Journal of Software Engineering Research and Development, 2023, 11:8, doi: 10.5753/jserd.2023.2671

Identification and Management of Technical Debt: A Systematic Mapping Study Update

María Isabel Murillo [University of Costa Rica | maria.murilloquintana@ucr.ac.cr]
Gustavo López [University of Costa Rica | gustavo.lopezherrera@ucr.ac.cr]
Rodrigo Spínola [Salvador University | rodrigo.spinola@unifacs.br]
Julio Guzmán [University of Costa Rica | julio.guzman@ucr.ac.cr]
Nicolli Rios [Federal University of Rio de Janeiro | nicolli@cos.ufrj.br]
Alexia Pacheco [University of Costa Rica | alexia.pacheco@ucr.ac.cr]

Abstract

Technical debt is a concept used to describe the lack of good practices during software development, leading to several problems and costs. Identification and management strategies can help reduce these difficulties. In a previous study, Alves et al. (2016) analyzed the research landscape of such strategies from 2010 to 2014. This paper replicates and updates their study to explore the evolution of technical debt identification and management research landscape over a decade, including literature from 2010 until 2022. We analyzed 117 papers from the ACM Digital Library, IEEE Xplore, Science Direct, and Springer Link. Newly suggested strategies include automatically identifying admitted debt in comments, commits, and source code. Between 2015 and 2022, more empirical evaluations have been performed, and the general research focus has changed to a more holistic approach. Therefore, the research area evolved and reached a new level of maturity compared to previous results from Alves et al. (2016). Not only are code aspects considered for technical debt, but other aspects have also been investigated (e.g., models for the development process).

Keywords: *Technical Debt Management, Technical Debt Identification, Software Development Process.*

1 Introduction

Technical debt (TD) is the consequence of taking shortcuts during the software development process, providing short-term benefits but potentially bringing more difficulties and costs in later stages (Izurieta et al., 2012). When developers take these shortcuts, deficiencies may be inserted. The cost of fixing previous work increases as the development continues because correcting the defects becomes more complex when technical debt is not timely paid (Akbarinasaji & Bener, 2016)

The interest is the additional cost that may have to be assumed because of the delayed payment. On the other hand, the principal is the amount over which interests are paid. In technical debt, the principal is the original cost of fixing the software (Ampatzoglou et al., 2015). When developers cannot pay the existing technical debt, bankruptcy may occur (Akbarinasaji & Bener, 2016)

Several activities can help manage debt during the software development process. Management activities may include measuring, prioritizing, preventing, monitoring, documenting, communicating, and paying the debt. (Li et al., 2015). The purpose of performing these actions is to avoid major problems that may lead to significant consequences, such as the failure of software projects.

Management strategies can help determine the appropriate time to pay the debt before interests become very costly. Consequently, it is possible to make faster deliveries in a controlled manner (Freire et al., 2020). Also, strategies allow even to recognize if the debt needs to be paid because there may be cases when there is no need to pay it, for example, when there is the certainty that a module will not change in the future (Guo et al., 2014).

Technical debt management is complex because there may be uncertainty during software development. Also, many factors must be considered for its management, such as the present and future costs, as well as the risks that are implied (Guo et al., 2014).

The identification of TD comprises the activities or actions taken to detect the presence of debt in software artifacts. Technical debt identification is the first step that needs to be taken to start managing it and avoid its possible high costs (Guo et al., 2014). For instance, TD identification is essential to prevent unwanted consequences of debt.

Alves et al. (2016) investigated the technical debt identification and management landscape between 2010 and 2014 by analyzing 100 primary studies. They found that strategies mainly addressed types of technical debt associated with source code. Nevertheless, few empirical evaluations demonstrated the proposals' actual benefits, limitations, and applicability. Alves et al. (2016) also presented an initial taxonomy of technical debt types and a list of indicators for their identification. In that study, TD management was understood as the activities that follow its identification. The findings of Alves et al. (2016) provided valuable contributions for both researchers and practitioners, while they also characterized the state of the art in the research area.

In this paper, we update the mapping study of Alves et al. (2016) to find proposals done between 2015 and 2022 about managing and identifying technical debt. Additionally, this paper provides a comparison with the previous results obtained by Alves et al. (2016) and an analysis of the research landscape comprising more than a decade. Keeping the results updated is essential because it helps to understand the

evolution of the research topic and new findings (Nepomuceno & Soares, 2019).

We consider the application of the same research questions, search string, search strategy, sources, inclusion, and exclusion criteria as an update of the previous systematic mapping. We intend to answer the same research questions (without adaptations) since changing them could be considered a new mapping instead of an update (Nepomuceno & Soares, 2019). Likewise, we considered TD management as the activities following debt identification to be conceptually consistent. The main difference in the protocol was the time-frame delimitation since our update only included papers published after the original study's year of inclusion. We also provide a more detailed definition than the original study of what we considered as "general" technical debt papers for their classification. Furthermore, two original authors assisted in the update process to ensure the compatibility between the concepts, procedure, and results of both systematic studies.

The high-level research question we aim to answer is:

- What strategies have been proposed to identify or manage technical debt in software projects?

Similarly, the complementary research questions are:

- **RQ1**. What are the types of technical debt found in the literature?
- **RQ2**. What are the strategies proposed to identify technical debt?
  - **RQ2.1**. Which empirical evaluations have been performed?
  - **RQ2.2**. Which artifacts and data sources have been proposed to identify technical debt?
  - **RQ2.3**. Which software visualization techniques have been proposed to identify technical debt?
- **RQ3.** What strategies have been proposed for the management of technical debt?
  - **RQ3.1**. Which empirical evaluations have been performed?
  - **RQ3.2**. Which software visualization techniques have been proposed to manage technical debt?

We analyzed 117 primary studies and identified new proposals and indicators between 2015 and 2022. Empirical evaluations of the analyzed papers include case studies, controlled experiments, and action research, but more evaluations are still required. Also, we found that technical debt visualization is yet an area that researchers have not extensively studied. This is a relevant finding since visualization techniques may be useful to aid decision-making for TD.

This paper's results benefit researchers since we provide knowledge about state-of-the-art and open problems that are future research opportunities. It is also helpful to practitioners since we present identification, management, and visualization strategies applicable to software projects to prevent technical debt unwanted negative consequences.

Future research opportunities include investigating new ways to use developers' knowledge about debt (not only through commits and comments) and exploring new strategies with a less-technical approach (such as incentives and TD guilds). Moreover, analyzing the applicability of strategies in different contexts, such as public or private organizations, is a future research opportunity.

The structure of this article is as follows. Section 2 describes previous literature related to this work. Section 3 presents the methodology used to perform the literature review. Section 4 presents the results obtained. Section 5 includes the discussion, Section 6 the threats to validity, and Section 7 covers the conclusions.

2 Related work

This section presents several previous works performed by other authors, particularly those that have addressed the identification and management of technical debt during software development. **Table 1** gives an overview of each authors' contributions.

Rios et al. performed a tertiary study to identify the state of the art regarding technical debt between 2012 and 2018 (Rios et al., 2018). The authors studied the understanding of the technical debt concept and the research efforts on its identification and management. They found nine secondary studies about technical debt management and two regarding its identification. Until 2018 there was little knowledge about the benefits and limitations of the proposed management strategies and indicators.

Another systematic mapping studied the concept of technical debt and its management activities and tools (Li et al., 2015). The authors analyzed 94 studies published between 1992 and 2013 and identified activities and tools for technical debt. Some of the mentioned tools are Checkstyle, DebtFlag, SonarQube, CodeVizard, and FindBugs. Likewise, the activities include code analysis, cost categorization, calculation models, code metrics, and portfolio approach. Also, the authors proposed a classification of technical debt types and argued that there needs to be more literature about what should not be considered technical debt.

The work of Fernández-Sánchez et al. consisted of a systematic mapping to identify the elements to consider for the management of technical debt, based on the literature until 2015 (Fernández-Sánchez et al., 2017). The authors identified the main aspects of technical debt management. They found that the business organizational perspective has not been addressed much in the literature, while research has focused more on the technical point of view.

Another systematic literature review focused on technical debt in the digital government area (Nielsen et al., 2020). This paper aimed to discover what fields of technical debt management are being studied and the focus of the performed research. The authors analyzed 31 pieces, from which a third proposed a tool, method, technique, or model for technical debt management. The authors found several gaps, including a lack of research on the public sector and a limited abstraction level of the analyses. Authors conclude that technical debt management is mainly studied either on open software projects or in the private sector.

Macit et al. performed a systematic mapping study regarding methods for identifying architectural debt based on the analysis of 28 papers published between 2011 and 2020 (Macit et al., 2020). The authors mention that architectural debt identification has been increasingly investigated in recent years. Also, code mining and expert opinion are common methods.

Alfayez et al. performed a systematic literature review on technical debt prioritization (Alfayez et al., 2020). The authors aimed to identify the current prioritization approaches, the decision factors, and the artifacts on which these approaches are based. A total of 23 papers published between 1992 and 2018 were analyzed. As a result, 24 strategies were found for technical debt prioritization. These approaches mainly addressed code, general, and design technical debt.

Lenarduzzi et al. performed a literature review regarding strategies and tools for technical debt prioritization (Lenarduzzi et al., 2021). In this study, they analyzed 44 primary studies published until 2020. Code, architecture, and design were the most frequent types of technical debt addressed. The authors found a lack of consensus on the factors to consider when prioritizing and measuring technical debt. Also, they show a lack of validated and reliable tools for technical debt prioritization.

Alves et al. present a systematic mapping regarding technical debt identification and management (Alves et al., 2016). In that study, the authors analyzed 100 primary studies, discussed a taxonomy of TD types, presented a list of the strategies found in the literature, and created a list of indicators that can help identify technical debt.

Table 1. Contributions by other authors.

Authors	Research topic	Analyzed period	Contributions
Li et al., 2015	Technical debt management	1992 – 2013	• Analyzed the technical debt concept on 94 existing research efforts. • Proposed a classification of ten technical debt types. • Identified the quality attributes compromised by technical debt. • Determined activities and tools for technical debt management.
Alves et al., 2016	Technical debt identification and management	2010 - 2014	• Analyzed 100 papers and determined a classification for technical debt types. • Listed strategies to identify or manage technical debt. • Determined the empirical evaluations, artifacts, and data sources cited in the literature for technical debt identification and management.
Fernández-Sánchez et al., 2017	Elements to manage technical debt	2010 - 2015	• Provided a taxonomy of elements for technical debt management by analyzing 63 papers. • Identified the proposed methods and techniques to manage technical debt. • Analyzed technical debt management elements from the perspective of stakeholders.
Rios et al., 2018	Technical debt	2012 – 2018	• Studied 13 secondary studies and their TD research topics. • Proposed a taxonomy of technical debt types. • Identified activities, strategies, and tools to support technical debt management.
Nielsen et al., 2020	Technical debt management in digital government	2017 - 2020	• Analyzed 31 papers about technical debt management research in the public sector. • Determined a research agenda for the digital government area.
Alfayez et al., 2020	Technical debt prioritization	1992 - 2018	• Identified approaches and decision factors for technical debt prioritization by studying 23 papers. • Analyzed the type of human involvement and artifacts needed for technical debt prioritization.
Lenarduzzi et al., 2021	Technical debt prioritization	2011 - 2020	• Determined the prioritization strategies for technical debt by analyzing 44 primary studies. • Analyzed factors and measures considered for technical debt prioritization. • Identified tools for technical debt prioritization.

