questions. Moreover, the data extracted from each paper has been checked before answering to the research questions.

Theoretical validity. To identified this threat, we considered entire SLR process. To minimize it, the selection processes were conducted by at least three researchers and all the disagreement were in deep discussed. Moreover, some inclusion and exclusion criteria are refined to make them more objective. Additionally, the search string may not include all terms related to the search topic. However, pilots were previously performed to adjust the presented string. The inclusion and exclusion criteria have been verified by testing with the same bibliographic databases and the same search string.

Interpretive validity. We are aware that the non identification and inclusion of primary studies are correlated, causing the loss of evidence related to the study's objective. In order to mitigate this threat, different data sources were first considered to include the largest possible number of related studies to minimize the mentioned threat. In addition, specific and reputable scholarly sources and digital libraries in the field of computer science and on the topics related to the objective of this work were considered. The search strategies were carefully applied, so that a relevant number of articles were found and consequently included in the study.

Repeatability. The procedure adopted in this study are carefully reported in the paper. Moreover, we include the rawdata with the complete obtained results in our replication package to allow replication and extension of our work by other researchers.

7. Conclusion

When software development stages are pending or not properly executed, they cause a phenomenon known as TD, which if not managed at the time and in the correct way, leads to financial and operational losses. TD can be present in the different phases of the software life cycle. However, specific types of this phenomenon still need to be investigated to assist in its management, for example, the TD of requirements. In this context, this work presents the results of a secondary study that aimed to answer the research question "How to assist in the identification and measurement of the Requirements Technical Debt in software development?".

The focus of the research was to identify the leading causes attributed to the emergence of Requirements Technical Debt, existing strategies that help identify and measure it, metrics that can be used as support during measurement, and difficulties reported in performing these activities. Rigorous steps and detailed analysis of evidence were used to conduct the systematic literature review. 66 primary studies were included through manual and automatic searches, together with the snowballing method. These primary studies were returned from renowned research sources and digital databases and provide an overview and update of the current state of the art over the past 10 years (2010 to 2020).

The results of this systematic review show different contexts that can lead to the emergence of the TD of requirements, involving causes for intentional or unintentional TD, a collaboration of clients and stakeholders, elicitation and documentation of requirements, and pressure of schedule, for example. Addition to these findings, other strategies for identifying and measuring TD of requirements were mapped, emphasizing manual management, with different application proposals. However, the results highlight the lack of automated resources focused on this type of specific TD.

Various metrics have been identified to assist in measuring TD requirements. However, it still requires validation and refinement in the context of requirements. Also, they already provide guidance and insights into the factors involved in measuring TD and how to apply and adapt them in specific contexts. Finally, the main difficulties in performing these activities were found to be precisely related to the measurement stage, in addition to non-technical aspects such as team morale and engagement.

In conclusion, the results provide considerable evidence for the objective of this work. Thus, the main contributions of this work are: (i) the availability of information to help software professionals identify and measure the TD of requirements in their projects; (ii) the broadening of understanding of topics related to TD and its management process, allowing identify how specific contents of this phenomenon can be fragmented; (iii) the availability of metrics that can be used in the measuring a TD; (iv) by identifying the difficulties and gaps that are encountered when performing requirements TD management, become an opportunity for the development of new research; (v) the presentation of findings that go beyond the code TD, for instance, the study of a different type of TD, which provides additional results in the area.

As future proposals, research will be invested based on the gaps identified in this work, adjusting with the evidence already placed and new empirical studies. Also, the development of a guide to support the TD requirements' identification and measurement is currently at an early stage. The guide will consist of evidence from the literature and information gathered through the survey in the software industry. Its objective is to help professionals identify and measure the existing TD requirements in their projects, by knowing and being able to measure the data required for the solution.

CRediT authorship contribution statement

Ana Melo: Conceptualization, Methodology, Data extraction, Data analysis, Writing – original draft, Editing. **Roberta Fagundes:** Data analysis, Review. **Valentina Lenarduzzi:** Methodology, Writing – review & editing. **Wylliams Barbosa Santos:** Supervision, Data analysis, Writing – review.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001, for the financial support for the development of this research.

Appendix. Selected primary studies

[P4] Hassouna, A., & Tahvildari, L. (2010). An effort prediction framework for software defect correction. Information and Software Technology, 52(2), 197–209.

