The Journal of Systems & Software 196 (2023) 111554

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

In practice

Software practitioners' point of view on technical debt payment ☆

Sávio Freire [a,b], Nicolli Rios [c], Boris Pérez [d,e], Camilo Castellanos [d], Darío Correal [d], Robert Ramač [f], Vladimir Mandić [f], Nebojša Taušan [f], Gustavo López [g], Alexia Pacheco [g], Manoel Mendonça [a], Davide Falessi [h], Clemente Izurieta [i], Carolyn Seaman [j], Rodrigo Spínola [k,l,*]

[a] Federal University of Bahia, Salvador, Brazil
[b] Federal Institute of Ceará, Morada Nova, Brazil
[c] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[d] University of Los Andes, Bogotá, Colombia
[e] Francisco de Paula Santander University, Cúcuta, Colombia
[f] University of Novi Sad, Novi Sad, Serbia
[g] University of Costa Rica, San José, Costa Rica
[h] University of Rome Tor Vergata, Rome, Italy
[i] Montana State University, Bozeman, MT, United States
[j] University of Maryland Baltimore County, Baltimore, MD, United States
[k] Salvador University, Salvador, Brazil
[l] Virginia Commonwealth University, Richmond, VA, United States

ARTICLE INFO

Article history:
Received 6 December 2021
Received in revised form 20 October 2022
Accepted 31 October 2022
Available online 5 November 2022

Keywords:
Technical debt
Technical debt payment
Technical debt management

ABSTRACT

Context: Technical debt (TD) payment refers to the activity of expending maintenance effort and resources to make up for the effects of previous technical compromises.
Aims: To investigate if software practitioners have paid debt items off in their projects, the practices that have been used for paying off debt items, and the issues that hamper the implementation of these practices.
Method: We analyze 653 responses collected by surveying practitioners from six countries about TD payment.
Results: Practitioners have not paid off TD items in most cases. We identified 27 reasons for not paying off those items and 32 payment-related practices. Practices are mainly related to internal quality issues, while reasons for not paying TD off are mostly associated with planning and management issues. Lastly, we identified relationships between practices and between reasons, indicating that both can appear in combination.
Conclusion: . We use different views to consolidate the set of information on TD payment, extending the conceptual model for TD and organizing the set of practices and reasons into a TD payment map. We believe that the model and the map can support practitioners in planning their TD payment strategy.

1. Introduction

Technical debt (TD) contextualizes technical compromises that can bring short-term benefits (e.g., higher productivity and lower costs) but may negatively impact the long-term health of software projects (Guo and Seaman, 2011; Izurieta et al., 2012). The negative impacts yield risks associated with, among others, unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project (Spínola et al., 2013; Rios et al., 2020b).

TD can be a good investment as long as the project team is aware of its presence and the increased risks it imposes on the

☆ Editor: Dr Daniel Mendez.

* Corresponding author at: Virginia Commonwealth University, Richmond, VA, United States.
E-mail addresses: savio.freire@ifce.edu.br (S. Freire), nicolli@cos.ufrj.br (N. Rios), br.perez41@uniandes.edu.co (B. Pérez), cc.castellanos87@uniandes.edu.co (C. Castellanos), dcorreal@uniandes.edu.co (D. Correal), ramac.robert@uns.ac.rs (R. Ramač), vladman@uns.ac.rs (V. Mandić), nebojsa.tausan@ef.uns.ac.rs (N. Taušan), gustavo.lopezherrera@ucr.ac.cr (G. López), alexia.pacheco@ucr.ac.cr (A. Pacheco), manoel.mendonca@ufba.br (M. Mendonça), falessi@ing.uniroma2.it (D. Falessi), clemente.izurieta@montana.edu (C. Izurieta), cseaman@umbc.edu (C. Seaman), spinolaro@vcu.edu (R. Spínola).

https://doi.org/10.1016/j.jss.2022.111554
0164-1212/

project (Kruchten et al., 2012). Through TD management (TDM), project teams can handle these effects, decreasing risks imposed by the presence of debt. If effectively managed, TD can help the project achieve its goals sooner or more cheaply (Guo et al., 2014). On the other hand, if debt items are unmanaged, they can cause financial and technical problems, increasing software maintenance and evolution costs, leading to a situation where the whole future of the software product is jeopardized (Nord et al., 2012; Martini et al., 2014).

TDM is comprised of activities that include the identification, measurement, prioritization, prevention, monitoring, documentation, communication, visualization, time-to-market analysis, scenario analysis, and payment of debt items (Li et al., 2015; Rios et al., 2018b). TD payment refers to the activity of expending maintenance effort and resources to make up for the effects of previous decisions to incur technical debt (Rios et al., 2018b). For example, a TD item that was incurred because shortcuts were taken during testing can be paid by carrying out the testing that was previously skipped. TD payment also refers to the strategies for supporting decision-making about the most appropriate time to pay debt items off and the practices to pay off debt items (Rios et al., 2018b). For example, by using the *portfolio management* strategy (Guo and Seaman, 2011; Seaman et al., 2012), TD items are organized as assets and populate the portfolio of the software development organization. Each item is analyzed in an iterative way to decide whether this item will or will not be paid. The decision is based on risk assessment. Once a decision is taken, a payment practice (e.g., code refactoring, design refactoring, testing) can be employed to pay off the debt item.

Related work has sought to identify strategies, practices, and tools for TD payment (see Section 2.2.1). For example, Samarthyam et al. (2017), Toledo et al. (2019), and Rios et al. (2020a) identified practices to pay off test, architectural, and documentation debt items, respectively. Li et al. (2015) and Behutiye et al. (2017) identified a set of categories of TD payment practices from a systematic mapping study and a systematic literature review, respectively. Yli-Huumo et al. (2014), Abad et al. (2016) and Bomfim and Santos (2017) performed case studies, while Apa et al. (2020) conducted a survey to investigate practices used in TD payment. Only the case study from Bomfim and Santos (2017) investigated reasons for the non-payment of debt items. These reasons merit further study, as the potential impediments to payment strategies (or to payment in general) are important factors for a team to understand when designing their own TD payment strategy. Choosing payment practices without understanding and planning for the inherent risks to TD payment is likely to result in failure to execute the plan.

We also have investigated TD payment in our previous work (see more details in Section 2.2.3) by using survey data collected by the *InsighTD* Project,[1] which is a globally distributed family of industrial surveys on TD causes, effects, and management (Rios et al., 2020b). In Freire et al. (2020a), we present a set of TD payment practices and reasons for TD non-payment by analyzing 432 answers from practitioners from the Brazilian, Chilean, Colombian, and North American software industries. Using the same data set, Pérez et al. (2020) investigate the relation between TD payment-related practices and types of debt. Pérez et al. (2021) analyze a subset of this same data set, consisting of the 72 answers provided by software architects, revealing the software architect perspective on TD payment. Finally, Freire et al. (2021c) investigate these same themes in an overlapping *InsighTD* data set that adds responses from Costa Rica and Serbia, but is restricted to respondents practicing an Agile approach.

[1] http://www.td-survey.com/.

Although related work (including our own) provides evidence about the nature of TD payment, much remains to be investigated. First of all, much of our previous work using *InsighTD* data on this subject was performed on more restricted data sets than is being used in this study. Extending our prior work with a larger and broader data set helps to show that decisions made by practitioners in different roles, following different development models, and in different countries, are likely to generalize, and further likely to generalize to real-world scenarios. This increases ecological validity (Andrade, 2018), as well as confidence in the validity of our results (Wohlin et al., 2012). We had to extend our work to include answers from all practitioners to help validate this.

Further, previous analyses of the *InsighTD* data that address TD payment (summarized above and detailed in Section 2.2.3) reported broad analyses that did not provide the depth needed for actionable guidance for practitioners. Software teams, when designing their own TD payment strategies, need evidence of what payment practices are available (including those they have not considered in the past), what factors should be taken into account when choosing between different practices, and what obstacles might arise that would hinder TD payment. With a rich body of empirical work that addresses all of these aspects in detail, a team can plan a strategy that incorporates risk planning and novel techniques, and is thus more likely to succeed.

For example, having information on whether a practice or reason for TD non-payment is managerial or technical (as we have included in our analysis) can clarify the importance of managerial vs. technical decisions to eliminate TD items (or not) in a particular project context. This helps a team understand the organizational context around TD payment. Investigating which practices are used together or which reasons for non-payment of TD are considered in combination can help software practitioners identify new practices to eliminate debt or identify reasons they had not considered that hinder TD payment initiatives, respectively. Finally, organizing all of this body of knowledge can provide valuable artifacts that practitioners and researchers can use to drive their activities.

This work significantly extends our previous work by investigating when software practitioners have or have not paid debt items off in their projects, the practices that have been used for paying off debt items, and the issues that hamper the implementation of these practices. We use a data set composed of 653 answers from practitioners from Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States. This work not only uses a larger data set than previous work, but also runs a broader analysis of the data, expanding our previous work by providing:

- a comprehensive list of TD payment practices and reasons for TD non-payment supported by a larger corpus of data, improving the confidence of our results (Wohlin et al., 2012).
- an analysis of practices and reasons for TD non-payment, revealing their types, nature, categories, and relationships with types of debt.
- an extension of the conceptual model for TD initially proposed by Avgeriou et al. (2016) and Izurieta et al. (2016) and evolved by Rios et al. (2018b), including the concepts of reasons and payment-related practices and their types, nature, and categories.
- a TD payment map that organizes the set of payment practices and non-payment reasons, supporting the practical use of the results by software practitioners.
- an assessment of the TD conceptual map and the TD payment map by considering the perception of software practitioners.

- a set of correlations among non-payment reasons and among payment practices.
- takeaways to drive practitioners in their TD payment initiatives.

Our results show that practitioners have not paid off TD items in most cases, and we have extracted a total of 27 reasons for this. By grouping the reasons, we identified that *planning and management* and *organizational issues* are the categories that most hamper the application of TD payment practices. We found that some of the identified reasons are factors that influenced decisions not to pay off debt, while others were insurmountable obstacles to TD payment. We also found correlations between many reasons, indicating that they often do not appear in isolation. We also identified 32 TD payment-related practices. Like the reasons, the practices are also often related to each other and are used in combination.

This work has several contributions for practitioners and researchers. Practitioners can use the list of TD payment practices to decide which ones better fit their needs. The list of reasons that hamper the application of these practices can support practitioners to foresee difficulties and better plan for TD payment in their work environment. Understanding practices and reasons together allow practitioners to develop well-formed TD payment strategies, that include not only practices but plans for addressing potential risks. For example, *code refactoring* is the most cited practice. But our results also indicate to a team that *lack of time* is a common hindrance to successfully carrying out refactoring. If the team determines that they do not have sufficient time, then our results also point to alternative strategies, such as *negotiating a deadline extension*. In this example, the team can use our results (packaged into the map presented later) to identify both potential obstacles to successfully carrying out their plan, as well as alternative strategies. Further, practitioners can use the TD payment map to identify the reasons and practices that are related to each other. For example, if a team normally performs *code refactoring* to pay off TD items, our TD payment map will show that practices such as *investing effort in testing activities* and *prioritizing TD items* are used to support refactoring activities. This helps the team make decisions about further process improvements that will contribute to better TD payment strategies. Similarly, if a team perceives that *cost* is a reason for TD non-payment, the TD payment map can reveal that *complexity of the project* and *complexity of the TD item* are correlated reasons that can also affect the project, which again allows the team to have a fuller understanding as a basis for making process improvements. Lastly, in addition to the map of TD payment practices and reasons, we summarize the main results of our work in a set of takeaways which can drive software teams in planning and performing TD payment in their projects. For researchers, our results provide a grounded view of software industry needs with respect to TD payment. The new version of the conceptual model for TD, the TD payment map, and situations describing the correlation among practices and among reasons can guide new research efforts aligned with the demands and current context of TD payment as experienced by practitioners.

This paper is organized into seven other sections. Section 2 presents the background, addressing TD payment and related work. Section 3 presents the research method. Then, Section 4 presents results reached. After, Section 5 discusses the main findings, presents the TDM landscape organized according to the results, presents the assessment of the TDM landscape, and shows the implications of the study for researchers and practitioners. Section 6 compares our findings to ones reported by related work. Section 7 discusses the threats to the study validity. Lastly, Section 8 presents the final remarks and discusses the next steps of this work.

2. Background

We start this section with an introduction to TD payment. Then, we describe a number of studies that looked at the following central topics approached in our work: TD payment practices and reasons for not paying off TD. Lastly, we briefly describe our own previous work (Freire et al., 2020a; Pérez et al., 2020, 2021; Freire et al., 2021c).

2.1. Technical debt payment

Technical debt management (TDM) is decisive for increasing the success of software projects (Seaman and Guo, 2011). TDM includes activities supporting decisions about the need and the appropriate time to pay off debt items from a software project (Guo et al., 2014). These decisions can cause negative and positive impacts on a project. A positive impact can be achieving project goals sooner, while negative impacts can be increasing cost and technical problems (Rios et al., 2020b).

TDM is composed of the following activities: identification, measurement, prioritization, prevention, monitoring, payment, documentation, communication, visualization, time-to-market analysis, and scenario analysis (Li et al., 2015; Rios et al., 2018b). More specifically, TD payment refers to the activities undertaken with the goal of supporting decision-making about the most appropriate time to pay off debt items (Rios et al., 2018b) or practices for paying off TD items (Li et al., 2015). An example of a payment practice commonly discussed in the literature is *refactoring* (Yli-Huumo et al., 2014; Li et al., 2015; Abad et al., 2016; Behutiye et al., 2017). Using this practice, software practitioners can pay down TD by restructuring the codebase or system architecture without altering the external behavior of the software system under development (Behutiye et al., 2017). An example of a strategy for TD payment is *portfolio management* (Guo and Seaman, 2011; Seaman et al., 2012). It allows for TD items to be organized as assets and populates the portfolio of the software development organization. Using risk assessment, each item is analyzed to decide whether that item will or will not be paid off.

2.2. Related work

In this section, we present related work on practices used for paying off TD items and reasons for the non-application of these practices.

2.2.1. Research on TD payment practices

Yli-Huumo et al. (2014) conducted a case study in one middle-size Finnish software company with two independent product lines. The authors intended to identify the sources of TD and the practices and strategies for TDM. Specifically, research question "RQ2: What management and reduction strategies/practices are being used for technical debt?" was related to TD payment. The data collection was performed through 11 interviews with practitioners. The authors identified that *refactoring*, *bug fixing days*, *code reviews*, *coding standards and guides*, and *communication structure between business* are practices used for paying off and preventing TD. Although our work also investigates the practices used to eliminate the debt, we also approach the reasons for TD non-payment.

Li et al. (2015) conducted a systematic mapping study on TD and its management. One of the research questions was "RQ7: What approaches are used in each TDM activity?". They organized TD payment approaches into seven categories, such as *refactoring*, *rewriting*, *automation*, *reengineering*, *repackaging*, *bug fixing*, and *fault tolerance*. They highlighted that refactoring was the most

used approach. Our work also categorizes the practices for TD payment, but it also reports and categorizes the reasons for TD non-payment. As this set of practices and reasons for TD non-payment was identified from a practitioners' point of view, they tend to be closer to the state of practice of TD payment.

In other related work, Abad et al. (2016) reported results from two studies. The first was based on in-depth interviews with stakeholders about development, testing, and product management teams. The second was a quantitative survey on effort estimation, quality management processes, and understanding of the risks and benefits of TD. Performing grounded theory, the authors aimed to respond to "RQ3: What suggestions do testers and developers have for reducing TD in their projects?" From 48 responses, the authors found that *allocating more resources (time, budget, and infrastructure)* was considered the most appropriate solution for paying off TD items. The other solutions found were *technical solutions (e.g., refactoring, using design patterns, and improving design and architecture)*, *prioritizing TD*, *having more flexible schedules*, and *quantifying TD*. Our study also investigates TD payment practices, but differently, it is not limited to developers' and testers' points of view.

Samarthyam et al. (2017) provided an overview on test debt, presenting factors that lead software teams to incur test debt items, strategies for managing them, and two case studies on managing them in real-world projects. Regarding TD payment, the authors indicated that the following practices should be used for paying off test debt items: *pair programming*, *clean coding*, and *refactoring*. We also investigated TD payment practices, but taking into consideration the different types of debt reported in the technical literature (Rios et al., 2018b).

Behutiye et al. (2017) performed a systematic literature review. Through research question "RQ3: What are the strategies proposed in the literature to manage TD in agile software development?", the authors sought to identify TDM strategies for controlling TD in an agile context. These strategies were divided into 12 categories, such as *specific approaches*, *tools*, and *models* to manage TD in agile software development. The authors indicated that *refactoring* is the most popular TDM strategy. Also in 2017, Bomfim and Santos (2017) investigated strategies and TD reduction practices adopted by agile teams through a case study. The authors interviewed six agile teams from four companies. In each team, only its Scrum Masters and/or technical leaders participated in the interview. The authors reported the following practices for reducing TD: *splitting methods to make them more reusable*, *using design patterns*, *refactoring older code*, and *using palliative solutions*. Differently, our study does not consider only the point of view of agile software development practitioners. We go further into the analyses by investigating, among other things, correlations among non-payment reasons and among payment practices.

More recently, Toledo et al. (2019) investigated the relationship between architecture TD and microservices performing an exploratory case study in a real-life project. The authors identified a set of payment practices responding to the research question "RQ1.2: What is a solution for the identified ATD in microservices and its associated refactoring cost (principal)?". The list is composed of 13 practices as, for example, *rewrite the communication layer*, *migrate the services to use the new architecture*, and *remove the business logic inside the communication layer*. Our work is not limited to investigating the practices used to eliminate architecture TD items. Apa et al. (2020) investigated the startup's characteristics that can lead to TD incurrence and the concepts related to TD and its management. The authors presented the results of a survey performed in Uruguayan software startups. Based on 33 responses, they indicated that *refactoring*, *redesign*, and *rewrite of code* are the actions used for reducing TD in Uruguayan software startups. Differently, our work is not limited to startups, but rather investigates the TD payment practices used in the software industry in general.

In other recent work in this area, Rios et al. (2020a) presented the results of the triangulation of the results of two complementary studies on causes, effects, preventive, and payment actions related to documentation debt. From answering the research question "RQ5: How can development teams react to the presence of documentation debt?", the authors found the following payment practices: *adopt TD payment prioritization criteria*, *keep the documentation updated*, and *review outdated documentation*. Our work is not limited to investigating TD payment practices to eliminate documentation debt items. Indeed, we take into consideration the different types of debt reported in the technical literature (Rios et al., 2018b).

Thus, there is already evidence in the technical literature that reveals TD payment practices. Table 1 summarizes relevant information about these articles, reporting whether they found TD payment practices or categories of them, types of debt related to them, the representativeness of each study in terms of the sample size and the number of organizations represented by the sample, and the research method used. We notice that the sample sizes tend to be quite small. The ones with the largest sample are systematic reviews or mapping studies (Li et al., 2015; Behutiye et al., 2017), which are still limited by the number of different organizations (Abad et al., 2016). Besides, one-third of related work focused on only one type of debt (architecture, test, or documentation debt), when the technical literature (Rios et al., 2018b) has reported 15 different types.

2.2.2. Research on reasons for not paying off TD

Bomfim and Santos (2017) investigated what factors influence the non-payment of TD payment, in the context of agile software projects. To this end, the authors performed a case study with six participants and four software organizations. They found that *members engagement by software quality*, *company image degradation risk*, *compliance with contractual clauses and/or end customer requirement*, and *opportunity for software improvements* are factors that influence the payment of TD items. However, *concern of impacting some module because the team does not know all parts of the code to carry out a deeper impact analysis*, *lack of test coverage or excessive manual testing*, *low impact for business,* and *high effort* discourage the application of payment practices. Our work also investigates the reasons for TD non-payment, but it is not limited to agile software development projects.

Overall, the findings presented in the related work reveal the point of view of a small number of practitioners and organizations. We triangulated its findings with our results and observe how they complement each other in Section 6.

2.2.3. Prior work

The context of this work is the *InsighTD* project, which is a globally distributed family of industrial surveys on TD. The project seeks to organize an open and generalizable set of empirical data on the practical problems of TD (Rios et al., 2020b). Its design supports the replications of the survey in different countries. At the time of this writing, the project has concluded data collection for the *InsighTD* replications in Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States. Results from *InsighTD* have been disseminated in the technical literature on causes and effects of TD (Rios et al., 2018a, 2019a,b; Pérez et al., 2019; Ramač et al., 2020; Rios et al., 2020a,b; Souza et al., 2020; Berenguer et al., 2021; Freire et al., 2021a; Ramač et al., 2022a), TDM (Pacheco et al., 2019; Freire et al., 2020a,b; Mandić et al., 2020; Ramač et al., 2020; Freire et al., 2021c; Mandić et al., 2021; Pérez et al., 2020, 2021; Rios et al., 2021; Rocha et al., 2021; Barbosa et al., 2022;

Table 1
Limitations of related work on TD payment practices.

Related work	TD payment		Type of debt	Representativeness		Research method
	Practice	Category		Sample size	Number of organizations	
Yli-Huumo et al. (2014)	Yes	No	General	11	1	Case study
Li et al. (2015)	No	Yes	General	–	–	Systematic review
Abad et al. (2016)	Yes	No	General	48	1	Case study
Samarthyam et al. (2017)	Yes	No	Test	–	–	Overview
Behutiye et al. (2017)	No	Yes	General	–	–	Systematic review
Bomfim and Santos (2017)	Yes	No	General	6	4	Case study
Toledo et al. (2019)	Yes	No	Architecture	1	1	Case study
Apa et al. (2020)	Yes	No	General	33	–	Survey
Rios et al. (2020a)	Yes	No	Documentation	4	1	Case study

Berenguer et al., 2022; Ramač et al., 2022b; Rocha et al., 2022), and the relation between TD effects and TD payment (Freire et al., 2021b). The interested reader is welcome to visit http://www.td-survey.com/publication-map and examine the picture we are painting with the project data.

In our first investigation on the practices for reducing TD items and the reasons for the non-application of these practices (Freire et al., 2020a), we analyzed the *InsighTD* data set formed by 432 answers collected in Brazil, Chile, Colombia, and the United States. We verified that TD payment seems to not be a common practice adopted by software teams. Concerning TD payment-related practices, we identified a set of 34 practices divided into four types: *payment practice, defining a favorable setting for TD payment, TD prevention,* and *TD prioritization*. Also, we identified 28 reasons for the non-application of these practices, divided into two categories: *decision factor* and *impediment*. A decision factor refers to reasons for deciding not to pay off the TD while an impediment points to situations in which the development team wanted to pay off the TD, but they could not pay it off for some reason. We also noticed that practices and reasons can be related to technical or non-technical software activities. Lastly, we organized the set of practices and reasons into eight categories and represented all this information in a conceptual map.

In other previous work, we performed an initial investigation of TD payment-related practices and their relationship with types of debt (Pérez et al., 2020). By analyzing the same data set used by Freire et al. (2020a), we pointed out that practices related to code and design debt are more similar among them. Additionally, we investigated if the practices are the same with respect to the system size and age, revealing that *refactoring* is more commonly used in small and large software systems and systems that are less than one year old. An investigation of the relationship between practices and TD causes was also performed. We found that *refactoring* is the main practice to pay off TD items regardless of the TD cause.

More recently, we investigated the point of view of 72 software architects on TD payment (Pérez et al., 2021). By filtering only answers given to software architects in the *InsighTD* survey from Brazilian, Chilean, Colombian, and North American software industries, we found that *refactoring* and *design improvements* were the most cited practices used by architects for paying off debt items. Lastly, Freire et al. (2021c) analyzed the practices and reasons for TD non-payment used by 274 agile software practitioners collected by *InsighTD* replications performed in Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States. To support the use of those practices and reasons, we organized them into IDEA (*impediments, decision factors, enabling practices, and actions*) diagrams.

Fig. 1 summarizes the results we have assembled on our previous work with *InsighTD* data on TD payment. All works are complementary to each other. Freire et al. (2020a) presented the list of practices and reasons for TD non-payment in general and a specialization of this list considering the agile process model is discussed in Freire et al. (2021c). Similarly, the works conducted by Pérez et al. (2020) and Pérez et al. (2021) presented the relationships between TD payment practices and causes of debt, and the point of view of software architects on TD payment, respectively. Although these results reveal evidence on TD payment practices and reasons for not applying them, in this paper we extend them, thus increasing the confidence of the results by adding another 224 answers collected in *InsighTD* replications conducted in Costa Rica and Serbia. Moreover, we performed a full data analysis which significantly improves on prior work, including:

1. A detailed analysis of the TD payment-related practices and reasons for non-application of these practices including:
    a. the updated list of identified practices/reasons ranked by the most cited (see Sections 4.2 and 4.3).
    b. the relationship between types of debt and practices/reasons (see Sections 4.2.1 and 4.3.1).
    c. the correlation between practices and between reasons. This correlation demonstrates what practices or reasons are used together for paying TD items or not in software projects, respectively (see Sections 4.2.2 and 4.3.2).
2. An extension of the conceptual model for TD (Rios et al., 2018b) adding our findings on TD payment (see Section 5.2.1).
3. Updates to the conceptual map initially proposed by Freire et al. (2020a) (see Section 5.2.2).
4. An assessment of the conceptual model and the TD payment map (see Section 5.2.3).
5. A set of possible interpretations of correlations among practices and among reasons (see Section 5.2.4).
6. Triangulation of the results with related work (see Section 6).
7. A set of takeaways for practitioners to support them in their TD payment initiatives (see the boxes presented in Sections 4 and 5.3).

This added analysis and presentation will help drive new research efforts by furthering the understanding of TD payment practices and the reasons for not applying these practices from the practitioners' perspective. It also helps practitioners to make use of the body of knowledge on TD payment to take steps to enable future TD payment. In the next section we discuss our research questions and procedures for data collection and analysis.

3. Research method

This section presents the research questions posed in this work and discusses our data collection and data analysis procedures.