Three previous studies analyzed and proposed a classification of technical debt types (Alves et al., 2016; Li et al., 2015; Rios et al., 2018). However, there is still no consensus on these taxonomies. This paper does not aim to provide a consensus but to find if new TD types have been mentioned recently and should be considered for new or refined taxonomies.

More recent efforts in the research area were those made by Nielsen et al. (2020) and Lenarduzzi et al. (2021). Their studies focused on technical debt prioritization and technical debt management in the digital government area. This paper focuses on technical debt identification and management, a related but different scope than their contributions.

Alves et al. (2016) and Li et al. (2015) made previous efforts specifically about technical debt identification or management. However, they analyzed literature published between 1992 and 2014. Our study aims to replicate and update the work of Alves et al. (2016) to integrate the obtained results by including papers published between 2015 and 2022. This delimitation is the main difference between this work and previous contributions made by other authors.

The relevance of performing this study is justified by the application of the framework proposed by (Mendes et al., 2020) for updating Systematic Literature Reviews:

- Does the previous study still address a current question?

  The high-level research question of this paper is: What strategies have been proposed to identify or manage technical debt in software projects? Any software may contain technical debt issues, regardless of the developing company's size or resources. The Consortium for Information and Software Quality (CISQ) reports that the cost of poor software quality in the US is at least $2.41 trillion and the accumulated technical debt principal is about $1.52 trillion in 2022 (Consortium for Information & Software Quality, 2022). Therefore, TD remains an expensive issue. Identifying and managing TD is still a major problem in the software development industry. Thus, investigating these topics is relevant for both practitioners and researchers.

- Has the previous study had good access or use?

  The work of Alves et al. (2016) was published in the Information and Software Technology Journal and is fully available through the Science Direct library. By March 2023, this paper has 589 reads and 238 citations (according to ResearchGate metrics). Thus, the previous study has good access and use.

- Has the previous study used valid methods and it was well conducted?

  Alves et al. (2016) based their methods on the standard process for conducting systematic mapping studies by Petersen et al. (2008). They provide a full explanation of the study implementation (research questions, search strategy, selection criteria, and classification scheme). The study presents a clear view of each step's outcome. Hence, it provides sufficient details and data to replicate the procedures. Moreover, two of the original authors participated in the update process.

- Are there any new relevant studies, methods, or new information?

  Research on technical debt is constantly being published in different venues, such as conferences and journals. For example, the International Conference on Technical Debt (TechDebt) is held annually since 2018, which is two years after the publication of the previous study in 2016. Consequently, there are plenty of new pieces on TD.

- Will the inclusion of new studies/information/data change the findings, conclusions, or credibility?

  Since the publication of the previous study in 2016, the concepts and focus of the TD research area have evolved. In this paper, we will discuss in detail these changes. One of the most important aspects is the increase of research efforts that address technical debt management with a more holistic perspective.

By updating the previous study by Alves et al. (2016), we provide the following contributions:

- An analysis of the technical debt identification and management research landscape between 2010 and 2022;
- Analysis of the previously proposed technical debt types and identification of new potential types mentioned recently in the literature;
- List of the strategies or techniques for technical debt identification, management, and visualization;
- An analysis of the empirical evaluations of the proposed methods, including the artifacts, programming language, and data sources used.
- Discussion on technical debt concepts and their evolution from 2010 to 2022.

The contributions presented herein provide insights to both practitioners and researchers regarding the most recent proposals for identifying and managing technical debt. This may help for further industry application of new proposals and for finding new research opportunities. The following section presents the methodology applied to perform this study.

3 Research Method

This paper aims to analyze the research landscape on technical debt identification and management. This section details the search strategy, study selection process, and synthesis methods.

3.1 Research questions

In this section, we present the rationale and importance of the research questions.

- RQ1. What are the types of technical debt found in the literature?

This question aims to determine if there are new technical debt types described in the literature different from

those proposed by Alves et al. (2016). Also, we aim to know which types of technical debt have been most studied in the literature between 2015 and 2022.

This research question is important because there is still no consensus on the different technical debt types. We aim to analyze the evolution of these concepts between 2010 and 2022 by integrating our results and those provided by Alves et al. (2016). However, the intent of this paper is not to establish a consensus but to find out the TD types mentioned in the literature.

- RQ2. What are the strategies proposed to identify technical debt?

This research question aims to determine new artifacts or data sources mentioned in the literature. Also, we aim to know which artifacts and data sources are the most cited. This allows us to determine trends or changes in recent years. We also aim to analyze the empirical evaluations of previously mentioned strategies since Alves et al. (2016) describe the need for more assessments to determine the applicability of such strategies.

Visualization techniques for technical debt identification are also crucial because they may help communication between developers and stakeholders, affecting decision-making during software development.

- RQ3. What strategies have been proposed for the management of technical debt?

This research question aims to determine the strategies for technical debt management and how they have been empirically tested to determine their applicability. Also, we aim to identify the visualization techniques proposed for technical debt management. Alves et al. (2016) found few visualization strategies. This study analyzes how this specific research topic has evolved from 2010 to 2022.

3.2 Search Strategy

We retrieved papers from the databases ACM Digital Library, IEEE Xplore, Science Direct, and Springer Link. We also consulted Engineering Village, Scopus, Citeseer, and DBLP, but no papers were included from these libraries. Since this paper updates the previous work of Alves et al. (2016), we used the same search string:

(“Technical debt”) AND (“Software”)

This search string was used in all the sources, restricting the results to publications between 2015 and 2022.

3.2.1 Inclusion Criteria

We considered papers that met the following inclusion criteria:

- Address the identification or management of technical debt in the context of software development;
- Explain one or more strategies, techniques, or activities for identifying or managing technical debt;
- The year of its publication is between 2015 and 2022 since the previous work of Alves et al. (2016) included papers from earlier years.

We also considered papers that address technical debt in a general manner or focus on a specific type of debt. Moreover, we included those that either provided empirical proof of their proposal or only a theoretical description.

3.2.2 Exclusion Criteria

Only the most recent paper was considered when several pieces reported the same study, and each study was considered separately when multiple studies were contained in a single paper. Also, we applied the next exclusion criteria:

- Papers that do not specify how to identify or manage technical debt with a strategy, activity, or technique. Therefore, we excluded exploratory studies of technical debt management;
- Papers in progress (incomplete) or those that do not provide full-text access;
- Papers published before the year 2015;
- Duplicate papers;
- Papers published in a language different than English.

Moreover, papers in the form of PowerPoint presentations, reports, and abstracts only were not considered.

3.3 Study Selection

The study selection was performed by following the following steps:

- **Search**: The first step of the process was to perform the search using the defined search string on the databases (ACM Digital Library, IEEE Xplore, Science Direct, Engineering Village, Springer Link, Scopus, Citeseer, and DBLP). As a result, we found 2517 papers in total.
- **Identification** (Filter 1): The second step was removing duplicate papers and applying the exclusion and inclusion criteria. In total, 466 duplicate studies were identified, leaving a total of 2051 articles (without duplicates).
- **Screening** (Filter 2): The next step was screening the articles. We read each of the 2051 titles and abstracts to find those that comply with the eligibility criteria. In this step, we identified 209 studies. In this step, 1852 papers were excluded because they did not comply fully with the inclusion criteria. This is explained because the search string is generic and returned articles that are not relevant to this study.
- **Inclusion and analysis** (Filter 3): All 209 papers were read in full at this stage. After reading them, only 111 were selected using the eligibility criteria. At this stage, we also extracted data from the selected papers as described in the following subsection (3.4 Synthesis Methods).
- **Backward snowballing**: During the final stage, we reviewed the references of each of the studies. As a result, we included seven more papers, which were analyzed and combined with the results of the study selection.

One researcher performed the search, identification, screening, and inclusion for every paper. Later, two other researchers were randomly assigned a set of papers each to independently review each paper and extract the data. The results were compared and discussed in case of disagreement. The process was performed in mid-March 2022.

3.4 Synthesis Methods

From each of the included papers, we extracted six categories of information. **Table 2** summarizes the data variables collected.

- **Metadata**: For a demographic characterization, we collected the title, authors, type and year of publication, and digital library from each included paper. These data were extracted as explicitly found on each corresponding database. We also considered two research topics: identification or management of technical debt. From each paper, we also collected the corresponding research topic.
- **Technical debt types**: We documented each paper's addressed type of technical debt. Some papers explicitly mentioned the studied type of technical debt, but in others, this was implicit. Also, many papers addressed technical debt without focusing on any specific type (77 in total). Therefore, we considered the types as follows:
  - **Direct:** Papers that explicitly mention the name of the type + debt.
  - **Indirect:** Determined from phrases in the text, such as *technical debt derived from issues on the documentation* (documentation debt).
  - **General:** We classified into general technical debt those papers that do not focus on a specific technical debt type directly or indirectly and consider only the concept of *technical debt*, such as *technical debt management approach*.
- **Indicators**: Indicators are elements that help identify technical debt items (Alves et al., 2016). We created a list of the indicators cited by authors on each paper and its associated type of technical debt, based on the indicators found by Alves et al. (2016). A new indicator was created when these previous indicators did not fit what was mentioned in a paper. We also collected data on how these indicators were empirically tested, including the artifact in which they are identified and data sources.
- **Management strategies**: We extracted the management strategies described in each included paper. We used the same criteria as Alves et al. (2016): to be considered a management strategy, it must support the decision-making about technical debt items. This definition includes activities for measuring, prioritizing, preventing, monitoring, documenting, and paying the debt. Each strategy and its definitions were collected as mentioned in each paper.
- **Evaluation studies**: Evaluations are needed to determine the feasibility of the proposed strategies. There are several types of evaluation studies. We classified them into case studies, controlled experiments, or ethnographic studies with the same criteria as in the previous study (Alves et al., 2016). Also, we documented the artifact considered, the programming language used, and the data sources used on each paper that performed an empirical evaluation.
- **Visualization techniques**: Several visualization techniques help understand the potential problems of technical debt in software projects. We extracted the visualization techniques described in the included papers for technical debt identification or management mentioned in each paper as Alves et al. (2016).

The aforementioned research method is based on the procedure performed by Alves et al. (2016) in their study. In this paper, we aim to answer the same research questions by applying the same study selection and synthesis methods. However, this update's protocol has two main differences from the original study methodology. One is the time-frame delimitation. We only considered publications between 2015 and 2022, a restriction that was added to the search strategy criteria. Moreover, we provide the definition of "general" technical debt classification. In the previous study, the authors refer to "type not specified" while we classify these papers as "general" technical debt to provide more clarity to the reader. However, we refer to the same type of papers (as described in the Synthesis Methods).

Table 2. Data collection variables and their purpose.