[P8] Guo, Y., & Seaman, C. (2011, May). A portfolio approach to technical debt management. In Proceedings of the 2nd Workshop on Managing Technical Debt (pp. 31–34).

[P9] Klinger, T., Tarr, P., Wagstrom, P., & Williams, C. (2011, May). An enterprise perspective on technical debt. In Proceedings of the 2nd Workshop on managing technical debt (pp. 35–38).

[P11] Theodoropoulos, T., Hofberg, M., & Kern, D. (2011, May). Technical debt from the stakeholder perspective. In Proceedings of the 2nd Workshop on Managing Technical Debt (pp. 43–46).

[P13] Seaman, C., & Guo, Y. (2011). Measuring and monitoring technical debt. In Advances in Computers (Vol. 82, pp. 25–46). Elsevier.

[P15] Nugroho, A., Visser, J., & Kuipers, T. (2011, May). An empirical model of technical debt and interest. In Proceedings of the 2nd workshop on managing technical debt (pp. 1–8).

[P23] Seaman, C., Guo, Y., Zazworka, N., Shull, F., Izurieta, C., Cai, Y., & Vetrò, A. (2012, June). Using technical debt data in decision making: Potential decision approaches. In 2012 Third International Workshop on Managing Technical Debt (MTD) (pp. 45–48). IEEE.

[P24] Snipes, W., Robinson, B., Guo, Y., & Seaman, C. (2012, June). Defining the decision factors for managing defects: A technical debt perspective. In 2012 Third International Workshop on Managing Technical Debt (MTD) (pp. 54–60). IEEE.

[P25] Ernst, N.A. (2012, June). On the role of requirements in understanding and managing technical debt. In 2012 Third International Workshop on Managing Technical Debt (MTD) (pp. 61–64). IEEE.

[P33] Curtis, B., Sappidi, J., & Szynkarski, A. (2012, June). Estimating the size, cost, and types of technical debt. In 2012 Third International Workshop on Managing Technical Debt (MTD) (pp. 49–53). IEEE.

[P34] Letouzey, J.L. (2012, June). The SQALE method for evaluating technical debt. In 2012 Third International Workshop on Managing Technical Debt (MTD). IEEE.

[P38] Spínola, R.O., Vetrò, A., Zazworka, N., Seaman, C., & Shull, F. (2013, May). Investigating technical debt folklore: Shedding some light on technical debt opinion. In 2013 4th International Workshop on Managing Technical Debt (MTD) (pp. 1–7). IEEE.

[P43] Izurieta, C., Griffith, I., Reimanis, D., & Luhr, R. (2013, June). On the uncertainty of technical debt measurements. In 2013 International Conference on Information Science and Applications (ICISA) (pp. 1–4). IEEE.

[P45] Zazworka, N., Spínola, R.O., Vetro', A., Shull, F., & Seaman, C. (2013, April). A case study on effectively identifying technical debt. In Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (pp. 42–47).

[P49] Codabux, Z., & Williams, B. (2013, May). Managing technical debt: An industrial case study. In 2013 4th International Workshop on Managing Technical Debt (MTD) (pp. 8–15). IEEE.

[P50] Falessi, D., Shaw, M.A., Shull, F., Mullen, K., & Keymind, M.S. (2013, May). Practical considerations, challenges, and requirements of tool-support for managing technical debt. In 2013 4th International Workshop on Managing Technical Debt (MTD) (pp. 16–19). IEEE.

[P61] Sneed, H.M. (2014, January). Dealing with Technical Debt in agile development projects. In International Conference on Software Quality (pp. 48–62). Springer, Cham.

[P72] Oliveira, F., Goldman, A., & Santos, V. (2015, August). Managing technical debt in software projects using scrum: An action research. In 2015 Agile Conference (pp. 50–59). IEEE.

[P74] Soares, H.F., Alves, N.S., Mendes, T.S., Mendonça, M., & Spinola, R.O. (2015, April). Investigating the link between user stories and documentation debt on software projects. In 2015 12th International Conference on Information Technology-New Generations (pp. 385–390). IEEE.

[P76] Suryanarayana, G., Samarthyam, G., & Sharma, T. (2015). Repaying Technical Debt in Practice.

[P77] Tools for Repaying Technical Debt

[P79] Li, Z., Liang, P., & Avgeriou, P. (2015, May). Architectural technical debt identification based on architecture decisions and change scenarios. In 2015 12th Working IEEE/IFIP Conference on Software Architecture (pp. 65–74). IEEE.

