Journal of Software Engineering Research and Development, 2019, 7:1, doi: 10.5753/jserd.2019.19

A Taste of the Software Industry Perception of the Technical Debt and its Management in Brazil

Victor Machado da Silva [Universidade Federal do Rio de Janeiro | victor0machado@gmail.com]
Helvio Jeronimo Junior [Universidade Federal do Rio de Janeiro | jeronimohjr@gmail.com]
Guilherme Horta Travassos [Universidade Federal do Rio de Janeiro | ght@cos.ufrj.br]

Abstract

Background: Technical Debt (TD) metaphor has been an exciting topic of investigation for the software industry and academia in the last year. Despite the increasing attention of practitioners and researchers, TD studies indicate that its management (TDM) is still incipient. Particularly in Brazilian Software Organizations (BSOs), there is still a lack of information regarding how software practitioners perceive and manage TD in their projects. **Objective:** To characterize TD and its management under the perspective of BSOs using their practitioners as proxies and extend the discussions presented at the 2018 Ibero-American Conference in Software Engineering. **Methods:** A survey was performed with 62 practitioners, representing about 12 organizations and 30 software projects. **Results:** The analysis of 40 valid questionnaires indicates that TD is still unknown to a considerable fraction of the participants, and only a small group of organizations adopts TD management activities in their projects. Besides, it was possible to obtain a set of technologies that can be used to support TDM activities and to make available a survey package to study TD and its management. **Conclusions:** Although the results provide an initial and representative landscape of the TDM scenario in BSOs, further research will support to observe how effective and efficient TDM activities can be in different software project contexts.

Keywords: *Technical debt, Software quality, Survey, Experimental software engineering*

1 Introduction

The software evolution is essential for the survival of a software product in the market since the environment in which it is immersed continually changes. As argued by Boehm (2008), in the face of an increasingly dynamic and competitive market, software development organizations need to support continuous and fast delivery of value to the customer in both short and long terms. In this scenario, many software organizations introduce agility practices into their development processes to handle the frequents requirements changes and the continuous delivery demand (de França et al. 2016).

This context reflects the challenges faced by software practitioners regarding the many decisions they take in their projects over time. At the same time, software practitioners should build high quality, low cost, on time, and useful software products.

This working environment brings challenges to practitioners regarding the decision-making, setting up a trade-off that can lead to the intentional or unintentional creation of "Technical Debt" in software projects over time. As argued by Tom et al. (2013) and Avgeriou et al. (2016), most, if not all, software projects face some TD.

TD refers to technical decisions taken in the software development scenario involving intertemporal choices (Becker et al. 2018), which influence positively (intentional and managed) or negatively (unintentional and not managed) to the software project ecosystem and the quality of their software products. When TD is perceived and managed in software projects, it has the potential to support deliveries of value to customers in a short time. On the other hand, in the long-term, some risks to internal software increase when the debt is not perceived and managed in the projects, hindering the software products maintenance and evolution (Avgeriou et al. 2016).

Currently, the TD metaphor interest and use have grown over the years (Li et al. 2015). Many studies have been discussing different knowledge areas of TD and supporting solutions to software engineers to achieve better results in their projects. Using an ad-hoc literature review, we observed some studies discussing the concept of TD and technologies to support the Technical Debt Management (TDM). As mentioned in Li et al. (2015) and Alves et al. (2016), only a few studies deal directly with the question of how software organizations perceive and apply the TD metaphor in their working environment. Also, the software development process is influenced by the country's culture, language, and beliefs (Prikladnicki et al. 2007), and it can influence how TD can emerge, be perceived, and managed.

Particularly in Brazil, there is a more latent gap regarding how the Brazilian software organizations (BSOs) perceive TD and how their practitioners handle it in their projects. Assuncao et al. (2015) reported that TDM is a topic of interest at Brazilian federal administration departments. However, there is scarce information on whether TD is adequately managed in BSOs. This information is useful since it can provide initial insights so that BSOs can improve their software processes to minimize the risks that TD can bring to the software project ecosystem and the quality of their software products.

This context motivated us to investigate how BSOs (represented by their practitioners) adopt and manage TD. Also, it is our interest to observe whether the perception from BSOs' practitioners on TD and its management match the findings of other TD investigations. Our study intends to

raise the level of knowledge of TD and its management in BSOs. Therefore, a survey was designed and conducted with software practitioners engaged in Brazilian software organizations. This paper presents the results of this survey, intending to provide the following initial contributions:

- To get an initial perception of the TD metaphor and its management in BSOs, using their engaged professionals as proxies;
- To make available a survey package with empirically evaluated instruments to support the gathering and aggregation of information regarding the TD perception and TDM activities, tailorable to other localities.

This paper is an extension of a previous publication at CIbSE 2018 (Silva et al. 2018b). It details the theoretical background on TD regarding its concepts, classification, TDM activities, and approaches for TD management. It offers a comparison between the obtained results and those concerned with the related works. The survey's design, analysis, and discussion of results are comprehensively presented, including the answers from three new survey's participants.

The remainder of this paper is structured as follows: Section 2 provides a background on TD; Section 3 summarizes the related works to our research; Section 4 presents the survey design; Section 5 explains the survey results; Section 6 presents the discussion about the main findings, the works related to our research and the threats to validity; and Section 7 presents the final considerations.

2 Theoretical background

Ward Cunningham (1993) first coined the term "Technical Debt" when discussing with stakeholders the consequences of releasing a poorly written piece of code to accelerate the development process. Although the code attends the core system requirements in the current release, in case of future changes, the consequences might spread over other software areas, affecting its evolvability.