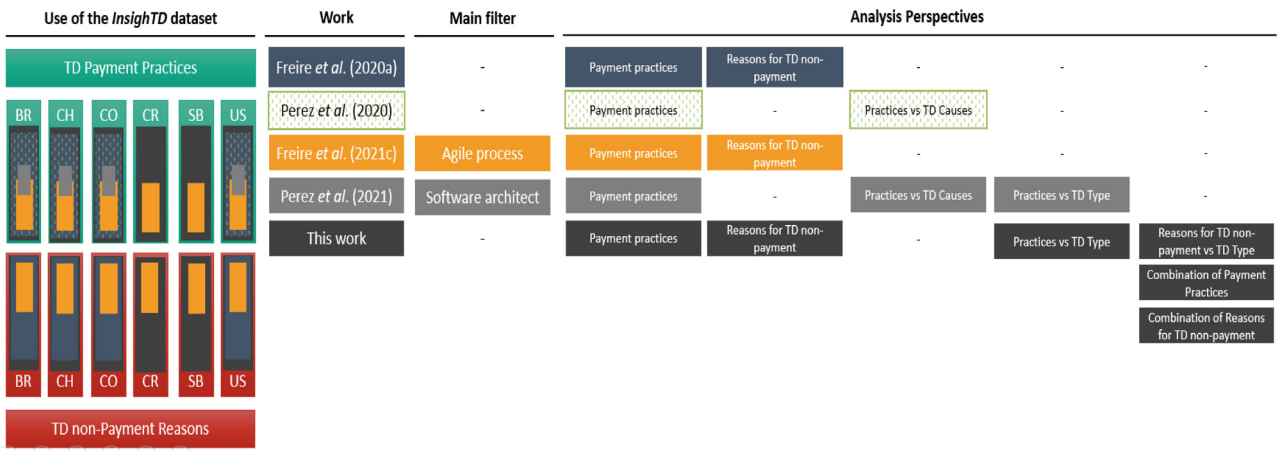


Fig. 1. Summary of our previous work using data from the *InsighTD* project.

3.1. Research questions

To better understand TD payment in practice, we address the following research questions (RQs):

- **RQ1**: *What are the TD payment-related practices used by software practitioners to deal with technical debt items in their projects?* This question aims to investigate the practices (techniques, strategies, tools, etc.) used to pay TD items in software industry. These results will give us a deeper understanding of what TD payment practices practitioners have available to them and which they prefer to use.
- **RQ2**: *What are the reasons considered by software practitioners for not paying off TD?* The purpose of this question is to investigate the possible reasons that hinder the implementation of TD payment practices in software industry.

3.2. Data collection

This work uses data collected by the *InsighTD* survey. The survey is composed of 28 questions, as previously described in Rios et al. (2020b). Table 2 presents the subset of these questions we use in this work together with their descriptions, answer options (if a closed question), and type.

In Q1 thru Q8, the participants characterize themselves and their work context. In Q10, the participant defines TD in his/her words and describes an example of a TD item that occurred in his/her software project (this example is used as a basis for answering TD payment questions) in Q13. Lastly, the discussion of participants' experiences with TD payment are captured in questions Q26 and Q27. The answers from Q26 and Q27 allowed us to answer RQ1 and RQ2.

The data collection strategy was to send e-mail invitations for the survey to software practitioners from Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States. In all cases, we followed the same strategy, using LinkedIn, industry-affiliated member groups, mailing lists, and industry partners as invitation channels. The data-gathering stage was done in 2018 in Brazil and the United States, and 2019–2020 in Chile, Colombia, Costa Rica, and Serbia.

All collected answers were validated by at least three researchers in each of the six replications. Two researchers separately validated the answers given to Q10, Q13, and Q27, and a third researcher resolved possible disagreements. The determination of a "valid answer" was based on the following acceptance criteria:

- *The participants should consider a TD perspective to answer the questions*. We analyzed the definition of TD described by the participants in Q10. If the definition was in conformance with the definition[2] of TD used in the *InsighTD* project (Rios et al., 2020b), then we moved to the second acceptance criterion.
- *The participants should provide a valid example of a TD item*. We analyzed the example of TD described by the participants in Q13. If the example was in conformance with the definition of TD used in *InsighTD* project (Rios et al., 2020b), then we concluded that the participant answered the other survey's questions considering the TD perspective. For example, answers like "I don't know what TD means" and "Errors that arise while fixing other issues" were discarded because they do not represent valid TD examples. On the contrary, answers from participants who provided examples of TD items like "Tests planned but not executed" and "Hard maintenance and future change due to poor documentation from the development team" passed to the next validation step.
- *The participants should provide valid answers to questions on TD payment*. We analyzed all answers given for Q27 to verify whether we can identify payment practices or reasons for TD non-payment. As we did not find any invalid answer, we concluded that the participants did not misunderstand this question. For example, we considered the answers given by a participant who reported that "the identified tech debt has been resolved through updated designs and refactors..." are TD payment practices.

3.3. Data analysis

As shown in Table 2, the questionnaire is composed of both open and closed questions. For closed questions, we used descriptive statistics and calculated the share of participants choosing each option. These strategies were used for the characterization questions. To analyze the open-ended questions, we applied qualitative data analysis techniques (Seaman, 1999; Strauss

[2] The TD definition used in *InsighTD* project is adapted from McConnell (2007): "Technical debt contextualizes the problem of outstanding software development tasks (for example, tests planned but not executed, pending code refactoring, pending documentation update, use of bad design practices, code that does not exhibit good coding practices) as a kind of debt that brings a short-term benefit to the project (normally in terms of higher productivity or shorter release time of software versions), that may have to be paid later in the development process with interest (for example, a poorly designed class tends to be more difficult and costly to maintain than if it had been implemented good object-oriented practices)".

Table 2
Subset of the *InsighTD* survey's questions related to TD payment.
Source: Adapted from Rios et al. (2020b).

No.	Question (Q) Description	Answer options	Type
Q1	What is the size of your company?	• 1–10 employees • 11-50 employees • 51–250 employees • 251–500 employees • 501–1000 employees • 1001–2000 employees • More than 2000 employees.	Closed
Q2	In which country you are currently working?	All countries in the world	Closed
Q3	What is the size of the system being developed in that project? (LOC)	• Less than 10KLOC • 10–100KLOC • 100KLOC–1MLOC • 1–10MLOC • 10+ MLOC	Closed
Q4	What is the total number of people of this project?	• Less than 5 people • 5–9 people • 10–20 people • 21–30 people • More than 30 people	Closed
Q5	What is the age of this system up to now or to when your involvement ended?	• Less than 1 year • 1-2 years • 2-5 years • 5-10 years • More than 10 years	Closed
Q6	To which project role are you assigned in this project?	• Business analyst • DBA/Data analyst • Developer • Process analyst • Project leader/Project manager • Requirements analyst • Software architect • Test manager/Tester • Other:	Closed
Q7	How do you rate your experience in this role?	• Novice (Minimal or "textbook" knowledge without connecting it to practice) • Beginner (Working knowledge of key aspects of practice) • Competent (Good working and background knowledge of area of practice) • Proficient (Depth of understanding of discipline and area of practice) • Expert (Authoritative knowledge of discipline and deep tacit understanding across area of practice)	Closed
Q8	Which of the following most closely describes the development process model you follow on this project?	• Agile (a lightweight process that promotes iterative development, close collaboration between the development team and business side, constant communication, and tightly-knit teams) • Hybrid (is the combination of agile methods with other non-agile techniques. For example, a detailed requirements effort, followed by sprints of incremental delivery) • Traditional (conventional document-driven software development methods that can be characterized as extensive planning, standardization of development stages, formalized communication, significant documentation and design up front)	Closed
Q10	In your words, how would you define TD?	–	Open
Q13	Give an example of TD that had a significant impact on the project that you have chosen to tell us about:	–	Open
Q26	Has the debt item been paid off (eliminated) from the project?	• Yes • No	Closed
Q27	If yes, how? If not, why?	–	Open

and Corbin, 1998). More specifically, because we did not provide a predetermined list of practices and reasons for respondents, we performed open coding, as explained by Strauss and Corbin (1998) as the "analytic process through which concepts are identified and their properties and dimensions are discovered in data". Thus, we identify the concepts, i.e., the central ideas in the data, and their categories and characteristics that are present in the data. The categories are composed of properties, that define the category meaning, and dimensions, which are variations in the category (Strauss and Corbin, 1998).

In answers given to Q27, we initially identified the concepts related to TD payment, i.e., the practices and reasons for TD non-payment, resulting in a set of codes. We divided those codes into two subsets based on the answers to Q26 (a yes/no question).

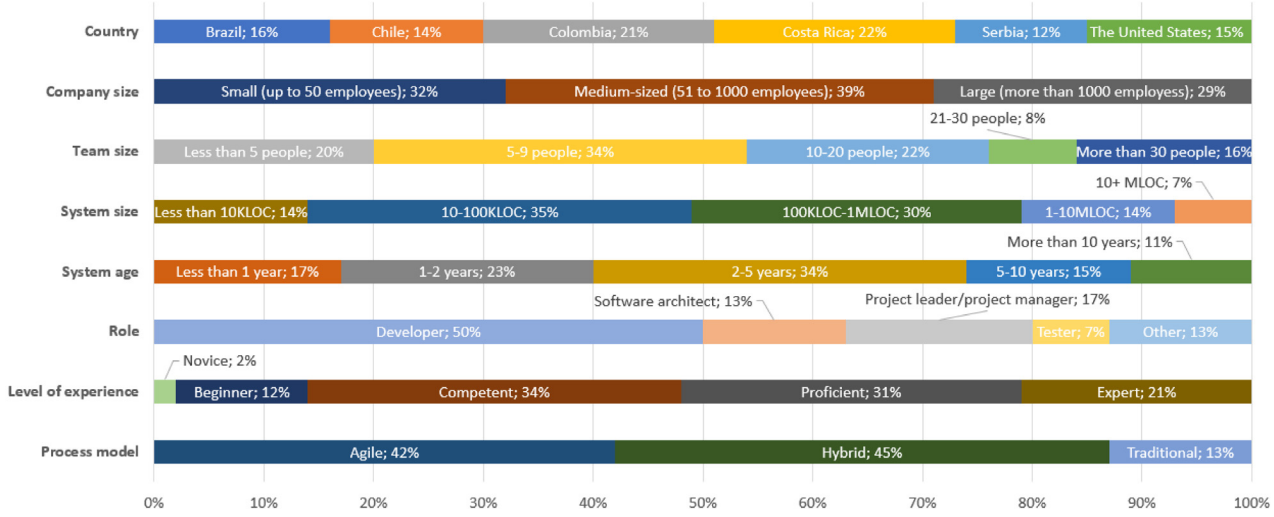


Fig. 2. Summary of participant's characterization.

If the answer was positive, the code was related to the *TD payment practices*, supporting a response to *RQ1*. If the answer was negative, the code was related to *reasons* for not paying off the TD, supporting an answer to *RQ2*. The process was performed iteratively revising and unifying codes at each cycle of analysis until reaching a state of saturation, i.e., a point where no new codes were identified. At the end of the analysis, we obtained a stable list of codes[3] along with their citation frequency.

At least three researchers conducted the coding in each of the six *InsighTD* replications. Each researcher could assume one of the following roles: (i) **code identifier** – the person responsible for extracting the codes from the answers, (ii) **code reviewer** – responsible for reviewing all extracted codes, and (iii) **referee** – responsible for resolving disagreements in codes identified by the code identifier and code reviewer. In total, we had six code identifiers, six code reviewers, and six referees. Also, the last author reviewed the code collected in all replications to guarantee the consistency among all analyses.

Code unification required some effort. For instance, participants cited the following TD practices: "*to make it solid and defect-free system*" and "*(…) corrective maintenance of artifacts.*" The initially extracted codes were *to make defect-free system* and *corrective maintenance of artifacts*, respectively. Then, as these codes had different nomenclature but shared a common meaning, we unified them as *bug fixing*.

3.3.1. Relationships with types of debt

We followed the same process performed by Rios et al. (2020b) to identify a TD type presented in the TD example given to Q13. This process uses the definitions of TD types reported in Rios et al. (2018b) and the list of TD indicators given in Alves et al. (2016). To illustrate this process, let us consider the following answer given for Q13: "*using an external framework that took more time and resources where a native framework would have worked better.*" By analyzing this text, we see that the answer describes issues in the architecture, representing a scenario of architecture debt. This determination was performed by at least three researchers in each *InsighTD* replication. As the respondents used the TD example provided by them in Q13 as context to answer the questions on TD payment, we used the TD type identified in Q13 to associate this type with practices and reasons for not paying off TD items.

3.3.2. Correlation analysis

A participant could cite more than one practice or reason, revealing that these practices or reasons were used together for paying off TD items or for justifying the non-application of TD practices, respectively. To investigate this, we calculated a correlation matrix using Kendall's τ coefficient, and interpreted the matrix values using Hopkins's categorization (Hopkins, 2003): *almost perfect* (0.9 to 1.0, or −0.9 to −1.0), *very large* (0.7 to 0.9, or −0.7 to −0.9), *large* (0.5 to 0.7, or −0.5 to −0.7), *moderate* (0.3 to 0.5, or −0.3 to −0.5), *small* (0.1 to 0.3, or −0.1 to −0.3), and *insubstantial* (less than 0.1 and greater than −0.1). We also applied the Kendall correlation test on top of that. This test is indicated for non-normal distributions and small samples. Our data fit those indications because: (i) we confirmed that our data does not follow a normal distribution by performing the Shapiro–Wilk normality test, and (ii) we identified more than one practice or reason in 65 and 48 answers, respectively.

4. Results

In this section, initially, we present the demographic data followed by the results that address each of the research questions.

4.1. Demographic data

In total, we obtained 653 valid (according to the criteria presented in Section 3.2) answers.[4] including data from Brazil (107), Chile (89), Colombia (134), Costa Rica (145), Serbia (79), and the United States (99). Fig. 2 summarizes the participants' characterization by country, company and team size, system size and age, role, level of experience, and process model. Fig. 3 shows how participant characteristics are spread among countries. In each diagram, the possible values for each participant characteristic are shown in the left, while the countries are presented in the right side. For instance, by considering the "company size", we can see that most of the participants worked in medium-sized companies (40%) and that most of them were from the Costa Rican software industry (23%). Overall, Fig. 3 indicates that the distribution of each characterization variable is similar from country to country.

[3] It is important to say that, although this work reuses the coded data set from our previous work (Freire et al., 2020a; Pérez et al., 2020, 2021; Freire et al., 2021c), we reviewed all coded data from Brazil, Chile, Colombia, and the United States. This was necessary due to the coding consolidation we performed to build the whole data set while including the data collected from the Costa Rican and Serbian software industries.

[4] The raw data is available at https://doi.org/10.5281/zenodo.7168424

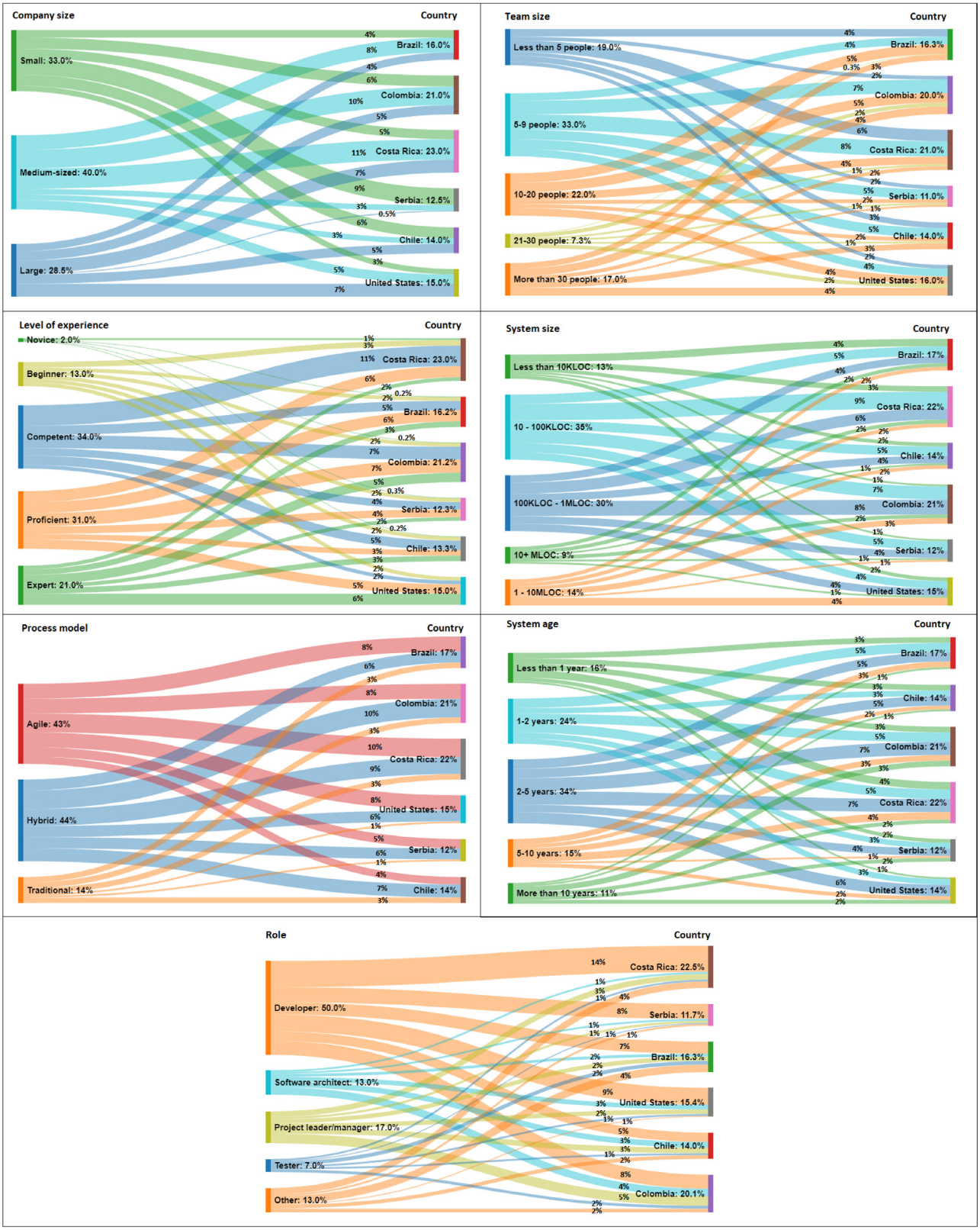


Fig. 3. Demographics per country.

Overall, the collected data includes a wide variety of projects from the software development industry in Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States, including projects of different ages, sizes, team sizes, process models, and composed of several participant roles and levels of experience from organizations of different sizes.

Table 3
Top 10 cited payment-related practices for TD payment.

Payment-related practice	#CPRP	%PRPP
Code refactoring	81	37.2%
Investing effort on TD payment activities	33	15.1%
Design refactoring	25	11.5%
Investing effort on testing activities	22	10.1%
Prioritizing TD items	15	6.9%
Negotiating deadline extension	14	6.4%
Adjusting code to follow good programming practices	10	4.6%
Monitoring and controlling project activities	10	4.6%
Improving the development process	9	4.1%
Increasing the project budget	9	4.1%

Caption:
#CPRP - Count of payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218).

4.2. RQ1: What are the TD payment-related practices used by software practitioners to deal with technical debt items in their projects?

In total, ∼40% (259) of the participants indicated in Q26 that the TD item mentioned was paid off and ∼84% (218) of them described how TD was paid off in Q27. Thus, this set of responses (218) was used for answering RQ1.

> **Takeaway #1**: Only 40% of the TD items described in our data were paid off. Thus, paying off TD should not be the only focus of TDM.

We identified 32[5] payment-related practices for the payment of TD items. These practices are all listed in Fig. 13 in Section 5.2.2. Table 16 in the Appendix presents the identified payment-related practices and quotes from participants. Table 3 summarizes the 10 most commonly cited ones. This table reports the **payment-related practice name** and the total number (i.e., count) of the 218 responses that mentioned that practice for paying off TD items **(#CPRP)**. Column **%PRPP** presents the percentage of #CPRP in relation to the total of all responses (i.e., 218), revealing how frequently each practice was used in software projects. The most cited payment-related practice *code refactoring* impacts ∼37% of the projects. This result was expected, as *code refactoring* has been widely used for paying TD items off (Li et al., 2015; Behutiye et al., 2017).

After identifying the practices for paying off TD, we analyzed them to understand better the complexities of the concept. The data revealed several dimensions related to the concept of TD payment practices, resulting in the following groupings:

- Types of TD payment practices. The data showed that there were several fundamentally different groups of responses that differed in the role they played in the software development process. For instance, some practices can pay off TD items directly (e.g., *refactoring*) while others can either create a favorable scenario for future TD payment (e.g. *negotiating a deadline extension*) or are practices from other TDM activities, such as TD prevention (e.g. *investing effort on testing activities*) or prioritization (e.g. *prioritization of TD items*). Thus, we followed open coding (Strauss and Corbin, 1998), grouping the practices into types. The first author assigned a type for each practice and reason, and the last author reviewed this assignment.
- Nature of TD payment practices. All the practices we discovered are related to either technical or managerial activities of software development. The difference between these groups is related to the nature of the practice. The first author identified the nature (technical or managerial) of each practice and each reason, and the last author reviewed this identification.
- Categories of TD payment practices. We found that many of the codes for practices were related to each other, so we followed a grouping process to organize them into categories reflecting the main concern of each subset. This process followed open coding (Strauss and Corbin, 1998). The first author analyzed each code comparing its meaning with that of the other codes. When he identified a relation between them, he grouped them into a category. To name the categories, we used the list proposed by Rios et al. (2019a). For consistency, once again, the entire process was reviewed by the last author.

Fig. 4 summarizes how our analysis process evolved to reveal the dimensions described above, as well as other complexities of our central concepts. It shows that for TD payment practices (and for reasons for not paying TD, as we discuss in the next section), we used open coding (Strauss and Corbin, 1998) to identify the practices and reasons, and also to identify the types, natures, and categories of practices and reasons by grouping them into different properties and dimensions. This allowed us to more deeply understand the central concepts. Further, we discovered the relationships between practices and types of debt, and between reasons and types of debt. Lastly, we identified the correlation between practices and between reasons for TD non-payment by performing correlation analysis.

Looking at the top 10 cited payment-related practices, presented in Table 3, only the practices *code refactoring*, *design refactoring*, and *adjusting code to follow good programming practices* allow the direct payment of TD items. The other most cited practices only support TD payment initiatives. For example, the practice *investing effort on TD payment activities* can create a favorable setting for TD payment, but TD items are not necessarily paid off when this practice is applied. Thus, we identify one dimension of TD payment-related practices, ***type***, having the following values:

- **Payment action**: includes practices directly resulting in TD item removal, such as *code refactoring*, *design refactoring*, and *update system documentation*.
- **Defining a favorable setting for TD payment**: includes practices that improve the capacity of development teams to pay off debt items. Some examples are *investing effort on TD payment activities*, *negotiating deadline extension*, and *increasing the project budget*.
- **TD prevention**: refers to practices intended to curb potential TD from being incurred. Among them, we have *monitoring and controlling project activities*, *investing effort on testing activities*, and *using short feedback iterations*.
- **TD prioritization**: is related to practices that support the ranking of TD items according to classification criteria. Only *prioritization of TD items* composes this category.

Indeed, the type dimension shows the inherent and unavoidable connection between TD management activities. Table 4 shows the number of unique TD payment practices (**#PRP**) in each type, and the total number of responses citing a practice (**#CPRP**) in each type. Column **%PRPP** corresponds to the percentage of all projects that cited a payment practice of that type, revealing how prevalent each type is in software projects. Table 17 in the Appendix presents, for each type, the payment-related practices ranked by number of citations.

[5] Although we identified 34 payment-related practices in our previous work (Freire et al., 2020a), we realized that the practices *improving the communication among team members* and *using automated testing* have the same meaning as *improving the team collaboration* and *investing effort on testing activities*, respectively. Then, we only used the latter practices, resulting in 32 payment-related practices.

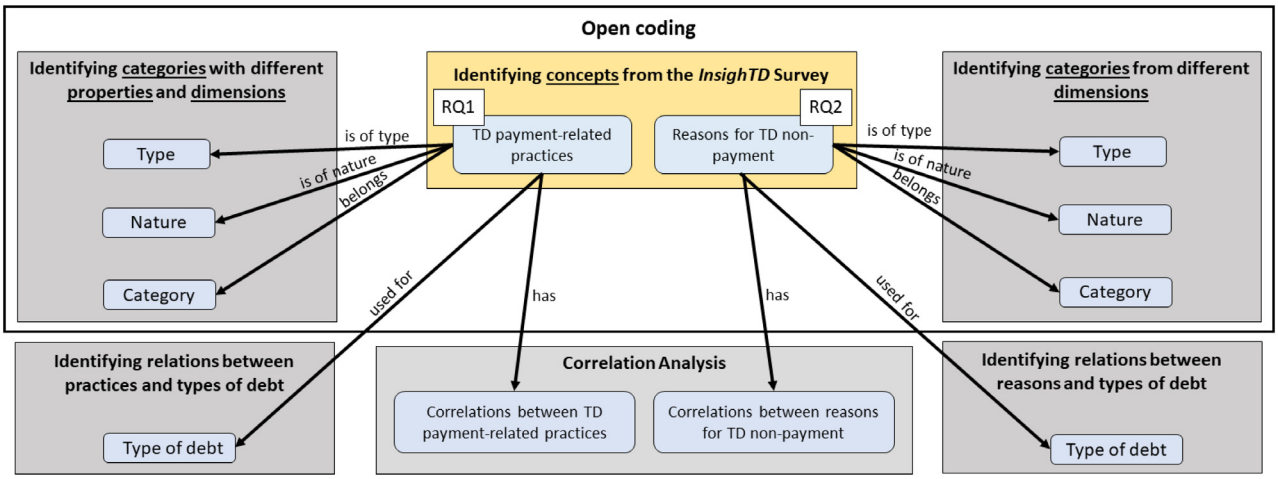


Fig. 4. Schema of research questions and their corresponding findings.

Table 4
Type of payment-related practices for TD payment.

Type of payment-related practice	#PRP	#CPRP	%PRPP
Payment action	8	146	67.0%
Defining a favorable setting for TD payment	11	93	42.7%
TD prevention	12	64	29.4%
TD prioritization	1	15	6.9%

Caption:
#PRP - Count of unique cited payment-related practices.
#CPRP - Count of payment-related practices.
%PRPP - Percentage of CPRP in relation to the total of all projects (218).

Takeaway #2: Not all TD payment activities result directly in the elimination of TD. Implementing practices to enable future TD payment, to prevent TD, and to prioritize TD payment are also necessary.

The most cited type, *payment action,* was applied in ∼67% of the projects, but consisted of only eight unique actions, ranging from *code refactoring* to *update system documentation*. Turning attention to another dimension, the ***nature*** of the payment practices, two of them (*restarting the project from scratch* and *system retirement*) are related to managerial decisions while the others are technical practices. Most of the technical practices can be applied during software implementation, but some, such as *bug fixing* and *refactoring,* are applicable during maintenance to improve the external or internal quality of the system.

Among the top 10 cited TD payment-related practices, the subset that are technical in nature is composed of *code refactoring*, *design refactoring*, and *adjusting code to follow good programming practices*, representing ∼53% of all projects (column %PRPP at Table 3). The managerial subset is composed of *investing effort on TD payment activities*, *investing effort on testing activities*, *prioritizing TD items*, *negotiating deadline extension*, *monitoring and controlling project activities*, *improving the development process*, and *increasing the project budget*, representing ∼51% of all projects (column %PRPP at Table 3). In other words, technical and managerial practices are chosen in almost equal measure in TD payment decisions in software organizations, although there is a wider variety of managerial payment practices. Table 18 in the Appendix presents the TD payment practices of each nature ranked by number of citations.

