Empirical Software Engineering
https://doi.org/10.1007/s10664-020-09832-9

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

Nicolli Rios[1] · Rodrigo Oliveira Spínola[2] · Manoel Mendonça[3] · Carolyn Seaman[4]

Published online: 13 June 2020

Abstract

Context Studying the causes of technical debt (TD) could aid in TD prevention, thus easing the job of TD management. On the other hand, better understanding of the effects of TD could also aid in TD management by facilitating more informed decisions about incurring and paying off debt.
Objective Create a deeper understanding, and confirming existing evidence, of the causes and effects of TD by collecting new evidence from real-world TD examples.
Method InsighTD is a globally distributed family of industrial surveys on the causes and effects of TD. It is designed to run as a large-scale study based on continuous and independent replications in different countries. The survey instrument asks practitioners to describe in detail a real example of TD from their experience. We present in this paper the design of InsighTD, which has the primary goal of replication at a large-scale, with the results of the study in Brazil as a small part of the larger puzzle.
Results The first iteration of the InsighTD survey, carried out in Brazil, yielded 107 responses. We identified a total of 78 causes and 66 effects, which confirm and also extend the current knowledge on causes and effects of TD. Then, we organized the identified set of causes and effects in probabilistic cause-effect diagrams. The proposed diagrams highlight the causes that can most contribute to the occurrence of TD as well as the most common effects that occur as a result of debt.
Conclusion We intend to reduce the problem of isolated TD investigations that are not yet representative and build a continuous and generalizable empirical basis for understanding practical problems and challenges of TD.

Keywords Technical debt · Technical debt causes · Technical debt effects · Survey · Family of surveys · InsighTD

✉ Nicolli Rios
nicollirioss@gmail.com

Extended author information available on the last page of the article

1 Introduction

Technical debt (TD) contextualizes the problem of pending development tasks as a type of debt that brings a short-term benefit to the project, usually in terms of increased development speed or shortened time to market, but which may have to be paid with interest later in the development process (Guo and Seaman 2011; Kruchten et al. 2012; Alves et al. 2016). It is common for a software project to incur debt during its development. However, its presence brings risks to the project and makes it difficult to manage (Guo and Seaman 2011; Guo et al. 2014).

According to Kruchten et al. (2012), TD can be a good investment as long as the project team knows about its presence and the increased risks it imposes on the project. If properly managed, it can help the project achieve its goals sooner or more cheaply. TD management activities include identification, monitoring and payment of the debt items incurred in a system (Griffith et al. 2014). Its main goal is to enable decision-making about the need to eliminate a debt item and the most appropriate time to do this (Guo et al. 2014). Thus, the management of TD focuses on reducing its negative impact, which can be a decisive factor for the success of software projects (Seaman and Guo 2011). 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 crisis point where the whole future of the software project is jeopardized (Nord et al. 2012; Martini et al. 2014).

Several secondary studies have identified proposed TD management (TDM) strategies in the literature (Rios et al. 2018a). All of these strategies assume a list of existing debt items, which are identified and documented in various ways. However, studying the underlying causes of TD, thus helping to identify actions that could prevent the TD items in the first place, is not yet common in the literature. This is a point that deserves investigation because it is expected that TD prevention could sometimes be "*cheaper*" than TD repayment. Further, increased TD prevention also facilitates other TDM activities, and setting up TD prevention practices helps especially in catching inexperienced developers' 'not-so-good' solutions (Yli-Huumo et al. 2016). Knowing the causes for TD can support development teams in defining TD prevention actions. The literature also lacks careful examinations of the effects of TD, which would then inform prioritization strategies for paying off debt (Alves et al. 2016; Yli-Huumo et al. 2016; Rios et al. 2018a). Like TD causes, TD effects are also important to investigate, as the implications of TD can affect projects in different ways. Having this information could aid in prioritization of TD items to pay off, by supporting a more precise impact analysis and the identification of corrective actions to minimize possible negative consequences for the project. Thus, while TDM is an important topic (Guo et al. 2014), it is also worthwhile to understand the motivations that could lead a development team to incur different types of TD and the implications of the presence of TD items in software projects, in other words TD causes and effects.

Recently, some publications have addressed causes and effects (Martini et al. 2014; Yli-Huumo et al. 2014; Ernst et al. 2015; Yli-Huumo et al. 2015; Martini and Bosch 2017; Rios et al. 2018b; Besker et al. 2018a), however, the existing evidence is still limited. Three of these (Martini et al. 2014; Martini and Bosch 2017; Besker et al. 2018a) are focused only on architectural TD, in detriment to the other types of debt. Ernst et al. (2015) evaluated a predefined list of 13 causes of TD, which is limiting because this list does not necessarily represent all sources of pain for the participants. Many of these studies (Martini et al. 2014; Yli-Huumo et al. 2014; Yli-Huumo et al. 2015; Martini and Bosch 2017; Rios et al. 2018b;

Besker et al. 2018a) were very limited in scope, with a maximum of 17 participants from a maximum of 5 different organizations in each study. Ernst et al. (2015) had a large number of participants, but they came from only three software organizations. The discussion around causes and effects of TD deserves a more comprehensive investigation so we can understand in a more generalizable way the reasons that lead software teams to incur debt and the pain that developers suffer because of its presence in their projects. Understanding TD from the practitioners' perspective is critical to guide research directions.

This paper presents the design of *InsighTD Survey – Investigating causes and implications of TD,* as well as the results of its first execution in Brazil. InsighTD is a globally distributed family of industrial surveys on TD. It is designed to run as an incremental large-scale study based on continuous and independent replications of the questionnaire in different countries. Its goal is to investigate the state of practice and industry trends in the TD area including the causes that lead to TD occurrence, the effects of its existence, how these problems manifest themselves in the software development process, and how software development teams react when they are aware of the presence of debt items in their projects. The InsighTD instrument is grounded in existing TD studies (Martini et al. 2014; Yli-Huumo et al. 2014; Ernst et al. 2015; Martini and Bosch 2015; Yli-Huumo et al. 2015; Alves et al. 2016; Yli-Huumo et al. 2016; Martini and Bosch 2017; Rios et al. 2018a; Rios et al. 2018b). Currently, researchers from 11 countries (Brazil, Chile, Colombia, Costa Rica, Finland, India, Italy, Norway, Saudi Arabia, Serbia, and United States) have already joined the initiative. Therefore, this project has the primary goal of replication at a large-scale, with the study reported here being a small part of the larger puzzle.

In the first execution of InsighTD in Brazil, a total of 107 software industry practitioners answered the questionnaire between December/2017 and January/2018. Results show that most of the participants are familiar with the concept of TD, even though they were not all familiar with the term itself. Moreover, we elaborated a conceptual map of the current understanding of the participants about TD, based on the words they used to describe the concept. The conceptual map implies an understanding that is aligned with the TD definitions disseminated by the technical literature. Surprisingly, none of the participants reported possible benefits of incurring TD. This is an indication that software practitioners are more concerned about the drawbacks of having TD in their projects and that a clearer view about its benefits is still missing.

From a total of 78 identified causes, *deadlines, inappropriate planning, lack of technical knowledge,* and *lack of a well-defined process* are among the top 10 cited and most likely causes that lead to the occurrence of TD. On the other side, from a total of 66 identified effects, *low external quality, delivery delay, low maintainability, rework, and financial loss* are among the top 10 most commonly cited and impactful effects of TD. The identified set of causes and effects confirms existing literature on the topic, while broadening our understanding by presenting a more fine-grained view.

We also organized the causes and effects of TD into categories that represent the sources of TD pain in software projects (e.g., planning and management, quality, development issues, lack of knowledge, methodology). We also organized the causes and effects by type of debt. Finally, we organized the identified causes and effects of TD in probabilistic cause-effect diagrams, which represent knowledge about the common causes and effects of the presence of the debt based on the experiences of practitioners who answered the survey.