Data collection variable	Purpose
Title Author Type of publication (workshop, conference, journal) Year of publication Digital library (database) Research topic (identification or management)	Demographic characterization
Technical debt type	RQ1
Indicators	RQ2
Artifact considered (identification studies)	RQ2
Data source (identification studies)	RQ2
Management strategy (management studies)	RQ3
Evaluation type (if applicable: case studies, controlled experiments, ethnographic studies, action research)	RQ2 and RQ3
Visualization techniques	RQ2 and RQ3

4 Results

This section presents the integration of our results and those obtained by Alves et al. (2016), which included 100 papers published between 2010 and 2014. Our study analyzes 117 additional articles dating from 2015 to 2022 (see Appendix A1). **Figure 1** shows the number of studies included by publication type and year.

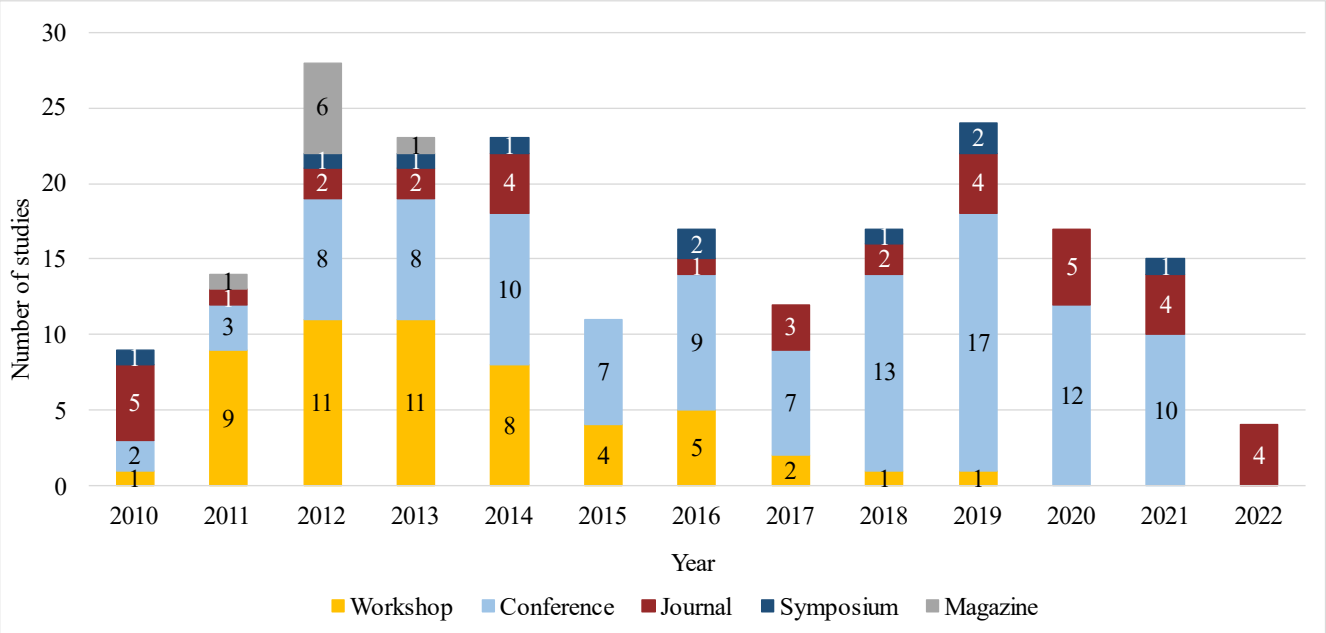

Figure 1. Number of studies by year and publication type.

We searched the same databases using the same search string and applied the established selection criteria. Papers were published in symposia, journals, magazines, workshops, and conferences. From 2010 to 2014, workshops and conferences were the most common publication types. Between 2015 and 2022, the most common publication types were conferences and journals. The decrease in publication on workshops and the rise of conferences shows that the theme has developed certain maturity over the years.

The number of publications on technical debt identification and management has been irregular during the last decade. Overall, the number of articles included in conferences has been rising. In 2010 only two papers were from conferences, while this number increased to 17 in 2019. However, there may have been an impact on the publications done in 2020 and 2021 due to the coronavirus pandemic.

In this study, we performed the search on ACM Digital Library, IEEE Xplore, Science Direct, Springer Link, Engineering Village, Scopus, Citeseer, and DBLP. Overall, since 2010 most papers have been published in the IEEE Xplore and ACM Digital Library. However, the number of papers on Springer Link has increased considerably since 2015**.** **Figure 2** shows the number of studies by digital library.

4.1 Technical debt types (RQ1)

Alves et al. (2016) proposed a taxonomy of technical debt that includes: design, architecture, documentation, test, code, defect, requirements, infrastructure, people, test automation, process, build, service, usability, and versioning debts.

From 2010 to 2014, the most common technical debt types studied in the literature were design, architecture, and documentation. Also, a high concentration of studies addressed the test, code, and defect debt.

Between 2015 and 2022, 77 studies did not focus on a particular type but addressed the topic in a general manner. In contrast, other papers focused on a specific technical debt type, such as code, design, or architecture. Consequently, technical debt is increasingly studied with a holistic approach rather than distinct kinds of debts that need to be managed differently. **Figure 3** shows the number of papers included by type of technical debt.

Of the 117 included papers (2015 - 2022), 34 addressed self-admitted technical debt, a concept commonly mentioned in the literature. Self-admitted technical debt (SATD) refers to situations in which developers are aware and admit that technical debt has been incurred. These scenarios are different from those with no consciousness that debt is present.

When SATD exists, issues may correspond to various technical debt types, such as code, architecture, documentation, etc. For this reason, those papers that addressed SATD were classified into the general technical debt (TD) category.

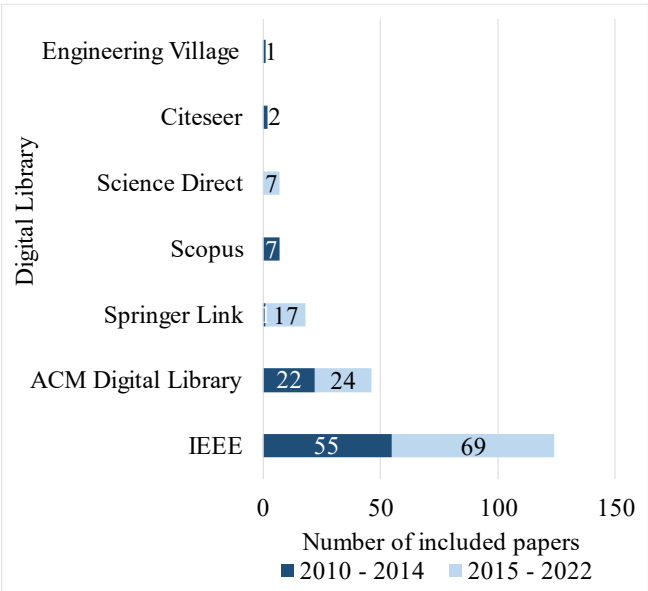

Figure 2. Number of included papers by digital library.

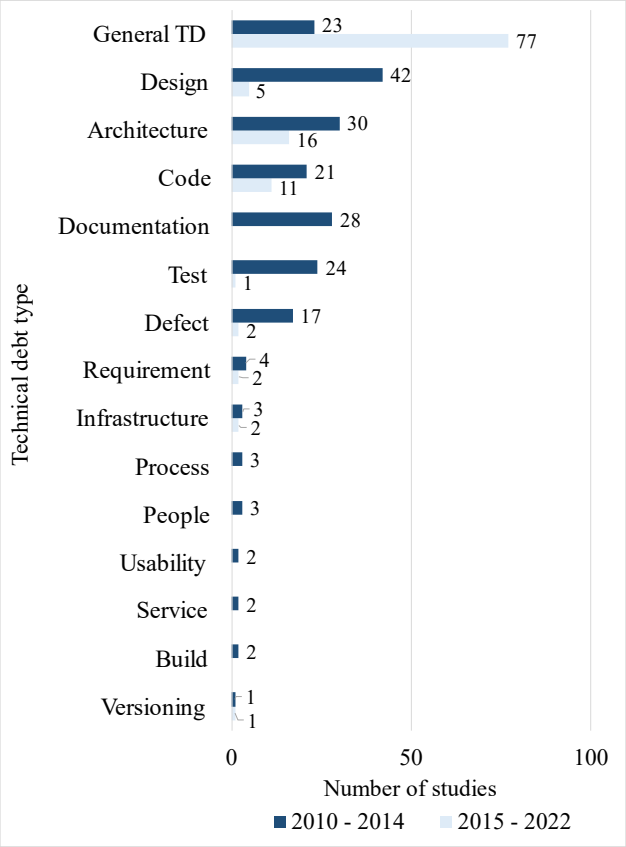

Figure 3. Number of included papers by technical debt type and year.

Forty out of the 117 papers addressed a specific technical debt type, as described by (Alves et al., 2016). From 2015 to 2022, architecture, code, and design debt were the most common types. However, there has been a significant reduction in studies focused on these types over the last seven years. For example, there was a reduction of nearly half of the publications on architecture and code debts, while design debt went from 42 publications to only five compared to the previous period (2010 – 2014).

We found no articles about documentation, people, build, services, or usability debt during the last seven years. These types of technical debt have not been as extensively studied compared to others, so they represent potential areas for further investigation in future work.

As a result of the literature review, we did find two new types mentioned in the literature: security and elasticity debts. Security debt refers to security issues in the software, such as vulnerabilities or exploitable weaknesses (Izurieta et al., 2018). Elasticity debt is a concept that describes non-effective or non-efficient resource provisioning resulting from the lack of dynamical adaptation to resource consumption (Mera-Gómez et al., 2016). These two types of technical debt have been mentioned in few studies. Consequently, they cannot still be considered widely accepted types of technical debt. Both may be subtypes of requirement (security) and infrastructure (elasticity) papers and classified them as such when performing the literature review.

4.2 Technical debt identification (RQ2)

An essential step for technical debt management is its identification. Identification comprises activities or actions to detect the presence of debt in software artifacts. Out of the 117 included papers, 47 addressed technical debt identification. We extracted the indicators and type of technical debt associated with each paper. Indicators are symptoms that help identify technical debt items (Alves et al., 2016).

From 2010 to 2014, forty-five indicators were found and presented by Alves et al. (2016). In this study, we found 11 indicators mentioned in the literature between 2015 and 2022. **Table 3** shows the eleven indicators and the top 5 most common indicators presented previously (Alves et al., 2016): code smells, documentation issues, software architecture issues, violation of modularity, and automatic static analysis issues. These indicators were either just mentioned or analyzed in the included papers.

The results show significant differences between both periods. Code smells were the most common indicator in previous years, while the comments and commits were mostly mentioned during the last seven years. This fact is due to the considerable number of papers (34 in total) that addressed self-admitted technical debt in recent years, which used several strategies to analyze comments or commits to identify different types of technical debt, not only those related to source code. This represents a more holistic view, in which not only code issues are intended to be identified.