Table 5 presents the relation between types and natures of all TD payment-related practices. The majority of TD payment and prevention practices are technical, while most of the practices for defining a favorable setting for TD payment are managerial.

Takeaway #3: Eliminating debt items cannot be solely a technical concern. Management practices are necessary as well for implementing TD payment initiatives, and making them part of the TD management strategy is necessary.

Following the categorization from our previous work (Freire et al., 2020a), we organized the set of practices related to the payment of TD items into the following ***categories***:

- **Development issues**: encompasses six practices that are applied during the implementation of software, such as *update system documentation*, *adjusting code to follow good programming practices*, and *solving technical issues*.
- **External quality issues**: groups practices that are related to software quality aspects that can be perceived by users. Only the practice *bug fixing* is part of this category.
- **Infrastructure**: groups two practices related to tools, technologies, and development environment, *external tools* and *organizing the project repository*.
- **Internal quality issues**: includes two practices that can be employed to address limitations that compromise the internal quality of the software, *code refactoring* and *design refactoring*.
- **Methodology**: encompasses 12 practices associated with processes followed by a software team. Among them, we highlight *investing effort on TD payment activities*, *investing effort on testing activities*, and *using short feedback iterations*.
- **Organizational**: refers to two practices associated with organizational decisions, *hiring specialized professionals* and *changing the project management*.
- **People**: includes two practices directly related to the members of software development teams, *improving the team collaboration* and *communicating with the customer about TD items*.
- **Planning and management**: groups five practices associated with management activities. Examples of practices in this category are *monitoring and controlling project activities*, *prioritizing TD items*, and *negotiating deadline extension*.

Table 6 presents the categories of payment-related practices, reporting the **category's name**, the number of unique payment-related practices cited (**#PRP**) and the total number (i.e., count) of responses (or projects) that cite practices in each category (**#CPRP**). Lastly, the column **%PRPP** corresponds to the percentage of #CPRP in relation to the total of all projects. Table 19 in the Appendix presents, for each category, the payment-related practices ranked by number of citations.

Table 5
Relation between type and nature of payment-related practices.

Type of payment-related practice	Nature of payment-related practice	
	Technical	Managerial
Payment action	6 (44%)[a]	2 (2%)
TD prevention	8 (7%)	4 (13%)
Defining a favorable setting for TD payment	1 (1%)	10 (28%)
TD prioritization	0 (0%)	1 (5%)
Total	15 (52%)	17 (48%)

[a]The number in parentheses represents the percentage of the total number (i.e., count) of payment-related practices cited in each nature in relation to the total of all cited practices (318).

Table 6
Categories of payment-related practices for TD payment.

Category of payment-related practices	#PRP	#CPRP	%PRPP
Internal quality issues	2	106	48.6%
Methodology	12	96	44%
Planning and management	5	50	22.9%
Development issues	6	39	17.9%
Organizational	2	10	4.6%
People	2	7	3.2%
External quality issues	1	6	2.8%
Infrastructure	2	4	1.8%

Caption:
#PRP - Count of unique cited payment-related practices.
#CPRP - Count of payment-related practices.
%PRPP - Percentage of CPRP in relation to the total of all projects (218).

Table 6 shows that practices in the *internal quality issues* category were cited in almost half of the projects, highlighting the central role of internal quality in the payment of TD items. This is consistent with the view that TD primarily affects internal quality issues in software products (Avgeriou et al., 2016). The categories *methodology*, *planning and management*, and *development issues* were also commonly mentioned, appearing in ∼85% of the responses (column %PRPP).

Table 6 also shows that few participants use improvement of the organizational environment (*organizational* category) or technical knowledge of the team (*people* category) as part of their TD payment initiatives. Finally, practices from the categories *external quality issues* and *infrastructure* were very seldom mentioned.

4.2.1. TD payment-related practice per type of debt

Fig. 5, Table 7, and Fig. 6 all present the relationships among TD payment practices and types of TD. Fig. 5 focuses on the top 10 TD payment-related practices, while Table 7 and Fig. 6 break down the relationship with TD types according to the types and categories, respectively, of TD payment practices. For simplicity, we included in Figs. 5 and 6 only those debt types with at least five data points.

We notice in Fig. 5 that all practices composing the top 10 were applied to design debt items, and most were also applied to code and test debt items. *Code refactoring* was cited as a TD payment practice for every type of TD, and the practice *investing effort on TD payment activities* (the second most cited) was cited for nearly all types, except defect debt.

From Table 7, we can observe that not all types of debt were directly paid off (e.g., automation test, process, usability, and versioning debt), while payment actions (marked in gray) were commonly cited for paying off design, code, and architecture debt items. Perhaps software practitioners focused on these items because they impact technical activities and thus can impact the software's internal and external quality.

Practitioners have also taken steps to enable TD payment (marked in green), as well as to prevent TD (marked in blue), for types of TD closely related to the code (i.e., design, test, code, architecture, and requirements). One might conclude that software practitioners have spent effort on improving the artifacts produced in technical, code-related activities as these artifacts are strongly related to or can directly impact the code. Therefore, the improvement of those activities can facilitate the application of practices for paying off TD items, and can potentially prevent TD items from being incurred.

Fig. 6 presents the relationship between the categories of TD payment-related practices and types of debt. We notice that nearly all categories of TD payment practices have been applied to code, design, and requirements debt. In general, the categories *development*, *internal quality*, *methodology*, and *planning and management issues* are very common for most types of TD, indicating that practitioners can invest in these practices and leverage them for multiple types of debt.

Takeaway #4: Choosing practices from the categories *development issues*, *internal quality issues*, *methodology*, and *planning and management* can support software practitioners to address multiple types of debt.

4.2.2. Correlation among TD payment-related practices

Participants reported more than one TD payment-related practice in 65 out of 218 responses given to Q27, allowing us to investigate practices that are commonly used in combination. We calculated the Kendall's τ coefficient for each combination of the top 10 practices. Fig. 7 presents those coefficients organized in a correlation matrix (in which the rows and columns are the practices) with the strength of each coefficient following the Hopkins' categorization (Hopkins, 2003). We used colors to represent a positive (blue), negative (red), or no (white) correlation. A positive correlation indicates that two practices tend to co-occur, while a negative correlation indicates the contrary. We highlighted with an asterisk the pairs of practices with statistically significant correlation. The complete correlation matrix containing all practices is available in Appendix Fig. 16, while the complete table containing the correlations and related statistics is available in Appendix Table 20.

We can observe that the practices *investing effort on testing activities* and *prioritizing TD items* have a strong correlation, meaning that they are often reported to be used together in TD activities. The practice *design refactoring* has a moderate correlation with

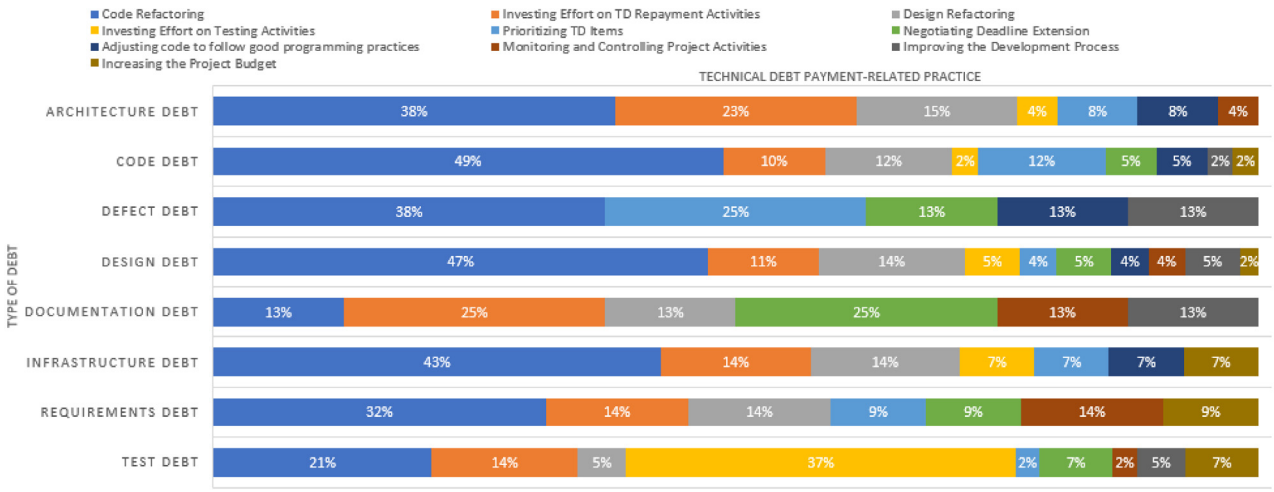


Fig. 5. Relationship among the top 10 payment-related practices and types of debt.

Table 7
Relationship among the types of technical debt payment-related practices and types of debt.

Type of debt	Type of TD payment-related practice			
	Payment action	Defining a favorable setting for TD payment	TD prevention	TD prioritization
Architecture debt	21	8	5	2
Automation test debt	0	1	1	0
Build debt	1	1	0	0
Code debt	29	9	12	5
Defect debt	4	2	0	2
Design debt	41	22	18	2
Documentation debt	8	7	1	0
Infrastructure debt	12	3	2	1
People debt	2	3	0	0
Process debt	0	5	2	0
Requirements debt	13	12	5	2
Service debt	4	1	0	0
Test debt	14	19	17	1
Usability debt	0	1	0	0
Versioning debt	0	0	1	0

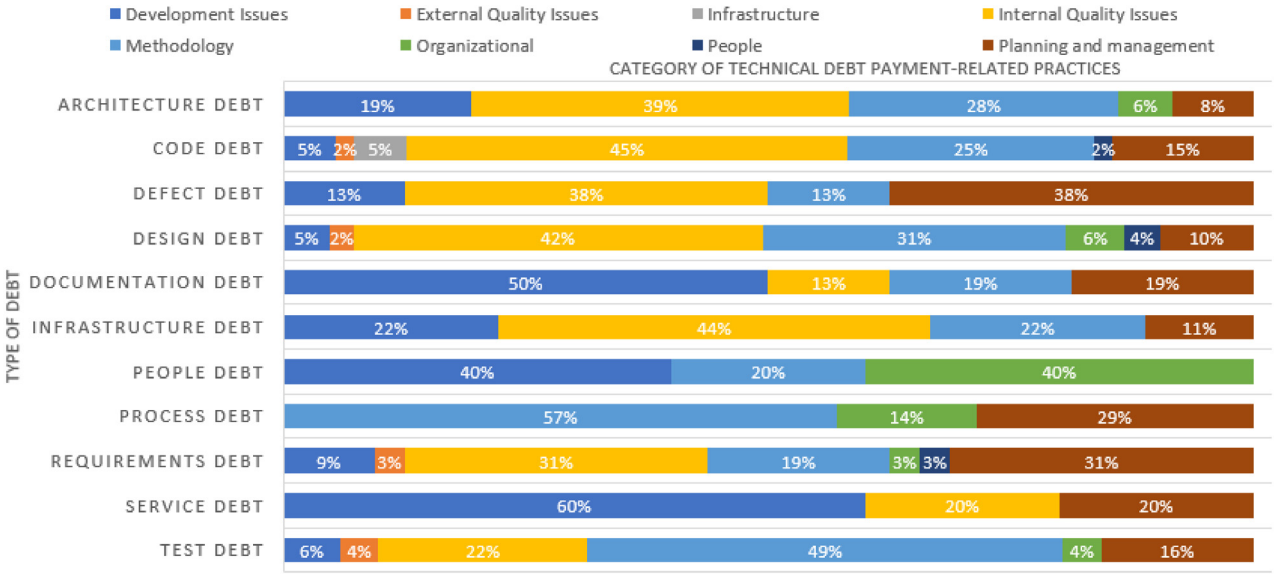


Fig. 6. Relationship between the category of technical debt payment-related practices and types of debt.

the practices *investing effort on TD payment activities, investing effort on testing activities*, and *improving the development process*. The same occurs with the pairs *investing effort on TD payment activities* and *negotiating deadline extension*, and *monitoring and controlling project activities* with the practices *investing effort on testing activities, prioritizing TD items*, and *increasing the project budget*. The negative correlation between *increasing the project budget* and other payment-related practices possibly signals that

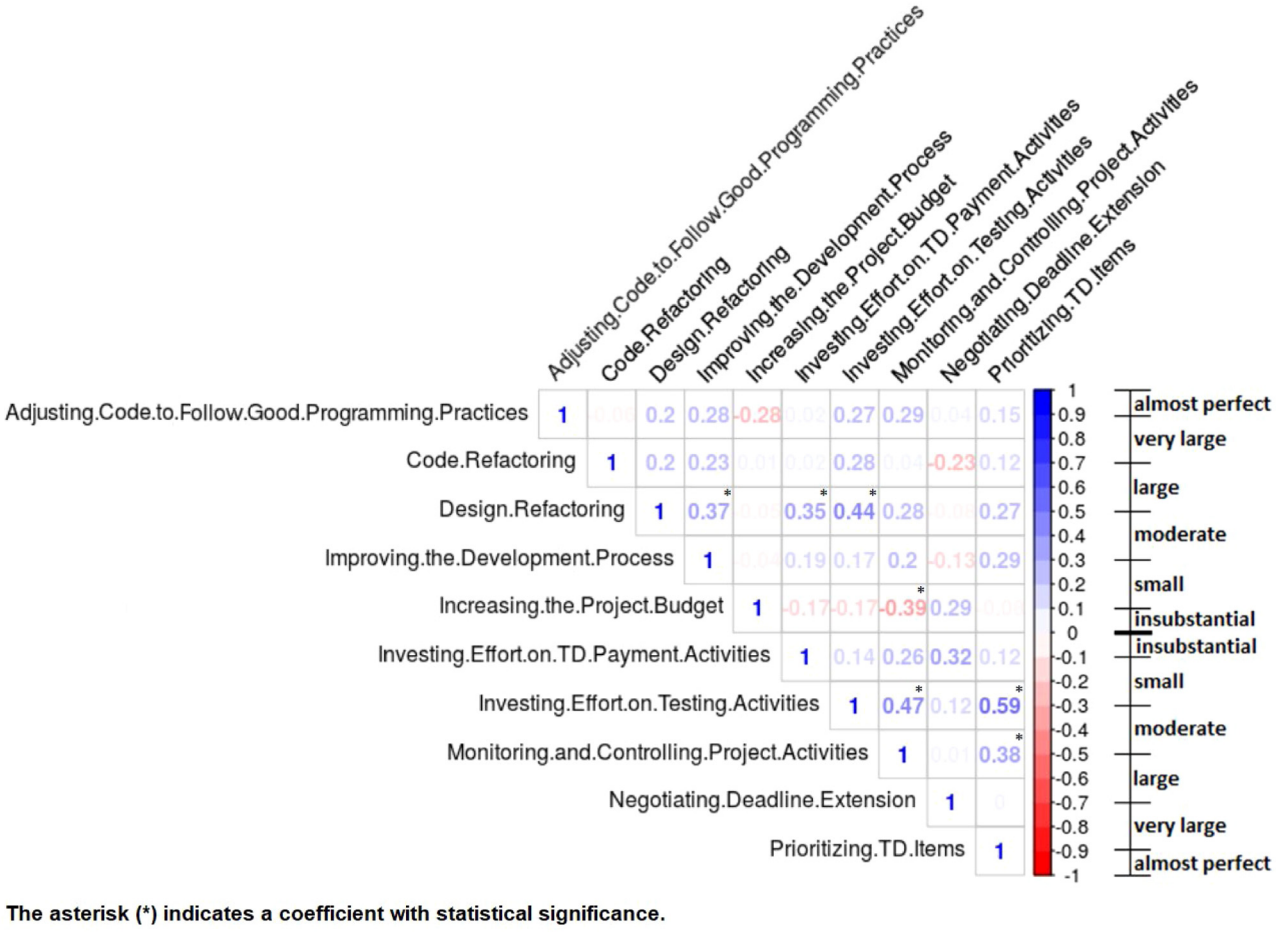


Fig. 7. Correlation matrix between the top 10 TD payment-related practices. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

there is some subset of projects that try to just throw money at the problem without taking any other more thoughtful steps. Finally, the presence of the practices related to the prioritization of TD items, the monitoring of the project's activities, and design refactoring practices in the correlation set signals a more mature development process adopted by a software team, indicating that the team uses a combination of advanced practices.

Surprisingly, although the practice *code refactoring* is the most cited practice for paying off TD items in our data, it has no significant correlation to other practices composing the top 10. However, we verified that *code refactoring* has a moderately significant correlation with the non-top-10 practices *using short feedback iterations*, *improving requirement elicitation process*, and *using automated deployment*.

4.3. RQ2: What are the reasons considered by software practitioners for not paying off TD?

In total, ∼60% (394) of the participants indicated that the TD item mentioned was not paid off and ∼66% (261) of those explained why. This set of responses (261) was used for answering RQ2.

Table 8 shows the 10 most commonly cited reasons for not paying off TD (out of a total of 27[6]) identified in Q27. The complete list of reasons is presented in Fig. 13 in Section 5.2.2. Also, all the identified reasons and quotes from participants are available in Appendix Table 21. Table 8 reports the **reason name** and the total number (i.e., count) of responses citing each reason (**#CR**). Column **%RP** shows the percentage of #CR in relation to the total of all projects.

Table 8
Top 10 cited reasons for TD non-payment.

Reasons for TD non-payment	#CR	%RP
Focusing on short term goals	69	26.4%
Lack of organizational interest	48	18.4%
Lack of time	41	15.7%
Cost	34	13.0%
Lack of resources	19	7.3%
Customer decision	13	5.0%
Complexity of the TD item	12	4.6%
Effort	11	4.2%
Complexity of the project	10	3.8%
Insufficient management view about TD payment	10	3.8%

Caption:
#CR - Count of citations of each reason.
%RP - Percentage of CR in relation to the total of all projects (261).

The most cited reason *focusing on short term goals* impacts ∼26% of the projects. This was expected, as focusing on short term goals, software teams can deliver faster and this is commonly seen as the primary cause of TD (Spínola et al., 2013; Rios et al., 2020b). These results show that it is also the most common cause of perpetuating TD by not paying it off.

[6] Although we identified 28 reasons for not paying off TD in our previous work (Freire et al., 2020a), we have since determined that the reason *low priority* has the same meaning as *focusing on short term goals*. Thus, we only use the latter, resulting in 27 reasons.

Table 9
Type of reasons for TD non-payment.

Type of reasons for TD non-payment	#R	#CR	%RP
Decision factor	12	149	57%
Impediment	15	170	65%

Caption:
#R - Count of unique cited reasons.
/#CR - Count of reasons.
%RP - Percentage of CR in relation to the total of all projects (261).

Takeaway #5: *Focusing on short term goals* is not only a leading cause of incurring TD, but also a primary reason for perpetuating it.

As with practices, the data revealed distinct ***types*** of reasons. Some reasons appear to refer to instances when a team decides not to pay TD items off. All the other reasons are related to the inability to pay off TD items, even if the team wants to pay them off. Thus, following our previous categorization schema defined in Freire et al. (2020a), we organized the reasons for not paying TD items into the following types:

- **Decision factor**: refers to reasons for deciding not to pay off the TD. Among them, we have *focusing on short-term goals*, *lack of organizational interest*, and *insufficient management view about TD payment*.
- **Impediment**: points to situations in which the development team wanted to pay off the TD, but they could not pay it off for some reason. Examples are *lack of time*, *cost*, and *lack of resources*.

Table 9 presents the types of reasons, reporting the **type's name**, the number of unique reasons cited (**#R**), and the total number of projects citing reasons (**#CR**) in each type. Column **%RP** corresponds to the percentage of #CR in relation to the total of projects. Table 22 in the Appendix presents, for each type, the reasons for TD non-payment ranked by number of citations.

Table 9 indicates that decision factors are common, but most of the time impediments prevent payment of TD items. About 8% of the participants indicated a combination of an impediment and a team decision factor. This indicates that team decisions can sometimes align with other factors that influence TD non-payment in software projects.

As with the payment-related practices, the reasons can also be related to managerial or technical activities as in *focusing on short term goals* and *complexity of the TD item,* respectively. Thus, the reasons can be categorized according to their ***nature***:

- **Technical reasons**: are related to reasons involved in the technical activities of a software project, such as requirement analysis, design, coding, and testing.
- **Managerial reasons**: are related to reasons involved in management activities of the development of a software project, such as hiring, project management, and cost estimation.

Considering only the top 10 cited reasons, the managerial subset is composed of *focusing on short term goals*, *lack of organizational interest*, *lack of time*, *cost*, *lack of resources*, *customer decision*, *effort*, *complexity of the project*, and *insufficient management view about TD payment*, representing ∼98% of all projects (column %RP at Table 8). The technical subset is composed of the reason *complexity of the TD item*, representing only ∼5% of all projects (column %RP at Table 8).

Table 10 presents the relation between types and natures of all reasons for not paying TD. We have 21 managerial reasons, representing 93% of the total number of citations, and only six technical reasons, corresponding to 7% of the total number of citations. The complete list of reasons and their corresponding types is available in Fig. 13 in Section 5.2.2. This result clearly suggests that dealing with managerial issues is quite decisive if we are interested in improving the use of TD payment practices. Table 23 in the Appendix presents, for each nature, the reasons for TD non-payment ranked by number of citations.

Takeaway #6: In most cases, the reasons for TD non-payment have a managerial nature regardless of their type (decision factor or impediment), revealing that software practitioners must disseminate the importance of TD payment to their managers.

We also classified the identified reasons for not paying off TD into eight ***categories***, following the initial categorization proposed in Freire et al. (2020a):

- **Development issues**: groups two reasons related to software development activities, *complexity of the project* and *decision to not change the framework*.
- **External factors**: organizes four reasons related to factors that are out of control of the development team, such as *customer decision*, *the project was discontinued*, and *TD items do not affect the user*.
- **Internal quality issues**: encompasses two reasons related to characteristics of system code and structure. Those reasons are *complexity of the TD item* and *number of TD items*.
- **Lack of knowledge**: groups two reasons associated with the need for technical knowledge. We have *lack of technical knowledge* and *lack of knowledge about TD*.
- **Methodology**: groups five reasons associated with process activities. Among them, we highlight *lack of adoption of lessons learned*, *lack of testing*, and *non-application of mitigation actions on TD causes*.
- **Organizational**: includes three reasons associated with organizational decisions. Those reasons are *lack of organizational interest*, *lack of resources*, and *high team turnover*.
- **People**: includes three reasons related to team characteristics, *insufficient management view about TD payment*, *team overload*, and *lack of committed team*.
- **Planning and management**: encompass six reasons related to management activities, such as *focusing on short term goals*, *lack of time*, and *cost*.

Table 11 presents the categories of reasons, reporting the **category's name**, the number of unique reasons cited (**#R**), and the total number (i.e., count) of projects citing reasons (**#CR**) in each category. The column **%RP** corresponds to the percentage of #CR in relation to the total of all projects. Table 24 in the Appendix presents, for each category, the reasons for TD non-payment ranked by number of citations.

The most cited category, *planning and management*, affects ∼63% of the software projects. The category *organizational* was also commonly found, affecting ∼26% of the projects. Although one may expect that development issues are crucial for determining the non-payment of TD items, planning and management, and organizational issues seem to be more decisive.

The categories *external factors* and *people* correspond to ∼8% of reasons considered in software projects. It indicates that few participants have perceived those factors as decisive for not paying off debt items. The other categories (*internal quality issues*, *development issues*, *lack of knowledge*, and *methodology*) are even less commonly cited. Among them, *development issues* category affects only ∼4% of the software projects, revealing that software practitioners did not recognize technical issues as impediments to paying debt items off.

Table 10
Relation between type and nature of reasons for TD non-payment.

Type of reason for TD non-payment	Nature of reason for TD non-payment	
	Technical	Managerial
Decision factor	2 (1%)[a]	10 (46%)
Impediment	4 (6%)	11 (47%)

[a]The number in parentheses represents the percentage of the total number (i.e., count) of reasons cited in each nature in relation to the total of all cited reasons.

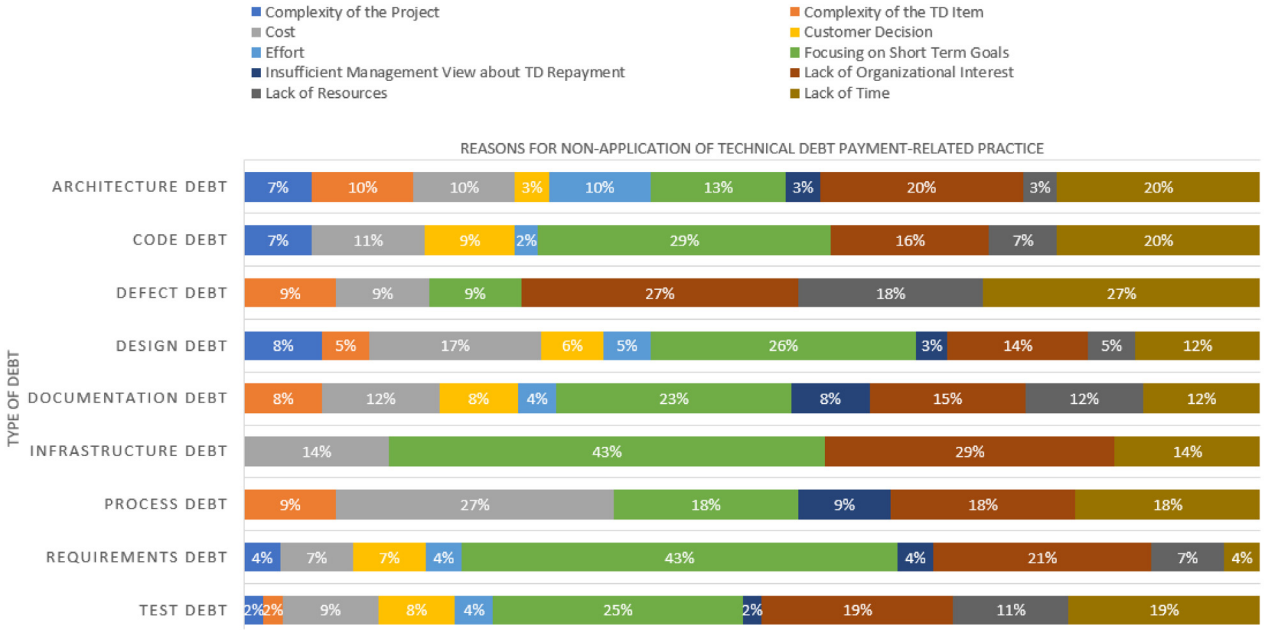


Fig. 8. Relationship between the top 10 reasons for non-payment and types of debt.

Table 11
Categories of reasons for TD non-payment.