[P81] Abad, Z.S.H., & Ruhe, G. (2015, August). Using real options to manage technical debt in requirements engineering. In 2015 IEEE 23rd International Requirements Engineering Conference (RE) (pp. 230–235). IEEE.

[P91] Mendes, T.S., de F. Farias, M.A., Mendonça, M., Soares, H.F., Kalinowski, M., & Spínola, R.O. (2016, April). Impacts of agile requirements documentation debt on software projects: a retrospective study. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 1290–1295).

[P96] Martini, A., & Bosch, J. (2016, May). An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt. In 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C) (pp. 31–40). IEEE.

[P99] Akbarinasaji, S., Bener, A.B., & Erdem, A. (2016, May). Measuring the principal of defect debt. In Proceedings of the 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (pp. 1–7).

[P101] Ghanbari, H. (2016, January). Seeking technical debt in critical software development projects: An exploratory field study. In 2016 49th Hawaii International Conference on System Sciences (HICSS) (pp. 5407–5416). IEEE.

[P110] Guo, Y., Spínola, R.O., & Seaman, C. (2016). Exploring the costs of technical debt management–a case study. Empirical Software Engineering, 21(1), 159–182.

[P111] Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016). The Effects of Software Process Evolution to Technical Debt—Perceptions from Three Large Software Projects. In Managing Software Process Evolution (pp. 305–327). Springer, Cham.

[P117] Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016). How do software development teams manage technical debt?–An empirical study. Journal of Systems and Software, 120, 195–218.

[P119] Ghanbari, H., Besker, T., Martini, A., & Bosch, J. Looking for Peace of Mind? Manage your (Technical) Debt.

[P125] Femmer, H., Fernández, D.M., Wagner, S., & Eder, S. (2017). Rapid quality assurance with requirements smells. Journal of Systems and Software, 123, 190–213.

[P126] Mohagheghi, P., & Aparicio, M.E. (2017). An industry experience report on managing product quality requirements in a large organization. Information and Software Technology, 88, 96–109.

[P132] Heikkilä, V.T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empirical Software Engineering, 22(6), 2892–2936.

[P133] Beer, A., Junker, M., Femmer, H., & Felderer, M. (2017, September). Initial investigations on the influence of requirement smells on test-case design. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 323–326). IEEE.

[P134] Femmer, H., Unterkalmsteiner, M., & Gorschek, T. (2017, September). Which requirements artifact quality defects are automatically detectable? A case study. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 400–406). IEEE.

[P135] Ferrari, A., Spoletini, P., Donati, B., Zowghi, D., & Gnesi, S. (2017, September). Interview review: detecting latent ambiguities to improve the requirements elicitation process. In 2017 IEEE 25th International Requirements Engineering Conference (RE) (pp. 400–405). IEEE.

[P146] Ampatzoglou, A., Michailidis, A., Sarikyriakidis, C., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2018, May). A framework for managing interest in technical debt: an industrial validation. In Proceedings of the 2018 International Conference on Technical Debt (pp. 115–124).

[P154] Charalampidou, S., Ampatzoglou, A., Chatzigeorgiou, A., & Tsiridis, N. (2018, August). Integrating traceability within the ide to prevent requirements documentation debt. In 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 421–428). IEEE.

[P158] Tsoukalas, D., Siavvas, M., Jankovic, M., Kehagias, D., Chatzigeorgiou, A., & Tzovaras, D. (2018, September). Methods and Tools for TD Estimation and Forecasting: A State-of-the-art Survey. In 2018 International Conference on intelligent systems (IS) (pp. 698–705). IEEE.

[P161] Conejero, J.M., Rodríguez-Echeverría, R., Hernández, J., Clemente, P.J., Ortiz-Caraballo, C., Jurado, E., & Sánchez -Figueroa, F. (2018). Early evaluation of technical debt impact on maintainability. Journal of Systems and Software, 142, 92–114.

[P165] de O. Passos, A.F., de Freitas Farias, M.A., de Mendonça Neto, M.G., & Spínola, R.O. (2018, October). A Study on Identification of Documentation and Requirement Technical Debt through Code Comment Analysis. In Proceedings of the 17th Brazilian Symposium on Software Quality.