Authors have recently studied SATD through natural language processing, neural networks, deep learning, and machine learning. SATD may be identified using these different approaches on commits, comments, and issue trackers to be further prioritized and managed.

4.2.1 Evaluation studies

Most studies on technical debt identification have performed empirical evaluations through case studies in recent years. However, there has been an increase in this type of study during the last seven years. A possible explanation for this is that the knowledge consolidated before 2015 gave the necessary foundations to perform empirical evaluations, such as case studies. The execution of case studies helps provide more information about the context in which the different identification strategies are applicable. The growth in the number of case studies is relevant because it is vital to have multiple sources of empirical data to generalize results.

We also found a significant increase in the number of controlled experiments. **Figure 4** shows the number of empirical evaluations performed by the number of papers and year of publication.

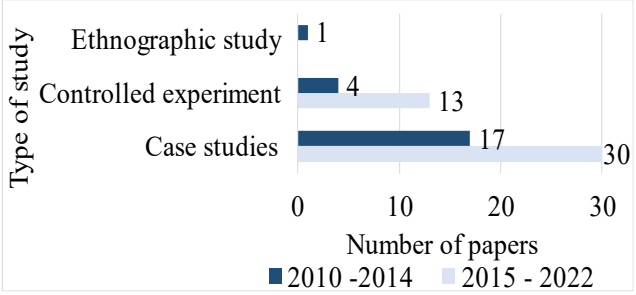

Figure 4. Empirical evaluations on technical debt identification studies.

Table 3. Indicators organized by technical debt (TD) type and period.

Indicator	2010 - 2014		2015 - 2022	
	# Papers	Technical debt types	# Papers	Technical debt types
Code smells	52	Code, Design	1	General TD, Architecture
Documentation issues	17	Documentation	-	-
Software architecture issues	9	Architecture	9	Architecture
Violation of modularity	9	Architecture	-	-
Automatic static analysis issues	9	Code, Design	3	Architecture, Code, General TD
Comments	1	Documentation	26	Code, Requirements, General TD
Uncorrected known defects	6	Defect, Test	1	General TD
Immature software	-	-	1	General TD
Feature usage and maintenance costs	-	-	1	General TD
Insufficient resource provisioning	-	-	1	Infrastructure
Low external/internal quality	1	Design	1	General TD
Software design issues	4	Design	1	Design

4.2.2 Artifacts and data sources

We extracted the data source and artifact considered in each paper that performed an empirical evaluation. **Figure 5** shows the number of studies by artifact. From 2010 to 2014, the most common artifact was source code. The obtained results show that source code remains the primary artifact used to perform empirical evaluations; this may be because static analysis tools can help for these purposes. However, the number of studies considering source code decreased from 58 between 2010 and 2014 to 39 from 2015 to 2022.

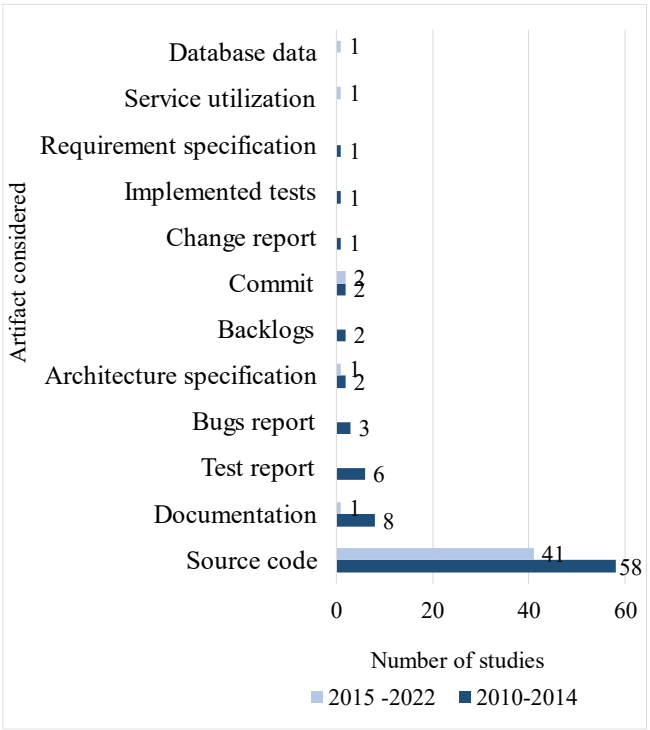

Figure 5. Number of studies by artifact considered for technical debt identification.

Researchers have started investigating by mining the repositories to extract metadata about technical debt in recent years. Alves et al. (2016) identified four different data sources: CMS (Configuration Management Systems), software repositories, and bug tracking. The CMS were the most used in that period. In contrast, we found six different data sources from 2015 to 2020. Software repositories predominated, which makes sense since the most common artifact was source code. **Figure 6** shows the number of papers by the data source used.

4.2.3. Visualization techniques

Only two papers on technical debt identification mentioned a visualization technique. The proposed methods are not mature because there is not much validation. Therefore, the visualization of technical debt is still an area that requires further investigation. The proposed visualization techniques were the assessment graph (Shapochka & Omelayenko, 2016) and coupling probability matrix (L. Xiao et al., 2016).

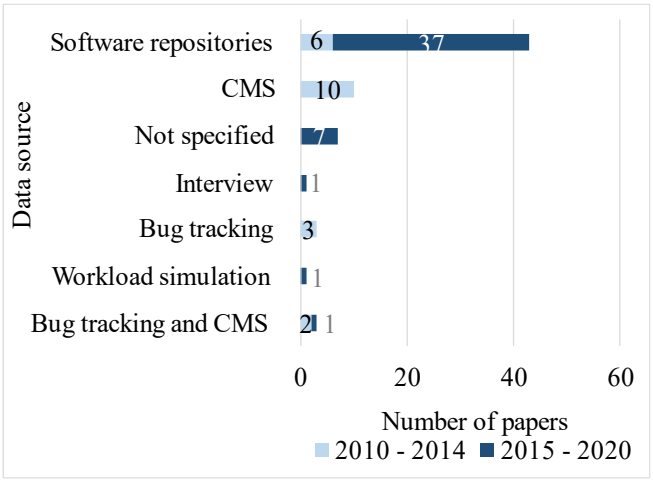

Figure 6. Number of papers by the data source for technical debt identification.

4.3 Technical debt management (RQ3)

The management of technical debt includes several activities to control debt during the software development process. These activities aim to avoid bankruptcy situations in which the debt becomes uncontrollable.

Out of the 117 included papers, 70 addressed technical debt management. This section presents the strategies proposed by authors in the literature, the evaluation studies performed, and the visualization techniques mentioned.

4.3.1 Strategies for managing technical debt

The first step for technical debt management is to identify its presence. Then, several strategies could be used for its timely administration to reduce interests' impact. **Table 4** shows the complete list of management strategies found during the literature review. The top 5 most studied strategies between 2015 and 2022 were the following:

- **Automated analysis of code issues**: Recent studies mention several tools to aid TD management: Sonargraph for analyzing software architecture (von Zitzewitz, 2019), Teamscale to analyze software quality based on data from version control systems, issue trackers and other tools (Haas et al., 2019), Sonarqube to analyze code and get several code metrics (Baldassarre et al., 2020), and CodeScene to perform behavioral code analysis, which can be helpful for debt prioritization and communication with stakeholders (Tornhill, 2018). These papers report code, architecture, test, and general technical debt management supported by tools that automatically identify code issues. The generated metrics or reports may be used by developers and stakeholders to prioritize, monitor, and perform the necessary management actions. One of the advantages of such tools is that they need minor human intervention to measure several code issues while creating awareness that TD exists.
- **Calculation of technical debt (TD) interest**: Interest is the additional cost that will have to be assumed because of the delayed payment. Authors have proposed methods for its calculation to prioritize technical debt items according to the interest that will have to be assumed (Chatzigeorgiou et al., 2015; Falessi & Reichel, 2015). This allows decision-making about the appropriate moment to pay the debt, depending on the acceptable costs of each scenario.
- **Portfolio approach**: In finance, the portfolio comprises the assets that an investor has. Portfolio management is carried out to decide what investments to make with the assets considering risks and return of investment. A TD portfolio approach brings financial concepts to TD and considers it a potential investment, whose final goal is to get more gains than losses (Guo & Seaman, 2011). The TD portfolio approaches are based on the financial portfolio theory and consider principal, interest, or correlations with other TD items to help decision-making. Authors proposed a glossary of financial technical debt concepts (Akbarinasaji & Bener, 2016), while others present frameworks that consider portfolio theory (Nielsen & Skaarup, 2021; Rindell et al., 2019). Papers that consider more than only interest calculation and reference the portfolio theory were classified into portfolio approaches rather than just interest calculation, while those that only mention interest formulas were classified as calculation of TD interest.
- **Prioritization approach**: Authors suggest different methods to prioritize TD items. The purpose of prioritization is to determine the order in which technical debt will be paid. The proposed strategies include code smells ranking through automated tools (Alfayez & Boehm, 2019; Vidal et al., 2016), backlogs managed considering risks and business needs (Besker et al., 2019), and approaches that focus on the business perspective (Stochel et al., 2020).
- **SATD removal approach**: Authors have also suggested management strategies for self-admitted technical debt removal. For example, natural language processing could analyze source code comments and later compare their evolution among different versions of each file (da Maldonado et al., 2017). It is also possible to use deep neural networks to provide recommendations for SATD removal (Zampetti et al., 2020).

Some of the included papers addressed strategies or techniques identified in previous years (Alves et al., 2016). These proposals are the portfolio approach, options analysis, calculation of the principal and interest, and TD management in database schemas. However, the number of empirical evaluations performed on each strategy is still small. Overall, the authors have proposed their own strategies and tested them empirically instead of validating or comparing them to previous proposals.

4.3.2 Evaluation studies

Case studies have been the most frequent type of empirical evaluation performed on technical debt management. This is true for both periods, as shown in **Figure 7**. Nevertheless, the number of this type of study raised to more than double from 2015 to 2022. Also, ten papers presented action research and controlled experiments in recent years, adding some diversity to the type of evaluation studies.

From 2010 to 2014, few empirical studies were performed in real settings. In contrast, subsequent years show more case studies and action research in real settings. The number of these evaluations is still small for every management strategy. Still, it is essential to highlight that researchers have started to acknowledge the need for empirical testing.

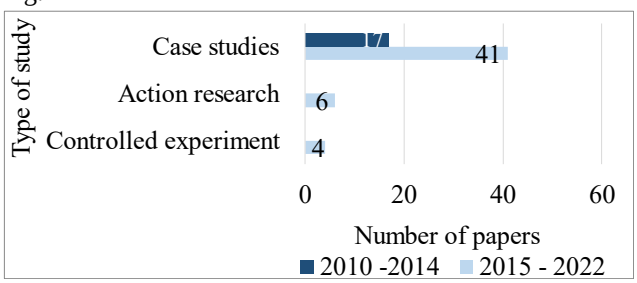

Figure 7. Number of papers by type of study.