Category of reasons for TD non-payment	#R	#CR	%RP
Planning and management	6	164	62.8%
Organizational	3	69	26.4%
External factors	4	22	8.4%
People	3	20	7.7%
Internal quality issues	2	15	5.7%
Development issues	2	11	4.2%
Lack of knowledge	2	9	3.4%
Methodology	5	9	3.4%

Caption:
#R - Count of unique cited reasons.
#CR - Count of reasons.
%RP - Percentage of CR in relation to the total of all projects (261).

4.3.1. Reasons for TD non-payment per types of debt

Figs. 8, 9, and 10 all present the relationships among reasons for non-payment of TD and types of TD. Fig. 8 focuses on the top 10 reasons, while Figs. 9 and 10 break down the relationship with TD types according to the types and categories, respectively, of reasons. For simplicity, we included in Figs. 8 and 10 only those debt types with at least five data points.

We notice in Fig. 8 that all the reasons composing the top 10 were used for justifying the non-payment of architecture, design, and test debt items. For documentation and requirements items, all reasons except *complexity of the project* and *complexity of the TD item* were considered. We can also observe that *focusing on short term goals, lack of organizational interest,* and *lack of time* are commonly related to almost all types of debt, indicating that dealing with them could have a positive impact on initiatives to pay debt items off.

In Fig. 9, we can observe that decision factors are responsible for failing to pay almost all debt types, except build, people, service, and usability debt. Impediments are cited as reasons in particular for the non-payment of design, test, and code debt items.

Fig. 10 presents the relationship among categories of reasons for not paying off TD and types of debt We notice that design and test debt are related to all categories of reasons. Architecture, documentation, and requirements debt are associated with seven of eight categories. The *planning and management* category is related to all types of debt, and the categories *organizational* and *people* are also very common.

Takeaway #7: Payment of all types of TD face obstacles from the category *planning and management*, implying that involving project managers in TD payment initiatives is crucial.

4.3.2. Correlations among reasons for not paying TD items

Participants reported more than one reason in 48 out of 261 responses given to Q27, allowing us to investigate reasons that are commonly considered in combination. We calculated the Kendall's τ coefficient for each combination of reasons. Fig. 11 presents those coefficients organized in a correlation matrix (where its rows and columns are reasons for TD non-payment) and the strength of each coefficient following the Hopkins's categorization (Hopkins, 2003) for the top 10 reasons. We used colors to represent a positive (blue), negative (red), or no (white) correlation. A positive correlation indicates that two reasons tend to co-occur, while a negative correlation indicates the contrary. We highlight with an asterisk the pairs of reasons

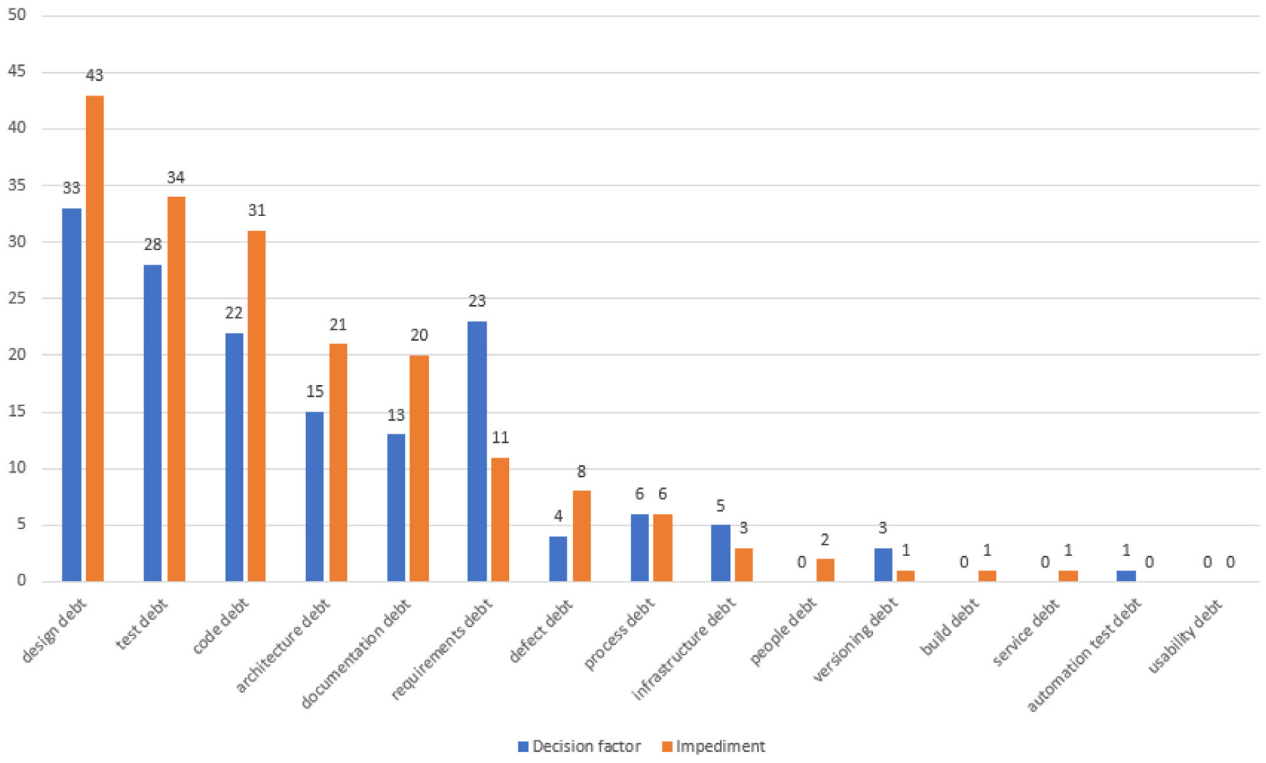


Fig. 9. Relationship among the types of reasons and types of debt.

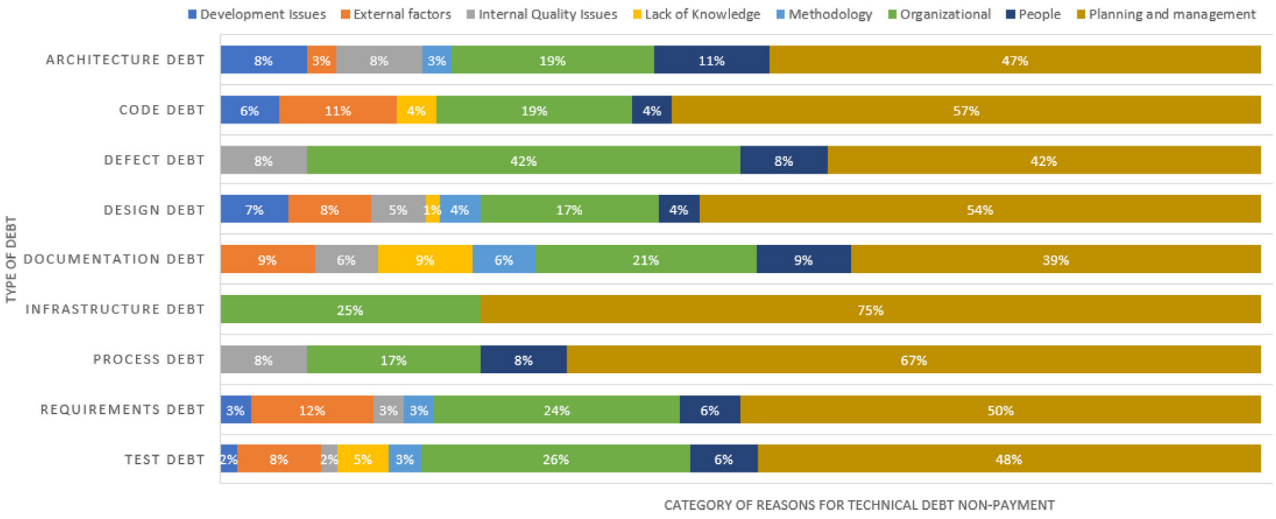


Fig. 10. Relationship among the category of reasons for non-application of technical debt payment-related practices and types of debt.

with statistically significant correlations. The complete correlation matrix containing all reasons is available in the Appendix Fig. 17 and the complete correlation table is in Appendix Table 25.

We can observe that *complexity of the TD item* and *complexity of the project* are highly correlated, indicating that the structure and relationships between the software entities (in the project or in the specific item) matter in the decision to pay or not a TD item. Another group that stands out is composed of *customer decision* and its following correlated reasons: *complexity of the TD item*, *complexity of the project*, *effort*, and *insufficient management view about TD payment*, revealing that an external factor to the team may be associated with internal factors to motivate the non-payment of the debt.

5. Discussion

This section discusses the findings of this study. Firstly, we revisit the research questions and summarize our results for each one. After, we go further into the results and organize the TDM landscape, which is composed of:

- an extension of the conceptual model for TD (Avgeriou et al., 2016; Rios et al., 2018b) using our findings on TD payment.
- a map encompassing the set of information on TD payment-related practices and reasons for not paying off TD items along with their types, categories, and nature. The map was created following the principles of evidence briefings in software engineering (Cartaxo et al., 2016).

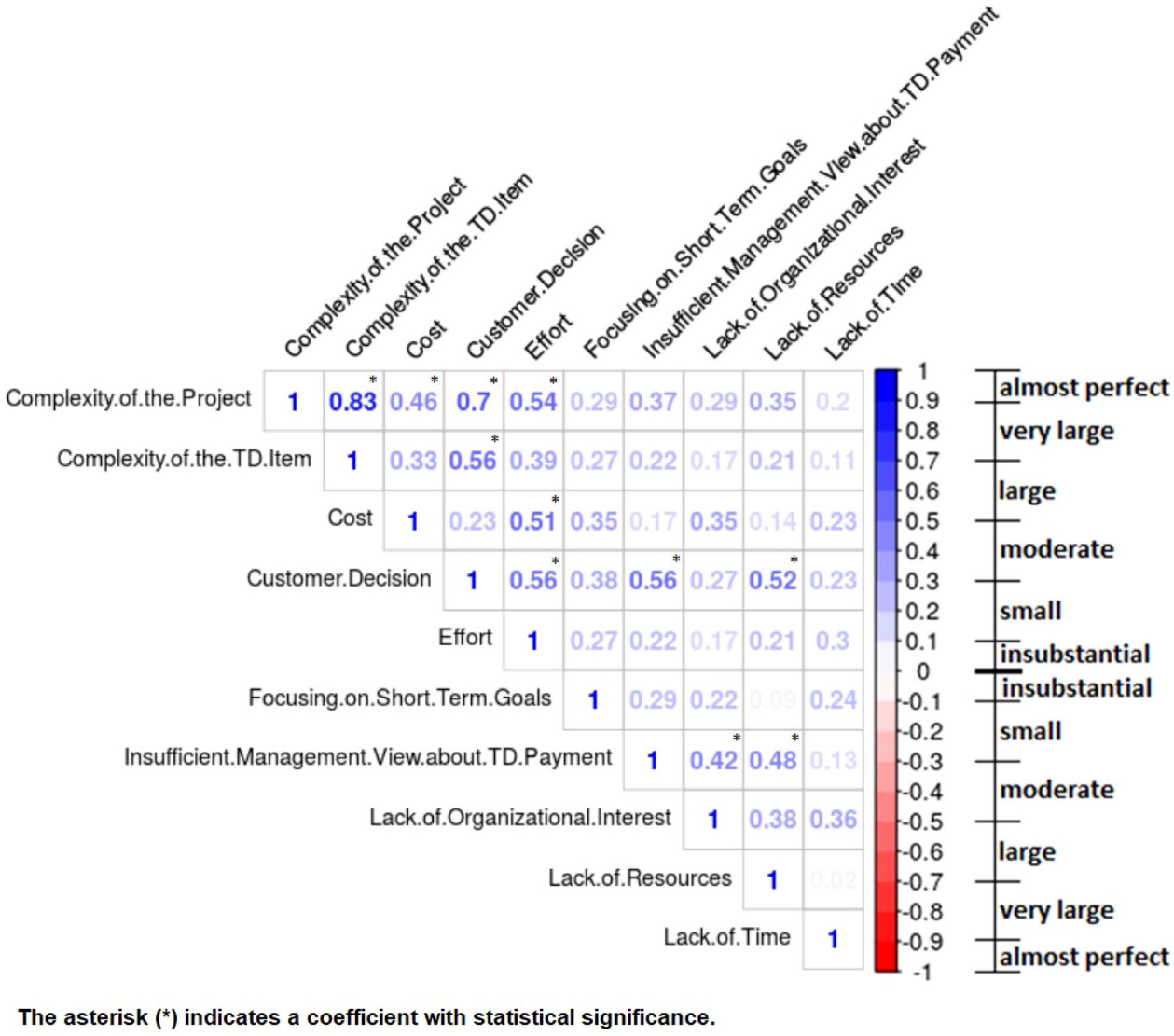


Fig. 11. Correlation matrix between the top 10 reasons for not paying TD items off. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- a set of correlations we found among the practices and among the reasons.

Lastly, we present the assessment of the conceptual model and the map we performed with software practitioners and the implications of this work for researchers and practitioners.

5.1. Revisiting the research questions

In this section, we answer the research questions posed in this work, summarizing the main results, and showing some insights from them.

5.1.1. TD payment-related practices (RQ1)

Although the respondents (∼60%) have not commonly paid off TD items in their projects, we could find a set of 32 practices related to the payment of TD, which fall into four broad types: practices that directly result debt item payment, practices that help create a favorable scenario for future debt payment, TD prevention practices, and TD prioritization practices. *Code refactoring*, *investing effort on TD payment activities*, and *design refactoring* are the most cited practices.

We found eight practices for directly paying off TD items: *code refactoring*, *design refactoring*, *adjusting code to follow good programming practices*, *update system documentation*, *solving technical issues*, *restarting the project from scratch*, *system retirement*, and *bug fixing*. These practices have been used in combination with others, for example, we found that when software practitioners pay debt items off, they are also concerned with the prevention and prioritization of debt items. Moreover, they have changed their work to enable practices related to TD payment in the future. These changes are related to the practices that define a favorable scenario for debt payment.

We also found that TD payment practices are roughly balanced between technical and managerial practices, indicating that TD payment involves different types of roles in software projects. However, most of the payment actions and TD prevention practices are technical, while most of the practices for defining a favorable setting for TD payment and the practices for TD prioritization are managerial. This implies that technical practitioners spend effort in activities for paying off and curbing TD items, but the execution of these activities needs the support of managers who must spend effort in making changes in the work environment to create conditions for TD payment.

We also found that the identified practices are more commonly concentrated in *methodological* issues of software development, and practitioners have used them for paying off several types of debt: architecture, build, code, defect, design, documentation, infrastructure, people, requirements, service, and test. This is an indication that the process followed by software practitioners plays a central role for TD payment. Improvements in processes seem to be a good starting point for TD payment initiatives.

Among the 32 identified practices, we found that 27 of them are statistically significantly correlated with each other, meaning that they usually appear in combination when practitioners are paying debt items off. For example, the practices *code refactoring* and *using short feedback iterations* have a moderate correlation, while the practices *investing effort on TD payment activities* and *changing project scope* are highly correlated. These correlations demonstrate that TDM requires a joint effort from a variety of activities. We found relationships between practices related to payment, prevention, and prioritization activities. New strategies for TDM should take into consideration the integration of TDM activities to provide useful strategies for supporting the management of TD items.

5.1.2. Reasons for not paying off TD (RQ2)

The study identified 27 reasons for not paying TD. Among them, *focusing on short term goals*, *lack of organizational interest*, and *lack of time* are the most cited reasons, revealing that managerial decisions are quite decisive.

We also found that these reasons can be either a decision taken by the team to intentionally not pay off debt, or an impediment that hinders the payment of debt items regardless of the practitioners' intentions. *Impediments* are slightly more commonly faced (∼65%) than *decision factors*. The reasons are more used for justifying the non-payment of design, test, code, architecture, documentation, and requirements debt items. Also, we found more managerial reasons than technical ones, indicating that the management view is decisive for the non-payment of TD.

Among the 27 identified reasons, 17 of them are statistically significantly correlated with each other, indicating that they usually appear in combination to curb the payment of debt items. For example, the reasons *high team turnover* and *number of TD items* have an almost perfect correlation, while the reasons *customer decision* and *effort* are highly correlated with each other. These correlations can reveal new impediments or decision factors related to those already perceived by a team.

5.2. Organizing the TDM landscape

In this section, we present an extension of the conceptual model for TD (Avgeriou et al., 2016; Rios et al., 2018b). We also organize the set of information on TD payment-related practices and reasons for not paying off TD items along with their types, categories, and nature into a map, following the principles of evidence briefings in software engineering (Cartaxo et al., 2016). We performed a complementary survey to assess the model and the map. Lastly, we identified a set of situations for contextualizing the correlations we found among the practices and among the reasons.

5.2.1. Extending the conceptual model for technical debt

Fig. 12 shows our contribution (in blue) to the conceptual model for TD. The classes in white were defined by Avgeriou et al. (2016) and Rios et al. (2018b). These classes represent the properties, artifacts, and elements associated with TD items. According to the conceptual model, a system has many concerns, where TD is one of them. TD can be specialized into 15 TD types (e.g., architecture, code, design, and test debt) and is composed of TD items. A TD item can be caused by different factors, like a decision, schedule pressure, inappropriate processes, and others. These factors are specializations of the *Cause* class. Also, TD items can bring consequences to the projects, affecting their business goals. These consequences can impact cost, value, schedule, or quality of a feature or the project continuance. A TD item is associated with one or more artifacts of the software development process (e.g., code, test, and documentation) represented as specializations of the *DevelopmentArtifact* class. Finally, a TD item can be related to TD management, which is represented as an aggregation of management strategies, support tools, and activities. The latter can be specialized into 11 types, such as prevention, payment, monitoring, and measurement.

Our extension to the conceptual model includes all the organized empirical evidence in this work related to the concept of TD payment (class in green). This class is associated with two lists. One is formed by the reasons for the non-payment of TD (*Reason* class) and the other represents the payment-related practices (*PaymentRelatedPractice* class). The *Reason* class can assume two different roles: an impediment or a decision factor. Also, a reason has a nature, which can be technical or managerial. The reasons that lead to the non-payment of TD items are from different categories (e.g., *ExternalFactors*, *LackOfKnowledge*, and *DevelopmentIssues*). The *PaymentRelatedPractice* class also has a nature (technical or managerial) and the practices are categorized into eight categories (e.g., *ExternalQualityIssues*, *InternalQualityIssues*, and *PlanningAndManagement*). Lastly, a payment related practice can be a payment action, a definition of a favorable scenario for the payment of TD, a prioritization activity, or a prevention activity.

Through the generalization of the *PaymentRelatedPractice* class into the classes *Prevention* and *Prioritization*, we found a relationship among the payment, prevention, and prioritization activities. This relationship indicates an integration of such kinds of TDM activities. The importance of such relationships was previously discussed in Rios et al. (2018b), when the authors reported that current tools and strategies for TDM are limited in the sense that they only cover isolated TDM activities. These tools and strategies can be updated considering the different types of practices we described in the *PaymentRelatedPractice* class and its specializations. For example, suppose a strategy is only focused on supporting TD prevention practices. In that case, other practices for TD payment and prioritization can be included in the strategy to support the application of combined practices from these three TDM activities (prevention, payment, and prioritization). Also, the practices from the three TDM activities could be combined in a tool, and the user can choose one or more practices to apply together in their project. This approach would not guarantee the use of practices from different activities but would present the opportunity to do so.

The conceptual model reveals valuable information for researchers and practitioners. Analyzing the model, researchers and practitioners can understand the big picture related to TD and its entities and relationships. After extending the model, they can analyze the payment, prevention, and prioritization activities and realize that integrating these activities can be performed by combining practices from these activities. Thus, the map can drive new research on this integration and support the definition or improvement of tools for TDM. Lastly, the model can help researchers characterize the context of the projects they are studying. Adding these new classes and relationships will help ensure that payment issues will be considered and described in detail in future studies.

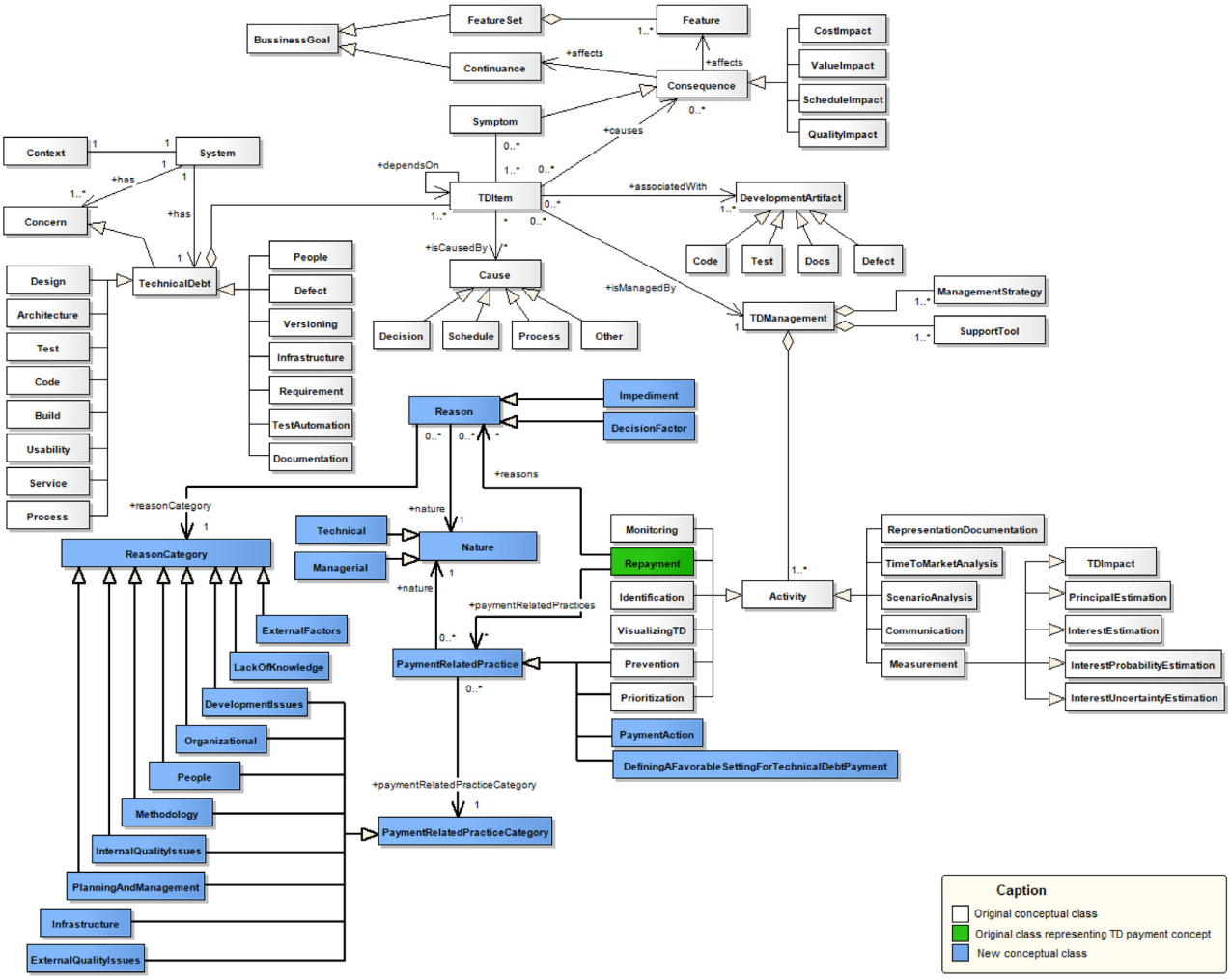


Fig. 12. Extended conceptual model for technical debt. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.2.2. Technical debt payment map

As evidence briefings have been used to disseminate research findings with practitioners (Cartaxo et al., 2016), we use the same approach to summarize our findings. Fig. 13 shows our map for supporting development teams in TD payment activities. The map organizes practices and reasons grouped by category. To compute the percentages associated with each practice and reason, we summed up the number of occurrences for each of them. Then, we divided this value by the number of projects for which any practice or reason was cited. For example, the payment action *code refactoring* was cited by 81 participants. As we had 218 participants indicating that a TD item was paid off in their projects, *code refactoring* was used in 37.2% (81/218 $*$ 100) of them. We followed the same process for computing the percentage for each category.

Analyzing the percentages in the map, we notice that TD payment-related practices from the categories *internal quality issues* and *methodology* are more commonly used in 48.6% and 44% of the projects, respectively. The payment action *code refactoring* stands out in the category *internal quality issues* and was used in 37.2% of the projects. On the right side of the figure, we can observe that the reasons from the categories *planning and management* and *organizational* are more used for explaining the non-payment of TD items in 62.8% and 26.4% of projects, respectively. The decision factor *focusing on short term goals* stands out in the category *planning and management,* being considered in 26.4% of the projects.

Small rectangles indicate the nature of a practice or reason, with black rectangles indicating a technical nature while white rectangles denote managerial. For example, the practice category *development issues* has three managerial (*changing project scope*, *restarting the project from scratch*, and *system retirement*) and three technical practices (*adjusting code to follow good programming practices*, *update system documentation*, and *solving technical issues*). Small circles represent the type of practice (payment action, defining a favorable setting for TD payment, TD prevention, or TD prioritization) or reason (decision factor or impediment). We used a specific color for each type. For instance, the reason category *planning and management issues* has four orange reasons (*lack of time*, *cost*, *effort*, and *risk for the project*) of the type *impediment*, and two yellow reasons (*focusing on short term goals* and *TD items do not have "interest"*) of the type *decision factor*.

Software practitioners can use the map to identify and understand reasons for the non-application of TD payment practices. For example, if a team realizes that the non-payment of TD items occurs due to *cost*, through the map, the team can perceive that this reason is from the category *planning and management* and identify other reasons composing this category like, for example, *effort*, *risk for the project*, and *lack of time*. A complete view of the reasons for TD non-payment can support the team in defining strategies to increase its capacity to manage TD items.

Through the analysis of the nature (technical or managerial) of the reasons, software practitioners can see if their TD payment difficulties are more due to technical or managerial issues. This can help the organization focus improvements. For example, primarily managerial reasons might indicate that communication about technical issues needs to be improved so that managers understand the impact of those issues on business concerns. Conversely, primarily technical reasons might indicate that new training, processes, or tools would help.

The map also supports the identification of appropriate TD payment practices. For example, if a team is starting to pay TD items off and is looking for a starting point, it can consult the map for the most used practices. On the other hand, if a team has previous experience in performing TD payment activities, the map can reveal new practices that the team has not yet used or considered. In particular, the categorization of practices can lead teams to identify new practices that are similar to ones they already employ, thus making it easier to adopt such new practices.