[P167] Bano, M., Zowghi, D., Ferrari, A., Spoletini, P., & Donati, B. (2018, August). Learning from mistakes: An empirical study of elicitation interviews performed by novices. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 182–193). IEEE.

[P171] Martini, A., Sikander, E., & Madlani, N. (2018). A semi-automated framework for the identification and estimation of architectural technical debt: A comparative case-study on the modularization of a software component. Information and Software Technology, 93, 264–279.

[P175] Conejero, J.M., Rodríguez-Echeverría, R., Hernández, J., Clemente, P.J., Ortiz-Caraballo, C., Jurado, E., & Sánchez-Figueroa, F. (2018). Early evaluation of technical debt impact on maintainability. Journal of Systems and Software, 142, 92–114.

[P177] Kouros, P., Chaikalis, T., Arvanitou, E.M., Chatzigeorgiou, A., Ampatzoglou, A., & Amanatidis, T. (2019, April). Jcaliper: search-based technical debt management. In Proceedings of the 34th ACM/SIGAPP Symposium on applied computing (pp. 1721–1730).

[P178] Rindell, K., Bernsmed, K., & Jaatun, M.G. (2019, August). Managing Security in Software: Or: How I Learned to Stop Worrying and Manage the Security Technical Debt. In Proceedings of the 14th International Conference on Availability, Reliability and Security (pp. 1–8).

[P183] Rios, N., Spínola, R.O., Mendonça, M., & Seaman, C. (2019, May). Supporting analysis of technical debt causes and effects with cross-company probabilistic cause–effect diagrams. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt) (pp. 3–12). IEEE.

[P192] Lenarduzzi, V., & Fucci, D. (2019, September). Towards a holistic definition of requirements debt. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1–5). IEEE.

[P196] Mendes, T.S., Gomes, F.G., Gonçalves, D.P., Mendonça, M.G., Novais, R.L., & Spínola, R.O. (2019). VisminerTD: a tool for automatic identification and interactive monitoring of the evolution of technical debt items. Journal of the Brazilian Computer Society, 25(1), 1–28.

[P203] Freire, S., Rios, N., Mendonça, M., Falessi, D., Seaman, C., Izurieta, C., & Spínola, R.O. (2020, March). Actions and impediments for technical debt prevention: results from a global family of industrial surveys. In Proceedings of the 35th Annual ACM Symposium on Applied Computing.

[P208] Freire, S., Rios, N., Gutierrez, B., Torres, D., Mendonça, M., Izurieta, C., …& Spínola, R. O. (2020). Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items. In Proceedings of the Evaluation and Assessment in Software Engineering (pp. 210–219).

[P209] Behutiye, W., Seppänen, P., Rodríguez, P., & Oivo, M. (2020). Documentation of quality requirements in agile software development. In Proceedings of the Evaluation and Assessment in Software Engineering (pp. 250–259).

[P214] Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., & Abhervé, A. (2020, August). How agile software development practitioners perceive the need for documenting quality requirements: a multiple case study. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 93–100). IEEE.

[P216] Li, Y., Soliman, M., & Avgeriou, P. (2020, August). Identification and Remediation of Self-Admitted Technical Debt in Issue Trackers. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 495–503). IEEE.

[P225] Lenarduzzi, V., Fucci, D., & Mendéz, D. (2020). On the perceived harmfulness of requirement smells: An empirical study. In Joint 26th International Conference on Requirements Engineering: Foundation for Software Quality Workshops, Doctoral Symposium, Live Studies Track, and Poster Track, Pisa; Italy, 24 March 2020 through 27 March 2020 (Vol. 2584).

[P228] Besker, T., Ghanbari, H., Martini, A., & Bosch, J. (2020). The influence of Technical Debt on software developer morale. Journal of Systems and Software, 167, 110586.

[P229] Rios, N., Spínola, R.O., Mendonça, M., & Seaman, C. (2020). The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. Empirical Software Engineering, 25(5).

[P233] de Freitas Farias, M.A., de Mendonça Neto, M.G., Kalinowski, M., & Spínola, R.O. (2020). Identifying self-admitted technical debt through code comment analysis with a contextualized vocabulary. Information and Software Technology, 121, 106270.

[P247] Dar, H.S. (2020, August). Reducing Ambiguity in Requirements Elicitation via Gamification. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 440–444). IEEE.