This work has several contributions for the discussion of TD. First, the InsighTD survey study is designed to facilitate replication by researchers around the world and to provide a

global characterization against which practitioners can compare their own context. Secondly, knowledge about causes and effects provides useful information for decision makers. Development teams can use the lists of identified causes to understand the factors that contribute to TD and, if necessary, identify preventative actions. The identified effects allow a clearer view of the possible consequences of TD in projects. These findings confirm and complement the existing literature. However, this work goes further by providing a more detailed discussion on the topic, by analyzing the relationships among causes, effects, and types of debt, and by identifying the most common and impactful causes and effects. Thirdly, we discuss the organization of the information on causes and effects of TD in probabilistic cause-effect diagrams. Such a representation can be easily interpreted by development teams and highlights the causes that can most contribute to the occurrence of the analyzed debt scenario as well as the most common effects that occur as a result of the TD. Thus, our contribution is intended to serve two audiences: researchers who will participate in the replications of the surveys and researchers and professionals interested in the results of the research.

Besides this introduction, this paper is organized in six more sections. Section 4 presents related work. Next, Section 5 discusses the design of InsighTD and presents the current status of its execution. The results of its first execution in Brazil are presented in Section 6. In Section 5, we discuss the organization of the identified causes and effects of TD in probabilistic cause-effect diagrams, we represent the relationship among causes, types of debt and its effects, we present the implications of the study for researchers and practitioners, and present the comparison to previous work. Then, we discuss threats to validity in Section 6. Finally, Section 7 presents final remarks and the next steps of this work.

2 Related work

In this section, we describe a number of studies that looked at causes and effects of TD and concentrate on the analyses of their results on these topics. We also briefly describe our own previously published article (Rios et al. 2018c).

2.1 Research on causes and effects of TD

Codabux and Williams (2013) performed a case study with a partner company during the implementation of agile development practices in order to identify best practices regarding the management of TD. One of the research questions investigated by the authors was "*RQ2: What are the consequences of TD on the development process?*". Some of the participants of the study indicated that they are aware that incurring TD in the short term will help them to achieve the goal of getting released resources but are not able to predict what the long-term impact would be. Some of them cited that the presence of debt can "*kill*" the project.

Ernst et al. (2015) reported the results of a survey of software engineers and architects working in software projects from two large multinational corporations and one government research lab, and addressed the following research questions: (i) do professional software engineers have a shared definition of TD?; (ii) are issues with architectural elements among the most significant sources of TD?, and (iii) are there practices and tools for managing TD? The results indicated that while participants believe the metaphor is itself important for communication, existing tools are not currently helpful in managing the details. Specifically talking about sources of TD, the authors asked participants to rank a randomly ordered predefined list

of 14 choices with respect to the amount of debt they represent on their projects. The authors found that immature decisions about the design architecture are the key source of TD.

Li et al. (2015), through a systematic mapping study, identified quality attributes that are compromised due to the presence of debt. Some of the impacted quality attributes stand out: maintainability, reliability, security, portability, and performance efficiency.

In other related work, Martini et al. (2014) conducted a multiple-case embedded case study in seven sites at five large companies to investigate the current causes for the accumulation of architectural TD (ATD). The authors investigated two research questions: (i) what factors cause the accumulation of ATD?, and (ii) what are the current trends in practice in the accumulation and recovery of ATD over time? As one of the results, the authors provided a taxonomy of causes and their influence in the accumulation of ATD.

Still in the context of this same study, Martini and Bosch (2017) investigated three additional research questions: (i) what are the most dangerous ATD items in terms of effort paid later?, (ii) what are the effects triggered by such ATD items?, and (iii) are there socio-technical patterns of events that trigger the creation of ATD leading to particularly dangerous interest to be paid? The authors reported that TD items can be contagious, causing other parts of the system to be contaminated with the same problem, which may lead to nonlinear growth of interest. The authors also presented a model of ATD effects that can be used for TD repayment prioritization.

This last topic was revisited in more detail in (2015), where Martini and Bosch presented the results of a multiple case study involving six cases in four large companies to investigate two research questions: (i) what is the information needed by product owners and architects to prioritize ATD with respect to feature development? and (ii) what are the differences between architects and product owners when prioritizing ATD with respect to features? According to the authors, delivery time, maintenance costs and risk, would benefit greatly from information related to the effects of architecture debt. The authors also highlighted how measures of ATD effects, especially contagious debt, quality issues and "*double*" effort would be appreciated by software developers.

Yli-Huumo et al. performed several studies in industrial environments to investigate the role of TD in software development (Yli-Huumo et al. 2014; Yli-Huumo et al. 2015). In (Yli-Huumo et al. 2014), the authors investigated the causes and effects of TD and what management and reduction strategies/practices are being used for TD. Through an exploratory case study with two independent software product lines in a mid-sized Finnish software company, they interviewed 12 practitioners with both business and development background. The results indicated that the primary reasons for incurring TD were management decisions that were made during the project to reach deadlines, or unknowingly due to lack of technical knowledge. In the long term, if TD is not paid back, it may generate quality issues in the software, which will later show as economic losses, such as extra work and decreased productivity. In (Yli-Huumo et al. 2015), Yli-Huumo et al. conducted semi-structured interviews with 17 representatives from two software organizations and concluded that workarounds (TD) are often intentional decisions and forced by time-to-market requirements. However, the stakeholders are not always familiar with the negative consequences of taking workarounds, like additional hours, costs, and poor quality.

In the report from the Dagstuhl Seminar 16,162, Avgeriou et al. (2016) indicated that the consequences of a TD item are many: technical debt can affect the value of the system, the costs of future changes, the schedule, and system quality. Also, they report that the cause of TD can be a process, a decision, an action (or lack thereof), or an event that triggers the

existence of that TD item, such as schedule pressure, unavailability of a key person, or lack of information about a technical feature.

In other work in the area, Besker et al. (2017a) investigated how software practitioners perceive and estimate the level of negative effects due to ATD. Through the analysis of survey data, the authors found that architectural TD has a significant negative impact on software practitioners' daily work. In addition, they also showed that ATD negatively affects all roles involved in software development and does not necessarily correlate with the age of the system and the level of negative effects generated by this type of debt. Complementing their previous work (Besker et al. 2017a), Besker et al. (2017b) investigated the issue of wasted development time due to TD. The results indicated that on average, 36% of all development time is wasted due to TD. Architectural and requirements debt generate the most negative effects. In addition, the results also indicate that all functional roles are affected by TD interest in different ways. And, the age of the software affects the amount of wasted time and the activities where time is spent.

More recently, Besker et al. (2018b) further explored wasted software development time due to TD through a longitudinal study. This study found that developers waste about 23% of their time due to TD and that they are frequently forced to introduce new TD due to already existing TD. The most common activities on which additional time is spent are additional testing, code analysis, and refactoring. In addition, this study indicated that developers are aware of the amount of time wasted on debt.

Besker et al. (2018a) conducted a systematic review of the literature in the ATD area in order to synthesize research efforts to create new knowledge and to create a common platform for future research. As result, the authors presented a descriptive model to support the management process. Concerning one of the research questions, RQ2.2 What are the major negative effects caused by ATD, the authors reported aspects such as flexibility, maintenance and evolvability, innovation and system growth, performance degradation, and reliability.

In other recent work in this area, Rios et al. (2018b) investigated causes that lead to the occurrence of TD, if these causes occur in isolation or in combination, if TD can be prevented, and, in terms of effort, if it is better to prevent debt, or incur it and pay it off later. Through an interview-based case study with ten practitioners from two software organizations, the authors identified 57 causes that lead a development team to incur debt. For most TD types, these causes occur in combination. It was also indicated that debt can be prevented, and it is better to work on prevention activities than to pay off debt later.

Thus, there is already evidence in the technical literature that reveals causes and effects of TD in software projects. Table 1 summarizes relevant information about this subset of related work that investigates our central questions. Table 1 shows which studies investigate causes and which look at effects (C or E in the second column), the comprehensiveness of each study in terms of whether it address TD in general or just ATD (third column), whether it uses a predefined list of causes and effects, or identifies causes and effects in vivo (fourth column), and the representativeness of each study in terms of the sample size and the number of organizations represented by the sample (last two columns). We can observe in Table 1 that the sample sizes tend to be quite small. The one with the largest sample (Ernst et al. 2015) is still limited by the number of different organizations, and by the fact that it used a pre-defined list of TD causes, which constrained participants to fit their experience into this format. Further, almost half of the relevant studies focused on architectural TD, just one type of debt, when we have 15 types in total (Rios et al. 2018a).