Since then, the TD metaphor use spread to allow better communication with non-technical stakeholders (e.g., corporate managers, clients, among others). Moreover, it has been used as a quality improvement instrument, bringing to the software development context terms such as "principal" (used to refer to the required effort to eliminate the TD source) and "interest" (the additional effort needed on software maintenance due to the presence of TD) (Alves et al. 2016).

Although reasonably disseminated, up until 2016, there was no standard definition of the TD concept, creating several inconsistencies in the technical literature (Tom et al. 2013). Some definitions of TD over the years are "a way to characterize the gap between the current state of a software system and some hypothesized 'ideal' state in which the system is optimally successful in a particular environment" (Brown et al. 2010), "any side of the current system that is considered suboptimal from a technical perspective" (Ktata and Lévesque 2010), and "a tradeoff between implementing some piece of software in a robust and mature way (the 'right' way) and taking a shortcut which may provide short-term benefits, but which has long-term effects that may impede evolution and maintainability" (Klinger et al. 2011).

This imprecision on the TD definition could cause several misinterpretations and even a TD metaphor misuse and damage to the concept. Tom et al. (2013) affirmed that "it is evident that the boundaries of technical debt, as reflected in academic literature, are fuzzy – they lack clarity and definition – and represent a barrier to efforts to model, quantify and manage technical debt."

The lack of consensus on the TD definition was brought to attention during the Dagstuhl Seminar 16162, "Managing Technical Debt in Software Engineering" (Avgeriou et al. 2016). This seminar gathered members from academia and industry to discuss many relevant points regarding the TD concept. At the end of the seminar, the participants came up with a TD definition: *"in software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible."* TD is also acknowledged for being restricted to internal software quality issues, like maintainability or evolvability.

As it can be observed, some differences between the definition of TD and its first association with financial debt appeared along the years. The primary divergence is on the optionality to repay the TD item (Guo et al. 2016). However, some similarities remain. Similar to financial debt, strategical, controlled decisions that opt to postpone some tasks to obtain short-term gains such as shortening time-to-delivery can be decisive for a product's success (Yli-Huumo et al. 2016).

Nowadays, the definition proposed in the Dagstuhl Seminar is the most accepted among the researchers, and it is adopted throughout this paper. This definition contradicts, though, with some previous concepts of what should be considered and what should not be considered TD. For instance, unfinished tasks in the development process are considered as a type of non-TD, as reported by Li et al.'s (2015) secondary study. However, it fits the Dagstuhl's TD definition and should be considered as such in this paper.

In other words, TD can be associated with technical decisions about the shortcuts and workarounds taken in software development. Such decisions can influence positively (strategic and managed) or negatively (unintentional and not managed). Depending on the perspective, the presence of TD can influence positively or negatively a software project and the quality of its software products. Strategical, controlled decisions that opt to postpone some tasks to obtain short-term gains such as shortening time-to-delivery can be decisive for a product's success. However, TD can cause damage to the project, since it might be incurred unintentionally throughout the software development cycle. TD items of this nature can be incurred due to many factors, such as a lack of knowledge of team members on writing the source code without following a specific programming style. Therefore, it is crucial that software organizations perceive and manage TD in their projects.

2.1 Classification

Even before the academic interest in the TD metaphor, the industry already had presented alternatives to classify it. McConnell (2007) divided TD into two different types: unintentional debt (in which TD does not incurr due to a strategical purpose, like a bad-written piece of code created by an inexperienced programmer); and intentional TD (when it is usually incurred with a strategical approach, when the team or the organization decides to achieve a short-term gain at the cost of a long-term effort). An example of intentional TD is the decision on developing a simplified architecture solution for the software, knowing that it might not attend the project's future needs.

Martin Fowler (2009) expanded the classification created by McConnell (2007), considering that, beyond the TD being intentional (deliberate) or unintentional (inadvertent), it can also be reckless or prudent. TD quadrants structure these classifications, as shown in **Figure 1**. A reckless debt (either deliberate or inadvertent) incurs in the project and is not adequately planned, creating unnecessary risks. On the other hand, prudent debt items receive attention from the developing team, which assess their risks and make a plan to repay them.

Another perspective is to observe the original artifacts that the TD item incurred or the TD item nature. Tom et al. (2013) named this classification scheme as dimensions of TD, naming five types of TD. Posteriorly, Li et al. (2015) conducted a systematic mapping study, expanding the TD dimensions into ten types. The most recent study attempting to classify the TD according to its nature or origin artifact, to our knowledge, is by Alves et al. (2016). In this study the authors provide classification of TD in fifteen types, like design debt (associated with violations of the principles of good object-oriented design, documentation debt (issues observed in the software documentation) and code debt (problems found in the source code, that can make it harder to maintain, usually related to inadequate coding practices).