Lastly, as a communication device, the map is able to define a favorable environment for reducing TD items, making the practices and reasons visible. Software teams can use the map to demonstrate their technical and managerial problems to managers. Then, managers can be aware that their decisions can support or not TD payment initiatives, and can reduce, or increase, the difficulty of paying off TD items. Also, they can use what they know about their work and project context to find improvements that are focused on that context. For example, if there is a desire to improve TD payment and that TD issues often arise from organizational decisions, they could consult reasons or practices for setting a favorable context for TD payment from the category *organizational* on the map.

5.2.3. Assessing the technical debt conceptual model and technical debt payment map

We conducted a complementary survey to investigate the perception of software practitioners on the accuracy and completeness of the proposed TD conceptual model and TD payment map.

5.2.3.1. Data collection. The complementary survey[7] was composed of eleven questions, as shown in Table 12 divided into two sections. In the first, the participants provided their perception of the TD conceptual model (Q1 to Q6), and the latter captures the participants' perceptions of the TD payment map (Q7 to Q11).

As we only invited practitioners who had answered the *InsighTD* survey, we only needed to ask participants to provide their email address, which we could then use as a participant identifier. This allowed us to match the answers collected in *InsighTD* to ones collected in the complementary survey. We then instructed the practitioners to watch a 5.3-min explanatory video[8] that summarizes the TD conceptual model. We decided to use an explanatory video instead of text to facilitate the practitioners' participation. The focus of the video is on our extension to the model. Also, we provided the video transcript and the image of the model in high resolution[9] The participants then answered Q1 to Q6, in which we asked if anything about their perception of TD payment changed and what they learned, if they disagreed with any concept or relationship proposed in the model's extension, and how accurately and completely the model represents TD payment concepts.

Next, we instructed the practitioners to watch an 8.5-min explanatory video[10] summarizing the TD payment map. We also provided the video transcript and the image of the map in high resolution[11] The participants answered Q7 to Q10, in which we ask if the map would influence participant's decisions about how and when to pay debt items. Finally, the participants analyzed three statements (Q11) associated with the usefulness and self-predicted future use of the map indicating the option that best represented their point of view, according to a 5-point scale from (1) strongly agree to (5) strongly disagree. For example, the statement "I find the TD payment map easy to read, follow and use to support decisions about TD payment" is related to usefulness of the map.

5.2.3.2. Data analysis. We matched the data collected in the complementary survey with the data collected in the *InsighTD* survey, allowing us to identify the demographic data of each participant. For closed questions and demographic data, we used descriptive statistics and calculated the share of participants choosing each option to obtain a better understanding of the data. For open-ended questions, we coded the answers to identify their central ideas. For example, consider the following answer to Q1: "*I realize now that reasons for delay in paying of TD may be much more complex than it seems*". We defined the code *complexity of TD payment reasons*. The coding process was performed by two authors. One of them coded all answers and other checked the extracted codes. The divergences were resolved in a consensus meeting.

5.2.3.3. Results. In total, we sent the survey to twenty-seven practitioners considering those we have contact and could quickly participate of this stage of the research. We received eight answers,[12] representing a ∼30% response rate. Most of them work in small (38%; organizations with up to 50 employees) and medium-sized (38%; 51 to 1000 employees) organizations, and two participants (25%) work in large (more than 1000 employees) organizations. Most of the participants identified themselves as developers (63%), but requirements analysts (25%) and software architects (13%) also answered the survey. Regarding the participants' experience level, we did not receive answers from novices or beginners. Most of them are competent (50%), followed by proficient (25%) and expert (50%). The participants mainly adopted the agile software development process (50%), but others followed hybrid (25%) and traditional (25%) ones. These distributions are very close to the larger data set used in this paper.

Regarding the **TD conceptual model**, only one participant indicated that his/her perception did not change due to the model (Q1). However, this participant's response revealed their appreciation that the TD payment elements were "*better defined in categories*". The remaining participants indicated that their perception had changed as the model (i) introduced new concepts (three answers, e.g. "*there some concepts related to payment and TD reasoning that I didn't know previously*"), (ii) provided a more complete view on TD payment (e.g., "*I now understand more completely about TD payment*"), and (iii) revealed TD payment as complex (e.g., "*I realize now that reasons for delay in paying of TD may be much more complex than it seems*").

Most participants (5) reported that they learned more about TD payment by studying the model (Q2). Some quotes from the responses indicate new understandings of concepts, e.g. that the part of the model about "*the definition of a favorable context for the payment of technical debt*" added to the respondents' knowledge. Others indicate a greater appreciation of the importance of this

[7] The complete survey is available at https://bit.ly/3BEsYix.
[8] https://www.youtube.com/watch?v=h9Tlx3ZxuRM.
[9] https://bit.ly/3P0xB9S.
[10] https://youtu.be/OL5dCSPpXWM.
[11] https://bit.ly/3uhoE4k.
[12] The raw data is available at https://bit.ly/3QmiPes.

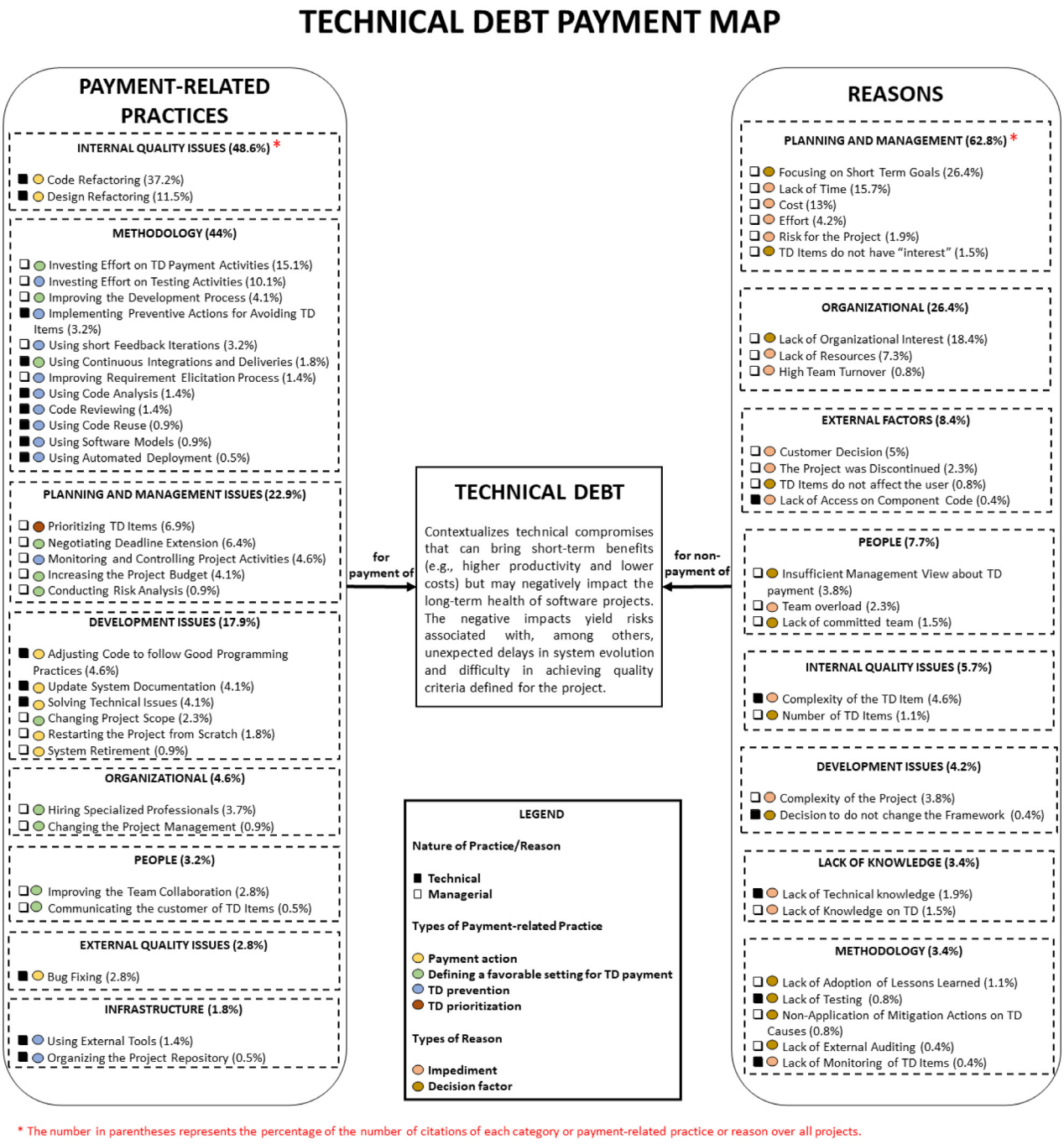


Fig. 13. The TD payment map.

knowledge for paying off TD efficiently (e.g. "*I learned that understanding the reasons behind TD and defining the payment strategy in terms of activities, is indeed prerequisite for efficient TD payment*", or "*I understood that the categorization strategy helps a lot in the payment and prioritization of TD*").

Only one participant disagreed with something in the model (Q3), specifically the lack of relationship between *ExternalFactors* and *PaymentRelatedPracticeCategory*, pointing out that "*various external factors can put pressure on the payment of technical debt*". The other participants agreed with the model and indicated that it is well organized and described.

Lastly, the participants rated the model in terms of its accuracy (Q5) and completeness (Q6). In the participants' opinion, the model seems accurate (answers ranging from 7 to 10) and complete (answers ranging from 8 to 10). For both, the mode was 9 (4 answers).

When asked about the **TD payment map**, most of the participants (7) indicated that it would influence their decisions about how to pay TD items (Q7). They explained (Q8) that the map can be used as a supporting tool for TD payment activities, e.g. "*it gives some insights or even a checklist to support a strategy to avoid and solve technical debts*", and "*it could be an initial guide for planning technical debt payment activities*". Also, one participant indicated that the percentages could contribute to comparing the TD payment initiatives of projects from the same organization. Only one participant claimed that the map would not influence their decisions on how to pay TD items and cautions that *"it is hard to generalize this data... Percent are different from case to case*".

Table 12
Complementary survey's questions.

Section	No.	Question (Q) Description	Options	Type
1: Technical debt conceptual model	Q1	After watching the video, what changed in your perception about TD payment? For example, are there concepts or ideas that you did not know about before, or that you now think about differently, or that you now understand more completely?	–	Open
	Q2	What have you learned about TD payment from the model?	–	Open
	Q3	Do you disagree with any concept or relationship presented in the conceptual model on TD payment (classes in blue)?	• Yes • No	Closed
	Q4	If yes, how? If not, why?	–	Open
	Q5	In your opinion, and considering a 0–10 scale, with 0 representing very low accuracy and 10 representing high accuracy, do you think the conceptual model is accurate to represent the TD payment concepts?	• 0–10 scale	Closed
	Q6	In your opinion, and considering a 0–10 scale, with 0 representing very low completeness and 10 representing high completeness, do you think the conceptual model is complete to represent the TD payment concepts?	• 0–10 scale	Closed
2: Technical debt payment map	Q7	Would our results, as captured in this map, influence any of your decisions about HOW to pay TD items?	• Yes • No	Closed
	Q8	If yes, how? If not, why?	–	Open
	Q9	Would our results, as captured in this map, influence any of your decisions about WHEN to pay TD items?	• Yes • No	Closed
	Q10	If yes, how? If not, why?	–	Open
	Q11	Concerning the map, choose the option that best represents your opinion. (a) I find the TD payment MAP easy to read, follow and use to support decisions about TD payment. (Useful and Controllable). (b) I find the TD payment MAP easy to read, follow and use to support decisions about TD non-payment. (Useful and Controllable). (c) Assuming the payment map were available at my job, I predict that I would use it on a regular basis in the future.	• Strongly agree • Agree • Neutral • Disagree • Strongly disagree	Closed

Most of the participants (6) affirmed that the map could influence decisions about when to pay TD items (Q9) and explained (Q10) that "*when analyzing the map, the first thing I think about is: Paying technical debt and managing to reduce technical debts should be constant activities. In other words, the 'when' is today, not to do everything at once, but to create the culture of starting to visualize the map and apply constant actions to make decisions about the technical debt and its payment*" and "*yes. It presents what should be improved and which phase you should tackle problems to avoid technical debts*". On the other hand, a participant reported that "*most of time I'm not a decision maker when and if that will be done. Anyhow, the map gives valuable insights to make such decision*".

Finally, in Q11, most of the participants indicated that the map is easy to read, follow, and use to support decisions about TD payment (strongly agree: 4 answers, agree: 3 answers, and neutral: 1 answer) and about TD non-payment (strongly agree: 2 answers, agree: 5 answers, and neutral: 1 answer). Also, five participants predicted that they would use the map regularly in the future if the map is available at their jobs (strongly agree: 3 answers, agree: 2 answers, and neutral: 3 answers).

In summary, results indicate that the TD conceptual model is well organized, accurate and complete, as well as providing valuable information to define strategies for TD payment. Also, the TD payment map seems to be useful as a support tool in TD payment activities, but it must be adapted according to practitioners' context.

5.2.4. Possible interpretations of the findings

In this section, we provide our interpretations of the correlations we identified among practices and among reasons. Although these interpretations can support us to contextualize the evidence we found, they are not findings of our study. Further studies should be performed to investigate if they really hold in practice.

We start interpreting the correlations among practices shown in Fig. 7. By analyzing the positive correlation between *increasing the project budget* and *negotiating deadline extension,* it seems to mean that the more time a team needs to finish its activities, the more money could be necessary to support it. That is, extending a deadline in many cases necessitates a budget increase. However, the data tell us that there are negative correlations between the practice *increasing the project budget* and the practices *investing effort on TD payment activities*, *investing effort on testing activities*, *adjusting code to follow good programming practices*, and *monitoring and controlling project activities,* which indicates that many projects that are increasing the budget to pay debt are blindly throwing money at the problem because they are not applying other practices to directly eliminate the debt.

Analyzing the cluster formed by *code refactoring* and its positive correlated practices (*design refactoring*, *improving the development process*, *investing effort on testing activities*, and *prioritizing TD items*) indicates that these practices are essential for supporting the execution of *code refactoring*. Further, it might indicate that the use of *code refactoring* is an indicator of a level of project maturity in which other advanced practices are employed. However, the negative correlation between *code refactoring* and *negotiating deadline extension* can reveal that software teams did not include time in their projects to perform *code refactoring*, and this can compromise the quality of the code refactoring activities. The practice *negotiating deadline extension* is negatively correlated to *improving the development process*. This can indicate that process improvement activities are only performed if there is sufficient time in the project. On the contrary, for performing the positively correlated practices (*increasing the project budget* and *investing effort on TD payment practices*), it is necessary to have more time. In summary, we can interpret these correlations to say that, if projects really intend to pay TD items off, they need to plan its execution in the project ahead of time.

Now, we interpret the correlations among reasons for not paying TD shown in Fig. 11. The reason *customer decision* is correlated with all ten most cited reasons. This may mean that the customer always has a role in making TD payment decisions. This seems to contradict the common notion in the literature that TD

is invisible to the user, and deserves further examination. Among the reasons that are correlated with *customer decision* are both technical and managerial reasons, indicating that combinations of both internal and external factors affect the non-payment of TD. The cluster composed of *cost* and its correlated reasons *complexity of the project*, *complexity of the TD item*, and *effort* can indicate TD payment is likely to have a high cost on projects where other costs are also high, as is the case when the work is highly complex. Lastly, the reason *lack of time* is correlated to *cost*, *customer decision*, *focusing on short term goals*, and *lack of organizational interest*. This combination of reasons might indicate an overall project context in which the primary focus is on getting functionality to customers as quickly as possible, as in a startup context, for example.

5.3. Implications for practitioners and researchers

The conceptual map for TD payment (Fig. 13) is designed to help practitioners who are planning their TD payment strategies. Practitioners can discover new practices they can apply in a particular area, and can focus on particular types (e.g., direct payment practices) or natures (e.g., technical) that are appropriate for their context. Similarly, they can identify reasons for non-payment that are relevant to their project, and use the map to identify other related reasons that might come up in the future, as well as understand those reasons better in terms of their type, nature, and category. This will help them plan better to avoid or counteract these reasons in the future. Practitioners can also analyze the frequency of practices and reasons in their workspace to choose the better options for paying off TD items or draw conclusions about the reasons that curb the application of these practices. It is also possible to use the information about the relationship between practices and types of debt to identify the practices that are most appropriate for reducing a specific type of debt that is relevant in a particular context. Finally, the identified set of correlations among practices and among reasons could further support practitioners in understanding what practices could be used in combination or what reasons contribute together for the non-payment of debt items. This last type of information can provide valuable knowledge on how to create a favorable scenario for debt payment.

To help support practitioners in their use of the map for planning TD payment strategy, we also offer the following key takeaways that arose from our data:

1. Only 40% of the TD items described in our data were paid off. Thus, paying off TD should not be the only focus of TDM.
2. Not all TD payment activities result directly in the elimination of TD. Implementing practices to enable future TD payment, to prevent it, and to prioritize it are also necessary.
3. Eliminating debt items cannot be solely a technical concern. Management practices are necessary as well for implementing TD payment initiatives, and making them part of the TD management strategy is necessary.
4. Choosing practices from the categories *development issues*, *internal quality issues*, *methodology*, and *planning and management* can support software practitioners to address multiple types of debt.
5. *Focusing on short term goals* is not only a leading cause of incurring TD, but also a primary reason for perpetuating it.
6. In most cases, the reasons for TD non-payment have a managerial nature regardless of their type (decision factor or impediment), revealing that software practitioners must disseminate the importance of TD payment to their managers.
7. Payment of all types of TD face obstacles from the category *planning and management*, implying that involving project managers in TD payment initiatives is crucial.

Many of these key highlights, which arise from various parts of our data analysis, point to the important relationship between technical and managerial concerns that are central to the management of TD. This is an important consideration not only for practitioners, but for researchers as well, as TD research focusing only on technical concerns will be limited in its impact. To that end, the conceptual model presented in Fig. 12 can guide researchers to new investigations on the relationship between activities for TDM. As demonstrated in the model, there is empirical evidence that the activities payment, prevention, and prioritization are related to each other when software practitioners reduce TD items in their projects. Further, our findings can motivate new pieces of research in a problem-driven way, considering the TD payment map and the correlated practices and reasons as starting point. The development of new approaches for TD payment could more closely reflect the practitioners' needs.

6. Comparison to related work

This section presents a comparison to related work on TD payment practices and the reasons for not paying off TD.

6.1. Technical debt payment-related practices

We compared the TD payment-related practices found in this work and the ones reported by Yli-Huumo et al. (2014), Abad et al. (2016), Samarthyam et al. (2017), Bomfim and Santos (2017), Toledo et al. (2019), Apa et al. (2020) and Rios et al. (2020a). The complete comparison is available in Appendix Table 26.

During the comparison, we realized that some practices were at different abstraction levels in different studies. For instance, the practices *remove the business logic inside the communication layer* and *move the business logic to the services* were reported by Toledo et al. (2019). On the other hand, we have the practice *adjusting code to follow good programming practices*, which is at a higher abstraction level than those from Toledo et al. (2019), but is similar.

The practices for defining a favorable setting for TD payment that were found both in our work and in at least some other past studies are: *conducting risk analysis*, *improving the team collaboration*, *increasing the project budget*, *investing effort on TD payment activities*, and *negotiating deadline extension*. TD payment actions that appear both in our work and related work are *adjusting code to follow good programming practices*, *bug fixing*, *code refactoring*, *design refactoring*, and *update system documentation*. The only TD prevention practices reported both in our study and past work are *code reviewing* and *improving requirement elicitation process*. The practice *prioritizing TD items* was confirmed in two related works, in addition to ours. However, some practices recognized in related work were not identified in our study. These practices are *quantifying TD*, *using palliative solutions*, and *service developers must learn new technologies*. The distinctions and overlaps between the unique TD payment-related practices found in our work and the related work is shown in Fig. 14.

Table 13 presents the comparison between the categories we identified in this work and the categories reported in Li et al. (2015) and Behutiye et al. (2017). We can observe that we confirmed the categories *internal quality issues*, *methodology*, *development issues*, and *external quality issues*. The other categories (*planning and management*, *people*, *organizational*, and *infrastructure*) were only identified in our work.

In summary, our findings on TD payment-related practices and their categories complement and extend the set of information already reported in the technical literature.

Table 13
Comparison to related work on categories of TD payment-related practices.

Our categories	Categories from Li et al. (2015)	Categories from Behutiye et al. (2017)	Overlapping degree
Internal quality issues	Refactoring Rewriting	Refactoring –	Total Total
Methodology	Automation Repackaging	Test automation –	Partial Partial
Development issues	Reengineering Fault tolerance	Code analysis –	Partial Partial
External quality issues	Bug fixing	–	Total
Planning and management	–	–	–
People	–	–	–
Organizational	–	–	–
Infrastructure	–	–	–

Table 14
Comparison to previous work on TD payment-related practices.

Practices from	
Our study	Pérez et al. (2021)
Adjusting code to follow good programming practices (10)[a] Code refactoring (81) Code reviewing (3)	Adoption of good practices (3) Refactoring (13) Code reviewing (2)
Design refactoring (25)	Architectural changes (1) Improve design (6)
Improving requirement elicitation process (3) Improving the development process (9) Improving the team collaboration (6) Increasing the project budget (9)	Improve requirements elicitation (1) Tech independent implementation (1) Improve communication (2) Budget increase (3)
Investing effort on TD payment activities (33)	Extra effort (2) Incremental payment (2)
Investing effort on testing activities (22) Prioritizing TD Items (15) Solving technical issues (9) System retirement (2) Using external tools (3)	Improve testing (3) Backlog inclusion (1) Technology/tool/platform change (1) System retirement (1) External tools (2)

[a]The number in parenthesis indicates the number of citations.

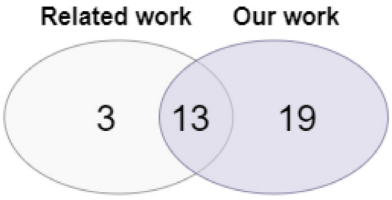


Fig. 14. Number of unique TD payment-related practices found in our work and the related work.

6.1.0.1. Comparing the identified practices with those reported by Pérez et al. (2021). By analyzing 72 answers given by software architects, our previous work (Pérez et al., 2021) identified 16 practices associated with TD payment, focusing on *InsighTD* responses from architects only. These practices were also recognized in the present study, but we increased the list to 32 practices. Table 14 presents the practices and their numbers of citations as reported in both studies. Some practices have different nomenclatures. This happens because we reanalyzed the answers during the consolidation of the answers collected in Costa Rica and Serbia to those previously collected in Brazil, Chile, Colombia, and the United States.

We further investigated how similar our list of TD payment-related practices is to the one reported by Pérez et al. (2021) by performing the rank-biased overlap (RBO) analysis (Webber et al., 2010), which quantitatively measures how similar the ranked lists are. RBO gives a value ranging from 0 to 1. The closer this value is to 1, the greater the similarity between the lists. As RBO supports top-weighted ranked lists, the first elements of a list have more impact on the similarity index than the last ones. We can configure what elements will be compared by setting the *p-value*, which, differently than the p statistic, refers to a level of overlapping and the degree of top-weightedness. In the analysis, we chose p-value ranging from 0.5 (only the very initial elements of a rank are considered) to 0.9 (almost all elements are considered). Before performing the analysis, we consolidated the list of practices reported by Pérez et al. (2021) to assure that the different nomenclature would not influence the comparison.

Fig. 15 presents the result of the comparison between the ranked lists of TD payment-related practices for each study. The RBO analysis reveals that the similarity level is about 70%–80% between the two lists. It indicates that the most cited practices in both studies are similar, but the similarity diminishes when more practices are included in the analysis, i.e., the *p-value* increases. Our study confirms and extends the list of practices reported in the previous work (Pérez et al., 2021), supporting us to increase the ecological validity. Extending our prior work beyond a focus on architects helps us validate that decisions made by other practitioners and architects alike are likely to generalize.

6.2. Reasons for not paying off TD

Table 15 presents the comparison of the reasons found in this work with the ones reported by Bomfim and Santos (2017). Only the reasons *lack of testing*, *complexity of the project*, and *effort* are reported previously. Thus, our results confirm and extend the

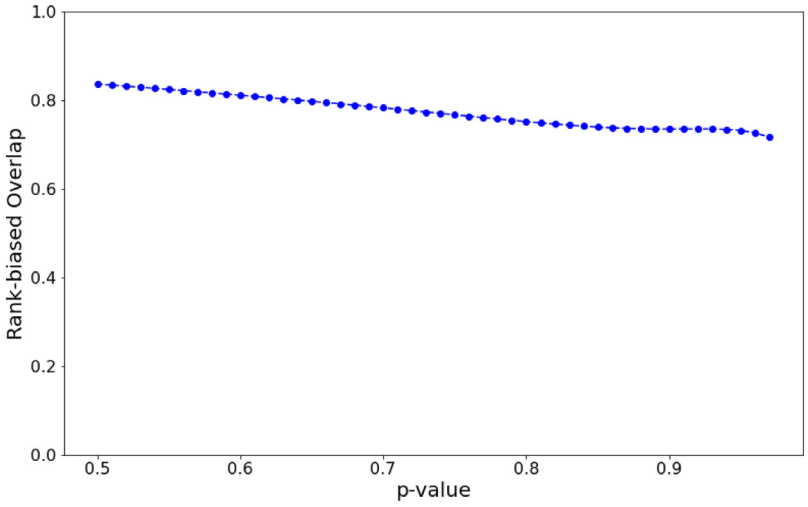


Fig. 15. RBO comparing TD payment-related practices from our and previous study.

Table 15
Comparison to related work on reasons for not reducing TD items.

	Our study	Bomfim and Santos (2017)
Decision factor	Decision to do not change the framework	–
	Focusing on short term goals	–
	Insufficient management view about TD payment	–
	Lack of adoption of lessons learned	–
	Lack of committed team	–
	Lack of external auditing	–
	Lack of organizational interest	–
	Lack of testing	Lack of test coverage or excessive manual testing
	Non-application of mitigation actions on TD causes	–
	Number of TD items	–
	TD Items do not affect the user	–
	TD Items do not have "interest"	–
Impediment	Complexity of the project	Concern of impacting some module because the team does not know all parts of the code to carry out a deeper impact analysis
	Complexity of the TD item	–
	Cost	–
	Customer decision	–
	Effort	Low impact for business and high effort
	High team turnover	–
	Lack of access on component code	–
	Lack of knowledge on TD	–
	Lack of monitoring of TD items	–
	Lack of resources	–
	Lack of technical knowledge	–
	Lack of time	–
	Risk for the project	–
	Team overload	–
	The project was discontinued	–

current knowledge on reasons for not paying off TD from software projects.