4.3.3. Visualization techniques

Only four papers on technical debt management mentioned visualizations techniques, which were the following: dynamic graphic (Pacheco et al., 2018), line chart (Falessi & Reichel, 2015), portfolio matrix (Plösch et al., 2018), and probabilistic cause-effect diagrams (Rios et al., 2019). Each technique was only mentioned once. Therefore, they still require further research to determine their applicability.

5 Discussion

This paper studied the technical debt identification and management research landscape from 2015 to 2022 and integrated our results with previous investigation efforts that analyzed the period 2010 - 2014 (Alves et al., 2016). This section presents a discussion of the obtained results.

5.1 Technical debt types (RQ1)

Technical debt as an analogy with financial debt is well known among authors in the academic literature. Overall, there is a common understanding of the technical debt concept itself as taking shortcuts during software development, leading to several future costs. However, different technical debt classifications exist, and there is no clarity on which are the accepted types.

Alves et al. (2016) proposed a taxonomy that includes: design, architecture, documentation, test, code, defect, requirements, infrastructure, people, test automation, process, build, service, usability, and versioning debts. Still, other studies mention different classifications, but there is a lack of consensus on some technical debt types.

To the best of our knowledge, besides the proposal of Alves et al. (2016), only three other papers address technical debt types or propose a classification (Li et al., 2015; Rios et al., 2018; Tom et al., 2013). One of these classifications was presented as a result of a non-academic literature review and interviews with people in the software development industry (Tom et al., 2013). Others were derived from a systematic mapping and a tertiary study of academic literature (Alves et al., 2016; Rios et al., 2018).

Some types of technical debt are presented in the three studies: code, design, architecture, and test debt. In fact, we found that from 2015 to 2022, the most addressed types correspond to design, architecture, code, and test debts. We observed that authors use these terms consistently, agreeing with their general meaning. Therefore, these particular types may be considered accepted technical debt types. On the other hand, the concept of self-admitted technical debt (SATD) is overall consistent among papers and referred to as a technical debt type.

Other types are much less established in the literature. For example, between 2010 and 2022, process and people debts were only mentioned in three papers each, while usability, service, build, and versioning debts were only cited in two papers each. There is also another new concept mentioned in the literature: variability debt. It was not identified through the performed review because papers mentioning it do not meet the acceptance criteria proposed in this study. However, it may be considered for future research. Variability debt refers to software's characteristics that allow it to adapt (create variants) for different needs (Wolfart et al., 2021). These concepts are still not widely accepted since not much literature is available on them. In some cases, they may not even represent technical debt categories themselves but subcategories. The same may be true for security and elasticity debts, which could be subcategories of other types of debt. Another relevant aspect is a position in the literature that considers defects and processes as non-technical debt (Li et al., 2015). However, this does not imply that the elements addressed by these types of technical debt lack importance.

Figure *8* presents a heatmap showing the number of publications by technical debt type and year, including papers from 2010 to 2022. The lack of clarity on some technical debt types and the number of existing categories may have influenced the authors' choice of categorizing their work. Still, the number of papers that presented typifications of technical debt dropped in 2016. Authors may have inadvertently reached the consensus that technical debt is an issue to be managed without necessarily specifying its type. There was a turning point between 2014 and 2015, in which authors left aside the classification and began to focus their studies on technical debt management.

5.2 Technical debt identification (RQ2)

Technical debt identification comprises actions to detect debt presence; it is the first step necessary for its management. In recent years, source code has been the most common artifact that helps technical debt identification since it is possible to implement several techniques, algorithms, or tools to detect debt automatically. However, other artifacts may be used, such as test cases.

Figure *9* summarizes the findings on technical debt identification between 2010 and 2022. When technical debt exists, there are several indicators that show symptoms of its presence. Identification approaches help find indicators through several artifacts and data sources for further management.

Comments on source code were the most common indicator from 2015 to 2022. Analyzing comments helps to identify self-admitted technical debt. The increasing number of studies on SATD suggests that there is valuable information that developers themselves can provide through comments. Nevertheless, it is a future research opportunity to explore how to take advantage of developers' knowledge of the code issues in other ways, different from only comments or commit messages.

	Publication year												
TD type	**2010**	**2011**	**2012**	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**
Requirement			2		2	1			1				
Usability				1	1								
Architecture	2	3	11	5	9	2	3	1	2	6	1	1	
Service				2									
Design	5	8	11	9	8	2			2		1		
Code	3	1	9	5	3	3	2	1	2	2	1		
Build			1		1								
Versioning				1					1				
Test	2	2	8	6	6				1				
Defect	1	5	3	3	5		1		1				
Test automation				2	1								
Process				2	1								
Documentation	2	4	6	4	12								
People			1		2								
Infrastructure	1		1	1			1			1			
Technical debt (General)	1	1	5	6	10	3	10	10	7	15	14	14	4

Figure 8. Heatmap showing the number of publications by technical debt type and year.

The automatic detection of technical debt allows getting a variety of quantitative measurements. Additionally, organizations may be interested in also having qualitative measures on technical debt, which has not been much explored in the literature and constitutes a future work opportunity. Depending on every project's business objectives and needs, organizations may identify those measurements that may help them further manage technical debt in their contexts.

In the academic literature, the number of empirical evaluations on technical debt identification has increased in recent years, which is beneficial for both researchers and practitioners because they help discover the indicators' applicability in different contexts. However, research has concentrated on identification based on source code, while other artifacts and data sources may be further investigated.

Few studies proposed visualization techniques for technical debt identification between 2010 and 2022. This is still an open issue and research opportunity. Technical debt visualization is important because it may support communication between developers and stakeholders while aiding decision-making on further technical debt management and prevention.

5.3 Technical debt management (RQ3)

Technical debt management comprises actions or activities performed to control debt once it has been identified. Authors in the literature have proposed several strategies for debt management. In general, papers present new proposals and test them empirically instead of testing others previously described in the literature. However, the number of empirical evaluations, especially case studies, has increased during the last seven years, along with the number of proposals. **Table 4** shows the complete list of strategies found in this replication study.

From 2015 to 2022, many strategies proposed for technical debt management were supported by automatic tools applied to source code, such as Sonargraph, CodeScene, Sonarqube, and Teamscale (Baldassarre et al., 2020; Haas et al., 2019; Tornhill & Ab, 2018; von Zitzewitz, 2019). The obtained measurements through such tools help to prioritize and support decision-making. Other authors compared penalties and gamification techniques for technical debt using automated tools in educational contexts, showing that rewards may be a suitable option for TD management (Crespo et al., 2021).

Other papers presented novel frameworks or models for managing technical debt during the software development process with a more holistic perspective, including several process elements or phases. For example, some authors propose creating a guild, a group of people that help address TD management and guide its payment (Detofeno et al., 2021). Moreover, another paper mentioned encouraging and rewarding incentives for developers to manage technical debt (Besker et al., 2022). Other authors evaluate a business prioritization approach that allows an alignment between business and technical stakeholders for prioritizing TD items (Reboucas De Almeida, 2019), while additional research efforts report using TD tickets that allow TD management and prevention (Wiese et al., 2021). Nevertheless, few papers specifically address the human resources involved during software development, which is essential because it is known that people issues can also lead to technical debt (Rios et al., 2020).

Between 2010 and 2014, twenty-two papers on technical debt management described a visualization technique. In contrast, we only found four visualization techniques in papers about technical debt management from 2015 to 2022. This shows a significant decrease in research efforts, even when only a few studies address this topic.

Although it is not part of the research questions of this paper, it is worth mentioning that there are different perspectives regarding the definition of TD management in the

literature. In this paper, we used the same definition of TD management as Alves et al. (2016) to be conceptually consistent. However, some authors consider that TD management includes its recognition, analysis, monitoring, and measurement (Izurieta et al., 2016), while others consider its identification, assessment, and remediation (Griffith et al., 2014). Furthermore, Li et al. (2015) present nine activities for technical debt management: identification, measurement, prioritization, prevention, monitoring, repayment, documentation, and communication.

As several definitions of TD management suggest, there are plenty of actions that help to control debt during software development. However, the concepts mentioned by the different authors agree that the first step for managing TD is to identify or recognize the presence of debt and start measuring (quantifying) it. These two activities alone are not the solution for TD issues. Subsequent strategies are needed to take effective actions toward TD management. It may be necessary to prioritize, monitor, repay, and document debt (Li et al., 2015). Prioritization includes deciding the order of importance or urgency to pay debt items. Monitoring refers to supervising several aspects related to TD, such as historical costs and resolution times. It is not possible to monitor debt if metrics have not been established and measured. Also, the progress on debt issues cannot be tracked if there is no monitoring. Moreover, debt repayment or remediation is the resolution of a TD item. Also, documentation and communication with stakeholders may be needed. Lastly, organizations may be interested in establishing prevention actions.

Technical debt is a context-dependent issue (Fernández-Sánchez et al., 2017). Therefore, the context must be well understood to take appropriate actions for debt management. Gathering and analyzing data (not only about TD) may be useful for establishing a TD management plan. For example, debt management may be different in an agile organization than in a traditional one. Also, the team size and type of software that is developed are variables that may be considered. Moreover, determining the main debt issues perceived by the software developers could be a starting point. Regardless of which definition of TD management is used, the appropriate strategies will depend solely on the specific needs, issues, and objectives of the organizational context. Furthermore, the selected strategies may vary or be adapted in time depending on the obtained outcomes.

The following sections present the threats to the validity and conclusions of this paper.

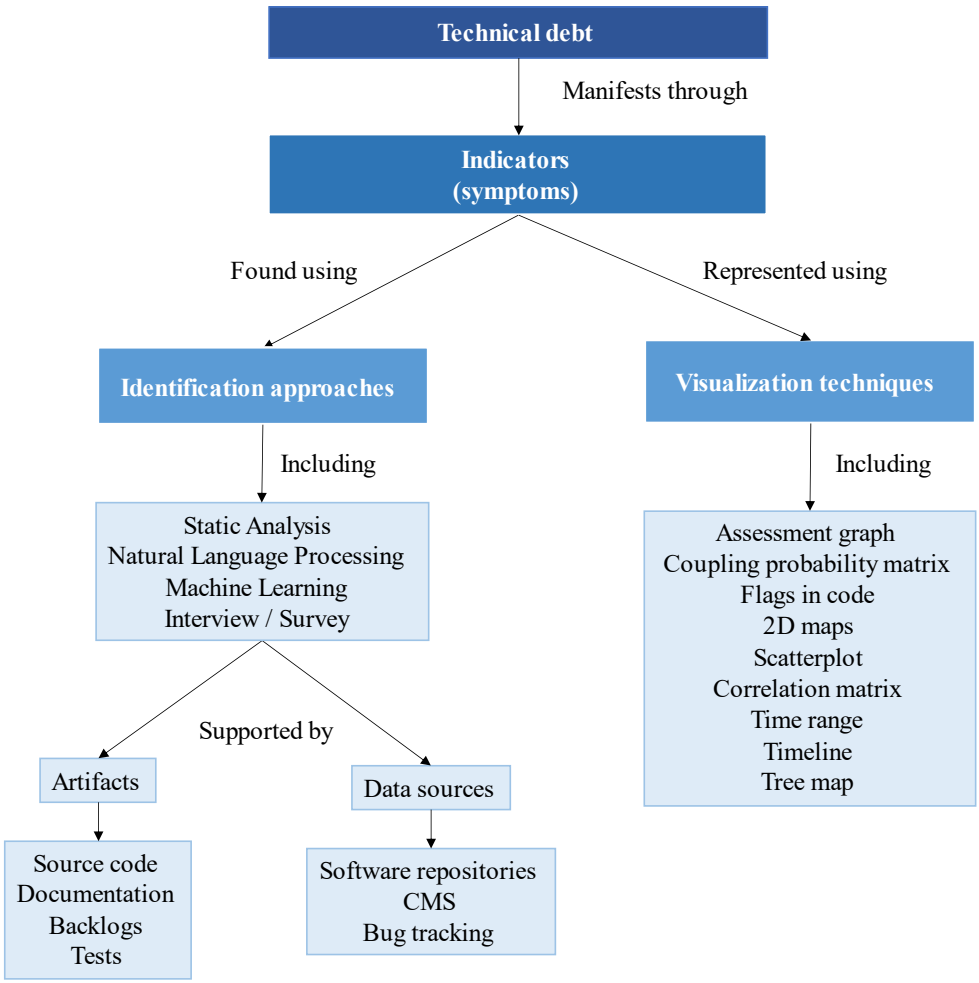

Figure 9. Technical debt identification concept map.