Thus, the current evidence on causes and effects of TD reported in the literature reflects the point of view of a small set of professionals from an even smaller set of organizations, limiting

Table 1 Related work

References	Cause/Effect	Comprehensiveness		Representativeness	
		TD Type	Predefined list of causes/effects?	Sample size	# of organizations
(Martini et al. 2014)	C	ATD	N	–	5
(Ernst et al. 2015)	C	General	Y	536	3
(Li et al. 2015)	E	General	N	–	–
(Yli-Huumo et al. 2014)	C/E	General	N	12	2
(Yli-Huumo et al. 2015)	C/E	General	N	17	2
(Martini and Bosch 2017)	E	ATD	N	–	6
(Rios et al. 2018b)	C	General	N	10	2
(Besker et al. 2018a)	E	ATD	N	–	–
(Codabux and Williams 2013)	E	General	N	10	1

our understanding on why TD is incurred in software projects and its consequences. This initial work (Martini et al. 2014; Ernst et al. 2015; Yli-Huumo et al. 2014; Yli-Huumo et al., 2015; Martini and Bosch 2017; Rios et al. 2018b; Besker et al. 2018a; Codabux and Williams 2013) is fundamental, not only to ground the InsighTD studies, but also to triangulate InsighTD results and observe how they complement each other. The result of the triangulation with the initial Brazilian InsighTD results is described in the *Section 5.4 - Comparison to related work*.

Lastly, different from related work, InsighTD has as one of its main goals replicability. As the study is replicated (at this time in 11 countries), we will incrementally define a corpus of knowledge on the causes and effects of TD that more closely reflects the state of the practice.

2.2 Preliminary work

In our first investigation on the causes of TD (Rios et al. 2018b), we performed a small interview-based evaluation with ten practitioners. We identified an initial list of causes. We also found, for the majority of TD types, that causes occur in combination. The participants also felt that debt could be prevented, and that it would be better to work on prevention activities than to pay it off later. Despite the study's limitations, the results motivated us to go further and pursue a research project that could support a deeper investigation on the causes of TD as well as its effects and, also, how development teams react to the presence of debt. This motivation led to the initiation of the InsighTD project.

In Rios et al. (2018c), we discussed the basic survey design and the preliminary results of the first round of InsighTD execution in Brazil. In that paper, we focused only on the discussion of the top 10 causes and effects of TD. Those preliminary results compose what we call the baseline report of InsighTD, that reports our interpretation of the initial results with basic descriptive statistics and reasoning. In this paper, we go further by extending that work with a full data analysis including:

1. A detailed analysis of participants' perceptions of TD, presented as a conceptual map;
2. A detailed analysis of the causes and effects of TD including: (i) the full list of identified causes/effects ranked by the most cited and, also, by the causes that most likely lead to the occurrence of TD and by the most impactful effects, (ii) the organization of identified

causes/effects into categories, and (iii) the relationships between TD types and causes/effects of debt;
3. The organization of the list of identified causes and effects into probabilistic cause-effect diagrams;
4. Triangulation of the results with related work.

Understanding the causes and effects of TD from the developers' perspective is critical to guide research directions. Practitioners can provide relevant perspectives and data on these topics, thus pursuing more such data is a worthwhile endeavor (Rios et al. 2018b). The current state of the literature has stimulated us to deeply investigate these topics. To the best of our knowledge, InsighTD will the first large-scale study in the TD area, involving researchers from different institutions around the world. Its design and the results from its first execution in Brazil will be discussed in the next sections.

3 A global family of surveys on td

We present, in this section, the design of the family of surveys on TD (InsighTD). The design enables the survey to be continuously replicated in different countries. The goal is to produce generalizable results about the state of practice including the causes that lead to TD occurrence, the effects of its existence and how these problems manifest themselves in the software development process. The rationale behind choosing countries as scopes of replication is twofold:

- organizing the work and making the dissemination of the survey wider by utilizing the local industry contacts of a large set of researchers;
- investigating whether differences in local development practices could influence how participants experience the TD concept.

To support the dissemination of results and collaboration among researchers, we provide for each replication a shared survey infrastructure using the same questionnaire and a set of instruments to guide the data analysis. We also have a communication plan that defines how the results and achievements will be communicated among the project participants. Finally, we also have a website (td-survey.com) where further details about the project and news about its replications are continuously updated.

In the following, we discuss our research questions and the methodology design. Then, we define the target population, the questionnaire itself, and the data analysis procedures.

3.1 Research questions

The overall design of InsighTD is based on the research questions presented in Table 2. The goal of RQ1 is to investigate how software practitioners define TD (RQ1.2), if this definition is close to the one commonly disseminated in the technical literature (RQ1.2), and how disseminated among professionals the concept of TD is (RQ1.1). We also investigate if the participants are able to describe representative TD items from their projects (RQ1.3). By representative we mean: (i) consistent with a TD definition adapted from McConnell (2007) (presented in the survey and in section 3.4) and the taxonomy of TD types defined by Alves

Table 2 Research questions

ID	Research questions
RQ1	Are software professionals familiar with the concept of TD?
- RQ1.1	How disseminated is the concept of TD among practitioners?
- RQ1.2	How do software practitioners conceptualize TD?
- RQ1.3	Are software professionals able to provide representative examples of TD items in their projects?
RQ2	What causes lead software development teams to incur TD?
- RQ2.1	Can TD causes be organized to broadly represent the main sources of factors that lead to the occurrence of TD?
- RQ2.2	Are the identified causes specific or shared among different types of debt?
RQ3	What effects does TD have on software projects?
- RQ3.1	Can TD effects be organized to broadly represent the main points of pain caused by the presence of debt?
- RQ3.2	Are the identified effects specific or shared among different types of debt?
RQ4	How do software development teams react when they are aware of the presence of debt items in their projects?

et al. (2016) and Rios et al. (2018a), and (ii) how commonly the development team faces that type of debt in its project. Thus, RQ1.3, gives us another way to ensure that participants have a consistent understanding of TD, which helps promote construct validity.

RQ2 aims at identifying possible causes and the causes most likely to lead to the occurrence of TD. RQ3 aims at identifying possible effects that debt items have on software projects, and those with bigger impact. RQ2 and RQ3 also allow us to investigate how TD causes and effects can be organized to broadly represent the main pain points in software projects (RQ2.1 and RQ3.1), and if the identified causes/effects are specific or shared among different types of debt (RQ2.2 and RQ3.2). Finally, RQ4 investigates how software development teams react (e.g. monitor, pay off, or define preventive actions) when debt items are identified in their projects. Besides these questions, we also define some specific questions to characterize the survey participants and their respective organizations and software projects.

3.2 Overall structure of InsighTD

InsighTD has been planned cooperatively with several TD researchers. Its design comprises four stages: conception, validation, initiation, and international replication. The first two stages, described in more detail in (Rios et al. 2018c), comprise the planning step of InsighTD. The conception stage included the definition of the research questions, design of the family of surveys, elaboration of survey instruments, definition of target population, and initial discussion about data analysis. In this stage we also defined the Core Team, responsible for executing the first round of the survey in Brazil and leading the project, and Replication Teams. Table 3 shows who is currently involved in InsighTD.

For the validation stage, we performed an internal validation, an external validation, and a pilot study before the first deployment of the survey instrument. The objective of the validation activities was to check the survey questions for clarity and completeness, i.e., internal and the construct validity. In total, the team took about six months to complete all the validations.