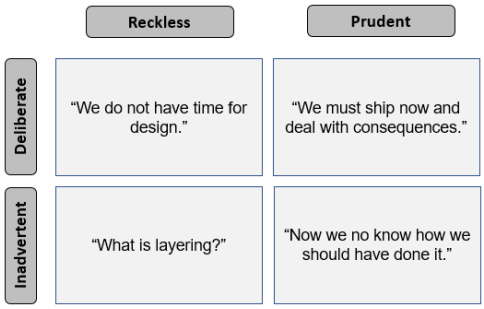

Figure 1. TD Quadrants (Adapted from Fowler (2009))

2.2 TD Management

Li et al. (2015) state that TDM includes activities that prevent potential TD (both intentional and unintentional) from being incurred, as well as those activities that deal with the accumulated TD to make it visible and controllable, and to keep a balance between cost and value of the software project. To our knowledge, their mapping study is the most recent on TDM activities, listing eight activities and the main approaches collected from the studies:

- TD identification: detects TD caused by technical decisions in software, either intentional or unintentional;
- TD measurement: evaluates the cost/benefit relationship of known TD items in software or estimates the overall TD;
- TD prioritization: adopts predefined rules to rank known TD items, to support the decision-making process;
- TD prevention: establishes practices to avoid potential TD from being incurred;
- TD monitoring: observes the evolution of known TD items over time;
- TD repayment: eliminates or reduces the TD impact (principal and interest) in a software system;
- TD representation/documentation: represents and codes TD in a pre-defined standard, to address the stakeholders' concerns;
- TD communication: disclose the identified TD to the stakeholders.

While searching for technologies to support TDM (Silva et al. 2018a), it was possible to observe that some studies are discussing and proposing different technologies, either approaches, tools, or techniques. **Table 1** presents some of the leading technologies identified in the literature to support the management of TD grouped by TDM activity.

3 Related works

Klinger et al. (2011) interviewed four software architects at IBM to obtain insights on how the organization perceives and manages TD. All four architects stated that the debt could incur unintentionally, showing up in the projects through, for example, acquisition, new alignment requirements, or changes in the market ecosystem. They claimed that unintentional debt is usually more problematic than intentional.

They also affirmed that the decision-making process on TDM is often informal and ad-hoc. Finally, the interviewees claimed that there was a gap between executive and technical stakeholders, indicating a lack of a channel or common vocabulary to explain the TD to non-technical stakeholders. Lim et al. (2012) conducted interviews with 35 practitioners with diverse industry experiences from the USA. The authors aimed to understand how TD manifested in software projects and to determine which practitioners adopted TD types in the industry. They also investigated the causes, symptoms, and effects of TD, and finally, they questioned how practitioners deal with TD. Seventy-five percent of the interviewees were not familiar with the TD metaphor. The participants described TD as tradeoffs between a short-term gain and an additional long-term effort. They affirmed that the effects of TD were not all negative, as the tradeoff depended on the product's value. Although they wanted a way to measure the TD, they claimed that measuring TD might not be that easy, as its impact is not uniform. Besides, they

the code level only. Among their research questions, they

Table 1. Some technologies to support the management of TD

TDM activity	Technologies and strategies
TD identification	Manual code inspection, SonarQube, CheckStyle, FindBugs (Yli-Huumo et al. 2016); CodeVizard (Zazworka et al. 2013); SonarQube, Understand, CPPCheck, FindBugs, Sloccount (Ernst et al. 2015).
TD documentation /representation	TD template (Seaman and Guo 2011); TD backlog/list, Documentation practice, JIRA, Wiki, TD template (Yli-Huumo et al. 2016).
TD communication	TD meetings (Yli-Huumo et al. 2016); TD board (Santos et al. 2013); Trello (Oliveira et al. 2015).
TD measurement	SonarQube, JIRA, Wiki; TD evaluation template (Yli-Huumo et al. 2016).
TD prioritization	Cost/benefit model, issue rating (Yli-Huumo et al. 2016).
TD repayment	Redesigning, refactoring, and rewriting (Yli-Huumo et al., 2016).
TD monitoring	SonarQube, JIRA, Wiki (Yli-Huumo et al. 2016); Vtiger and JIRA (Oliveira et al. 2015).
TD prevention	Coding standards, code reviews, Definition of Done (Yli-Huumo et al. 2016).

claimed the key to measure TD is to evaluate the cumulative effect over time. Finally, the authors suggested start managing the TD in an organization through "conducting audits with the entire development team to make technical debt visible and explicit; track it using a Wiki, backlog, or task board."

Ernst et al. (2015) executed a survey with 1,837 participants in three organizations in the United States and Europe. The authors found a "widespread agreement on high-level aspects of the technical debt metaphor, including some popular financial extensions of the metaphor." They also observed that the project context dramatically affects how the practitioners perceive TD. As they stated, only the software architecture was commonly seen as a source of TD, regardless of context. Sixty-five percent of the respondents in this survey report that they adopted only ad-hoc TDM practices in their projects. However, many respondents affirmed that they manage TD through existing practices, such as risk processes or product backlog. Forty-one percent of the participants affirmed not to use any tool for managing TD, while only 16% use tools to identify TD.

Ampatzoglou et al. (2016) conducted a study to understand how practitioners in organizations from the embedded systems domain perceive the TD. They performed an exploratory case study in seven organizations from four different countries. Among other research questions, the authors wanted to find what TD types are more frequently occurring in embedded systems.

Their findings about the most frequent TD types in the practitioners' point of view coincide with the taxonomy proposed by Alves et al. (2016), except regarding design debt, which is considered more relevant to researchers as it is for practitioners; and test debt and code debt, which seems to be more relevant to practitioners. The study did not identify the defect, people, process, service, and usability debts.

Rocha et al. (2017) surveyed with practitioners from BSOs to understand how the TD is dealt with in practice, at investigated which are the factors that lead developers to create TD at the code level, and which practices can prevent developers from creating TD at the code level. Seventy-four practitioners answered the survey, from which almost 72% affirmed to have low, very low or medium knowledge about the TD metaphor. The participants affirmed that developers should follow the best programming practices to help prevent the TD, despite admitting they indeed contribute to creating TD on their projects. Among the main reasons to incur in TD, the participants answered management pressure, tight schedule, developer's inexperience, and work overload. The code review was pointed to as the most relevant practice to prevent the occurrence of TD.