Table 4. Management strategies proposed in the academic literature from 2015 to 2022.

Strategy proposed	Number of papers	References
Automated analysis of code issues	7	(Anderson et al., 2019; Baldassarre et al., 2020; Fontana et al., 2016; Haas et al., 2019; Lahti et al., 2021; Sharma, 2019; Tornhill & Ab, n.d.; von Zitzewitz, 2019)
Calculation of TD interest	6	(Ampatzoglou, Ampatzoglou, Avgeriou, et al., 2015; Ampatzoglou et al., 2018; Chatzigeorgiou et al., 2015; Falessi & Reichel, 2015; Kontsevoi et al., 2019; Martini & Bosch, 2016, 2017a)
Portfolio approach	5	(Akbarinasaji & Bener, 2016; Aldaeej & Seaman, 2018; Nielsen & Skaarup, 2021; Plösch et al., 2018; Rindell et al., 2019)
Prioritization approach	5	(Alfayez & Boehm, 2019; Besker et al., 2019; de Lima et al., 2022; Stochel et al., 2020; Vidal et al., 2016)
SATD removal approach	3	(da Maldonado et al., 2017; T. Xiao et al., 2021; Zampetti et al., 2020)
Approach for technical debt decision making	3	(Codabux & Williams, 2016; Pacheco et al., 2018; Ribeiro et al., 2017)
Model for TD alignment with business	3	(Reboucas De Almeida, 2019; Reboucas De Almeida et al., 2018, 2019)
Calculation of TD principal	3	(Akbarinasaji et al., 2016; Kontsevoi et al., 2019; Kosti et al., 2017)
Process framework for managing TD	2	(Oliveira et al., 2015; Ramasubbu & Kemerer, 2019)
Model for optimizing technical debt	2	(Perez et al., 2019; Yli-Huumo et al., 2016)
Strategic TD management model	2	(Ciancarini & Russo, 2020; Martini et al., 2016)
Framework for TD management	2	(Borup et al., 2021; Wiese et al., 2021)
Continuous Architecting Framework for Embedded Software and Agile (CAFFEA)	1	(Martini & Bosch, 2017b)
Automated identification of refactoring candidates	1	(Tornhill, 2018)
Automated refactoring	1	(Mohan et al., 2016)
Automatic identification and interactive monitoring	1	(Fernandez-Sanchez et al., 2017)
Benchmarking-based model	1	(Mera-Gómez et al., 2016)
Continuous/extensive testing	1	(Trumler & Paulisch, 2016)
Estimation approach	1	(Lenarduzzi et al., 2019)
Linear-predictive lifecycle/incremental-predictive lifecycle application	1	(Fairley & Willshire, 2017)
Maintainability Model	1	(di Biase et al., 2019)
Managing TD in database schemas	1	(Albarak et al., 2020)
Metric for managing architectural technical debt	1	(Kouros et al., 2019)
Model-driven development (preemptive)	1	(Izurieta et al., 2015)
Model of maintenance cost growth	1	(Snipes & Ramaswamy, 2018)
Propagation model	1	(Holvitie et al., 2016)
Real options analysis	1	(Abad & Ruhe, 2015)
TD enhanced backlog	1	(Martini, 2018)
Visual Thinking	1	(Chicote, 2017)
TD cause-effect analysis	1	(Rios et al., 2019)
Normative Process Framework	1	(de Leon-Sigg et al., 2020)
TD predictive model	1	(Aversano et al., 2021)
Conceptual model for holistic debt management	1	(Malakuti & Heuschkel, 2021)
Automated identification of deprecation in metamodels	1	(Iovino et al., 2020)
TD Management Guild	1	(Detofeno et al., 2021)
Encouraging and rewarding incentives	1	(Besker et al., 2022)

6 Threats to validity

The results presented in this systematic mapping may have been affected by the following threats to validity:

- **Publication bias**: Relevant studies may not have been returned when performing the literature search. Several databases were consulted, and we did a backward snowballing process to find as many studies as possible to address this threat.
- **Search string**: It is possible that some papers in the literature propose a way to manage or identify technical debt but do not explicitly mention that they are suggesting an approach for technical debt. Therefore, these papers may have been left out when performing the search. Still, our focus was on literature regarding technical debt strategies.

Also, since this was an update to a previous systematic mapping study, other limitations and threats to validity include:

- **Consistency in integrating the results**: This paper updates the previous work by Alves et al. (2016). Different researchers performed the data extraction than the original study, and we cannot ensure that this could cause some differences in the updated results. However, our research method is based on the procedure performed by Alves et al. (2016) in their study. Also, two of the original authors contributed to the elaboration of this paper and reviewed the obtained results from the data extraction to ensure there was no misunderstanding of concepts between the two sets of primary sources to address this risk.

Lastly, we performed the paper selection process in March 2022, so the results of that year are not fully complete. These aspects are the limitations of this study.

7 Conclusion

This paper explored the evolution of technical debt identification and management research landscape over a decade. We searched for studies on eight databases and analyzed academic literature published between 2015 and 2022. By applying the defined search string and the inclusion criteria, we found 117 papers. We integrated our results with a previous study (Alves et al., 2016) that analyzed literature from 2010 to 2014.

In addition to the technical debt types mentioned in the taxonomy by Alves et al. (2016), there are three new terms in the literature: security, elasticity, and variability debt. The security type refers to security issues in the software, such as vulnerabilities or exploitable weaknesses (Izurieta et al., 2018). On the other hand, elasticity debt is a concept that refers to non-effective or non-efficient resource provisioning resulting from the lack of dynamical adaptation to resource consumption (Mera-Gómez et al., 2016). Lastly, variability debt comprises the lack of software characteristics that allow it to adapt (create variants) for different needs (Wolfart et al., 2021).

Unlike the previous mapping, most of the included papers addressed technical debt without focusing on specific types. This shows that the technical debt phenomenon is analyzed more holistically. Still, those papers that focused on specific types of technical debt studied those identifiable or measurable through code. The most frequent artifacts and data sources are the source code and repositories; this may be because there are various code and repository data analysis tools. There are no such abundant tools for analyzing other types of debts, like documentation, people, and infrastructure.

Over the years, several proposals have been developed for technical debt management. However, as in the previous systematic mapping, there is a need for more research to validate the effectiveness of the proposals and their applicability in different contexts. Another finding was that only a few studies included in the update proposed a visualization strategy. Therefore, the topic of technical debt visualization continues to be a future research opportunity.

The automatic identification of debt through the analysis of comments, commits, and source code is among the main proposals found in the literature published between 2015 and 2022. Several evaluations have been performed through case studies, controlled experiments, and action research. The number of evaluations has been rising through the years, which is particularly important for consolidating the knowledge gained in the research area. However, it is still required to perform more evaluations to generalize the obtained results.

The most relevant findings of this paper were the following:

- Investigations on technical debt identification and management have increasingly changed their focus to a more holistic perspective, considering technical debt as a global problem during the software development process instead of analyzing it as a set of different isolated problems. However, a significant number of investigations still focus on technical debt types closely related to source code.
- The number of empirical evaluations performed on each strategy is still small. In most cases, authors have proposed their own strategies and tested them empirically instead of testing previous proposals.
- Recent research on technical debt has focused on its management, while the proposal of new types has decreased dramatically since 2016. Creating new categories seems unnecessary, while authors may have inadvertently reached the consensus that technical debt is an issue to manage without specifying its type.
- Overall, authors agree on the general meaning of code, design, architecture, and test debt, which suggests that these are widely accepted technical debt types.

Likewise, future work possibilities include the following:

- Research on how to use developers' knowledge on existing technical debt, not only focused on their comments or commit messages. It is an opportunity to

explore this knowledge as a valuable asset for technical debt identification.

- Creating tools for analyzing certain types of debt, such as documentation, people, and infrastructure is a potential research opportunity since there is a lack of such tools.
- There is still a small number of proposals regarding technical debt visualization; this is a future research opportunity, particularly considering that visualization techniques can help to better communicate with stakeholders.
- Few studies have explored strategies with a less-technical approach but focus on human resources, such as creating guilds, communities of practice, and rewards or incentives. Therefore, performing such investigations is a future opportunity.
- There is a need to analyze which strategies are best in specific contexts (for example, public or private organizations).

The next steps in the research could be regarding how technical debt can be used as a competitive advantage, generating value rather than bringing undesired and costly consequences.

Acknowledgments

The authors thank Dr. Carolyn Seaman for her valuable suggestions and comments. This work was partially supported by CITIC at the University of Costa Rica. Grant No. 834-B4-412.