The first internal validation was performed by a senior Brazilian industry consultant and researcher (the third author of this paper), who then joined the *core team*. Further rounds of internal validation were performed by members of the international community: a senior US

Table 3 InsighTD Replication Teams

Researcher	Country	Researcher	Country
Nicolli Rios *(core team)*	Brazil	Dr. Vladimir Mandić	Serbia
Dr. Rodrigo Spínola *(core team)*		Dr. Nebojša Taušan	
Dr. Manoel Mendonça *(core team)*		Robert Ramač	
Dr. Carolyn Seaman *(core team)*	United States	Dr. Hernán Astudillo	Chile
Dr. Clemente Izurieta		Juan Pablo Brito	
Dr. Davide Falessi		Boris Rainiero Pérez Gutierrez	
Dr. Forrest Shull *(external reviewer)*		Dr. Darío Ernesto Correal Torres	
Dr. Antonio Martini	Norway	Dr. Francesca Arcelli Fontana	Italy
Boris Rainiero Pérez Gutierrez	Colombia	Dr. Valentina Lenarduzzi	Finland
Dr. Darío Ernesto Correal Torres		Dr. Davide Taibi	
David Chaves	Costa Rica	Dr. Kiran K Ravulakollu	India
Brenda Aymerich		Dr. Rajesh Kumar Upadhyay	
Julio Guzman		Mitali Chugh	
Alexia Pacheco		Neeraj Chugh	
Ignacio Diaz-Oreiro			
Gustavo Lopez			
Dr. Mohammad Alshayeb Abdullah Aldaeej	Saudi Arabia	–	–

researcher (the fourth author of this paper, who also joined the core team), and two other researchers from Finland. To perform the external validation, we invited an experienced researcher in the empirical software engineering and TD areas from the US. This researcher is not part of the InsighTD core or replication teams, acting as an independent reviewer of the instruments.

During internal and external validation, we received feedback concerning the design of the study, the research questions, and the questionnaire itself (e.g. clarity, ease of understanding, size). The feedback in many cases led to several changes, such as adjustments to the definition of TD used, and inclusion of definitions of traditional, hybrid, and agile process models.

After internal and external validation, we implemented the survey using the Google Forms infrastructure and then conducted the pilot study, in which we invited five pilot participants, through personal contacts, with varying levels of industry experience. After completing the questionnaire, the participants were asked to fill in a feedback form containing questions about 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. Their feedback and the analysis of the responses served to identify vague questions and incomplete answer options in the closed questions, thus increasing the internal and the external validity. Finally, after pilot execution, the members of the core team performed a last internal review of the questionnaire.

The third stage corresponds to the first execution of the survey, which has already been performed in Brazil, and the first replication of InsighTD, which is underway in the United States. Based on the results of the first execution, we have a baseline report (presented in (Rios et al. 2018c)) and a better definition of how we can execute the data analysis and the synthesis of the results. The last phase comprises the replication of the survey in other countries. After the first execution, the empirical package is now available for partners from other countries, so they can replicate the survey and reuse the whole set of instruments in the last stage. At the time of writing this paper, researchers from 11 countries (shown in Table 3) have already joined the project. The international replication of the questionnaire is currently under execution in Chile, Colombia, Costa Rica, Finland, Saudi Arabia, Serbia, and the United States.

Each replication of the survey will be performed independently by different researchers in different countries. All of them will use the same survey infrastructure and version of the questionnaire. The result of the continuous replication of the survey will be a rich empirical dataset on causes and effects of TD, and how software teams react when they are aware of the presence of debt items in their projects, that will be used by partners of the project to perform isolated or joint data analyses.

There are many advantages to the globally distributed nature of InsighTD, starting with the obvious advantages of replication in terms of increasing external validity and deepening understanding. With separate autonomous teams, work can proceed in different places both continuously and in parallel, where different teams learn from each other in real time and results and lessons are shared. Also, this structure allows the development of a cohesive research community working together around a shared goal, promoting cooperation and synergy.

3.3 Population

We use multiple strategies to reach the target population of practitioners. Each replication team will use the strategies most appropriate for their context. In most instances, we utilize social media (in particular LinkedIn), giving us direct access to a large number of professionals with whom we did not have previous contact. Specifically, LinkedIn allows the use of keywords to search for professionals with specific expertise. For example, to find professionals that could provide insightful answers about test debt, we used keywords like tester, test analyst, and test manager. The same applies to the other types of debt. Table 4 presents the keywords we have used to search for participants in LinkedIn for each type of debt.

Besides social media, participants are also being solicited from industry-affiliated member groups, mailing lists, and industry partners.

3.4 Questionnaire

The research questions guided the definition of the questionnaire, whose questions are summarized in Table 5 (the full questionnaire is available at https://goo.gl/zRwSGa). For each question, we define the research question it is related with, an identifier *"Q + number of the question"* that will be used to reference it in the text, and denote whether it is an open question or a closed one. In total, we defined 28 questions, including some questions to characterize the participants and their organizations (Q1 through Q8). We also presented the definitions of agile[1] (Beck et al. 2001; Alliance et al. 2016; Pressman and Maxim 2014), hybrid[2] (Alliance et al. 2016; Pressman and Maxim 2014), and traditional[3] (Pressman and

[1] Agile: a lightweight process that promotes iterative development, close collaboration between the development team and business side, constant communication, and tightly-knit teams (Beck et al. 2001; Alliance et al. 2016; Pressman and Maxim 2014).

[2] Hybrid: is the combination of agile methods with other non-agile techniques. For example, a detailed requirements effort, followed by sprints of incremental delivery (Alliance et al. 2016; Pressman and Maxim 2014).

[3] 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 (Pressman and Maxim 2014).

Table 4 Keywords used to search for participants in LinkedIn

TD Type	Keyword
Architecture debt	Software architect
Automation test, test, and defect debt	Test analyst, test manager, tester
Build, code, and design debt	Developer, programmer, project manager, software engineer
Documentation debt	All other keywords
Infrastructure debt	Configuration analyst, configuration manager, developer, programmer, software engineer, project manager
Process debt	Process analyst
Requirements debt	Requirements analyst, requirements engineer, product owner, project manager
Service debt	Software architect, developer, programmer, software engineer
Usability debt	Developer, programmer, software engineer
Versionning debt	Configuration analyst, configuration manager

Maxim 2014) process models and asked the participants which one was followed by the development team.

After completing the characterization questions, we ask (Q9) how familiar the participants are with the concept of TD. The available options are:

- "Never heard of it";
- "I have read about it in books / articles";
- "I have been on projects where I recognized TD but the project did not explicitly manage it", and;
- "I have been on projects where we attempted to actively manage TD".

Next, we ask participants to define TD in their own words. Then, we present a TD definition 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)". We chose to use this definition because it is already very disseminated in the technical literature and it is aligned with the types of debt identified by Alves et al. (2016) and Rios et al. (2018a). After, we ask (Q11) the participants how close this definition is to their understanding of the TD term, from the following options: "Very close", "Close", "Far", "Very far" and "Had no prior knowledge of TD".

We also ask (Q13) participants to provide an example TD item that occurred in their project (this example would then be used as the basis for answering later questions about causes and effects) and, next, we asked (Q15) how representative that example was, using the following options: "*It was a unique instance*", "*It is the type of thing that happens from time to time in the project*", and "*It is the type of thing that happens very often in the project*".

For RQ2, questions Q16 to Q19 support the identification of the causes that lead development teams to insert debt items into their projects. Initially, referring to the example TD item cited by the participant in Q13, we asked the participant what led the development team to incur the TD in that example. Then, in subsequent questions, we asked for more related causes, in an effort to dig further

Table 5 Survey questions (simplified)