Holvitie et al. (2018) conducted a multi-national survey to observe TD in practice, including practitioners from Finland, Brazil, and New Zealand. The authors opted to focus on practitioners managing TD in organizations adopting agile practices and methodologies. One hundred eighty-four practitioners answered the survey. Approximately 20% of the participants had little to no knowledge on the TD definition. Thirty-five percent of the Brazilian participants were able to provide an example of a TD instance. According to the study, the six leading causes of TD, selected by more than 50% of the participants, are inadequate architecture, structure, tests, and documentation, software complexity and violation of best practices or style guides. Finally, most of the participants perceived refactoring, coding standards, continuous integration, and collective code ownership as having a positive effect on reducing the TD in software projects. Regarding agile software development processes' and process artifacts, iteration reviews/retrospectives, iteration backlog, daily meetings, product backlog, iteration planning meetings, and iterations were all assigned as having a positive impact on reducing TD.

4 Survey design

4.1 Research objectives

Using the goal-question-metric (GQM) paradigm (van Solingen et al., 2002), the objective of this study is to ***analyze*** the TD and its management, ***with the purpose of*** characterizing, ***with respect to*** the level of knowledge and the adopted strategies, activities and technologies, ***from the point of view of*** software practitioners, ***in the context of*** Brazilian software organizations.

4.1.1 Research questions

The research questions are explained as follows:

- **RQ1: Is there a consensus on the perception of TD among software practitioners in BSOs?** It intends to determine whether the perception of TD is homogeneous among professionals in BSOs. If so, it can support the observation of the existence of a common perspective on TD between the industry and academia (a positive side effect of this survey).
- **RQ2: Do the practitioners in the BSOs perceive TD in their software projects?** Before characterizing the TDM activities, it is essential to confirm that the software organizations (through their practitioners) perceive, i.e., observe the presence of TD in their projects.
  - **RQ2.1: Do BSOs manage their TD?** If TD is perceived, it is essential to know whether BSOs manage the TD in their software projects.
    - **RQ2.1.1: What TDM activities are most relevant to software projects?** The goal of this question is to identify, among professionals, which TDM activities, among those proposed by Li et al. (2015), are more relevant, or at least more considered during the software projects.
    - **RQ2.1.2: Which technologies and strategies are adopted for each TDM activity?** For all eight TDM activities proposed by Li et al. (2015), which strategies and technologies are used to support them.

Even though this survey had been designed to identify the most common technologies used by the practitioners in their BSOs to support TDM activities, it is not possible to make any judgment regarding their efficiency and effectiveness. Furthermore, this survey did not look for the benefits of applying such technologies in BSOs.

4.2 Questionnaire design

The questionnaire was designed according to the guidelines presented in Linåker et al. (2015). We performed an ad-hoc literature review to gather specific information about the perception of TD and TDM. Concerning TDM, we organized the activities as proposed by Li et al. (2015), as such activities cover the ones mentioned in different studies. Moreover, we accepted them as consistent since no disagreements were observed during pilot trials.

For each activity, we identified a set of specific strategies and technologies used to conduct the activity, as well as a list of possible roles for each activity. From this information, a questionnaire was designed, and specific questions for each activity were included. For instance, on TD identification, a set of questions involving the TD classification was included – as by Alves et al. (2016).

Regarding the questionnaire structure, it is divided into fourteen sections, described in **Table 2**. It is composed mostly of closed-ended questions. A small number of open-ended questions was necessary to get further information from the participant. It also contains partially closed-ended questions to deal with issues related to tools and strategies for each TDM activity when the given options do not cover the entire possibility of the participant´s answers.

Table 3 presents an extract of our questionnaire translated into English, with some questions on TD identification. Each section starts with a brief explanation of its content and specific instructions. The LimeSurvey platform available in the Experimental Software Engineering (ESE) Group at COPPE/UFRJ (http://lens-ese.cos.ufrj.br/ese/) supported the questionnaire implementation and survey execution.

The questionnaire was configured to ensure the participant's anonymity. A welcome message describes the survey structure and explains its importance for BSOs. The participants are asked to answer the questions based on their current (or most recent) software project and organization. Each set of questions related to a specific TDM activity was conditionally presented to the participants only if they have some experience with that activity to minimize the problem of lengthy survey questionnaires. Other conditional breakpoints in the questionnaire were set to end the survey whether the participant is not familiar with the TD concept and whether the organization or the project do not apply any TDM activity.

4.2.1 Characterization sections

The three sections related to the participant's characterization include questions regarding its role in the projects, academic formation, working experience in software projects, its organization field, size, and any maturity model certificate in software processes.

To assess the size of the organizations, we adopted the SEBRAE/IBGE classification of organizations, consisting in micro (fewer than ten employees), small (between 10 and 49), medium (between 50 and 99) and large (more than 100) organizations. Although this grouping does not constitute a world-level standard, it attends the first necessity for this study, which is a means to estimate the total number of organizations represented by the participants that answered the survey.

Finally, the projects in which the participants work are also characterized, through their domain problem and their lifecycle model. In this last question, the agile software development method was included for simplification purposes, even though it does not characterize as a lifecycle model.