7. Threats to validity

There are threats to validity that could affect our work. We used the following categorization of Wohlin et al. (2012) to identify and analyze the threats: (i) construct validity (threats from this category are related to the design of the experiment or social factors), (ii) conclusion validity (this is associated with threats that affect the ability to draw correct conclusions from the results), (iii) internal validity (threats from this category are associated with other factors that could affect the results without the researcher's knowledge), and (iv) external validity (this is related to the possibility of generalizing the results). We tried to remove them when possible or mitigate their effects when removal was not possible.

7.1. The insightd survey

Construct validity. A threat arises from the validity of the participants' responses. As the survey was answered remotely, the participants can respond to it without understanding the concept of TD as intended. To reduce this threat, in Q13, the participants had to describe a TD item that occurred in their project. Then, only the responses of the participants with the example given in Q13 that actually described a TD item were considered. Another threat emerges from the *InsighTD* questionnaire, as it was answered remotely, allowing for potential misunderstandings of the questions that can lead to meaningless answers. To mitigate this threat, several reviews (three internal and one external) and a pilot study were performed before the questionnaire application. More details on these reviews and the pilot study are described in Rios et al. (2020b).

Conclusion validity. We identified three threats affecting the conclusion validity. First, a threat is related to the identification of TD type, the coding of practices and reasons, and the grouping processes used in this study. As the identification of TD types and the coding process are subjective and subject to inconsistencies, both processes were performed by three researchers of each replication team playing different roles: identifier, reviewer, and referee. For the grouping process, a researcher grouped the practices or reasons into types, natures, and categories, and an experienced researcher reviewed the grouping. Another threat is associated with the sample size used in the correlation test performed for practices and reasons. Although we handled a small sample, this characteristic aided us in choosing the appropriate statistical test, which was the Kendall correlation test. We carefully followed its assumptions to guarantee the validity of our test. Lastly, some of the data may seem old, but practices and reasons for TD payment are related to the what and not about how. For example, the practice *code refactoring* (the *what*) can be done by several means (the *how*), such as using external tools to automating refactoring, or by doing some small changes in the code. Therefore, the practice could remain updated no matter how it was performed. Similarly, a reason for TD non-payment such as *customer decision* (the *what*) can come from multiple sources (the *how*).

Internal validity. A threat to internal validity arises when causal conclusions are drawn. Our data and analysis are based on correlations, so it is not vulnerable to internal validity threats. Our interpretations of those correlations, however, are in fact vulnerable to internal validity threats and thus can only be viewed as pointers to future investigations.

External validity. We reduced this threat by targeting industry practitioners and seeking to achieve respondent diversity from several countries. Although this diversity brings a good view of TD payment and their practices and reasons for non-application of these practices, we cannot say how generalizable the results are because we are not able to estimate the representativeness of our sample given the lack of empirical data characterizing the population. However, an argument can be made that the ecological validity (Andrade, 2018) of the work, i.e., the extent to which these findings approximate other real-world scenarios, is likely to hold in other settings.

7.2. The complementary survey

Construct validity. A threat arises from the questionnaire because its questions could be misunderstood by the participants. To reduce this threat, we performed three validations conducted by researchers from the *InsighTD* project. The objective of the validation was to check the questions for clarity and completeness. Then, following the feedback we received, several adjustments were applied in the questionnaire, such as the inclusion of the video transcripts. Also, we piloted the questionnaire before its execution, by inviting three participants through personal contacts, with varying levels of industry experience. These participants were asked to tell us how much time it took to complete the task (the mean time was about 20 min), impressions about questions (e.g., clarity, ease of understanding, size), and improvement points. Finally, we performed a last review of the questionnaire.

Conclusion validity. A threat emerges from the coding process we performed to analyze the answers collected in the open-ended questions. As this process is subjective, it was performed by two different researchers. One of them coded the answers and the other checked the extracted codes. Inconsistencies were solved in a meeting.

External validity. A threat arises from the fact that study participants were chosen for convenience because we invited only practitioners who answered the *InsighTD* survey. Also, the results are not generalizable. To this end, further studies need to be performed to evaluate the extended TD conceptual model and the proposed TD payment map.

8. Final remarks

This work reveals the point of view of practitioners on TD payment. We reported the practices related to the payment of TD items, and the reasons for avoiding the application of these practices. All this information was organized in a conceptual map that can support software teams in the identification of practices for paying off TD items and reasons curbing the application of these practices. We used a correlation test for identifying practices and reasons that occur in combination and described a set of interpretations of these correlations using our background. We also evolved the conceptual model for TD including the empirical evidence on TD payment. Lastly, we performed an assessment of the model and the map considering the software practitioners' perceptions.

As future work, we intend to (i) investigate how the payment practices and reasons for TD non-payment are used in different contexts, such as process model, (ii) perform further studies to evaluate the extended version of the TD conceptual model and the proposed TD payment map, and (iii) develop a recommendation system to dynamically generate the conceptual map considering different parameters like company size, process model, and practitioners' level of experience.

CRediT authorship contribution statement

Sávio Freire: Writing – original draft, Formal analysis, Conceptualization, Methodology, Investigation. **Nicolli Rios:** Writing – original draft, Formal analysis, Conceptualization, Methodology, Investigation. **Boris Pérez:** Formal analysis, Conceptualization, Methodology, Investigation. **Camilo Castellanos:** Formal analysis, Conceptualization, Methodology, Investigation. **Darío Correal:** Formal analysis, Conceptualization, Methodology, Investigation. **Robert Ramač:** Formal analysis, Conceptualization, Methodology, Investigation. **Vladimir Mandić:** Formal analysis, Conceptualization, Methodology, Investigation, Writing – review & editing. **Nebojša Taušan:** Formal analysis, Conceptualization, Methodology, Investigation. **Gustavo López:** Formal analysis, Conceptualization, Methodology, Investigation. **Alexia Pacheco:** Formal analysis, Conceptualization, Methodology, Investigation. **Manoel Mendonça:** Project administration, Supervision, Writing – original draft, Formal analysis, Conceptualization, Methodology, Investigation. **Davide Falessi:** Formal analysis, Conceptualization, Methodology, Investigation. **Clemente Izurieta:** Writing – review & editing, Methodology, Investigation. **Carolyn Seaman:** Project administration, Supervision, Writing – original draft, Formal analysis, Conceptualization, Methodology, Investigation. **Rodrigo Spínola:** Project administration, Supervision, Writing – original draft, Formal analysis, Conceptualization, Methodology, Investigation.

Declaration of competing interest

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

Data availability

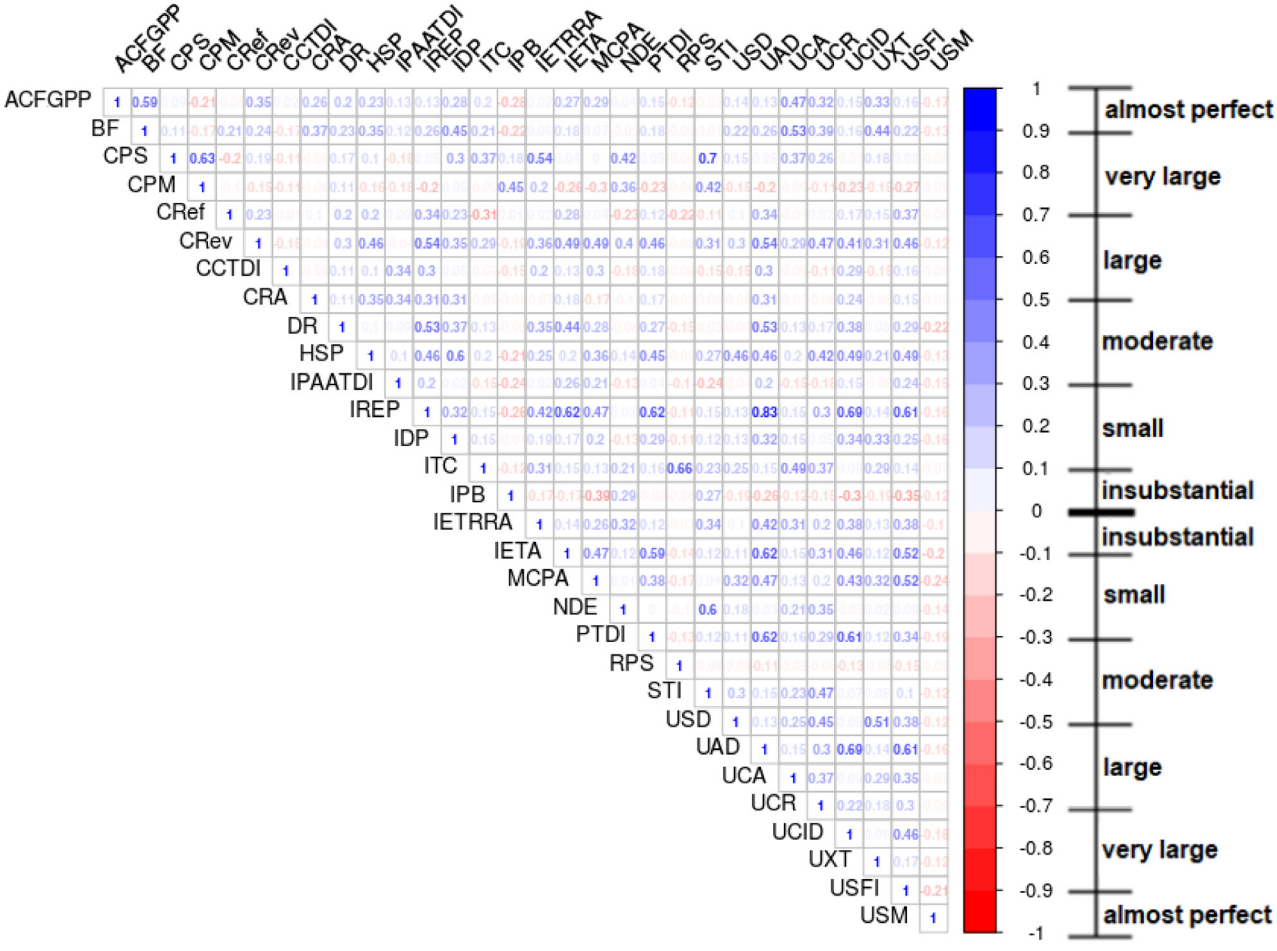
Data will be made available on request.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) Finance Code 001 and the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). This research was also supported in part by funds received from the David A. Wilson Award for Excellence in Teaching and Learning, which was created by the Laureate International Universities network to support research focused on teaching and learning. For more information on the award or Laureate, please visit www.laureate.net.

Appendix

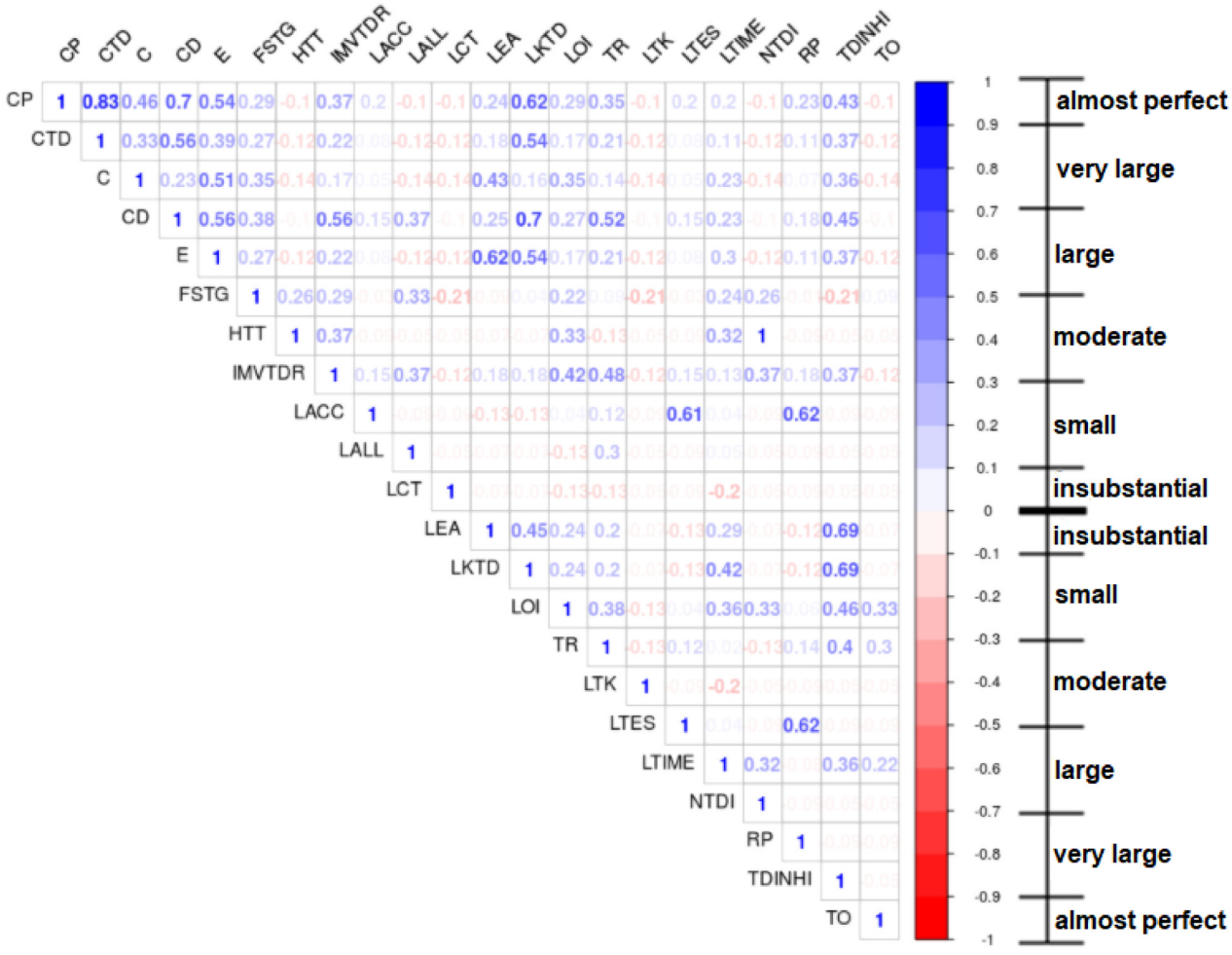
See Tables 16–26 and Figs. 16 and 17.



CAPTION:

ACFGPP: Adjusting Code to follow Good Programming Practices
BF: Bug Fixing
CPS: Changing Project Scope
CPM: Changing the Project Management
CRef: Code Refactoring
CRev: Code Reviewing
CCTDI: Communicating the customer of TD Items
CRA: Conducting Risk Analysis
DR: Design Refactoring
HSP: Hiring Specialized Professionals
IPAATDI: Implementing Preventive Actions for Avoiding TD Items
IREP: Improving Requirement Elicitation Process
IDP: Improving the Development Process
ITC: Improving the Team Collaboration
IPB:Increasing the Project Budget
IETRRA: Investing Effort on TD Payment Activities
IETA: Investing Effort on Testing Activities
MCPA: Monitoring and Controlling Project Activities
NDE: Negotiating Deadline Extension
PTDI: Prioritizing TD Items
RPS: Restarting the Project from Scratch
STI: Solving Technical Issues
USD: Update System Documentation
UAD: Using Automated Deployment
UCA: Using Code Analysis
UCR: Using Code Reuse
UCID: Using Continuous Integrations and Deliveries
UXT: Using External Tools
USFI: Using short Feedback Iterations
USM: Using Software Models

Fig. 16. Correlation matrix between the TD payment-related practices.



CAPTION:

CP: Complexity of the Project	LEA: Lack of External Auditing
CTD: Complexity of the TD Item	LKTD: Lack of Knowledge on TD
C: Cost	LOI: Lack of Organizational Interest
CD: Customer Decision	TR: Lack of Resources
E: Effort	LTK: Lack of Technical knowledge
FSTG: Focusing on Short Term Goals	LTES: Lack of Testing
HTT: High Team Turnover	LTIME: Lack of Time
IMVTDR: Insufficient Management View about TD Payment	NTDI: Number of TD Items
LACC: Lack of Access on Component Code	RP: Risk for the Project
LALL: Lack of Adoption of Lessons Learned	TDINHI: TD Items do not have Interest
LCT: Lack of Committed Team	TO: Team overload

Fig. 17. Correlation matrix between the reasons for not paying TD items off.

Table 16
TD payment-related practices and their examples of citation.

Payment-related practice	Quotes from participants
Adjusting code to follow good programming practices	"Benefits of shared library cutting down on dev time duplicating or maintaining extra code."
Bug fixing	"To make it solid and defect free system."
Changing project scope	"Redefining the scope of the project."
Changing the project management	"There were changes in team leaders (…)."
Code refactoring	"We rewrote the offending code."
Code reviewing	"The identified tech debt has been resolved through updated designs and refactors. There was a lot of post completion code reviewing to identified areas with performance impacts."
Communicating the customer of TD Items	"(…), communication of TD items to the customer, (…)."
Conducting risk analysis	"Through risk mitigation plans and action plans."
Design refactoring	"Refactoring and changing architecture."
Hiring specialized professionals	"There were cases where it was necessary to hire a specialized team (…)."
Implementing preventive actions for avoiding TD items	"(…) periodic reviews were established."
Improving requirement elicitation process	"The project managed to align itself with the milestones by aggressively simplifying the definition of requirements for the implementation process."
Improving the development process	"Improvement in the development and changes process."
Improving the team collaboration	"Relationship between development team and business team improves and gets better and better."
Increasing the project budget	"With cost overruns in money and time, updating the solution to the latest changes in technology."
Investing effort on TD payment activities	"Taking sprints to pay down the debt."
Investing effort on testing activities	"Additional development hours and code testing."
Monitoring and controlling project activities	"With the measurement bulletins, there was the metric of progress."
Negotiating deadline extension	"It was possible to negotiate time with the user to make improvements."
Organizing the project repository	"After organizing the repository, it was possible to deliver in a flow, (…)."
Prioritizing TD items	"Due to the project deadline, as soon as the DT is detected, we already plan its development for the next sprints of the project."
Restarting the Project from Scratch	"Starting the project 100% from scratch."
Solving technical issues	"Updated the version of the framework."
System retirement	"The delay of the project was so long that it was canceled."
Update system documentation	"The documentation was updated."
Using automated deployment	"The deployment and testing process was further automated to streamline developments."
Using code analysis	"Each of the reports generated by Sonar was analyzed and the improvements that made it possible to cover the debt were applied."
Using code reuse	"It was necessary to redo the code, design common functions, reuse code (…)."
Using continuous integrations and deliveries	"Everything that is integration and continuous deployment was implemented through Gitlab (…)"
Using external tools	"The proposed operations were performed by sonar."
Using short feedback iterations	"(…)shorter and shorter feedback cycles."
Using software models	"(…)The project models were developed (…)"

Table 17
TD payment-related practices per type.

Type	Payment-related practice	#CPRP	%PRPP
Defining a favorable setting for TD payment	Investing effort on TD payment activities	33	15.1%
	Negotiating deadline extension	14	6.4%
	Improving the development process	9	4.1%
	Increasing the project budget	9	4.1%
	Hiring specialized professionals	8	3.7%
	Improving the team collaboration	6	2.8%
	Changing project scope	5	2.3%
	Using continuous integrations and deliveries	4	1.8%
	Changing the project management	2	0.9%
	Conducting risk analysis	2	0.9%
	Communicating the customer of TD items	1	0.5%
Payment action	Code refactoring	81	37.2%
	Design refactoring	25	11.5%
	Adjusting code to follow good programming practices	10	4.6%
	Solving technical issues	9	4.1%
	Update system documentation	9	4.1%
	Bug fixing	6	2.8%
	Restarting the project from scratch	4	1.8%
	System retirement	2	0.9%

(*continued on next page*)

Table 17 (*continued*).

Type	Payment-related practice	#CPRP	%PRPP
TD prevention	Investing effort on testing activities	22	10.1%
	Monitoring and controlling project activities	10	4.6%
	Implementing preventive actions for avoiding TD items	7	3.2%
	Using short feedback iterations	7	3.2%
	Code reviewing	3	1.4%
	Improving requirement elicitation process	3	1.4%
	Using code analysis	3	1.4%
	Using external tools	3	1.4%
	Using code reuse	2	0.9%
	Using software models	2	0.9%
	Organizing the project repository	1	0.5%
	Using automated deployment	1	0.5%
TD prioritization	Prioritizing TD items	15	6.9%

Caption:
#CPRP - Count of payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218).

Table 18
TD payment-related practices per nature.

Nature	Payment-related practice	#CPRP	%PRPP
Managerial	Investing effort on TD payment activities	33	15.1%
	Investing effort on testing activities	22	10.1%
	Prioritizing TD items	15	6.9%
	Negotiating deadline extension	14	6.4%
	Monitoring and controlling project activities	10	4.6%
	Improving the development process	9	4.1%
	Increasing the project budget	9	4.1%
	Hiring specialized professionals	8	3.7%
	Using short feedback iterations	7	3.2%
	Improving the team collaboration	6	2.8%
	Changing project scope	5	2.3%
	Restarting the project from scratch	4	1.8%
	Improving requirement elicitation process	3	1.4%
	Changing the project management	2	0.9%
	Conducting risk analysis	2	0.9%
	System retirement	2	0.9%
	Communicating the customer of TD items	1	0.5%
Technical	Code refactoring	81	37.2%
	Design refactoring	25	11.5%
	Adjusting code to follow good programming practices	10	4.6%
	Solving technical issues	9	4.1%
	Update system documentation	9	4.1%
	Implementing preventive actions for avoiding TD items	7	3.2%
	Bug fixing	6	2.8%
	Using continuous integrations and deliveries	4	1.8%
	Code reviewing	3	1.4%
	Using code analysis	3	1.4%
	Using external tools	3	1.4%
	Using code reuse	2	0.9%
	Using software models	2	0.9%
	Organizing the project repository	1	0.5%
	Using automated deployment	1	0.5%

Caption:
#CPRP - Count of payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218).

Table 19
TD payment-related practices per category.

Category	Payment-related practice	#CPRP	%PRPP
Development issues	Adjusting code to follow good programming practices	10	4.6%
	Solving technical issues	9	4.1%
	Update system documentation	9	4.1%
	Changing project scope	5	2.3%
	Restarting the project from scratch	4	1.8%
	System retirement	2	0.9%
External quality issues	Bug fixing	6	2.8%
Infrastructure	Using external tools	3	1.4%
	organizing the project repository	1	0.5%

(*continued on next page*)

Table 19 (*continued*).

Category	Payment-related practice	#CPRP	%PRPP
Internal quality issues	Code refactoring	81	37.2%
	Design refactoring	25	11.5%
Methodology	Investing effort on TD payment activities	33	15.1%
	Investing effort on testing activities	22	10.1%
	Improving the development process	9	4.1%
	Implementing preventive actions for avoiding TD items	7	3.2%
	Using short feedback iterations	7	3.2%
	Using continuous integrations and deliveries	4	1.8%
	Code reviewing	3	1.4%
	Improving requirement elicitation process	3	1.4%
	Using code analysis	3	1.4%
	Using code reuse	2	0.9%
	Using software models	2	0.9%
	Using automated deployment	1	0.5%
Organizational	Hiring specialized professionals	8	3.7%
	Changing the project management	2	0.9%
People	Improving the team collaboration	6	2.8%
	Communicating the customer of TD items	1	0.5%
Planning and management	Prioritizing TD items	15	6.9%
	Negotiating deadline extension	14	6.4%
	Monitoring and controlling project activities	10	4.6%
	Increasing the project budget	9	4.1%
	Conducting risk analysis	2	0.9%

Caption:
#CPRP - Count of payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218).

Table 20
Correlation between the TD payment-related practices.