RQ	No.	Question	Type
–	Q1	What is the size of your company?	Closed
	Q2	In which country you are currently working?	Closed
	Q3	What is the size of the system being developed in that project? (LOC)	Closed
	Q4	What is the total number of people of this project?	Closed
	Q5	What is the age of this system up to now or to when your involvement ended?	Closed
	Q6	To which project role are you assigned in this project?	Closed
	Q7	How do you rate your experience in this role (at the time)?	Closed
	Q8	Which of the following most closely describes the development process model you follow on this project?	Closed
RQ1	Q9	How familiar you are with the concept of TD?	Closed
	Q10	In your words, how would you define TD?	Open
	Q11	How close to the above TD definition is your understanding about TD?	Closed
	Q12	Are there any parts of the definition above from McConnell that you disagree with?	Open
	Q13	Please give an example of TD that had a significant impact on the project that you have chosen to tell us about:	Open
	Q14	Why did you select this example?	Open
	Q15	About this example, how representative it is?	Closed
RQ2	Q16	What was the immediate, or precipitating, cause of the example of TD you just described?	Open
	Q17	What other cause or factor contributed to the immediate cause you described above?	Open
	Q18	What other motives or reasons or causes contributed either directly or indirectly to the occurrence of the TD example?	Open
	Q19	Considering all the cases of TD you've encountered in different projects, and the causes of those TD cases, which causes would you say are the most likely to lead to TD (ordered by likelihood of causing TD)? Please list up to 5 causes.	Open
RQ3	Q20	Considering the TD item you described in question 13, what were the impacts felt in the project?	Open
	Q21	Considering all the cases of TD you've encountered in different projects and the effects of that TD that you have personally experienced, which 5 effects would you classify as the effects that have a bigger impact (ordered by their level of impact).	Open
RQ4	Q22	Do you think it would be possible to prevent the type of debt you described in question 13?	Closed
	Q23	If yes, how? If not, why?	Open
	Q24	Once identified, was the debt item monitored?	Closed
	Q25	If yes, how? If not, why?	Open
	Q26	Has the debt item been paid off (eliminated) from the project?	Closed
	Q27	If yes, how? If not, why?	Open
	Q28	Considering your personal experience with TD management, what actions have you performed to prevent its occurrence?	Open

and further down, beyond the "*obvious*" causes, and also to find combinations of causes. Next, in Q19, we asked participants to list up to five causes he/she considered the ones that most likely lead to TD, considering all the cases of TD he/she has encountered.

Regarding RQ3, we defined questions Q20 and Q21 to identify effects of the presence of TD in software projects, both with respect to the example TD item (Q20) and in general (Q21). Finally, for RQ4 (Q22 to Q28), we intend to collect some data that helps us to understand how TD has been managed in practice, in particular with respect to prevention, repayment, and monitoring.

As we can observe in Table 5, open questions have an important role in our questionnaire (50% of the questions). As this is the first comprehensive study on causes and effects of TD, we decided not to present an initial list of causes and effects to avoid limiting the responses from the participants. We are more interested in hearing the voices of practitioners on their real problems than ask them if they agree or not with a predefined list that might not reflect their sources of pain.

3.5 Data analysis

The survey instrument is composed of a mix of closed and open questions. Thus, we need to rely on a variety of procedures for data analysis.

For the analysis of the answers to closed questions, we first relied on descriptive statistics to get a better understanding of the data. We used the mode and median for the central tendency of the ordinal and interval data. For the nominal data, we calculated the distribution of participants choosing each option.

To analyze answers given to the open questions on causes and effects of TD, we applied qualitative data analysis techniques (Strauss and Corbin 1998; Seaman 1999). Qualitative analysis is useful for answering questions of the form "*what is going on here?*", for when we want to learn about what people understand and how they deal with what is happening to them through time and changing circumstances (Schreiber and Stern 2001). Thus, it is an appropriate method when we are interested in explaining, for example, the causes of TD.

As the answers given for RQ2 and RQ3 were not related to any previous expectations, we followed an inductive approach to generate a new theory based on the given qualitative data. We applied manual coding on the RQ2 and RQ3 open questions as follows. Initially, the first and second authors individually coded the set of all answers for two subsets of related questions (RQ2: Q16 + Q17 + Q18 + Q19, and RQ3: Q20 + Q21). This involves open coding as described in (Strauss and Corbin 1998), then axial coding to derive higher-level categories. This allowed us to discover what the data was telling us about causes and effects, allowing the possibility of novel insights. Next, the two coders discussed possible differences in their coding until they reached consensus. The coding involved attaching codes to small coherent units in the answers, and categorizing the emerging concepts (causes/effects) in a hierarchy of categories. This process was performed iteratively until reaching a state of saturation (the point where no new codes or categories were identified).

Thus, from an initial dataset of raw causes (Q16 to Q19) and effects (Q20 and Q21), we started to code (step 1) and, then group (step 2) them by similarity. For example, three participants cited the following causes in raw form: "*Deliver the requested functionality in a tight deadline*", "*Short project time*", and "*The deadline forced the team to put the features in the systems, only ensuring that it would work*". We initially coded these three chunks with "tight *deadline*", "*short time*", and "*deadline*", respectively. In step 3, we could identify these three examples as different nomenclature for the same causes/effects. Then, we unified the names of sets of causes/effects using the most commonly used term in that subset, which was *deadline* in this example. After repeating these steps on the whole data set we had the final list of causes and effects. The step 3 was also individually performed by two researchers, who cross-checked their answers until they reached consensus.

The basic coding and analysis of causes and effects into categories has been, and is expected to continue to be, done similarly in all InsighTD replications. A standard, basic, well-defined set of analysis procedures helps to reduce the possibility of bias, and to ensure consistency and comparability of results. However, replication teams are free to develop and employ other analysis strategies to investigate other questions and issues as they arise. For example, in the first InsighTD instance in Brazil, described in Section 4, there are specific strategies that we used to analyze the answers for RQ1.2 and RQ1.3. These specific strategies are also qualitative and will be described in more detail with their respective results.

As InsighTD replications increase, it will also be possible to coordinate analysis between teams in order to investigate questions that require more data than is available in any one

replication, e.g. the question of whether causes and/or effects vary in correlation with the development process model being used, or the role of developer experience on their perspectives on TD. During the initial steps of the project, it is expected that the results will be published in isolation, as we have done with the results of the first instance in Brazil in this paper. As time passes, results of the project will be disseminated in combination (from multiple replication teams), as this strategy will allow us to achieve a broader and, at the same time, deeper spectrum of research questions.

3.6 Current status of the InsighTD project and next steps

Currently, researchers from 11 countries (Brazil, Chile, Colombia, Costa Rica, Finland, India, Italy, Norway, Serbia, Saudi Arabia, and United States) have already joined the InsighTD project. While Brazil hosted the initial study (described in this paper), replications are currently underway all of these countries except for India, Italy, and Norway. The Italian and Norwegian replications will begin shortly, and the Indian team is in the planning stages. Data collection is complete in Chile, Colombia, Costa Rica, Serbia, and the United States with, respectively 92, 133, 156, 90, and 100 responses. Researchers from each replication team are working on analyzing their respective datasets. The next steps of the InsighTD project include the coordination of the replications, their syntheses, and dissemination of the results.

Some very preliminary results from some of the replications have been disseminated (Rios et al. 2018c; Rios et al. 2019b; Pacheco et al. 2019; Peréz et al. 2019; Freire et al. 2020), and some InsighTD data has been used, along with other data, in analyses of particular research questions (Rios et al. 2020), but this article is the first complete report of an application of InsighTD.

Although much effort has already been invested, primarily in data collection activities, there is still significant contribution to be gained from InsighTD in the future. At the first work meeting of the InsighTD project, which occurred together with the TechDebt'2019 conference, team members discussed opportunities for data analysis with the goals of minimizing duplication of effort and maximizing collaboration opportunities. Among other analyses, the team decided to work towards a more comprehensive data synthesis. One of them will be precisely the synthesis of the causes and effects identified in all replications. Another will be the investigation of differences between countries, a largely unexplored area that InsighTD is uniquely positioned to address empirically. Certainly the countries involved so far in InsighTD are diverse in terms of culture, geography, educational and business environments, etc. These points of diversity may or may not manifest in differences in how TD is experienced, but exploring this issue is part of the InsighTD agenda.

Next steps for the Brazilian InsighTD team is to extend the analysis of the survey results for RQ4, which will allow us to have an understanding of how TD manifests itself in software development processes and how software development teams react when they are aware of the presence of debt items in their projects. We also intend to investigate how to best use the TD probabilistic cause-effect diagrams in the software development process and the costs and benefits of their use in TD management activities.

Finally, InsighTD is a large-scale project. In addition to the aforementioned future work, we are in constant communication with the other replication teams to jointly define the future research agenda of the project. We intend that each disseminated result composes a piece of a larger puzzle. For those who are interested in more details about the project and how to join or lead a replication team, please visit the InsighTD website at www.td-survey.com to learn about the rules and responsibilities of joining, and to get in touch with the project organizers.