Appendices

A1. Complete list of included papers

The complete bibliography of the 117 papers analyzed in the full-text review is available at:
https://drive.google.com/file/d/1g8THUuny-SVuhDWBr5A_ScVlTXWcCnta/view?usp=sharing

A2. List of included papers about technical debt identification

The complete list of included papers about technical debt identification and the artifact considered is available at;
https://drive.google.com/file/d/1tXn8Sv6og_N59dhZKjKtcMmAZcl4Ud3e/view?usp=sharing

References

Abad, Z. S. H., & Ruhe, G. (2015). Using real options to manage Technical Debt in Requirements Engineering. *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, 230–235. https://doi.org/10.1109/RE.2015.7320428

Akbarinasaji, S., & Bener, A. (2016). Adjusting the Balance Sheet by Appending Technical Debt. *Proceedings - 2016 IEEE 8th International Workshop on Managing Technical Debt, MTD 2016*, 36–39. https://doi.org/10.1109/MTD.2016.14

Akbarinasaji, S., Bener, A. B., & Erdem, A. (2016). Measuring the principal of defect debt. *Proceedings - 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, RAISE 2016*, 1–7. https://doi.org/10.1145/2896995.2896999

Albarak, M., Bahsoon, R., Ozkaya, I., & Nord, R. L. (2020). Managing Technical Debt in Database Normalization. *IEEE Transactions on Software Engineering*. https://doi.org/10.1109/TSE.2020.3001339

Aldaeej, A., & Seaman, C. (2018). From lasagna to spaghetti, a decision model to manage defect debt. *Proceedings - International Conference on Software Engineering*, 67–71. https://doi.org/10.1145/3194164.3194177

Alfayez, R., Alwehaibi, W., Winn, R., Venson, E., & Boehm, B. (2020). A systematic literature review of technical debt prioritization. *Proceedings - 2020 IEEE/ACM International Conference on Technical Debt, TechDebt 2020*, *10*, 1–10. https://doi.org/10.1145/3387906.3388630

Alfayez, R., & Boehm, B. (2019). Technical Debt Prioritization: A Search-Based Approach. *Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019*, 434–445. https://doi.org/10.1109/QRS.2019.00060

Alves, N. S. R., Mendes, T. S., de Mendonça, M. G., Spinola, R. O., Shull, F., & Seaman, C. (2016a). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, *70*, 100–121. https://doi.org/10.1016/J.INFSOF.2015.10.008

Alves, N. S. R., Mendes, T. S., de Mendonça, M. G., Spinola, R. O., Shull, F., & Seaman, C. (2016b). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, *70*, 100–121. https://doi.org/10.1016/J.INFSOF.2015.10.008

Ampatzoglou, A., Ampatzoglou, A., Avgeriou, P., & Chatzigeorgiou, A. (2015). A Financial Approach for Managing Interest in Technical Debt. *Lecture Notes in Business Information Processing*, *257*, 117–133. https://doi.org/10.1007/978-3-319-40512-4_7

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2015). The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology*, *64*, 52–73. https://doi.org/10.1016/J.INFSOF.2015.04.001

Ampatzoglou, A., Michailidis, A., Sarikyriakidis, C., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., Ampatzoglou, A., Michailidis, A., Sarikyriakidis, C., Chatzigeorigiou, A., & Avgeriou, P. (2018). A Framework for Managing Interest in Technical Debt: An Industrial Validation. *Proceedings of the 2018 International Conference on Technical Debt*, *10*. https://doi.org/10.1145/3194164

Anderson, P., Kot, L., Gilmore, N., & Vitek, D. (2019). SARIF-enabled tooling to encourage gradual technical debt reduction. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 71–72. https://doi.org/10.1109/TECHDEBT.2019.00024

Aversano, L., Bernardi, M. L., Cimitile, M., & Iammarino, M. (2021). Technical Debt predictive model through Temporal Convolutional Network. *Proceedings of the International Joint Conference on Neural Networks*, *2021-July*. https://doi.org/10.1109/IJCNN52387.2021.9534423

Baldassarre, M. T., Lenarduzzi, V., Romano, S., & Saarimäki, N. (2020). On the diffuseness of technical debt items and accuracy of remediation time when using SonarQube. *Information and Software Technology*, *128*, 106377. https://doi.org/10.1016/J.INFSOF.2020.106377

Besker, T., Martini, A., & Bosch, J. (2019). Technical debt triage in backlog management. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 13–22. https://doi.org/10.1109/TECHDEBT.2019.00010

Besker, T., Martini, A., & Bosch, J. (2022). The use of incentives to promote technical debt management. *Information and Software Technology*, *142*, 106740. https://doi.org/10.1016/J.INFSOF.2021.106740

Borup, N. B., Christiansen, A. L. J., Tovgaard, S. H., & Persson, J. S. (2021). Deliberative Technical Debt Management: An Action Research Study. *Lecture Notes in Business Information Processing*, *434 LNBIP*, 50–65. https://doi.org/10.1007/978-3-030-91983-2_5/TABLES/3

Chatzigeorgiou, A., Ampatzoglou, A., Ampatzoglou, A., & Amanatidis, T. (2015). Estimating the breaking point for technical debt. *2015 IEEE 7th International Workshop on Managing Technical Debt, MTD 2015 - Proceedings*, 53–56. https://doi.org/10.1109/MTD.2015.7332625

Chicote, M. (2017). Startups and Technical Debt: Managing Technical Debt with Visual Thinking. *Proceedings - 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups, SoftStart 2017*, 10–11. https://doi.org/10.1109/SOFTSTART.2017.6

Ciancarini, P., & Russo, D. (2020). The Strategic Technical Debt Management Model: An Empirical Proposal. *IFIP Advances in Information and Communication Technology*, *582 IFIP*, 131–140. https://doi.org/10.1007/978-3-030-47240-5_13

Codabux, Z., & Williams, B. J. (2016). Technical debt prioritization using predictive analytics. *Proceedings - International Conference on Software Engineering*, 704–706. https://doi.org/10.1145/2889160.2892643

Consortium for Information & Software Quality. (2022). *Cost of Poor Software Quality in the U.S.: A 2022 Report - CISQ*. https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/

Crespo, Y., Gonzalez-Escribano, A., & Piattini, M. (2021). Carrot and Stick approaches revisited when managing Technical Debt in an educational context. *Proceedings - 2021 IEEE/ACM International Conference on Technical Debt, TechDebt 2021*, 99–108. https://doi.org/10.1109/TECHDEBT52882.2021.00020

da Maldonado, E. S., Abdalkareem, R., Shihab, E., & Serebrenik, A. (2017). An empirical study on the removal of Self-Admitted Technical Debt. *Proceedings - 2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017*, 238–248. https://doi.org/10.1109/ICSME.2017.8

de Leon-Sigg, M., Vazquez-Reyes, S., & Rodriguez-Avila, D. (2020). Towards the use of a framework to make technical debt visible. *Proceedings - 2020 8th Edition of the International Conference in Software Engineering Research and Innovation, CONISOFT 2020*, 86–92. https://doi.org/10.1109/CONISOFT50191.2020.00022

de Lima, B. S., Garcia, R. E., & Eler, D. M. (2022). Toward prioritization of self-admitted technical debt: an approach to support decision to payment. *Software Quality Journal*, 1–27. https://doi.org/10.1007/S11219-021-09578-7/FIGURES/10

Detofeno, T., Malucelli, A., & Reinehr, S. (2021). Technical Debt Guild: When experience and engagement improve Technical Debt Management. *XX Brazilian Symposium on Software Quality*. https://doi.org/10.1145/3493244

di Biase, M., Rastogi, A., Bruntink, M., & van Deursen, A. (2019). The delta maintainability model: Measuring maintainability of fine-grained code changes. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 113–122. https://doi.org/10.1109/TECHDEBT.2019.00030

Fairley, R. E., & Willshire, M. J. (2017). Better Now Than Later: Managing Technical Debt in Systems Development. *Computer*, *50*(5), 80–87. https://doi.org/10.1109/MC.2017.124

Falessi, D., & Reichel, A. (2015). Towards an open-source tool for measuring and visualizing the interest of technical debt. *2015 IEEE 7th International Workshop on Managing Technical Debt, MTD 2015 - Proceedings*, 1–8. https://doi.org/10.1109/MTD.2015.7332618

Fernández-Sánchez, C., Garbajosa, J., Yagüe, A., & Perez, J. (2017). Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. *Journal of Systems and Software*, *124*, 22–38. https://doi.org/10.1016/J.JSS.2016.10.018

Fernandez-Sanchez, C., Humanes, H., Garbajosa, J., & Diaz, J. (2017). An open tool for assisting in technical debt management. *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, 400–403. https://doi.org/10.1109/SEAA.2017.60

Fontana, F. A., Roveda, R., & Zanoni, M. (2016). Tool support for evaluating architectural debt of an existing system: An experience report. *Proceedings of the ACM Symposium on Applied Computing*, *04-08-April-2016*, 1347–1349. https://doi.org/10.1145/2851613.2851963

Freire, S., Rios, N., Mendonça, M., Falessi, D., Seaman, C., Izurieta, C., & Spínola, R. O. (2020). Actions and impediments for technical debt prevention: Results from a global family of industrial surveys. *Proceedings of the ACM Symposium on Applied Computing*, 1548–1555. https://doi.org/10.1145/3341105.3373912

Griffith, I., Taffahi, H., Izurieta, C., & Claudio, D. (2014). A SIMULATION STUDY OF PRACTICAL METHODS FOR TECHNICAL DEBT MANAGEMENT IN AGILE SOFTWARE DEVELOPMENT. *Proceedings of the Winter Simulation Conference 2014*. https://doi.org/10.1109/WSC.2014.7019961

Guo, Y., & Seaman, C. (2011). *A Portfolio Approach to Technical Debt Management*.

Guo, Y., Spínola, R. O., & Seaman, C. (2014). Exploring the costs of technical debt management – a case study. *Empirical Software Engineering 2014 21:1*, *21*(1), 159–182. https://doi.org/10.1007/S10664-014-9351-7

Haas, R., Niedermayr, R., & Juergens, E. (2019). Teamscale: Tackle technical debt and control the quality of your software. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 55–56. https://doi.org/10.1109/TECHDEBT.2019.00016

Holvitie, J., Licorish, S. A., & Leppanen, V. (2016). Modelling Propagation of Technical Debt. *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, 54–58. https://doi.org/10.1109/SEAA.2016.53

Iovino, L., Di, A., Davide, S., Ruscio, D., Pierantonio, A., Salle, A. di, Ruscio, D. di, & Pieran, A. (2020). Metamodel deprecation to manage technical debt in model co-evolution. *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings*, 306–315. https://doi.org/10.1145/3417990.3419625

Izurieta, C., Ozkaya, I., Seaman, C., Kruchten, P., Nord, R., Snipes, W., & Avgeriou, P. (2016). Perspectives on managing technical debt : A transition point and roadmap from Dagstuhl. *CEUR Workshop Proceedings*, *1771*, 84–87.