TD payment-related practices	Adjusting Code to follow Good Programming Practices	Bug Fixing	Changing Project Scope	Changing the Project Management	Code Refactoring	Code Reviewing	Communicating the customer of TD Items	Conducting Risk Analysis	Design Refactoring	Hiring Specialized Professionals	Implementing Preventive Actions for Avoiding TD Items	Improving Requirement Elicitation Process	Improving the Development Process	Improving the Team Collaboration	Increasing the Project Budget	Investing Effort on TD Payment Activities	Investing Effort on Testing Activities	Monitoring and Controlling Project Activities	Negotiating Deadline Extension	Prioritizing TD Items	Restarting the Project from Scratch	Solving Technical Issues	Update System Documentation	Using Automated Deployment	Using Code Analysis	Using Code Reuse	Using Continuous Integrations and Deliveries	Using External Tools	Using short Feedback Iterations	Using Software Models
Adjusting Code to follow Good Programming Practices	**0***	**0.0012**	0.6343	0.2429	0.7281	0.0562	0.9311	0.1483	0.2359	0.2074	0.4616	0.4813	0.1199	0.2750	0.1188	0.9090	0.1275	0.1043	0.8312	0.3877	0.5153	0.7812	0.4437	0.4813	**0.0093**	0.0836	0.3971	0.0662	0.3498	0.3491
Bug Fixing	**0.0012**	**0**	0.5496	0.3694	0.2090	0.1865	0.3694	**0.0455**	0.1808	0.0573	0.5244	0.1554	**0.0149**	0.2548	0.2305	0.7585	0.3257	0.6821	0.7788	0.3192	0.6171	0.9681	0.2285	0.1554	**0.0040**	**0.0362**	0.3951	**0.0165**	0.2210	0.4718
Changing Project Scope	0.6343	0.5496	**0**	**0.0007**	0.2256	0.3105	0.5496	0.7389	0.3426	0.5847	0.3220	0.7867	0.1044	**0.0457**	0.3113	**0.0034**	0.8426	1.0000	**0.0218**	0.7782	0.7389	**0.0001**	0.4221	0.7867	**0.0457**	0.1627	0.9665	0.3113	0.9072	0.6314
Changing the Project Management	0.2429	0.3694	**0.0007**	**0**	0.5685	0.4233	0.5496	0.7389	0.5187	0.3711	0.3220	0.2790	0.7867	0.6316	**0.0123**	0.2659	0.1529	0.1013	**0.0441**	0.1842	0.7389	**0.0217**	0.4221	0.2790	0.6316	0.5496	0.2077	0.4241	0.1297	0.6314
Code Refactoring	0.7281	0.2090	0.2256	0.5685	**0**	0.1610	0.9716	0.5514	0.1933	0.2206	0.5781	**0.0423**	0.1682	0.0595	0.9545	0.9001	0.0849	0.7858	0.1528	0.4637	0.1901	0.5106	0.5661	**0.0423**	0.7972	0.9149	0.3111	0.3608	**0.0198**	0.6070
Code Reviewing	0.0562	0.1865	0.3105	0.4233	0.1610	**0**	0.4233	0.6556	0.0824	**0.0107**	0.8203	**0.0034**	0.0597	0.1087	0.2848	**0.0483**	**0.0050**	**0.0058**	**0.0266**	**0.0081**	0.6556	0.0865	0.1025	**0.0034**	0.1067	**0.0104**	**0.0239**	0.0794	**0.0088**	0.5209
Communicating the customer of TD Items	0.9311	0.3694	0.5496	0.5496	0.9716	0.4233	**0**	0.7389	0.5187	0.5847	0.0659	0.1044	0.7867	0.6316	0.4241	0.2659	0.4748	0.1013	0.3256	0.2954	0.7389	0.4233	0.4221	0.1044	0.6316	0.5496	0.1105	0.4241	0.3511	0.6314
Conducting Risk Analysis	0.1483	**0.0455**	0.7389	0.7389	0.5514	0.6556	0.7389	**0**	0.5258	0.0561	0.0699	0.0973	0.0973	0.7894	0.6561	0.6954	0.3195	0.3613	0.5840	0.3464	0.8527	0.6556	0.6547	0.0973	0.7894	0.7389	0.1824	0.6561	0.3986	0.7893
Design Refactoring	0.2359	0.1808	0.3426	0.5187	0.1933	0.0824	0.5187	0.5258	**0**	0.5704	0.5902	**0.0024**	**0.0325**	0.4654	0.7609	**0.0414**	**0.0080**	0.0993	0.6496	0.1030	0.3744	0.8789	0.9269	**0.0024**	0.4654	0.3426	**0.0258**	0.6264	0.0798	0.2011
Hiring Specialized Professionals	0.2074	0.0573	0.5847	0.3711	0.2206	**0.0107**	0.5847	0.0561	0.5704	**0**	0.5728	**0.0126**	**0.0011**	0.2824	0.2321	0.1613	0.2595	**0.0412**	0.4210	**0.0097**	0.6183	0.1295	**0.0117**	**0.0126**	0.2824	**0.0223**	**0.0069**	0.2321	**0.0053**	0.4733
Implementing Preventive Actions for Avoiding TD Items	0.4616	0.5244	0.3220	0.3220	0.5781	0.8203	0.0659	0.0699	0.5902	0.5728	**0**	0.2767	0.8982	0.4275	0.1858	0.9008	0.1509	0.2451	0.4857	0.8416	0.5812	0.1851	0.8495	0.2767	0.4275	0.3220	0.4212	0.7913	0.1683	0.4272
Improving Requirement Elicitation Process	0.4813	0.1554	0.7867	0.2790	**0.0423**	**0.0034**	0.1044	0.0973	**0.0024**	**0.0126**	0.2767	**0**	0.0866	0.4161	0.1481	**0.0235**	**0.0005**	**0.0095**	0.8489	**0.0004**	0.5465	0.4049	0.4678	**0.0000**	0.4161	0.1044	**0.0002**	0.4264	**0.0006**	0.3855
Improving the Development Process	0.1199	**0.0149**	0.1044	0.7867	0.1682	0.0597	0.7867	0.0973	**0.0325**	**0.0011**	0.8982	0.0866	**0**	0.4161	0.8283	0.3016	0.3322	0.2663	0.4848	0.0958	0.5465	0.5145	0.4678	0.0866	0.4161	0.7867	0.0640	0.0707	0.1624	0.3855
Improving the Team Collaboration	0.2750	0.2548	**0.0457**	0.6316	0.0595	0.1087	0.6316	0.7894	0.4654	0.2824	0.4275	0.4161	0.4161	**0**	0.5218	0.0912	0.3903	0.4818	0.2374	0.3581	**0.0003**	0.1995	0.1767	0.4161	**0.0072**	**0.0457**	0.6137	0.1093	0.4273	0.7007
Increasing the Project Budget	0.1188	0.2305	0.3113	**0.0123**	0.9545	0.2848	0.4241	0.6561	0.7609	0.2321	0.1858	0.1481	0.8283	0.5218	**0**	0.3478	0.3239	**0.0286**	0.1067	0.6285	0.6561	0.1343	0.2835	0.1481	0.5218	0.4241	0.0924	0.2856	**0.0430**	0.5216
Investing Effort on TD Payment Activities	0.9090	0.7585	**0.0034**	0.2659	0.9001	**0.0483**	0.2659	0.6954	**0.0414**	0.1613	0.9008	**0.0235**	0.3016	0.0912	0.3478	**0**	0.4416	0.1489	0.0697	0.4784	0.6954	0.0600	0.6041	**0.0235**	0.0912	0.2659	**0.0350**	0.4814	**0.0285**	0.5731
Investing Effort on Testing Activities	0.1275	0.3257	0.8426	0.1529	0.0849	**0.0050**	0.4748	0.3195	**0.0080**	0.2595	0.1509	**0.0005**	0.3322	0.3903	0.3239	0.4416	**0**	**0.0074**	0.4847	**0.0005**	0.4258	0.4831	0.5226	**0.0005**	0.3903	0.0806	**0.0085**	0.5039	**0.0020**	0.2517
Monitoring and Controlling Project Activities	0.1043	0.6821	1.0000	0.1013	0.7858	**0.0058**	0.1013	0.3613	0.0993	**0.0412**	0.2451	**0.0095**	0.2663	0.4818	**0.0286**	0.1489	**0.0074**	**0**	0.9562	**0.0281**	0.3613	0.9501	0.0786	**0.0095**	0.4818	0.2746	**0.0147**	0.0747	**0.0024**	0.1889
Negotiating Deadline Extension	0.8312	0.7788	**0.0218**	**0.0441**	0.1528	**0.0266**	0.3256	0.5840	0.6496	0.4210	0.4857	0.8489	0.4848	0.2374	0.1067	0.0697	0.4847	0.9562	**0**	1.0000	0.5840	**0.0007**	0.3090	0.8489	0.2374	0.0550	0.7675	0.9104	0.6033	0.4306
Prioritizing TD Items	0.3877	0.3192	0.7782	0.1842	0.4637	**0.0081**	0.2954	0.3464	0.1030	**0.0097**	0.8416	**0.0004**	0.0958	0.3581	0.6285	0.4784	**0.0005**	**0.0281**	1.0000	**0**	0.4594	0.4975	0.5382	**0.0004**	0.3581	0.0889	**0.0004**	0.4780	**0.0408**	0.2870
Restarting the Project from Scratch	0.5153	0.6171	0.7389	0.7389	0.1901	0.6556	0.7389	0.8527	0.3744	0.6183	0.5812	0.5465	0.5465	**0.0003**	0.6561	0.6954	0.4258	0.3613	0.5840	0.4594	**0**	0.6556	0.6547	0.5465	0.7894	0.7389	0.4828	0.6561	0.3986	0.7893
Solving Technical Issues	0.7812	0.9681	**0.0001**	**0.0217**	0.5106	0.0865	0.4233	0.6556	0.8789	0.1295	0.1851	0.4049	0.5145	0.1995	0.1343	0.0600	0.4831	0.9501	**0.0007**	0.4975	0.6556	**0**	0.1025	0.4049	0.1995	**0.0104**	0.7107	0.7322	0.5535	0.5209
Update System Documentation	0.4437	0.2285	0.4221	0.4221	0.5661	0.1025	0.4221	0.6547	0.9269	**0.0117**	0.8495	0.4678	0.4678	0.1767	0.2835	0.6041	0.5226	0.0786	0.3090	0.5382	0.6547	0.1025	**0**	0.4678	0.1767	**0.0160**	0.7868	**0.0053**	**0.0309**	0.5198
Using Automated Deployment	0.4813	0.1554	0.7867	0.2790	**0.0423**	**0.0034**	0.1044	0.0973	**0.0024**	**0.0126**	0.2767	**0.0000**	0.0866	0.4161	0.1481	**0.0235**	**0.0005**	**0.0095**	0.8489	**0.0004**	0.5465	0.4049	0.4678	**0**	0.4161	0.1044	**0.0002**	0.4264	**0.0006**	0.3855
Using Code Analysis	**0.0093**	**0.0040**	**0.0457**	0.6316	0.7972	0.1087	0.6316	0.7894	0.4654	0.2824	0.4275	0.4161	0.4161	**0.0072**	0.5218	0.0912	0.3903	0.4818	0.2374	0.3581	0.7894	0.1995	0.1767	0.4161	**0**	**0.0457**	0.6137	0.1093	**0.0447**	0.7007
Using Code Reuse	0.0836	**0.0362**	0.1627	0.5496	0.9149	**0.0104**	0.5496	0.7389	0.3426	**0.0223**	0.3220	0.1044	0.7867	**0.0457**	0.4241	0.2659	0.0806	0.2746	0.0550	0.0989	0.7389	**0.0104**	**0.0160**	0.1044	**0.0457**	**0**	0.2306	0.3113	0.0948	0.6314
Using Continuous Integrations and Deliveries	0.3971	0.3951	0.9665	0.2077	0.3111	**0.0239**	0.1105	0.1824	**0.0258**	**0.0069**	0.4212	**0.0002**	0.0640	0.6137	0.0924	**0.0350**	**0.0085**	**0.0147**	0.7675	**0.0004**	0.4828	0.7107	0.7868	**0.0002**	0.6137	0.2396	**0**	0.7364	**0.0080**	0.3124
Using External Tools	0.0662	**0.0165**	0.3113	0.4241	0.3608	0.0794	0.4241	0.6561	0.6264	0.2321	0.7913	0.4264	0.0707	0.1093	0.2856	0.4814	0.5039	0.0747	0.9104	0.4780	0.6561	0.7322	**0.0053**	0.4264	0.1093	0.3113	0.7364	**0**	0.3344	0.5216
Using short Feedback Iterations	0.3498	0.2210	0.9072	0.1297	**0.0198**	**0.0088**	0.3511	0.3986	0.0798	**0.0053**	0.1683	**0.0006**	0.1624	0.4273	**0.0430**	**0.0285**	**0.0020**	**0.0024**	0.6033	**0.0408**	0.3986	0.5535	**0.0309**	**0.0006**	**0.0447**	0.0948	**0.0080**	0.3344	**0**	0.2244
Using Software Models	0.3491	0.4718	0.6314	0.6314	0.6070	0.5209	0.6314	0.7893	0.2011	0.4733	0.4272	0.3855	0.3855	0.7007	0.5216	0.5731	0.2517	0.1889	0.4306	0.2870	0.7893	0.5209	0.5198	0.3855	0.7007	0.6314	0.3124	0.5216	0.2244	**0**

* The lines in gray presents the p value's Kendall correlation test <=0.05.

* The lines in gray presents the p value's Kendall correlation test $< = 0.05$.

Table 21
Reasons for non-TD payment and their examples of citation.

Reason for non-TD payment	Quotes from participants
Complexity of the project	"It is a very large project and the interconnectedness of the data layer throughout the project makes completely rewriting it very difficult."
Complexity of the TD item	"It is something that cannot be eliminated, the errors generated must be supported."
Cost	"It is too expensive to fix. 'Too big to succeed'."
Customer decision	"Because of the client's decision."
Decision to do not change the framework	"Because we are still using the framework."
Effort	"Would take a lot of time to migrate to a new back-end database and change the application code."
Focusing on short term goals	"Focus on next release."
High team turnover	"Staff turnover."
Insufficient management view about TD payment	"Resistance from the project manager."
Lack of access on component code	"(…) Also, other teams "own" the code that would need to be updated."
lack of adoption of lessons learned	"Team members who don't 'learn from past mistakes'."
Lack of committed team	"It depends, those who are more professional paid off, while others not."
Lack of external auditing	"The consultancy helped identify critical points to attack and prioritize them. However, this did not prevent changes from being made to the coding that were not contemplated in the initial project estimate."
Lack of knowledge on TD	"(…) The business side of the house is not technical enough to understand the impact that the TD has."
Lack of monitoring of TD items	"Lack of monitoring."
Lack of organizational interest	"We learned enough to move forward but no one wanted to spend the time."
Lack of resources	"We did not have the technical resources, or the time required to carry out the activity."
Lack of technical knowledge	"Lack of commitment and lack of knowledge of the team."
Lack of testing	"(…) Project deadlines prohibit a full search and fix, and there are no tests in the legacy code to ensure the changes result in no behavior change (…)."
Lack of time	"No time dedicated to significant redesign and rework."
Non-application of mitigation actions on TD causes	"The root causes were not mitigated."
Number of TD items	"There is too much. We are attacking as needed."
Risk for the project	"They are not corrected by the high risk involved in generating application errors in the most critical locations."
TD Items do not affect the user	"It's been considered fallout and If an end user runs across the problem then it gets fixed."
TD Items do not have "interest"	"Because it is easier for the project manager to put out fires and not to avoid them."
Team overload	"Precisely because even the system repair teams are overloaded."
The Project was Discontinued	"Because the project has been abandoned completely."

Table 22
Reasons for non-TD payment per type.

Type	Reason for non-TD payment	#CR	%RP
Decision factor	Focusing on short term goals	69	26.4%
	Lack of organizational interest	48	18.4%
	Insufficient management view about TD payment	10	3.8%
	Lack of committed team	4	1.5%
	TD items do not have "interest"	4	1.5%
	Lack of adoption of lessons learned	3	1.1%
	Number of TD items	3	1.1%
	Lack of testing	2	0.8%
	Non-application of mitigation actions on TD causes	2	0.8%
	TD items do not affect the user	2	0.8%
	Decision to do not change the framework	1	0.4%
	Lack of external auditing	1	0.4%
Impediment	Lack of time	41	15.7%
	Cost	34	13.0%
	Lack of resources	19	7.3%
	Customer decision	13	5.0%
	Complexity of the TD item	12	4.6%
	Effort	11	4.2%
	Complexity of the project	10	3.8%
	Team overload	6	2.3%
	The project was discontinued	6	2.3%
	Lack of technical knowledge	5	1.9%
	Risk for the project	5	1.9%
	Lack of knowledge on TD	4	1.5%
	High team turnover	2	0.8%
	Lack of access on component code	1	0.4%
	Lack of monitoring of TD items	1	0.4%

Caption:
#CR - Count of reasons.
%RP - Percentage of CR in relation to the total of all projects (261).

Table 23
Reasons for non-TD payment per nature.

Nature	Reason for non-TD payment	#CR	%RP
Managerial	Focusing on short term goals	69	26.4%
	Lack of organizational interest	48	18.4%
	Lack of time	41	15.7%
	Cost	34	13.0%
	Lack of resources	19	7.3%
	Customer decision	13	5.0%
	Effort	11	4.2%
	Complexity of the project	10	3.8%
	Insufficient management view about TD payment	10	3.8%
	Team overload	6	2.3%
	The project was discontinued	6	2.3%
	Risk for the project	5	1.9%
	Lack of committed team	4	1.5%
	Lack of knowledge on TD	4	1.5%
	TD items do not have "interest"	4	1.5%
	Lack of adoption of lessons learned	3	1.1%
	Number of TD items	3	1.1%
	High team turnover	2	0.8%
	Non-application of mitigation actions on TD causes	2	0.8%
	TD items do not affect the user	2	0.8%
	Lack of external auditing	1	0.4%
Technical	Complexity of the TD item	12	4.6%
	Lack of technical knowledge	5	1.9%
	Lack of testing	2	0.8%
	Decision to do not change the framework	1	0.4%
	Lack of access on component code	1	0.4%
	Lack of monitoring of TD items	1	0.4%

Caption:
#CR - Count of reasons.
%RP - Percentage of CR in relation to the total of all projects (261).

Table 24
Reasons for non-TD payment per category.

Category	Reason for non-TD payment	#CR	%RP
Development issues	Complexity of the project	10	3.8%
	Decision to do not change the framework	1	0.4%
External factors	Customer decision	13	5.0%
	The project was discontinued	6	2.3%
	TD items do not affect the User	2	0.8%
	Lack of access on component code	1	0.4%
Internal quality issues	Complexity of the TD Item	12	4.6%
	Number of TD items	3	1.1%
Lack of knowledge	Lack of technical knowledge	5	1.9%
	Lack of knowledge on TD	4	1.5%
Methodology	Lack of adoption of lessons learned	3	1.1%
	Lack of testing	2	0.8%
	Non-application of mitigation actions on TD causes	2	0.8%
	Lack of external auditing	1	0.4%
	Lack of monitoring of TD items	1	0.4%
Organizational	lack of organizational interest	48	18.4%
	Lack of resources	19	7.3%
	High team turnover	2	0.8%
People	Insufficient management view about TD payment	10	3.8%
	Team overload	6	2.3%
	Lack of committed team	4	1.5%
Planning and management	Focusing on short term goals	69	26.4%
	Lack of time	41	15.7%
	Cost	34	13.0%
	Effort	11	4.2%
	Risk for the project	5	1.9%
	TD items do not have "interest"	4	1.5%

Caption:
#CR - Count of reasons.
%RP - Percentage of CR in relation to the total of all projects (261).

Table 25
Correlation between the reasons for TD non-payment.

Reasons for TD non-payment	Complexity of the Project	Complexity of the TD Item	Cost	Customer Decision	Effort	Focusing on Short Term Goals	High Team Turnover	Insufficient Management View about TD Payment	Lack of Access on Component Code	Lack of Adoption of Lessons Learned	Lack of Committed Team	Lack of External Auditing	Lack of Knowledge on TD	Lack of Organizational Interest	Lack of Resources	Lack of Technical knowledge	Lack of Testing	Lack of Time	Number of TD Items	Risk for the Project	TD Items do not have "interest"	Team overload
Complexity of the Project	**0***	**0.0001**	**0.0268**	**0.0008**	**0.0107**	0.1465	0.6386	0.0816	0.3542	0.6386	0.6386	0.2688	**0.0038**	0.1729	0.0925	0.6386	0.3542	0.3213	0.6386	0.2863	**0.0459**	0.6386
Complexity of the TD Item	**0.0001**	**0**	0.1076	**0.0077**	0.0619	0.1689	0.5896	0.2861	0.6944	0.5896	0.5896	0.3898	**0.0124**	0.4228	0.3186	0.5896	0.6944	0.5902	0.5896	0.6012	0.0843	0.5896
Cost	**0.0268**	0.1076	**0**	0.2581	**0.0139**	0.0759	0.5027	0.3943	0.8162	0.5027	0.5027	**0.0374**	0.4454	0.0911	0.4790	0.5027	0.8162	0.2253	0.5027	0.7280	0.0848	0.5027
Customer Decision	**0.0008**	**0.0077**	0.2581	**0**	**0.0077**	0.0572	0.6392	**0.0077**	0.4769	0.0788	0.6392	0.2345	**0.0009**	0.1915	**0.0109**	0.6392	0.4769	0.2421	0.6392	0.3945	**0.0349**	0.6392
Effort	**0.0107**	0.0619	**0.0139**	**0.0077**	**0**	0.1689	0.5896	0.2861	0.6944	0.5896	0.5896	**0.0038**	**0.0124**	0.4228	0.3186	0.5896	0.6944	0.1359	0.5896	0.6012	0.0843	0.5896
Focusing on Short Term Goals	0.1465	0.1689	0.0759	0.0572	0.1689	**0**	0.2058	0.1445	0.8779	0.1090	0.3114	0.6688	0.8545	0.2728	0.6400	0.3114	0.8779	0.2070	0.2058	0.9593	0.3114	0.6732
High Team Turnover	0.6386	0.5896	0.5027	0.6392	0.5896	0.2058	**0**	0.0843	0.6911	0.8273	0.8273	0.7518	0.7518	0.1285	0.5450	0.8273	0.6911	0.1113	**0.0000**	0.6918	0.8273	0.8273
Insufficient Management View about TD Payment	0.0816	0.2861	0.3943	**0.0077**	0.2861	0.1445	0.0843	**0**	0.4713	0.0843	0.5896	0.3898	0.3898	**0.0451**	**0.0218**	0.5896	0.4713	0.5075	0.0843	0.3957	0.0843	0.5896
Lack of Access on Component Code	0.3542	0.6944	0.8162	0.4769	0.6944	0.8779	0.6911	0.4713	**0**	0.6911	0.6911	0.5647	0.5647	0.8536	0.5816	0.6911	**0.0049**	0.8388	0.6911	**0.0039**	0.6911	0.6911
Lack of Adoption of Lessons Learned	0.6386	0.5896	0.5027	0.0788	0.5896	0.1090	0.8273	0.0843	0.6911	**0**	0.8273	0.7518	0.7518	0.5431	0.1579	0.8273	0.6911	0.8015	0.8273	0.6918	0.8273	0.8273
Lack of Committed Team	0.6386	0.5896	0.5027	0.6392	0.5896	0.3114	0.8273	0.5896	0.6911	0.8273	**0**	0.7518	0.7518	0.5431	0.5450	0.8273	0.6911	0.3145	0.8273	0.6918	0.8273	0.8273
Lack of External Auditing	0.2688	0.3898	**0.0374**	0.2345	**0.0038**	0.6688	0.7518	0.3898	0.5647	0.7518	0.7518	**0**	**0.0392**	0.2707	0.3420	0.7518	0.5647	0.1450	0.7518	0.5657	**0.0016**	0.7518
Lack of Knowledge on TD	**0.0038**	**0.0124**	0.4454	**0.0009**	**0.0124**	0.8545	0.7518	0.3898	0.5647	0.7518	0.7518	**0.0392**	**0**	0.2707	0.3420	0.7518	0.5647	**0.0390**	0.7518	0.5657	**0.0016**	0.7518
Lack of Organizational Interest	0.1729	0.4228	0.0911	0.1915	0.4228	0.2728	0.1285	**0.0451**	0.8536	0.5431	0.5431	0.2707	0.2707	**0**	0.0677	0.5431	0.8536	0.0674	0.1285	0.7590	**0.0333**	0.1285
Lack of Resources	0.0925	0.3186	0.4790	**0.0109**	0.3186	0.6400	0.5450	**0.0218**	0.5816	0.1579	0.5450	0.3420	0.3420	0.0677	**0**	0.5450	0.5816	0.9383	0.5450	0.5017	0.0553	0.1579
Lack of Technical knowledge	0.6386	0.5896	0.5027	0.6392	0.5896	0.3114	0.8273	0.5896	0.6911	0.8273	0.8273	0.7518	0.7518	0.5431	0.5450	**0**	0.6911	0.3145	0.8273	0.6918	0.8273	0.8273
Lack of Testing	0.3542	0.6944	0.8162	0.4769	0.6944	0.8779	0.6911	0.4713	**0.0049**	0.6911	0.6911	0.5647	0.5647	0.8536	0.5816	0.6911	**0**	0.8388	0.6911	**0.0039**	0.6911	0.6911
Lack of Time	0.3213	0.5902	0.2253	0.2421	0.1359	0.2070	0.1113	0.5075	0.8388	0.8015	0.3145	0.1450	**0.0390**	0.0674	0.9383	0.3145	0.8388	**0**	0.1113	0.6848	0.0784	0.2759
Number of TD Items	0.6386	0.5896	0.5027	0.6392	0.5896	0.2058	**0.0000**	0.0843	0.6911	0.8273	0.8273	0.7518	0.7518	0.1285	0.5450	0.8273	0.6911	0.1113	**0**	0.6918	0.8273	0.8273
Risk for the Project	0.2863	0.6012	0.7280	0.3945	0.6012	0.9593	0.6918	0.3957	**0.0039**	0.6918	0.6918	0.5657	0.5657	0.7590	0.5017	0.6918	**0.0039**	0.6848	0.6918	**0**	0.6918	0.6918
TD Items do not have "interest"	**0.0459**	0.0843	0.0848	**0.0349**	0.0843	0.3114	0.8273	0.0843	0.6911	0.8273	0.8273	**0.0016**	**0.0016**	**0.0333**	0.0553	0.8273	0.6911	0.0784	0.8273	0.6918	**0**	0.8273
Team overload	0.6386	0.5896	0.5027	0.6392	0.5896	0.6732	0.8273	0.5896	0.6911	0.8273	0.8273	0.7518	0.7518	0.1285	0.1579	0.8273	0.6911	0.2759	0.8273	0.6918	0.8273	**0**

* The lines in gray presents the p value's Kendall correlation test <=0.05.

* The lines in gray presents the p value's Kendall correlation test $< = 0.05$.

Table 26
Comparison to related work on TD payment-related practices.

	Our study	Technical debt payment-related practices from						
		Yli-Huumo et al. (2014)	Abad et al. (2016)	Samarthyam et al. (2017)	Bomfim and Santos (2017)	Toledo et al. (2019)	Apa et al. (2020)	Rios et al. (2020a)
Defining a favorable setting for TD payment	Changing project scope	–	–	–	–	–	–	–
	Changing the project management	–	–	–	–	–	–	–
	Communicating the customer of TD items	–	–	–	–	–	–	–
	Conducting risk analysis	–	–	–	–	Define and execute a governance plan to handle the migration	–	–
	Hiring specialized professionals	–	–	–	–	–	–	–
	Improving the development process	–	–	–	–	–	–	–
	Improving the team collaboration	Communi-cation structure between business	–	–	–	–	–	–
	Increasing the project budget	–	Allocating more budget	–	–	–	–	–
	Investing effort on TD payment activities	–	Allocating more infrastruc-ture	–	–	–	–	–
	Negotiating deadline extension	–	Allocating more time Having more flexible schedules	–	–	–	–	–
	Using continuous integrations and deliveries	–	–	–	–	–	–	–

(*continued on next page*)

Table 26 (*continued*).

	Our study	Technical debt payment-related practices from						
		Yli-Huumo et al. (2014)	Abad et al. (2016)	Samarthyam et al. (2017)	Bomfim and Santos (2017)	Toledo et al. (2019)	Apa et al. (2020)	Rios et al. (2020a)
Payment action	Adjusting code to follow good programming practices	Coding standards and guides	Using design patterns	Pair programming Clean coding	Using design patterns	Remove the business logic inside the communication layer Move the business logic to the services Define a canonical model per domain Centralize the source code and documentation for all services in a common management system Provide a common middleware that can be used by all services Maintain the system working with different solutions during the migration period	–	–
	Bug fixing	Bug fixing days	–	–	–	–	–	–
	Code refactoring	Refactoring	Refactoring	Refactoring	Splitting methods to make them more reusable Refactoring older code	–	Refactoring Rewrite of code	–
	Design refactoring	–	Improving design and architecture	–	–	Rewrite the communication layer Migrate the services to use the new architecture Rewrite services to use the same middleware Update the services to use the newly defined canonical models	Redesign	–
	Restarting the project from scratch	–	–	–	–	–	–	–
	Solving technical issues	–	–	–	–	–	–	–
	System retirement	–	–	–	–	–	–	–
	Update system documentation	–	–	–	–	–	–	Keep the documentation updated Review outdated documentation
TD prevention	Code reviewing	Code reviews	–	–	–	–	–	–
	Implementing preventive actions for avoiding TD items	–	–	–	–	–	–	–
	Improving requirement elicitation process	–	–	–	–	New requirements should be down prioritized	–	–
	Investing effort on testing activities	–	–	–	–	–	–	–

(*continued on next page*)

Table 26 (*continued*).