4 Results from Brazil

In this section, we present the results from the first survey round conducted in Brazil. The survey in Brazil was online from December 7th, 2017, until January 7th, 2018. In total, we sent the survey invitation to about 513 professionals and 112 of them completed the full questionnaire. This represents an approximate response rate of 22%. However, this is a rough estimate because social media and mailing lists do not allow accurate measurement of the number of individuals that read our recruitment message or read it but chose not to participate. Five participants who answered the questionnaire were excluded from the final dataset because they were not working in Brazil at that time (3 participants) or did not provide a valid example of a TD item according to McConnell's TD definition in Q13 (2 - "*I don't know what TD means*" and "*Errors that arise while fixing other issues*").

In the following, we first summarize the information about the study population, before describing the results for each of the research questions. In this article, we focus on causes and effects of TD. Thus, we include here only research questions RQ1, RQ2, and RQ3. RQ4 will be considered for analysis in future work.

4.1 Demographics data

Several types of expertise were represented in the participant sample, as we can observe in Table 6. Most of the participants were developers, but we also have project managers, testers, software architects, and requirements analysts among others. Participants defined their level of experience in their role among the following options: 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), and Expert (Authoritative knowledge of discipline and deep tacit understanding across area of practice). Table 6 summarizes this demographic information. For each role, Table 6 shows the number of respondents in that role (second column) and the percentage of the entire sample in that role (third column). Table 6 also shows, for each level of experience, first the percentage of the entire sample at that level (e.g. 11% of the entire sample identifies as Beginners), and then the number of respondents in each role at each experience level (e.g. 17 Developers identied as

Table 6 Participants' roles

Role	#	%	Novice (1%)	Beginner (11%)	Competent (32%)	Proficient (36%)	Expert (20%)
Developer	44	41%	0	2	17	18	7
Project Leader / Project Manager	14	13%	0	3	6	4	7
Test Manager / Tester	12	11%	0	1	6	5	1
Software Architect	10	9%	0	0	2	6	2
Requirements Analyst	10	9%	0	4	3	3	0
Process Analyst	4	4%	0	1	1	1	1
Infrastructure analyst	3	3%	1	0	1	0	1
DBA	3	3%	0	0	0	1	2
Performs multiple functions	3	3%	0	0	1	1	1
Business Analyst	2	2%	0	1	1	0	0
Configuration Manager	2	2%	0	0	0	0	2

Competent). In summary, Table 6 shows that a significant portion of the sample are proficient, competent or expert (88% of the total), indicating that, in general, the questionnaire was answered by professionals with experience in their functions. On the other hand, responses from professionals with low level of experience (12%) were also obtained.

Organizations of different sizes are represented in the dataset. Figure 1 shows the distribution of organization size, both according to the choices on the questionnaire, but also in the more coarse-grained categories of small, mid, and large based on a classification adapted from Mendez-Fernandez et al. (2015). The participants are well distributed among small (28%), mid (45%), and large size (27%) companies. By observing the data in more detail, we can see that most participants (mode) work in organizations with more than 2000 employees, closely followed by enterprises with 11–50 and 51–250 employees. The median size is 251–500 employees. Therefore, the participants tend to work in larger companies, but we have representatives from companies of all sizes as we can observe in Fig. 1.

Table 7 shows that participants tend to work in small development teams, but we have representatives from teams of all sizes (Q4). We can observe that most of them (51%) are part of development teams with no more than nine professionals. We also have a good sample in teams with 10–20 people (31%) and larger development teams with more than 30 professionals involved in the construction of the software (16%).

Concerning the process models used, respondents answered a multiple choice question with the following options: agile, hybrid, and traditional. Out of the 107 participants who completed the survey, most projects were described as agile (48%) followed by a hybrid process model (36%). Less common is the use of a traditional process (16%). Thus, despite the fact that we have a concentration of participants in agile environments, the dataset also includes significant samples representing all process models.

The most common system age was 2 to 5 years old (30%), closely followed by 1–2 years old (28%). We also had a significant number of represented systems of less than 1 year old (19%) and 5–10 years old (18%) (Q5). Finally, the systems were typically between 10 KLOC and 1 million SLOC in size. But we also had good samples for smaller (<10 KLOC – 23%) and larger systems (>10 MLOC – 9%) (Q3).

Thus, overall, the collected data represents a very wide variety of software development contexts, including (i) several participants' roles and levels of experience, (ii) organizations of different sizes, and (iii) projects of different age, size, team size, and process models.

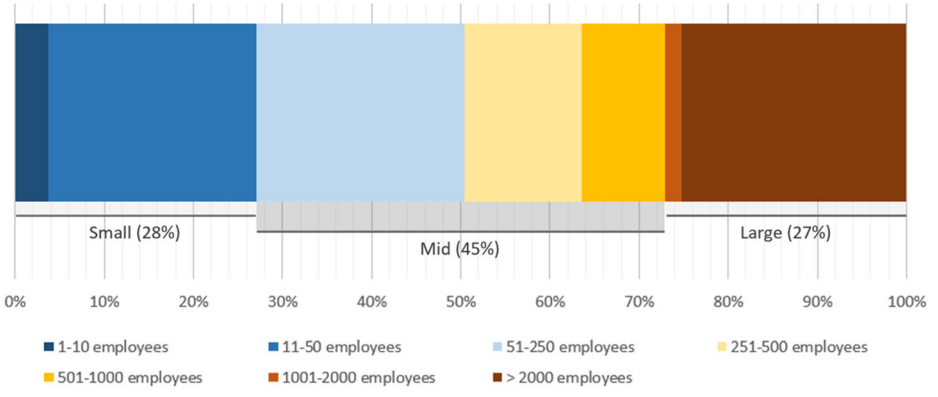

Fig. 1 Organization's size

Table 7 Team size

Teams Size	#	%
< 5 people	28	26%
5–9 people	27	25%
10–20 people	33	31%
21–30 people	2	2%
> 30 people	17	16%

4.2 Practitioners' point of view on the concept of technical debt (RQ1)

4.2.1 Dissemination of the concept of technical debt among practitioners (RQ1.1)

Initially, we asked (Q9) how familiar the participants were with the concept of TD. We can see in Fig. 2 (left side) that for each ten participants, six are somewhat familiar with the concept, i.e., 60% responded with something other than "*Never heard of it*". However, only 22% of the participants indicated that TD identification or management was part of their daily activities.

Next, we presented the TD definition adapted from McConnell (2007) and asked (Q11) the participants how close this definition is to their understanding of the TD term. Most participants (80%) indicated that their understanding is close to or very close to the TD definition we presented (the graph presented in the right side of Fig. 2). Looking at the relationship between the answers to these two questions, we find that, of the 40% of respondents who said they had "*never heard of*" TD before seeing a TD definition, 60% then indicated in their response to question Q11 that the TD definition we presented was close or very close to their understanding of TD. One can conclude that this group of respondents (24% of all participants) was in fact familiar with the concept of TD, but not the term itself. This is confirmed by the fact that all of these respondents did in fact provide a valid example of TD in response to Q13. This

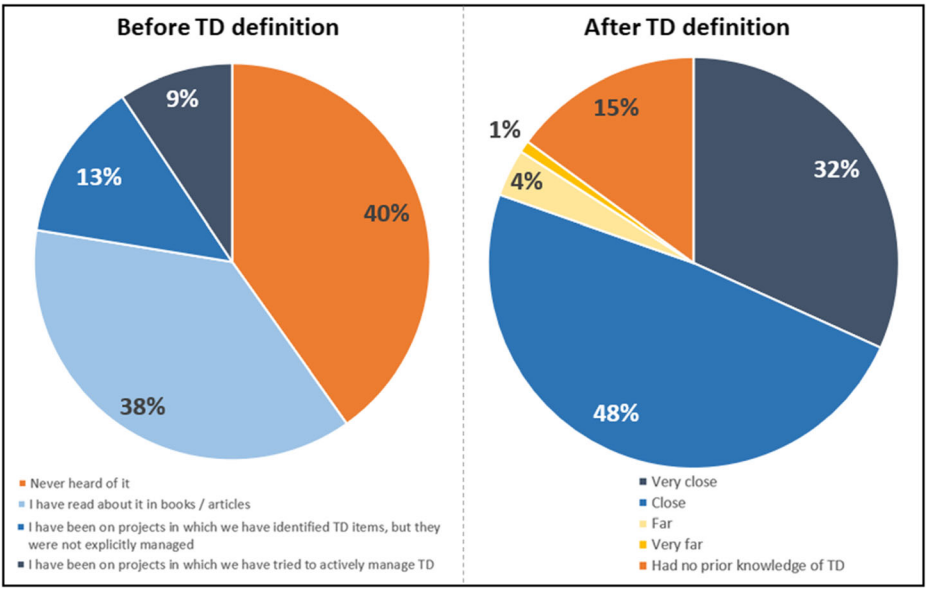

Fig. 2 Participants' familiarity with TD concept

leaves a small (16%) of respondents who were completely unfamiliar with TD, either before or after seeing a definition. Even so, they were able to provide a valid example of TD item in Q13.