Izurieta, C., Rice, D., Kimball, K., & Valentien, T. (2018). A Position Study to Investigate Technical Debt Associated with Security Weaknesses. *Proceedings of the 2018 International Conference on Technical Debt*. https://doi.org/10.1145/3194164

Izurieta, C., Rojas, G., & Griffith, I. (2015). Preemptive Management of Model Driven Technical Debt for Improving Software Quality. *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*. https://doi.org/10.1145/2737182

Izurieta, C., Vetrò, A., Zazworka, N., Cai, Y., Seaman, C., & Shull, F. (2012). Organizing the technical debt landscape. *2012 3rd International Workshop on Managing Technical Debt, MTD 2012 - Proceedings*, 23–26. https://doi.org/10.1109/MTD.2012.6225995

Kontsevoi, B., Soroka, E., & Terekhov, S. (2019). TETRA, as a set of techniques and tools for calculating technical debt principal and interest. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 64–65. https://doi.org/10.1109/TECHDEBT.2019.00021

Kosti, M. V., Ampatzoglou, A., Chatzigeorgiou, A., Pallas, G., Stamelos, I., & Angelis, L. (2017). Technical debt principal assessment through structural metrics. *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, 329–333. https://doi.org/10.1109/SEAA.2017.59

Kouros, P., Chaikalis, T., Arvanitou, E. M., Chatzigeorgiou, A., Ampatzoglou, A., & Amanatidis, T. (2019). JCaliper: Search-based technical debt management. *Proceedings of the ACM Symposium on Applied Computing, Part F147772*, 1721–1730. https://doi.org/10.1145/3297280.3297448

Lahti, J. R., Tuovinen, A. P., & Mikkonen, T. (2021). Experiences on Managing Technical Debt with Code Smells and AntiPatterns. *Proceedings - 2021 IEEE/ACM International Conference on Technical Debt, TechDebt 2021*, 36–44. https://doi.org/10.1109/TECHDEBT52882.2021.00013

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., & Arcelli Fontana, F. (2021). A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools. *Journal of Systems and Software*, *171*, 110827. https://doi.org/10.1016/J.JSS.2020.110827

Lenarduzzi, V., Martini, A., Taibi, D., & Tamburri, D. A. (2019). Towards surgically-precise technical debt estimation: Early results and research roadmap. *MaLTeSQuE 2019 - Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, Co-Located with ESEC/FSE 2019*, 37–42. https://doi.org/10.1145/3340482.3342747

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, *101*, 193–220. https://doi.org/10.1016/J.JSS.2014.12.027

Macit, Y., Giray, G., & Tüzün, E. (2020). Methods for Identifying Architectural Debt: A Systematic Mapping Study. *2020 Turkish National Software Engineering Symposium, UYMS 2020 - Proceedings*. https://doi.org/10.1109/UYMS50627.2020.9247070

Malakuti, S., & Heuschkel, J. (2021). The Need for Holistic Technical Debt Management across the Value Stream: Lessons Learnt and Open Challenges. *Proceedings -*

2021 IEEE/ACM International Conference on Technical Debt, TechDebt 2021, 109–113. https://doi.org/10.1109/TECHDEBT52882.2021.00021

Martini, A. (2018). AnaConDebt: A Tool to Assess and Track Technical Debt. *Proceedings of the 2018 International Conference on Technical Debt*. https://doi.org/10.1145/3194164

Martini, A., Besker, T., & Bosch, J. (2016). The introduction of technical debt tracking in large companies. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, *0*, 161–168. https://doi.org/10.1109/APSEC.2016.032

Martini, A., & Bosch, J. (2016). An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt. *Proceedings - International Conference on Software Engineering*, 31–40. https://doi.org/10.1145/2889160.2889224

Martini, A., & Bosch, J. (2017a). The Magnificent Seven: Towards a Systematic Estimation of Technical Debt Interest. *Proceedings of the XP2017 Scientific Workshops*. https://doi.org/10.1145/3120459

Martini, A., & Bosch, J. (2017b). Revealing social debt with the CAFFEA framework: An antidote to architectural debt. *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, 179–181. https://doi.org/10.1109/ICSAW.2017.42

Mendes, E., Wohlin, C., Felizardo, K., & Kalinowski, M. (2020). When to update systematic literature reviews in software engineering. *Journal of Systems and Software*, *167*, 110607. https://doi.org/10.1016/J.JSS.2020.110607

Mera-Gómez, C., Bahsoon, R., & Buyya, R. (2016). Elasticity Debt: A Debt-Aware Approach to Reason About Elasticity Decisions in the Cloud. *Proceedings of the 9th International Conference on Utility and Cloud Computing*. https://doi.org/10.1145/2996890

Mohan, M., Greer, D., & McMullan, P. (2016). Technical debt reduction using search based automated refactoring. *Journal of Systems and Software*, *120*, 183–194. https://doi.org/10.1016/J.JSS.2016.05.019

Nepomuceno, V., & Soares, S. (2019). On the need to update systematic literature reviews. *Information and Software Technology*, *109*, 40–42. https://doi.org/10.1016/J.INFSOF.2019.01.005

Nielsen, M. E., Østergaard Madsen, C., & Lungu, M. F. (2020). Technical Debt Management: A Systematic Literature Review and Research Agenda for Digital Government. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12219 LNCS*, 121–137. https://doi.org/10.1007/978-3-030-57599-1_10

Nielsen, M. E., & Skaarup, S. (2021). IT Portfolio management as a framework for managing Technical Debt; IT Portfolio management as a framework for managing Technical Debt. *14th International Conference on Theory and Practice of Electronic Governance*. https://doi.org/10.1145/3494193

Oliveira, F., Goldman, A., & Santos, V. (2015). Managing Technical Debt in Software Projects Using Scrum: An Action Research. *Proceedings - 2015 Agile Conference, Agile 2015*, 50–59. https://doi.org/10.1109/AGILE.2015.7

Pacheco, A., Marín-Raventós, G., & López, G. (2018). Designing a Technical Debt Visualization Tool to Improve Stakeholder Communication in the Decision-Making Process: A Case Study. *Lecture Notes in Business Information Processing*, *327*, 15–26. https://doi.org/10.1007/978-3-319-99040-8_2

Perez, B., Correal, D., & Astudillo, H. (2019). A proposed model-driven approach to manage architectural technical debt life cycle. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 73–77. https://doi.org/10.1109/TECHDEBT.2019.00025

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008*. https://doi.org/10.14236/EWIC/EASE2008.8

Plösch, R., Bräuer, J., Saft, M., & Körner, C. (2018). Design Debt Prioritization: A Design Best Practice-Based Approach. *Proceedings of the 2018 International Conference on Technical Debt*, *18*. https://doi.org/10.1145/3194164

Ramasubbu, N., & Kemerer, C. F. (2019). Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests. *IEEE Transactions on Software Engineering*, *45*(3), 285–300. https://doi.org/10.1109/TSE.2017.2774832

Reboucas De Almeida, R. (2019). Business-Driven Technical Debt Prioritization. *Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019*, 605–609. https://doi.org/10.1109/ICSME.2019.00096

Reboucas De Almeida, R., Kulesza, U., Treude, C., Cavalcanti Feitosa, D., & Lima, A. H. G. (2018). Aligning technical debt prioritization with business objectives: A multiple-case study. *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018*, 655–664. https://doi.org/10.1109/ICSME.2018.00075

Reboucas De Almeida, R., Treude, C., & Kulesza, U. (2019). Tracy: A Business-Driven Technical Debt Prioritization Framework. *Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019*, 181–185. https://doi.org/10.1109/ICSME.2019.00028

Ribeiro, L. F., Alves, N. S. R., de Mendonca Neto, M. G., & Spinola, R. O. (2017). A strategy based on multiple decision criteria to support technical debt management. *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications*,

SEAA 2017, 334–341. https://doi.org/10.1109/SEAA.2017.37

Rindell, K., Bernsmed, K., & Gilje Jaatun, M. (2019). Managing security in software or: How I learned to stop worrying and manage the security technical debt. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3339252.3340338

Rios, N., Mendonça Neto, M. G. de, & Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, *102*, 117–145. https://doi.org/10.1016/J.INFSOF.2018.05.010

Rios, N., Spinola, R. O., de Mendonça Neto, M. G., & Seaman, C. (2019). Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 3–12. https://doi.org/10.1109/TECHDEBT.2019.00009

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 2020 25:5*, *25*(5), 3216–3287. https://doi.org/10.1007/S10664-020-09832-9

Shapochka, A., & Omelayenko, B. (2016). Practical Technical Debt Discovery by Matching Patterns in Assessment Graph. *Proceedings - 2016 IEEE 8th International Workshop on Managing Technical Debt, MTD 2016*, 32–35. https://doi.org/10.1109/MTD.2016.7

Sharma, T. (2019). How deep is the mud: Fathoming architecture technical debt using designite. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 59–60. https://doi.org/10.1109/TECHDEBT.2019.00018

Snipes, W., & Ramaswamy, S. (2018). A Proposed Sizing Model for Managing 3rd Party Code Technical Debt. *Proceedings of the 2018 International Conference on Technical Debt*, *18*. https://doi.org/10.1145/3194164

Stochel, M. G., Cholda, P., & Wawrowski, M. R. (2020). Continuous Debt Valuation Approach (CoDVA) for Technical Debt Prioritization. *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, 362–366. https://doi.org/10.1109/SEAA51224.2020.00066

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. *Journal of Systems and Software*, *86*(6), 1498–1516. https://doi.org/10.1016/J.JSS.2012.12.052

Tornhill, A. (2018). Assessing technical debt in automated tests with codescene. *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*, 122–125. https://doi.org/10.1109/ICSTW.2018.00039

Tornhill, A., & Ab, E. (n.d.). Prioritize Technical Debt in Large-Scale Systems using CodeScene. *Proceedings of the 2018 International Conference on Technical Debt*, *18*. https://doi.org/10.1145/3194164

Trumler, W., & Paulisch, F. (2016). How "Specification by Example" and Test-Driven Development Help to Avoid Technial Debt. *Proceedings - 2016 IEEE 8th International Workshop on Managing Technical Debt, MTD 2016*, 1–8. https://doi.org/10.1109/MTD.2016.10

Vidal, S., Vazquez, H., Diaz-Pace, J. A., Marcos, C., Garcia, A., & Oizumi, W. (2016). JSpIRIT: A flexible tool for the analysis of code smells. *Proceedings - International Conference of the Chilean Computer Science Society, SCCC, 2016-February*. https://doi.org/10.1109/SCCC.2015.7416572

von Zitzewitz, A. (2019). Mitigating technical and architectural debt with sonargraph. *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 66–67. https://doi.org/10.1109/TECHDEBT.2019.00022

Wiese, M., Riebisch, M., & Schwarze, J. (2021). Preventing Technical Debt by Technical Debt Aware Project Management. *Proceedings - 2021 IEEE/ACM International Conference on Technical Debt, TechDebt 2021*, 84–93. https://doi.org/10.1109/TECHDEBT52882.2021.00018

Wolfart, D., Assunção, W. K. G., & Martinez, J. (2021). Variability Debt: Characterization, Causes and Consequences. *SBQS '21: Proceedings of the XX Brazilian Symposium on Software Quality*. https://doi.org/10.1145/3488042.3488048

Xiao, L., Cai, Y., Kazman, R., Mo, R., & Feng, Q. (2016). Identifying and Quantifying Architectural Debt. *Proceedings of the 38th International Conference on Software Engineering*. https://doi.org/10.1145/2884781

Xiao, T., Wang, D., Mcintosh, S., Hata, H., Kula, R. G., Ishio, T., & Matsumoto, K. (2021). Characterizing and Mitigating Self-Admitted Technical Debt in Build Systems. *IEEE Transactions on Software Engineering*, 1–1. https://doi.org/10.1109/TSE.2021.3115772

Yli-Huumo, J., Maglyas, A., Smolander, K., Haller, J., & Törnroos, H. (2016). Developing Processes to Increase Technical Debt Visibility and Manageability – An Action Research Study in Industry. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10027 LNCS*, 368–378. https://doi.org/10.1007/978-3-319-49094-6_24

Zampetti, F., Serebrenik, A., & di Penta, M. (2020). Automatically Learning Patterns for Self-Admitted Technical Debt Removal. *SANER 2020 - Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution, and Reengineering*, 355–366. https://doi.org/10.1109/SANER48275.2020.9054868