	Our study	Technical debt payment-related practices from						
		Yli-Huumo et al. (2014)	Abad et al. (2016)	Samarthyam et al. (2017)	Bomfim and Santos (2017)	Toledo et al. (2019)	Apa et al. (2020)	Rios et al. (2020a)
	Monitoring and controlling project activities	–	–	–	–	–	–	–
	Organizing the project repository	–	–	–	–	–	–	–
	Using automated deployment	–	–	–	–	–	–	–
	Using code analysis	–	–	–	–	–	–	–
	Using code reuse	–	–	–	–	–	–	–
	Using external tools	–	–	–	–	–	–	–
	Using short feedback iterations	–	–	–	–	–	–	–
	Using software models	–	–	–	–	–	–	–
TD prioritization	Prioritizing TD items	–	Prioritizing TD	–	–	–	–	Adopt TD payment prioritization criteria
–	–	–	Quantifying TD	–	–	–	–	–
–	–	–	–	–	Using palliative solutions	–	–	–
–	–	–	–	–	–	Service developers must learn new technologies	–	–

References

Abad, Zahra S.H., Karimpour, Reza, Ho, Jason, Didar-Al-Alam, S.M, Ruhe, Guenther, Tse, Edward, Barabash, Kevin, Hargreaves, Ian, 2016. Understanding the impact of technical debt in coding and testing: an exploratory case study. In: Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice (SER & IP' 16). Association for Computing Machinery, New York, NY, USA, pp. 25–31. http://dx.doi.org/10.1145/2897022.2897023.

Alves, Nicolli S.R., Mendes, Thiago S., de Mendonça, Manoel G., Spínola, Rodrigo O., Shull, Forrest, Seaman, Carolyn, 2016. Identification and management of technical debt: a systematic mapping study. Inf. Softw. Technol. 70 (February), 100–121. http://dx.doi.org/10.1016/j.infsof.2015.10.008.

Andrade, Chittaranjan, 2018. Internal, external, and ecological validity in research design, conduct, and evaluation. Indian J. Psychol. Med. 40 (5), 498–499. http://dx.doi.org/10.4103/IJPSYM.IJPSYM_334_18.

Apa, Cecilia, Jeronimo, Helvio, Nascimento, Luciana M., Vallespir, Diego, Travassos, Guilherme Horta, 2020. The perception and management of technical debt in software startups. In: Nguyen-Duc, A, Münch, J, Prikladnicki, R, Wang, X, Abrahamsson, P. (Eds.), Fundamentals of Software Startups. Springer, Cham, http://dx.doi.org/10.1007/978-3-030-35983-6_4.

Avgeriou, Paris, Kruchten, Philippe, Ozkaya, Ipek, Seaman, Carolyn, 2016. Managing technical debt in software engineering. Dagstuhl seminar 16162. In: Dagstuhl Reports. Vol. 6, N. 4. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Barbosa, Larissa, Freire, Sávio, Rios, Nicolli, Ramač, Robert, Taušan, Nebojša, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Pacheco, Alexia, López, Gustavo, Mandić, Vladimir, Maciel, Rita S.P., Mendonça, Manoel, Falessi, Davide, Izurieta, Clemente, Seaman, Carolyn, Spínola, Rodrigo, 2022. Organizing the TD management landscape for requirements and requirements documentation debt. In: Proceedings of the 25th Workshop on Requirements Engineering (WER 22).

Behutiye, Woubshet N., Rodríguez, Pilar, Oivo, Markku, Tosun, Ayşe, 2017. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Inf. Softw. Technol. 82, 139–158. http://dx.doi.org/10.1016/j.infsof.2016.10.004.

Berenguer, Clara, Borges, Adriano, Freire, Sávio, Rios, Nicolli, Ramac, Robert, Tausan, Nebojsa, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Pacheco, Alexia, López, Gustavo, Mendonça, Manoel, Falessi, Davide, Seaman, Carolyn, Mandic, Vladimir, Izurieta, Clemente, Spínola, Rodrigo, 2022. Investigating the relationship between technical debt management and software development issues. J. Softw. Eng. Res. Dev..

Berenguer, Clara, Borges, Adriano, Freire, Sávio, Rios, Nicolli, Tausan, Nebojsa, Ramac, Robert, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Pacheco, Alexia, López, Gustavo, Falessi, Davide, Seaman, Carolyn, Mandic, Vladimir, Izurieta, Clemente, Spínola, Rodrigo, 2021. Technical debt is not only about code and we need to be aware about it. In: XX Brazilian Symposium on Software Quality. SBQS, ACM, New York, NY, USA, 14. http://dx.doi.org/10.1145/3493244.3493285, 1–12.

Bomfim, Jr., Marcelo M., Santos, Viviane A., 2017. Strategies for reducing technical debt in agile teams. In: Silva da Silva, T., Estácio, B., Kroll, J., Mantovani Fontana, R. (Eds.), Agile Methods. WBMA 2016. In: Communications in Computer and Information Science, vol. 680, Springer, Cham, http://dx.doi.org/10.1007/978-3-319-55907-0_6.

Cartaxo, Bruno, Pinto, Gustavo, Vieira, Elton, Soares, Sérgio, 2016. Evidence briefings: towards a medium to transfer knowledge from systematic reviews to practitioners. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM' 16). Association for Computing Machinery, New York, NY, USA, 57. http://dx.doi.org/10.1145/2961111.2962603, 1–10.

Freire, Sávio, Rios, Nicolli, Gutierrez, Boris, Torres, Darío, Mendonça, Manoel, Izurieta, Clemente, Seaman, Carolyn, Spínola, Rodrigo O., 2020a. Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items. In: Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20). ACM, New York, NY, USA, pp. 210–219. http://dx.doi.org/10.1145/3383219.3383241.

Freire, Sávio, Rios, Nicolli, Mendonça, Manoel, Falessi, Davide, Seaman, Carolyn, Izurieta, Clemente, Spínola, Rodrigo O., 2020b. Actions and impediments for technical debt prevention: results from a global family of industrial surveys. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC '20). ACM, New York, NY, USA, pp. 1548–1555. http://dx.doi.org/10.1145/3341105.3373912.

Freire, Sávio, Rios, Nicolli, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Ramač, Robert, Mandić, Vladimir, Taušan, Nebojša, López, Gustavo,

Pacheco, Alexia, Falessi, Davide, Mendonça, Manoel, Izurieta, Clemente, Seaman, Carolyn, Spínola, Rodrigo, 2021a. How experience impacts practitioners' perception of causes and effects of technical debt. In: Proceedings of the IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering. CHASE, IEEE, Madrid, Spain, pp. 21–30. http://dx.doi.org/10.1109/CHASE52884.2021.00011.

Freire, Sávio, Rios, Nicolli, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Ramac, Robert, Tausan, Nebojša, Mandic, Vladimir, Hernandez, Alexia P., López, Gustavo, Mendonça, Manoel, Izurieta, Clemente, Falessi, Davide, Seaman, Carolyn B., Spinola, Rodrigo O., 2021c. Pitfalls and solutions for technical debt management in agile software projects. IEEE Softw. 38 (6), 42–49. http://dx.doi.org/10.1109/MS.2021.3101990, Nov.-Dec. 2021.

Freire, Sávio, Rios, Nicolli, Pérez, Boris, Correal, Darío, Mendonça, Manoel, Izurieta, Clemente, Seaman, Carolyn, Spínola, Rodrigo, 2021b. How do technical debt payment practices relate to the effects of the presence of debt items in software projects? In: Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2021). IEEE, Honolulu, USA, pp. 605–609. http://dx.doi.org/10.1109/SANER50967.2021.00074.

Guo, Yuepu, Seaman, Carolyn, 2011. A portfolio approach to technical debt management. In: Proceedings of the 2nd Workshop on Managing Technical Debt (MTD' 11). Association for Computing Machinery, New York, NY, USA, pp. 31–34. http://dx.doi.org/10.1145/1985362.1985370.

Guo, Yuepu, Spínola, Rodrigo O., Seaman, Carolyn, 2014. Exploring the costs of technical debt management–a case study. Empir. Softw. Eng. J. 1, 1–24. http://dx.doi.org/10.1007/s10664-014-9351-7.

Hopkins, Will G., 2003. A new view of statistics. Sport Sci. http://www.sportsci.org/resource/stats/.

Izurieta, Clemente, Ozkaya, Ipek, Seaman, Carolyn, Kruchten, Philippe, Nord, Robert, Snipes, Will, Avgeriou, Paris., 2016. Perspectives on managing technical debt. A transition point and roadmap from dagstuhl. In: The 1st International Workshop on Technical Debt Analytics (TDA). In Association with the 23rd Asia-Pacific Software Engineering Conference. APSEC, University of Waikato, Hamilton, New Zealand, December 6-9 2016.

Izurieta, Clemente, Vetrò, Antonio, Zazworka, Nico, Cai, Yuanfang, Seaman, Carolyn, Shull, Forrest, 2012. Organizing the technical debt landscape. In: Proceedings of Third International Workshop on Managing Technical Debt (MTD), Zurich. pp. 23–26. http://dx.doi.org/10.1109/MTD.2012.6225995.

Kruchten, Philippe, Nord, Robert L., Ozkaya, Ipek, 2012. Technical debt: from metaphor to theory and practice. IEEE Softw. Published by the IEEE Computer Society.

Li, Zengyang, Avgeriou, Paris, Liang, Peng, 2015. A systematic mapping study on technical debt and its management. J. Syst. Softw. 101, 193–220. http://dx.doi.org/10.1016/j.jss.2014.12.027.

Mandić, Vladimir, Taušan, Nebojša, Ramač, Robert, 2020. The prevalence of the technical debt concept in Serbian IT industry: results of a national-wide survey. In: Proceedings of the 3rd International Conference on Technical Debt (TechDebt). ACM, New York, NY, USA, pp. 77–86. http://dx.doi.org/10.1145/3387906.3388622.

Mandić, Vladimir, Taušan, Nebojša, Ramač, Robert, Freire, Sávio, Rios, Nicolli, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Pacheco, Alexia, Lopez, Gustavo, Izurieta, Clemente, Falessi, Davide, Seaman, Carolyn, Spínola, Rodrigo, 2021. Technical and nontechnical prioritization schema for technical debt: Voice of TD-experienced practitioners. IEEE Softw. 38 (6), 50–58. http://dx.doi.org/10.1109/MS.2021.3103121, Nov.-Dec. 2021.

Martini, Antonio, Bosch, Jan, Chaudron, Michel, 2014. Architecture technical debt: understanding causes and a qualitative model. In: Proceeding of 40th Euromicro Conf. on Software Engineering and Advanced Applications. SEAA, IEEE Computer Society, Washington, DC, USA, pp. 85–92. http://dx.doi.org/10.1109/SEAA.2014.65.

McConnell, Steve, 2007. Technical debt, 10x software development blog. In: Construx Conversations. URL= http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx.

Nord, Robert L., Ozkaya, Ipek, Kruchten, Philippe, Gonzalez-Rojas, Marco, 2012. In search of a metric for managing architectural technical debt. In: Proceeding of Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA), Helsinki. pp. 91–100. http://dx.doi.org/10.1109/WICSA-ECSA.212.17.

Pacheco, Alexia, Marín-Raventós, Gabriela, López, Gustavo, 2019. Technical debt in costa rica: an insightd survey replication. In: Franch, X, Männistö, T, Martínez-Fernández, S. (Eds.), Product-Focused Software Process Improvement. PROFES 2019. In: Lecture Notes in Computer Science, vol. 11915, Springer, Cham, http://dx.doi.org/10.1007/978-3-030-35333-9_17.

Pérez, Boris, Brito, Juan Pablo, Astudillo, Hernán, Correal, Darío, Rios, Nicolli, Spínola, Rodrigo O., Manoel Mendonça, Seaman, Carolyn, 2019. Familiarity, causes and reactions of software practitioners to the presence of technical debt: A replicated study in the chilean software industry. In: Proceedings of 38th SCCC. pp. 1–7. http://dx.doi.org/10.1109/SCCC49216.2019.8966424.

Pérez, Boris, Castellanos, Camilo, Correal, Darío, Rios, Nicolli, Freire, Sávio, Spínola, Rodrigo, Seaman, Carolyn, 2020. What are the practices used by software practitioners on technical debt payment? Results from an international family of surveys. In: Proceedings of the 3rd International Conference on Technical Debt (TechDebt' 20). Association for Computing Machinery, New York, NY, USA, pp. 103–112. http://dx.doi.org/10.1145/3387906.3388632.

Pérez, Boris, Castellanos, Camilo, Correal, Darío, Rios, Nicolli, Freire, Sávio, Spínola, Rodrigo, Seaman, Carolyn, Izurieta, Clemente, 2021. Technical debt payment and prevention through the lenses of software architects. Inf. Softw. Technol. 140, 106692. http://dx.doi.org/10.1016/j.infsof.2021.106692.

Ramač, Robert, Mandić, Vladimir, Taušan, Nebojša, Rios, Nicolli, Freire, Sávio, Pérez, Boris, Castellanos, Camilo, Correal, Darío, Pacheco, Alexia, Lopez, Gustavo, Izurieta, Clemente, Seaman, Carolyn, Spinola, Rodrigo, 2022a. Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. J. Syst. Softw. 184. http://dx.doi.org/10.1016/j.jss.2021.111114.

Ramač, Robert, Mandić, Vladimir, Taušan, Nebojša, Rios, Nicolli, Mendonça, Manoel, Seaman, Carolyn, Spínola, Rodrigo O., 2020. Common causes and effects of technical debt in Serbian IT: Insightd survey replication. In: Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Portoroz, Slovenia, Vol. 2020. pp. 354–361. http://dx.doi.org/10.1109/SEAA51224.2020.00065.

Ramač, Robert, Taušan, Nebojša, Freire, Sávio, Rios, Nicolli, de Mendonça Neto, Manoel G., Spínola, Rodrigo Oliveira, Mandić, Vladimir, 2022b. Technical debt payment practices and rationales behind payment avoidance in the Serbian IT industry. In: Lalic, B., Gracanin, D., Tasic, N., Simeunović, N. (Eds.), Proceedings on 18th International Conference on Industrial Systems – IS'20. In: Lecture Notes on Multidisciplinary Industrial Engineering, Springer, Cham, http://dx.doi.org/10.1007/978-3-030-97947-8_14.

Rios, Nicolli, Freire, Sávio, Pérez, Boris, Castellanos, Camilo, Correal, Dario, Mendonça, Manoel, Falessi, Davide, Izurieta, Clemente, Seaman, Carolyn, Spínola, Rodrigo Oliveria, 2021. On the relationship between technical debt management and process models. IEEE Softw. Published by the IEEE Computer Society.

Rios, Nicolli, Mendes, Leonardo, Cerdeiral, Cristina, Magalhães, Ana P.F., Perez, Boris, Correal, Darío, Astudillo, Hernán, Seaman, Carolyn, Izurieta, Clemente, Santos, Gleison, Spínola, Rodrigo O., 2020a. Hearing the voice of software practitioners on causes, effects, and practices to deal with documentation debt. In: Madhavji, N., Pasquale, L., Ferrari, A., Gnesi, S. (Eds.), Requirements Engineering: Foundation for Software Quality. REFSQ 2020. In: Lecture Notes in Computer Science, vol. 12045, Springer, Cham, http://dx.doi.org/10.1007/978-3-030-44429-7_4.

Rios, Nicolli, Mendonça, Manoel, Seaman, Carolyn, Spínola, Rodrigo O., 2019b. Causes and effects of the presence of technical debt in agile software projects. In: Proceedings of the Americas Conference on Information Systems (AMCIS). Cancún, Mexico.

Rios, Nicolli, Neto, Manoel Mendonça, Spínola, Rodrigo O., 2018b. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Inf. Softw. Technol. (ISSN: 0950-5849) 102 (June), 117–145. http://dx.doi.org/10.1016/j.infsof.2018.05.010.

Rios, Nicolli, Spínola, Rodrigo Oliveira, Mendonça, Manoel, Seaman, Carolyn, 2018a. The most common causes and effects of technical debt: first results from a global family of industrial surveys. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM' 18). Association for Computing Machinery, New York, NY, USA, 39. http://dx.doi.org/10.1145/3239235.3268917, 1–10.

Rios, Nicolli, Spínola, Rodrigo O., Mendonça, Manoel, Seaman, Carolyn, 2019a. Supporting analysis of technical debt causes and effects with cross-company probabilistic cause–effect diagrams. In: Proceedings of the 2nd International Conference on Technical Debt (TechDebt). IEEE Press, Piscataway, NJ, USA, pp. 3–12. http://dx.doi.org/10.1109/TechDebt.2019.00009.

Rios, Nicolli, Spínola, Rodrigo O., Mendonça, Manoel, Seaman, Carolyn, 2020b. 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. Empir. Softw. Eng. 25, 3216–3287. http://dx.doi.org/10.1007/s10664-020-09832-9.

Rocha, Verusca, Freire, Sávio, Mendonça, Manoel, Spínola, Rodrigo, 2022. Evaluating a conceptual framework for supporting technical debt management in testing activities – A feasibility study. In: The Proceedings of the 7th Brazilian Symposium on Systematic and Automated Software Testing. http://dx.doi.org/10.1145/3559744.3559753.

Rocha, Verusca, Freire, Sávio, Rios, Nicolli, Lima, Cleydiane, Ribeiro, Leilane, Perez, Boris, Neto, Arilo.Dias, Moura, Hermano, Correal, Dario, Mendonça, Manoel, Spinola, Rodrigo, 2021. A conceptual framework to support the management of technical debt in software testing. In: Proceedings of the Americas Conference on Information Systems. AMCIS.

Samarthyam, Ganesh, Muralidharan, Mahesh, Anna, Raghu K., 2017. Understanding test debt. In: Mohanty, H., Mohanty, J., Balakrishnan, A. (Eds.), Trends in Software Testing. Springer, Singapore, http://dx.doi.org/10.1007/978-981-10-1415-4_1.

Seaman, Carolyn, 1999. Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. 25 (4), 557–572.

Seaman, Carolyn, Guo, Yuepu, 2011. Measuring and Monitoring Technical Debt. 1, Vol. 82. Elsevier Inc., ed. [s.l.].

Seaman, Carolyn, Guo, Yuepu, Zazworka, Nico, Shull, Forrest, Izurieta, Clemente, Cai, Yuanfang, Vetrò, Antonio, 2012. Using technical debt data in decision making: potential decision approaches. In: 3rd International Workshop on Managing Technical Debt (MTD' 12). IEEE Computer Society, Zurich, Switzerland, pp. 45–48. http://dx.doi.org/10.1109/MTD.2012.6225999.

Souza, Lucinéia, Freire, Sávio, Rocha, Verusca, Rios, Nicolli, Spínola, Rodrigo O., Mendonça, Manoel, 2020. Using surveys to build-up empirical evidence on test-related technical debt. In: Proceedings of the 34th Brazilian Symposium on Software Engineering (SBES' 20). ACM, New York, NY, USA, pp. 750–759. http://dx.doi.org/10.1145/3422392.3422430.

Spínola, Rodrigo O., Vetrò, Antonio, Zazworka, Nico, Seaman, Carolyn, Shull, Forrest, 2013. Investigating technical debt folklore: shedding some light on technical debt opinion. In: Proceeding of the 4th International Workshop on Managing Technical Debt (MTD), San Francisco, CA. pp. 1–7. http://dx.doi.org/10.1109/MTD.2013.6608671.

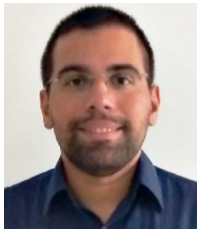
Strauss, Anselm, Corbin, Juliet M., 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications.

Toledo, Saulo S. de, Martini, Antonio, Przybyszewska, Agata, Sjøberg, Dag I.K., 2019. Architectural technical debt in microservices: a case study in a large company. In: Proceedings of the Second International Conference on Technical Debt (TechDebt '19). IEEE Press, pp. 78–87. http://dx.doi.org/10.1109/TechDebt.2019.00026.

Webber, William, Moffat, Alistair, Zobel, Justin, 2010. A similarity measure for indefinite rankings. ACM Trans. Inf. Syst. 28 (4), 1–38. http://dx.doi.org/10.1145/1852102.1852106.

Wohlin, Claes, Runeson, Per, Höst, Martin, Ohlsson, Magnus C., Regnell, Björn, Wesslén, Anders, 2012. Experimentation in Software Engineering: An Introduction. Springer.

Yli-Huumo, Jesse, Maglyas, Andrey, Smolander, Kari, 2014. The sources and approaches to management of technical debt: a case study of two product lines in a middle-size finnish software company. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (Eds.), Product-Focused Software Process Improvement. PROFES 2014. In: Lecture Notes in Computer Science, vol. 8892, Springer, Cham, http://dx.doi.org/10.1007/978-3-319-13835-0_7.



SÁVIO FREIRE is a Ph.D. candidate at the Department of Computer Science at the Federal University of Bahia, Salvador, 40170-110, Bahia, Brazil, and an assistant professor at the Federal Institute of Ceará, Morada Nova, Ceará, 62.940-000, Brazil. He is a researcher at the Technical Debt Research Team (www.tdresearchteam.com) at the Salvador University. His research interest encompasses technical debt and empirical software engineering. Contact him at savio.freire@ifce.edu.br.

NICOLLI RIOS is a postdoctoral researcher at PESC (Systems and Computer Engineering Program)/COPPE (The Alberto Luiz Coimbra Institute for Graduate Studies and Research in Engineering) at Federal University of Rio de Janeiro, Rio de Janeiro, 21941-972, Brazil. She is also a researcher at the Technical Debt Research Team (www.tdresearchteam.com). She holds a Ph.D. in Computer Science from the Federal University of Bahia. Her main research interests are technical debt and empirical software engineering. Contact her at nicolli@cos.ufrj.br.

BORIS PÉREZ is an assistant professor in Francisco de Paula Stder. University's Department of Systems and Informatics in Cúcuta, 12E-96, Colombia. Also, he is Ph.D. candidate at the Department of Systems and Computing Engineering, Universidad de Los Andes, Bogota, 111711, Colombia. His research interests include software architecture, model-driven architecture, and technical debt. Boris received his M.Sc. in software engineering from Universidad de Los Andes. Contact him at borisperezg@ufps.edu.co.

CAMILO CASTELLANOS is a Ph.D. candidate in the Department of Systems and Computing Engineering, Universidad de Los Andes, Bogota, 111711, Colombia. His research fields include software architecture, big data analytics, and model-driven engineering. Contact him at cc.castellanos87@uniandes.edu.co.

DARÍO CORREAL, Ph.D., is associate professor at Los Andes University Bogota, 111711, Colombia. His main research topics are software architecture, solutions architectures, service-oriented software engineering and blockchain solutions. More can be found at https://profesores.virtual.uniandes.edu.co/dcorreal/es/inicio/. Contact him at dcorreal@uniandes.edu.co.

ROBERT RAMAČ is a teaching assistant of Software Engineering at the University of Novi Sad, Faculty of Technical Sciences, 21000 Novi Sad, Serbia, and a software developer at TIAC ltd. Currently he is enrolled as a Ph.D. student at the University of Novi Sad, at the Faculty of Technical Sciences. He received two M.Sc. degrees at the University of Novi Sad, master of Information technologies and master of Engineering management. His areas of interest are: empirical software engineering, software development, technical debt, the improvement of the software development process. Contact him at ramac.robert@uns.ac.rs.

VLADIMIR MANDIĆ is an associate professor of software engineering at University of Novi Sad, Faculty of Technical Sciences, 21000 Novi Sad, Serbia. His research interests include empirical software engineering, software process improvement, software quality, and technical debt. He holds a Ph.D. degree in Information Processing Science and Software Engineering from University of Oulu, Finland, M.Sc. and B.Sc. degrees in Computer Science from University of Novi Sad, Serbia. Contact him at vladman@uns.ac.rs.

NEBOJŠA TAUŠAN (Ph.D.) is an assistant professor at the University of Novi Sad, Faculty of Economics Subotica. He received both Magisterial and Bachelor degrees from the University of Novi-Sad, Serbia and the Doctoral degree from the University of Oulu, Finland. Nebojša's research interests include: technical debt, business informatics and empirical research methods. For more information, contact him at nebojsa.tausan@ef.uns.ac.rs.

GUSTAVO LOPEZ is a professor of software engineering and human–computer interaction at the University of Costa Rica (San Pedro, Montes de Oca, San José, 11501, Costa Rica). He holds a Ph.D. in Computer Science from the same university. His research interests include agile software development, software process improvement, technical debt, user experience, and usability. Contact him at gustavo.lopez_h@ucr.ac.cr.

ALEXIA PACHECO (alexia.pacheco@ucr.ac.cr) is a Ph.D. candidate at the graduate program in Computer Science at the University of Costa Rica (San Pedro, Montes de Oca, San José, 11501, Costa Rica). Her research interests include technical debt, software analytics, and empirical software engineering. She holds an MS in Applied Mathematics and a Degree in Computer Science from the University of Costa Rica.

MANOEL MENDONÇA (manoel.mendonca@ufba.br) is a Professor of computer science at the Federal University of Bahia (UFBA). He acted as the Founding Director of the Fraunhofer Center for Software and Systems Engineering at UFBA.

DAVIDE FALESSI is an assistant professor (RTDb) at the University of Rome "Tor Vergata", Rome, 00133, Italy. He got his bachelor, master, and Ph.D. degrees from that same university. His research interests include technical debt and machine learning to support software engineering tasks. You can contact him at falessi@ing.uniroma2.it.

CLEMENTE IZURIETA is an associate professor of computer science in the Gianforte School of Computing at Montana State University, Bozeman, Montana, 59717, USA. His research focuses on quality assurance, technical debt, and cybersecurity. He is active in many collaborative projects where computing techniques enhance outcomes. He is also the chief technology officer of Authors A.I. (authors.ai). He is a Senior Member of IEEE. Contact him at clemente.izurieta@montana.edu.

CAROLYN SEAMAN is a Professor of Information Systems at the University of Maryland Baltimore County (UMBC), Baltimore, Maryland, 21250, USA. She is also the Director of the Center for Women in Technology, also at UMBC. Her research consists mainly of empirical studies of software engineering, with particular emphases on maintenance, organizational structure, communication, measurement, and technical debt. She also investigates qualitative research methods in software engineering, as well as computing pedagogy. She holds a Ph.D. in Computer Science from the University of Maryland, College Park, a MS from Georgia Tech, and a BA from the College of Wooster (Ohio). More can be found at https://userpages.umbc.edu/~cseaman/. Contact her at cseaman@umbc.edu.

RODRIGO SPÍNOLA is an Associate Professor at Virginia Commonwealth University, Richmond, Virginia, 23284, United States, where he leads the Technical Debt Research Team (www.tdresearchteam.com). His research interests include technical debt and empirical software engineering. He holds a Ph.D. and MS in Computer Science and Systems Engineering from the Federal University of Rio de Janeiro. More can be found at www.rodrigospinola.com. Contact him at spinolaro@vcu.edu.