The results for these two questions also indicate that although TD is a research topic close to the software industry, the use of the term still needs to be expanded. We also noticed that a large part of the participants, from different levels of experience, have an understanding close or very close to the provided definition. Experts indicated that their understanding is close or very close 86% of the time. For proficients we had 85%, 74% of the competents, and 83% of the beginners. For novices, we only had one answer ("*had no prior knowledge of TD*"). This result suggests that the level of experience does not impact their familiarity with the concept of TD. This aligns with the common perception that TD is easily understood and intuitive for practitioners (Rios et al. 2018d).

The answers to Q12 also reinforce the alignment between participants' perception of TD and the TD definition we presented to them in the questionnaire. Q12 asked whether participants would in any way modify the TD definition presented in Q11. Most of the participants agreed with the presented definition of TD with exception of two of them. One who disagreed said, "*I disagree at the point where it is said: type of debt that brings a short term benefit for the project*". The other wrote a long explanation about how inadequate testing and documentation should not be considered TD.

Some participants suggested modifications to the definition, mostly minor. Two more substantive examples are: "I would add that any project activity, not only software development, but also training, requirements gathering, etc., when poorly done or poorly executed can bring immediate benefit, but with a medium/long-term loss.", and, "The definition must be more contextual – it should not be only restricted to the debt of the software itself, but also to the development environment - developers, technical leadership, management and even strategic stakeholders of the organization.". These two participants indicated with their suggestions that TD goes beyond project development activities. They imply that issues such as lack of team training and problems in technical leadership and management can trigger problems in the medium and long term.

4.2.2 Software practitioners' expression of technical debt (RQ1.2)

Participants were asked to define TD in their own words (Q10), before being presented with the TD definition found in the technical literature. In total, 66% of the participants answered the question. The other 34% who did not answer Q10 were the same who, in Q9, reported that they "*had never heard of*" TD.

We coded the reported definitions to identify the main TD elements included by the participants. First, each definition was read and the elements identified were extracted. Then, the elements of all definitions were grouped and those that referred to the same element but had a different nomenclature were standardized using the most commonly cited term. In the end, we had a list of elements that participants included in their TD definitions. Table 8 shows the steps followed using two example TD definitions extracted from the set of answers. In the first column are the two example definitions, with fragments marked (in bold underlined) that represent elements of TD (as identified by the coder). Then, in the second column we list the marked fragments. Finally, in the third column, we present the standard term that we used to represent each identified element. For example, in the first line, the standardized term *low maintainability* refers to the fragment of text *harmful to the good evolution of the system*. This process was carried out by the first author, and carefully reviewed by the second.

Next, we analyzed each TD definition reported in search of explicitly described relationships among the elements. Considering the same examples presented in Table 8, Table 9 illustrates how this analysis was performed. The underlined texts refer to the identified elements. The bold fragments refer to the terms that implied relationship between the elements. In the second column, we show the identified relationships using the standard terms previously presented in the second column of the Table 8 and the relations identified in the first column of Table 9.

Figure 3 presents a conceptual map containing the elements, the frequency with which each element was mentioned, and their relationships identified in the TD definitions. The map shows that TD can be inserted intentionally or unintentionally (boxes in blue and dark blue). Lack of technical knowledge leads to unconscious decisions that end up leading to the presence of debt items. On the other side, short-term goals and lack of time impact team choices about how development activities will be carried out. These choices are sometimes characterized by issues such as planning failure, inappropriate prioritization of activities, and negligence in carrying out activities. Finally, TD is mainly characterized by unrealized activities or poorly performed activities (yellow boxes). Their presence can bring several consequences for the project such as low quality, negative impact on the project, rework, among others (orange boxes).

Thus, overall, the conceptual map represents in a simplified way the current understanding of the participants of the study about the concept of TD. According to the map, this understanding is structured in terms of causes that lead development teams to incur TD (represented in blue), the meaning of the TD concept itself (in yellow), and the effects of its presence (in orange).

The conceptual map also allowed us to observe that, although the participants provided their definition of TD before the definition described in the technical literature was presented in

Table 8 Examples of the TD definition analyses procedure

Step 1 – Extraction of elements	Step 1 – Extracted elements	Step 2 – Standardization
Technical debt refers to activities that, during the software development, are not performed due to unfamiliarity and time constraints, that in the future can become harmful to the good evolution of the system.	1. Activities that, during the software development, are not performed 2. Unfamiliarity 3. Time constraints 4. Harmful to the good evolution of the system	1. Not performed activity 2. Lack of technical knowledge 3. Deadline 4. Low maintainability
TD can be understood as a strategy to stop doing some activity/artifact in order to achieve a short-term goal. On the other hand, TD can also be understood as the activities / artifacts that were not developed by negligence or even lack of knowledge. When not properly managed, TD can impact the maintenance and evolution of the software.	1. Strategy 2. Stop doing some activity/artifact 3. Achieve a short-term goal 4. Negligence 5. Lack of knowledge 6. Impact the maintenance and evolution of the software	1. Team Choices 2. Not performed activity 3. Short-term goal 4. Negligence 5. Lack of technical knowledge 6. Low maintainability

the questionnaire, their perception is close to that definition. However, we could not find in any of the answers for Q10 an indication of the possible benefits of incurring TD. This can indicate that software practitioners represented by the population of this study are more concerned about the drawbacks of having TD in their projects and a clearer view about its benefits is still missing.

4.2.3 Representativeness of technical debt item examples (RQ1.3)

We asked (Q13) the participants to provide an example of a TD item that occurred in their projects, their reasons for selecting that example (Q14), and how representative it was for their project (Q15). When asking participants to provide an example of a TD item, we did not ask for a specific type, nor did we ask them to identify the type of debt they were talking about, because we do not have any way to assure that participants have a common understanding of the types of debt. Thus, we only asked for an example ("*Q13: Please give an example of TD that had a significant impact on the project that you have chosen to tell us about:"*). The association between the provided example and its corresponding type of debt was identified during the analysis of the results as described in the following paragraphs. The taxonomy of types of debt considered in this work is based on the list provided by Rios et al. (2018a).

To classify these examples as valid or not, we used two criteria: (i) the example needed to be compatible with the definition of TD we used in the survey, and (ii) the given example could be classified into one of the types of debt defined in Rios et al. (2018a). At the end, only two examples were discarded: "*I don't know what TD means*" and "*Errors that arise while fixing other issues*". The first was deleted because no example was provided at all. The second example was discarded because it does not fit any definition of TD found in the technical literature. Thus, our participants were able to provide valid examples of debt items, indicating that they have a reasonable understanding of TD and that the answers given to the other survey questions are based on valid examples.

Next, we analyzed the answers for Q13 and Q14 together to identify the type of debt associated with each example. We used the list of indicators[4] of TD defined by Alves et al. (2016), each of which is associated with a TD type. Thus, for each example of a TD item (Q13) and justification of why the participant chose that example (Q14), we looked for terms that could be mapped to indicators of TD. If we could perform the mapping, then we could identify the type of debt associated with that example. If we could not perform the direct mapping, then we identified the type of debt based on the overall description provided by the participant. This process was performed by the first author and reviewed by the second one. This process succeeded in all 107 cases to associate a specific TD type to each of the examples.