[P249] Panis, M.C. (2020, August). An Analysis of Requirements-Related Problems that Occurred in an organization Using a Mature Requirements Engineering Process. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 291–299). IEEE.

[P258] Curtis, B., Sappidi, J., & Szynkarski, A. (2012). Estimating the principal of an application's technical debt. IEEE software, 29(6), 34–42.

[P263] Femmer, H., Fernández, D.M., Juergens, E., Klose, M., Zimmer, I., & Zimmer, J. (2014, June). Rapid requirements checks with requirements smells: Two case studies. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (pp. 10–19).

[P273] Robiolo, G., Scott, E., Matalonga, S., & Felderer, M. (2019, November). Technical debt and waste in non-functional requirements documentation: An exploratory study. In International Conference on Product-Focused Software Process Improvement (pp. 220–235). Springer, Cham.

[P274] Lenarduzzi, V., Orava, T., Saarimäki, N., Systa, K., & Taibi, D. (2019, September). An empirical study on technical debt in a finnish sme. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1–6). IEEE.

[P275] Rios, N., Mendes, L., Cerdeiral, C., Magalhães, A.P.F., Perez, B., Correal, D., …& Spínola, R.O. (2020, March). Hearing the voice of software practitioners on causes, effects, and practices to Deal with documentation debt. In International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, Cham.

References

Abad, Z.S.H., Ruhe, G., 2015. Using real options to manage technical debt in requirements engineering. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE). IEEE, pp. 230–235.

Alves, N.S., Mendes, T.S., de Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C., 2016. Identification and management of technical debt: A systematic mapping study. Inf. Softw. Technol. 70.

Alves, M., Nunes, Gava, V., Luiz, 2018. Uma proposta para identificar, medir e gerenciar dívida técnica em requisitos de software. In: International Conference on Information Systems and Technology Management.

Alves, N.S., Ribeiro, L.F., Caires, V., Mendes, T.S., Spínola, R.O., 2014. Towards an ontology of terms on technical debt. In: 2014 Sixth International Workshop on Managing Technical Debt. IEEE, pp. 1–7.

Becker, C., Chitchyan, R., Betz, S., McCord, C., 2018. Trade-off decisions across time in technical debt management: a systematic literature review. In: Proceedings of the 2018 International Conference on Technical Debt.

Behutiye, W.N., Rodríguez, P., Oivo, M., Tosun, A., 2017. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Inf. Softw. Technol. 82, 139–158.

Benldris, M., 2020. Investigate, identify and estimate the technical debt: a systematic mapping study. Available at SSRN 3606172.

Besker, T., Martini, A., Bosch, J., 2018. Technical debt cripples software developer productivity.

Besker, T., Martini, A., Bosch, J., 2019. Software developer productivity loss due to technical debt—a replication and extension study examining developers' development work. J. Syst. Softw. 156, 41–61.

Bijlsma, D., 2010. Indicators of issue handling efficiency and their relation to software maintainability. (Ph.D. thesis). Master's thesis, University of Amsterdam.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., et al., 2010. Managing technical debt in software-reliant systems. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. ACM, pp. 47–52.

Chidamber, S.R., Darcy, D.P., Kemerer, C.F., 1998. Managerial use of metrics for object-oriented software: An exploratory analysis. IEEE Trans. Softw. Eng. 24 (8), 629–639.

Cunningham, W., 1992. The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger 4 (2), 29–30.

Curtis, B., Sappidi, J., Szynkarski, A., 2012a. Estimating the principal of an application's technical debt. IEEE Softw. 29 (6), 34–42.

Curtis, B., Sappidi, J., Szynkarski, A., 2012b. Estimating the size, cost, and types of technical debt. In: 2012 Third International Workshop on Managing Technical Debt (MTD). IEEE, pp. 49–53.

da Silva Maldonado, E., Shihab, E., Tsantalis, N., 2017. Using natural language processing to automatically detect self-admitted technical debt. IEEE Trans. Softw. Eng. 43 (11), 1044–1062.

de Melo, A.C.C., Fagundes, R., Lima, J.V.V., Alencar, F., Santos, W., 2021. Identificaçao e Mensuraçao da Dıvida Técnica de Requisitos: um survey na indústria de software. In: WER.

Dyba, T., Dingsoyr, T., Hanssen, G.K., 2007. Applying systematic reviews to diverse study types: An experience report. In: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). IEEE, pp. 225–234.