Table 2. Questionnaire sections

Sections	Topic	Description
1	Participant characterization	Obtain personal information regarding the participant, such as professional experience and academic degrees.
2	Organization characterization	Gather information about the organization the participant works for or has worked before.
3	Project characterization	Obtain information about the project considered by the participant in the survey.
4	TD perception	Collect information on the participant's knowledge regarding TD, including what can be considered TD. Also, determine if the organization or the project he works at has strategies for TDM.
5	TDM (general)	Ask the participant which TDM activities are adopted in the working project. Obtain information about the responsibilities and importance associated with each activity from the participant's point of view.
6-13	TDM (activities)	Gather information on several aspects regarding each of the TDM activities proposed in Li et al. (2015).
14	TDM (other)	Provide space for the participant to describe other activities that are executed in the organization.

Table 3. Survey – TD identification section

Question	Answer options
Is there a formal strategy to identify TD?	( ) Yes, we have a formal procedure to identify the TD. ( ) No, the TD identification is executed only informally.
Are all the stakeholders required to apply the TD identification strategy?"	( ) Yes, the strategy is mandatory for all stakeholders. ( ) No, the strategy is considered only a suggestion.
At what point in the project is the TD identified?	( ) There is no defined period; we identify the TD whenever we perceive some issue. ( ) We always identify the TD at the end of each iteration/sprint. ( ) The TD identification is continuous, i.e., occurs throughout the development process.
Mark below all tools or techniques that are used to identify TD.	[ ] Manual coding inspection [ ] Dependency analysis [ ] Checklist [ ] *SonarQube*/*SQALE* [ ] *CheckStyle* [ ] *FindBugs* [ ] *CodeVizard* [ ] *CLIO* [ ] Other (cite which)

4.2.2 TD perception section

This section aims to gather the general participant understanding, regarding the TD definition and its overall aspects. Its first question regards the participant understanding of TD. It was not our purpose to inquiry participants not knowing the meaning of TD, as they can provide wrong answers on the TDM sections. Thus, the participants without TD knowledge should finish their questionnaire in this question.

To the participants that claim they know TD, a follow-up question was designed, to assess which common issues in software development should be considered TD. We presented to the participants a list with items not considered TD (according to the mapping study conducted by Li et al. (2015)), and items considered TD (obtained through an ad-hoc literature review).

Following the general understanding of TD by the participants, they were questioned if TD was perceived in their most recent project, i.e., if they could notice any issues that could be associated with TD. An affirmative answer on this question allows the participants to answer two follow-up questions: if their organization adopts any TDM activity and if their manager (or themselves) adopt any TDM activity, regardless their organization adopting any. Answering "yes" to any of these two questions, allow the answering of the remaining questionnaire.

4.2.3 TD management section

The purpose of this section is to identify the adoption and relevance of TDM practices in the BSOs' projects. The participants were clarified that by "technical debt management," they should consider all activities that organize, monitor, and control the TD and its impacts on software projects.

The participants were asked to select which TDM activities were conducted in their projects, based on the list of TDM activities provided by Li et al. (2015). An additional option was included to provide space for the participants so that they can mention other TDM activities not discussed by Li et al. (2015).

For each TDM activity selected by the participants, an additional question was created to ask which roles were responsible for conducting that activity. The leading roles offered as answers were obtained from Yli-Huumo et al.'s study (2016), but the questionnaire provided an open-ended question so that the participants could elaborate in case another role should be considered responsible for that activity.

4.2.4 TDM activities sections

Eight sections follow the questionnaire, asking for information regarding each one of the TDM activities proposed by Li et al. (2015). They are only available to the participants if they select those activities in the previous section.

At the beginning of each section, the TDM activity is described, to improve the participants' knowledge and reduce the probability of misunderstanding of the proposed questions.

Those sections follow the structure briefly presented below, for each activity. All activities subsections included a question to obtain which tools or techniques were adopted to conduct that particular activity, based mainly on a list obtained mainly from Li et al. (2015) and Yli-Huumo et al. (2016).

- TD identification: The participants were asked about the use of any formal approach to identify TD, as well as their optionality. Next, they were asked when the TD was identified. A subsection was created to assess if and how the TD is classified after it has been identified.
- TD documentation/representation: The participants were asked if there was a standard to follow when documenting TD, and whether it was mandatory to all stakeholders. Then they were asked how the TD items are documented or cataloged.
- TD communication: The participants were only asked how the unresolved TD items were communicated between the project stakeholders.
- TD measurement: The participants were asked if there was any strategy previously defined to measure TD, and how it was measured efficiently. They were asked which information or variables were used to measure the TD items.
- TD prioritization: The participants were asked how the TD is prioritized. Finally, they were asked which criteria are used to support the TD prioritization.
- TD repayment: The participants were asked if there is any planning to repay TD.
- TD monitoring: Like the TD repayment, the participants were asked how the TD is monitored.
- TD prevention: For this section, the participants were asked if there are any formal practices conducted to prevent the TD and whether they are mandatory or optional to the stakeholders.
- Other TDM activities: One last section is provided to gather information regarding other TDM activities used in the participant's software organization, presenting similar questions to the previous sections.

4.3 Pilot execution

A pilot trial was conducted using the same artifacts and procedures designed for the final survey, including the survey questionnaire and the execution method, but with a small number of participants from the target population (Linåker et al. 2015).

Seven practitioners were invited to the pilot trials. Five of them work on software projects and come from the ESE Group at COPPE/UFRJ, which conducts this research. The other two participants also work on software projects but are from outside the research group. All of them have some prior experience with TD and/or TDM, mostly in the industry.