Table 10 illustrates the steps we followed to identify the types of debt, either by mapping between text fragments and TD indicators (column TD Indicator), or referring to the overall description. In the examples in Table 10, the terms highlighted in underlined and bold were decisive in making the association of the TD type to that example of TD item. In the first line, we can observe that there is a direct mapping between the text in underlined and bold from the example cited by the participant in Q13 (second column) and the TD indicator *outdated documentation*, from Alves et al. (2016), presented in column "*TD indicator*". As this

[4] TD indicators allow the discovery of TD items when analyzing different software development artifacts. An TD indicator is sometimes a metric, or sometimes something less formal, that can be used to point to areas with specific types of debt (Alves et al. 2016).

Table 9 Examples of the identification of relationships between the elements

TD definition	Identified relationships
Technical debt refers to activities that, during the software development, are not performed **due to** unfamiliarity and time constraints, that in the future **can become** harmful to the good evolution of the system.	• Lack of time ➔ Not performed activity ➔ Low maintainability • Lack of technical knowledge ➔ Not performed activity ➔ Low maintainability
TD can be understood as a strategy **to** stop doing some activity/artifact **in order to** achieve a short-term goal. On the other hand, TD can also be understood as the activities / artifacts that were not developed by negligence **or even** lack of knowledge. When not properly managed, TD **can impact the** maintenance and evolution of the software.	• Short-term goal ➔ Team choices ➔ Not performed activity ➔ Low maintainability • Negligence ➔ Not performed activity ➔ Low maintainability • Lack of technical knowledge ➔ Not performed activity ➔ Low maintainability

indicator is associated with documentation debt, we classified this example as documentation debt. For the second line, there is not a direct mapping between the described example and the list of TD indicators from Alves et al. (2016). However, by reading the example, we can observe that it refers to violation of modularity, which is an architectural issue and a well-known kind of architecture debt. Thus, we classified the second example as architecture debt.

We identified 11 types of debt, as shown in Table 11. One possible explanation for the larger concentration of examples in design, code, documentation, and requirements TD types is that they are all interrelated during development activities: design and code are implemented while requirements/documentation are used to support this implementation.

In addition, we also analyzed the relationship between the identified types and the role of the participant who mentioned them. We can see in Table 11 that roles as developer and test

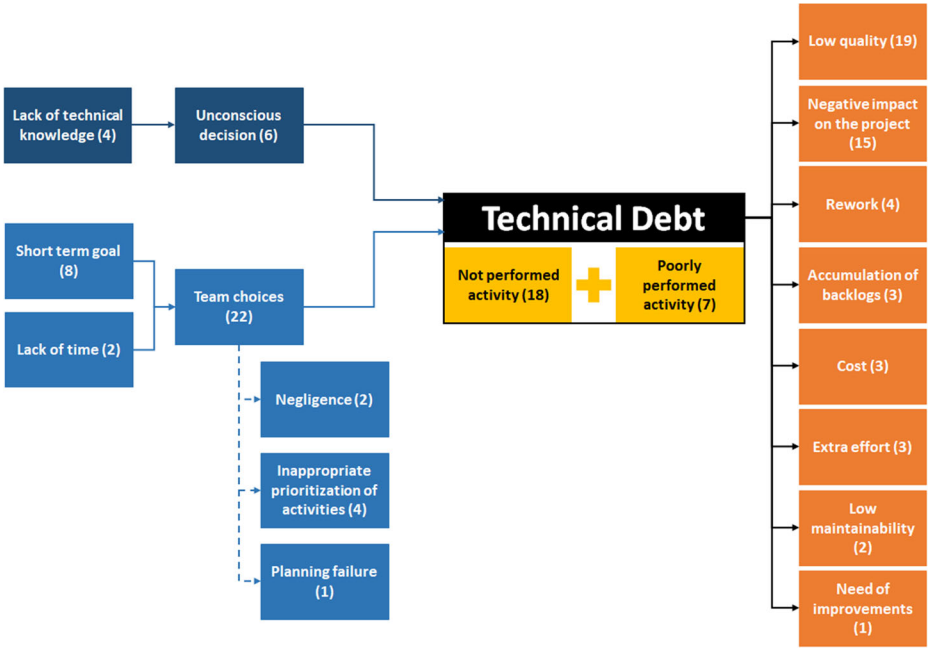

Fig. 3 Conceptual map of identified elements in TD definitions reported by study participants

Table 10 Identification of the type of debt related to the examples of TD items

	Answer for Q13	Answer for Q14	TD Indicator	TD Type
Mapping using TD indicator	"***Update pending documentation***"	"*As I work with tests, the non-update of documentation impacts directly on my daily work.*"	Outdated documentation	Documentation
Overall description	"*The refactoring of a system* ***module that impacted another module***"	"*Because this example makes clear the need to deal with TD earlier, minimizing future impacts*"	–	Architecture

manager/tester have contact with several types of debt, not only those related to coding and testing, respectively. We also observed that some types of debt (build, infrastructure, process, usability, and versioning) were cited by only one participant, which may indicate that participants have difficulty associating TD with artifacts different from those directly related to the source code of the project.

Finally, we also asked (Q13) participants to indicate how representative the provided example was. Most participants (42%) indicated that their example is a type of debt item that happens very often, closely followed by those who indicated that it was a kind of situation that occurs from time to time in the project (40%). Only 18% of participants indicated that the provided example is about a unique situation. These results indicate that debt items upon which this study's results are based are real and recurrent in software projects.

4.3 Causes of technical debt (RQ2)

The analyses presented in the following subsections are based on the answers to questions Q16 to Q19 of the questionnaire. Question Q16 asked the participant to cite the immediate, or precipitating, cause of the example of TD the participant described in Q13, while Q17 and Q18 asked for additional contributing causes. In Q19, we ask the participant to cite which causes they think are the most likely to lead to TD in general.

In this section, we present analyses that view this data from different perspectives. Initially, we will present the most cited causes, both with respect to the specific examples (Q16–18), and in general (Q19). Next, the identified causes will be organized into categories, which allows us a higher-level view of factors that lead to TD. Finally, we will organize the idenfied causes by type of debt.

4.3.1 Most cited causes with ranking of those that most contribute to the occurrence of debt (RQ2)

Overall, 78 causes were identified in this study. Two researchers analyzed each participant's answers to Q16-Q19, identifying the causes by directly copying the terms provided in the answers. To standardize the nomenclature, small adjustments were made without altering the semantics of the labels (for example, *deadline* and *short deadline* were mapped to *deadline*). The causes were then filtered to exclude duplicates. In the end, the overall frequency of each cause is the number of participants who mentioned it.

Table 11 TD Types x Project participants' role

TD Type	Total	Infrastructure analyst	Business analyst	Process analyst	Requirements analyst	Software architect	Performs multiple functions	DBA	Developer	Configuration manager	Test manager / Tester	Project Leader / Project Manager
Architecture debt	**11**	–	–	1	1	4	–	–	2	–	1	2
Build debt	**1**	–	–	–	–	–	–	–	1	–	–	–
Code debt	**15**	–	–	1	1	1	1	–	7	1	1	2
Defect debt	**2**	–	1	–	–	–	–	–	–	–	1	–
Design debt	**23**	2	–	–	1	3	–	–	10	–	3	4
Documentation debt	**15**	–	1	1	3	–	–	–	7	–	1	2
Infrastructure debt	**1**	–	–	–	–	–	–	1	–	–	–	–
Process debt	**1**	–	–	–	–	–	1	–	–	–	–	–
Requirement debt	**15**	–	–	–	2	2	1	–	5	–	1	4
Service debt	**2**	–	–	1	–	–	–	–	1	–	–	–
Test automation debt	**3**	–	–	–	–	–	–	–	2	–	1	–
Test debt	**13**	1	–	–	2	–	–	–	8	–	2	–
Usability debt	**1**	–	–	–	–	–	–	–	1	–	–	–
Versioning debt	**1**	–	–	–	–	–	–	–	–	1	–	–