Ernst, N.A., 2012. On the role of requirements in understanding and managing technical debt. In: 2012 Third International Workshop on Managing Technical Debt (MTD). IEEE, pp. 61–64.

Gama, E., Paixao, M., Freire, E.S.S., Cortés, M.I., 2019. Technical debt's state of practice on stack overflow: a preliminary study. In: Proceedings of the XVIII Brazilian Symposium on Software Quality, pp. 228–233.

Guo, Y., Spínola, R.O., Seaman, C., 2016. Exploring the costs of technical debt management–a case study. Empir. Softw. Eng. 21 (1).

Hassouna, A., Tahvildari, L., 2010. An effort prediction framework for software defect correction. Inf. Softw. Technol. 52 (2).

Kitchenham, B.A., Budgen, D., Brereton, O.P., 2011. Using mapping studies as the basis for further research–a participant-observer case study. Inf. Softw. Technol. 53 (6), 638–651.

Kitchenham, B., Charters, S., 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Citeseer.

Kruchten, P., Nord, R.L., Ozkaya, I., 2012. Technical debt: From metaphor to theory and practice. Ieee Softw. 29 (6), 18–21.

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., Fontana, F.A., 2021. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools. J. Syst. Softw. 171.

Lenarduzzi, V., Fucci, D., 2019. Towards a holistic definition of requirements debt. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, pp. 1–5.

Li, Z., Avgeriou, P., Liang, P., 2015a. A systematic mapping study on technical debt and its management. J. Syst. Softw. 101.

Li, Z., Liang, P., Avgeriou, P., 2014. Architectural debt management in value-oriented architecting. In: Economics-Driven Software Architecture. Elsevier, pp. 183–204.

Li, Z., Liang, P., Avgeriou, P., 2015b. Architectural technical debt identification based on architecture decisions and change scenarios. In: 2015 12th Working IEEE/IFIP Conference on Software Architecture. IEEE.

Lima, J.V., Júnior, M.d.M.A., Moya, A., Almeida, R., Anjos, P., Lencastre, M., Fagundes, R.A.d.A.F., Alencar, F., 2019. As metodologias ativas e o ensino em engenharia de software: uma revisão sistemática da literatura. In: Anais Do Workshop de Informática Na Escola. 25, (1), p. 1014.

McConnell, S., 2008. Managing technical debt. Construx Software Builders, Inc 1–14.

Molléri, J.S., Petersen, K., Mendes, E., 2019. Cerse-catalog for empirical research in software engineering: A systematic mapping study. Inf. Softw. Technol. 105, 117–149.

Muhr, T., 1991. Atlas/ti—A prototype for the support of text interpretation. Qualitative Sociol. 14 (4), 349–371.

Nascimento, R., Aranha, E., Kulesza, U., Lucena, M., 2018. Requirements Smells como indicadores de má qualidade na especificação de requisitos: Um Mapeamento Sistemático da Literatura. In: WER.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. 64, 1–18.

Riaz, M., Mendes, E., Tempero, E., 2009. A systematic review of software maintainability prediction and metrics. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE.

Rios, N., de Mendonça Neto, M.G., Spínola, R.O., 2018a. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Inf. Softw. Technol. 102, 117–145.

Rios, N., Spínola, R.O., Mendonça, M., Seaman, C., 2018b. The most common causes and effects of technical debt: first results from a global family of industrial surveys. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 1–10.

Rios, N., Spínola, R.O., Mendonça, M., Seaman, C., 2019. Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. In: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt).

Saha, S.K., Selvi, M., Büyükcan, G., Mohymen, M., 2012. A systematic review on creativity techniques for requirements engineering. In: 2012 International Conference on Informatics, Electronics & Vision (ICIEV). IEEE, pp. 34–39.