An invitation by e-mail included the main instructions and questionnaire link. They were asked to answer the questionnaire and return their feedback regarding response time, proper understanding, completeness, and other aspects.

All pilot participants answered the pilot survey within a week. The average answering time was 15.2 minutes. The relevant comments were associated with usability issues, clarity of questions, and some suggestions to improve some details and definitions throughout the questionnaire. These were later discussed internally, and modifications were applied to the final questionnaire. Overall, we did not observe negative comments or doubts about either the answer options or the questions descriptions, suggesting that the questionnaire was good enough to use in the study.

4.4 Target population and sampling

To achieve the research objectives and to answer the research questions, the practitioners from BSOs were selected as the target audience. The sampling design adopted is accidental, a non-probabilistic type of sampling, i.e., we cannot observe randomness on the selected units from the population. This decision can incur in a threat to validity, which will be further discussed in Section 6.3. An invitation to answer the survey was sent to a series of renowned software development groups in the country. Other invitations were sent through the LinkedIn professional social network.

Finally, the survey was disclosed to the participants on three software-related events: RioInfo (practitioners oriented) and SBQS (high participation from practitioners), in Rio de Janeiro/Brazil, and CBSoft (some practitioners participation), in Fortaleza/Brazil.

4.5 Final revisions and survey release

After the pilot trial, the final survey was released on June 2017. A lab package with the research plan and the survey questionnaire is available in English and Portuguese at https://doi.org/10.6084/m9.figshare.5923969.

5 Survey results

The survey was conducted between June 2017 and April 2018. In total, 62 participants answered the survey, with 36 complete answers. Four participants did not complete the survey but reached the questionnaire's section 4 (TD perception), so they were included in this initial analysis, totalizing 40 valid answers. The remaining 22 incomplete responses were not included in the analysis. **Figure 2** summarizes the survey responses.

5.1 Participants' characterization

The respondents have an average work experience of 14.15 years in software projects. Only four respondents reported having an incomplete undergraduate degree, while the remaining 36 respondents hold at least an undergraduate degree. Twenty-six participants reported holding a specialization degree (master or doctorate).

Regarding the BSOs in which the respondents work, most of them (23) are from the IT sector. Referring to the project development, most of them (35) adopt agile or incremental lifecycle models. Two projects adopt the spiral model, while three adopt the waterfall model. Due to the questionnaire anonymity, it is not possible to precise the number of organizations represented in the survey. However, it is possible to estimate it roughly, based on the information provided by the participants in this section. Therefore, we could estimate around 12 organizations and 30 projects included in the survey.

Only one organization characterized by the participants adopt de MPS.BR maturity model to evaluate its software processes, at level G, whereas two participants affirmed the organizations they work have CMMI level 5 and two others have CMMI level 2.

5.2 TD awareness and TD perception

Regarding the perception of TD, from 40 valid answers, we found that 16 respondents (40%) claimed not to be aware of the TD metaphor. The 24 remaining participants (60%) were asked to select the options that best matched the TD definition. Two of them did not answer this question. As it can be observed in **Table 4**, seven issues out of 12 were marked by 50% or more of the 22 participants: low internal quality, poorly written code, that violates code rules; "shortcuts" taken during design; the presence of known defects that were not eliminated; architectural problems; planned but unfinished or unplanned tasks; and issues associated with low external quality.

Regarding the TD perception, from the 24 participants that were aware of the TD meaning, 17 informed to perceive some issues associated with the TD concept in their projects, whereas four did not perceive the TD occurrence, and one participant did not answer this particular question.

From the 17 participants that informed that perceived the TD occurrence in their projects, ten answered that their organizations or the project managers adopt TDM activities. **Table 5** presents the distribution of these answers, grouped by the organization size. **Table 6** presents the same results among organizations that adopt any model to evaluate their maturity level on software processes.

5.3 TD management

From the eight TDM activities proposed by Li et al. (2015), as shown in **Figure 2**, only TD monitoring was not marked by the participants when asked about which TDM activities were conducted in their projects. TD identification and TD documentation are conducted in projects for six participants each, while five participants each marked TD prioritization, TD communication, and TD repayment. TD measurement and TD prevention are conducted in projects according to two participants. One participant did not mention any TDM activities. No participants mentioned any TDM activity besides those proposed by Li et al. (2015). **Table 7** presents the grouping of the results according to the sizes of the organizations.

5.3.1 TDM responsibilities

There was no consensus among participants on which roles should be responsible for each TDM activity. Moreover, some distinction was observed between the participants' responses and the TDM framework presented in Yli-Huumo et al.'s study (2016). For instance, the TDM framework presented in Yli-Huumo et al. (2016) states that software architects and the team leader are the responsible roles for the activity of TD measurement. However, in our survey, no respondent selected software architects as responsible for this activity. On the other hand, it was also identified that some responsible roles to perform some TDM activities are similar to results pointed in Yli-Huumo et al. (2016), for example, software architect and development team to perform TD identification. Therefore, we consider the findings concerning TDM responsibilities are coherent and complementary to presented in Yli-Huumo et al. (2016).

Table 8 presents our results, compared with the ones from Yli-Huumo et al. (2016).

5.3.2 TD identification

Two out of six participants answered that there is a mandatory strategy to conduct the TD identification activities, while one participant answered that there is a formal strategy, albeit not mandatory. Three participants claimed to adopt only simple strategies. Three out of six answers suggested that TD identification was conducted continuously throughout the project. Regarding TD identification, one participant affirmed that the TD was classified as Design Debt or Documentation Debt in the project, while one participant claimed to use the artifact that initially incurred the TD to classify it.

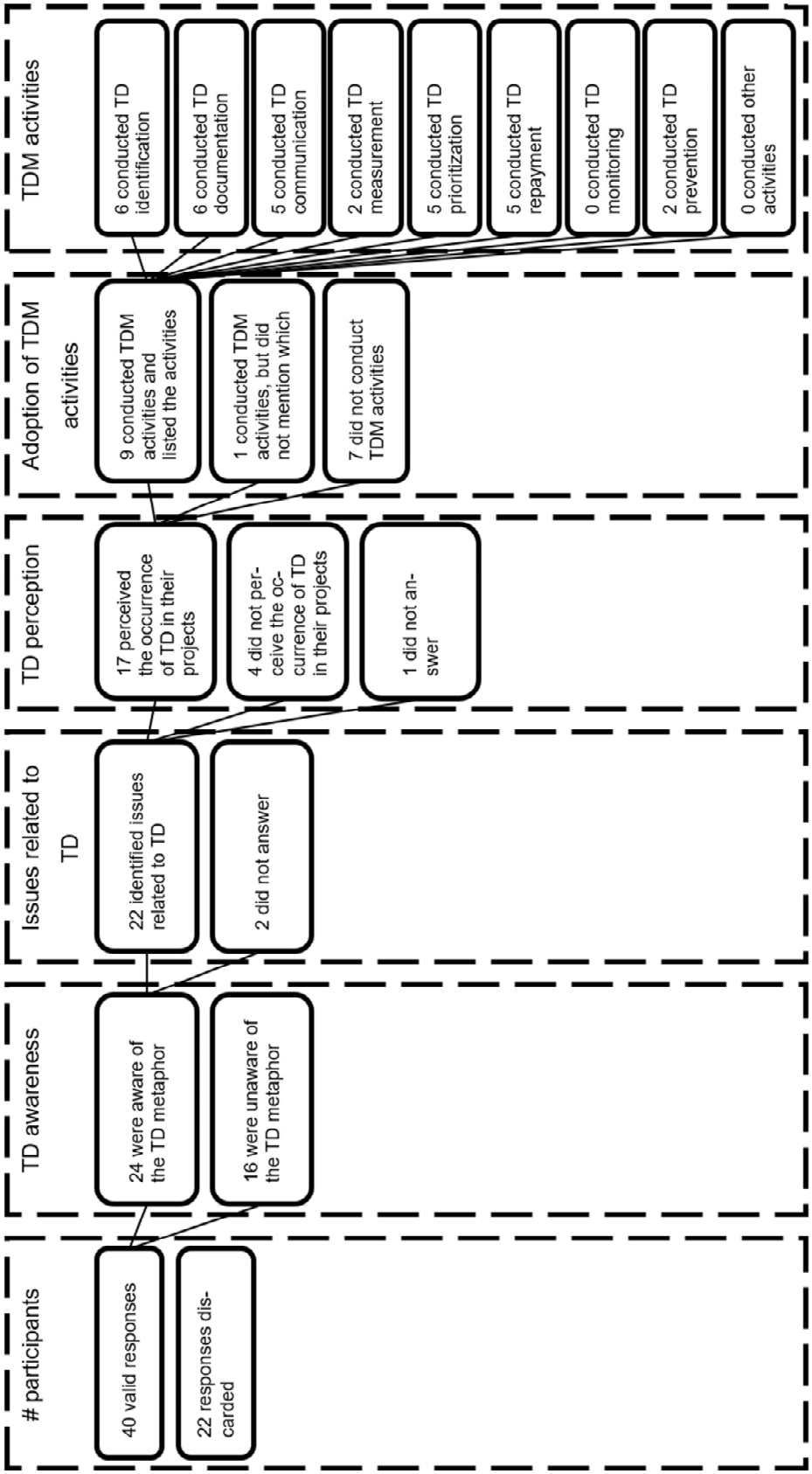

Figure 2. Summary of the survey responses

Table 4. Issues related to TD, according to the participants

Issue	% of participants
Low internal quality aspects, such as maintainability and reusability	77%
Poorly written code that violates code rules	68%
"Shortcuts" taken during design	68%
Presence of known defects that were not corrected	68%
Architectural problems (like modularity violation)	55%
Low external quality aspects, such as usability and efficiency	50%
Planned, but not performed, or unfinished, tasks (e.g., models, test plans, etc.)	50%
Trivial code that does not violate code rules	45%
Code smells	45%
Defects	36%
Lack of support processes to the project activities	23%
Required, but unimplemented, features	18%

Table 5. TD perception, grouped by organization size

Yes No No answer	Knows TD?	Perceived TD?	Adopts any TDM activity?	Total
Fewer than 10 employees	Yes: 1	Yes: 1	Yes: 0	3
	No: 2	No: 0	No: 1	
		No answer: 2	No answer: 2	
Between 10 and 49 employees	Yes: 5	Yes: 3	Yes: 3	11
	No: 6	No: 0	No: 0	
		No answer: 8	No answer: 8	
More than 100 employees	Yes: 18	Yes: 13	Yes: 7	26
	No: 8	No: 4	No: 6	
		No answer: 9	No answer: 13	
Total	40	40	40	40

5.3.3 TD documentation

From the total of six participants, two answered that they have a standard on documenting the TD that should be followed by all stakeholders. One participant answered that his/her project has a TD documentation standard, but it is not mandatory for the stakeholders. Two participants answered that the TD documentation is conducted only informally.