Seaman, C., Guo, Y., 2011. Measuring and monitoring technical debt. In: Advances in Computers. Vol. 82, Elsevier, pp. 25–46.
Seaman, C., Guo, Y., Zazworka, N., Shull, F., Izurieta, C., Cai, Y., Vetrò, A., 2012. Using technical debt data in decision making: Potential decision approaches. In: 2012 Third International Workshop on Managing Technical Debt (MTD). IEEE, pp. 45–48.
Spínola, R.O., Vetrò, A., Zazworka, N., Seaman, C., Shull, F., 2013. Investigating technical debt folklore: Shedding some light on technical debt opinion. In: 2013 4th International Workshop on Managing Technical Debt (MTD). IEEE, pp. 1–7.
Taylor, J.R., 1997. An introduction to error analysis, 327 pp. Univ. Sci. Books, Sausalito, Calif.
Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86 (6).
Tonin, G.S., 2018. Technical debt management in the context of agile methods in software development. (Ph.D. thesis). Universidade de São Paulo.
Tonin, G.S., Goldman, A., Seaman, C., Pina, D., 2017. Effects of technical debt awareness: A classroom study. In: International Conference on Agile Software Development. Springer, Cham, pp. 84–100.
Van Vliet, H., Van Vliet, H., Van Vliet, J., 2008. Software Engineering: Principles and Practice. Vol. 13, Citeseer.
Vazquez, C.E., Simões, G.S., 2016. Engenharia de Requisitos: Software Orientado Ao Negócio. Brasport.
Wang, Q., Huang, Y., 2020. Identification and management of requirements debt: Systematic mapping study and survey.
Wiegers, K., Beatty, J., 2013. Software Requirements. Pearson Education.
Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10.
Yli-Huumo, J., Maglyas, A., Smolander, K., 2016. How do software development teams manage technical debt?–an empirical study. J. Syst. Softw. 120, 195–218.

Ana Melo is a Master's student in the Post Graduate Program in Computer Engineering (PPGEC) at the University of Pernambuco (UPE), where she researches technical debt in software development. Bachelor's degree in Information Systems (2019) from the Federal Rural University of Pernambuco (UFRPE). Member of the REACT Research Labs Research Group. His research areas of interest include software development, technical debt, and requirements engineering. Contact her accm@ecomp.poli.br.

Roberta Fagundes has a Post-Doctorate in Statistics (2015) from the Federal University of Pernambuco (UFPE), Brazil. She also holds a doctorate (2013) and a master's degree (2006) in Computer Science from UFPE. Graduated in Telematics Technology (2002) from the Federal Center for Technological Education of Paraíba (CEFET-PB). She is currently an Adjunct Professor at the University of Pernambuco (2007) in Information Systems and Computer Engineering at the University of Pernambuco (UPE), Brazil. She is also a vice-coordinator and professor of the Graduate Program in Computer Engineering (PPGEC), where there are Masters and Doctorate courses. She is interested in researching in research in the area of Computer Science, with emphasis on Computational Intelligence. Contact her roberta.fagundes@upe.br.

Valentina Lenarduzzi is an assistant professor (tenure track) at University of Oulu (Finland). Her research activities are related to modern software development practices and methodologies, including data analysis in software engineering, software quality, software maintenance and evolution, focusing on Technical Debt as well as code and architectural smells. She got the Ph.D. in Computer Science in 2015 and was a postdoctoral researcher at the Free University of Bozen-Bolzano, (Italy), at the Tampere University (Finland), and at LUT University (Finland). Moreover, she was visiting researcher at the University of Kaiserslautern (TUK) and the Fraunhofer Institute for Experimental Software Engineering IESE (Germany). She served as a program committee member of various international conferences (e.g., ICPC, ICSME, ESEM), and for various international journals (e.g., TSE, EMSE, JSS, IST) in the field of software engineering. She has been program co-chair of OSS 2021 and TechDebt 2022. She was also one of the organizer of the last edition of MaLTeSQuE workshop (2022) collocated with ESEC/FSE. Dr. Lenarduzzi is recognized by the Journal of Systems and Software (JSS) as one of the most active SE researcher in top-quality journals in the period 2013 to 2020. Contact her valentina.lenarduzzi@oulu.fi

Wylliams Santos is adjunct professor at the University of Pernambuco (UPE), Brazil, where he leads the REACT Research Labs. Ph.D. in Computer Science (2018), Informatics Center (CIn) at Federal University of Pernambuco (UFPE), Brazil. M.Sc. in Computer Science (2011), Informatics Center at Federal University of Pernambuco, Brazil. He undertook his sandwich Ph.D. (2015–2016) research at the Department of Computer Science and Information Systems (CSIS) of the University of Limerick, Ireland and in collaboration with Lero — the Irish Software Research Centre. His research areas of interest include management of software projects, agile software development, technical debt and industry-academia collaboration. Contact her wbs@upe.br.