When asked how the TD is documented, four participants answered that they use a general task backlog, with no specific details, while one affirmed she uses a specific backlog of TD items.

One participant did not provide any details on TD documentation, despite informing that it is conducted in her project.

5.3.4 TD communication

From the five participants that answered the TD communication section, four affirmed that the TD was discussed during project meetings, but with the participation of only a few of the necessary stakeholders. One participant said that the TD was only discussed informally.

5.3.5 TD measurement

Out of the two participants that answered the TD measurement section, one affirmed that the TD measurement was conducted informally, through the analysis of metrics and indicators based on specific information regarding the TD item. The other participant indicated that there is a mandatory strategy to measure TD, based on direct information, like person-hours to repay the TD item or the item LOC.

5.3.6 TD prioritization

Regarding the TD prioritization, three of five participants answered that the TD items were prioritized according to "guesses" or simplified estimative based on previous experiences, while the other one used the TD item criticality to prioritize it. Four participants affirmed that they tend to prioritize the TD items that most impact the client, and three answered that they prioritize the TD items that could cause the most impact on the project. One did not provide any details on TD prioritization, despite adopting it in his/her project.

Table 6. TD perception, grouped by the adoption of maturity models to evaluate software processes

Yes No No answer	**Knows TD?**	**Perceived TD?**	**Adopts any TDM activity?**	**Total**
MPS.BR level G	Yes: 1	Yes: 0	Yes: 0	1
	No: 0	No: 1	No: 0	
		No answer: 0	No answer: 1	
CMMI level 2	Yes: 2	Yes: 2	Yes: 0	2
	No: 0	No: 0	No: 2	
		No answer: 0	No answer: 0	
CMMI level 5	Yes: 0	Yes: 0	Yes: 0	2
	No: 2	No: 0	No: 0	
		No answer: 2	No answer: 2	
Others	Yes: 2	Yes: 1	Yes: 1	4
	No: 2	No: 0	No: 0	
		No answer: 3	No answer: 3	
Total	9	9	9	9

Table 7. TDM activities conducted in the participants' projects, grouped by organization size

TDM activity	**Between 10 and 49 employees**	**More than 100 employees**	**Total**
Identification	2	4	6
Documentation/ Representation	3	3	6
Communication	1	4	5
Measurement	0	2	2
Prioritization	2	3	5
Repayment	2	3	5
Monitoring	0	0	0
Prevention	1	1	2
Identification	2	4	6
Documentation/ Representation	3	3	6

5.3.7 TD repayment

From the five participants that answered the TD repayment section, two of them answered that the TD repayment is planned according to the current project necessities, while one answered that the TD repayment is planned continuously, with specific periods during the development process destined to this activity. One participant answered that the TD is only repaid when it is not possible to avoid it anymore. One participant did not provide any details on TD repayment, despite informing that it is conducted in his/her project.

5.3.8 TD prevention

Both respondents answering the TD prevention section mentioned that it is an activity conducted only by each member of the team individually.

5.3.9 Technologies and strategies for TDM

Table 9 presents a list of practices, techniques, and tools used in each TDM activity. The numbers in parentheses represent the number of participants answering that specific section (column "TDM activity") and the number of participants that affirmed using that tool or technique (column "Tools and techniques"). We can observe that different technologies support TDM, and there is no consensus about which one to use. Most of such technologies are similar to those identified in the technical literature (see **Table 1**).

6 Discussion

6.1 Revisiting the findings

The analysis of the survey's results, presented in Section 5, allowed us to answer reasonably the RQs, which we discuss next.

6.1.1 RQ1: Consensus on the perception of TD

We did not observe consensus in the overall TD perception. Each participant was asked to select which of the 12 issues suggested on TD should be associated with the TD concept, as presented in **Table 4**. Out of those options, 75% or more

Table 8. TDM responsibilities

TDM activity	Our study	Yli-Huumo et al. (2016)
Identification	Team leader; Software architect; Development team	Software architect; Development team
Documentation/ Representation	Project manager; Team leader; Software architect; Development team	Software architect; Development team
Communication	Team leader; Software architect; Development team	Project manager; Software architect; Development team
Measurement	Team leader; Development team	Software architect; Development team
Prioritization	Project manager; Team leader; Software architect; Development team	Project manager; Software architect
Repayment	Team leader; Software architect; Development team	Software architect; Development team
Prevention	Project manager; Team leader; Software architect; Development team	Software architect; Development team

Table 9. TDM activities – technologies and strategies

TDM activity	Technologies and strategies
TD identification (6)	Manual code inspection (4), dependency analysis (1), checklist (2), *SonarQube*[1]/*SQALE*[2] (3), *CheckStyle* (1), *FindBugs*[3] (1)
TD documentation /representation (6)	TD backlog (3), specific artifacts for TD documentation (1), *JIRA*[4] (1), others - *Trello* (1)
TD communication (5)	Discussion forums (3), specific meetings about TD (1), others - *GitLab* (1), others - *Trello*[5] (1)
TD measurement (2)	Manual measurement (1), *SonarQube* (2), *JIRA* (1)
TD prioritization (5)	Cost/benefit analysis (1), classification of issues (3)
TD repayment (5)	Refactoring (3), redesign (1), code rewriting (4), meetings/workshops/training (1)
TD monitoring (0)	N/A
TD prevention (2)	Guidelines (2), coding standards (2), code revisions (1), retrospective meetings (1), Definition of Done (2)

of the 22 respondents evaluated only one issue. From the issues associated with the TD concept by 50% or more of participant's, only one ("issues associated with low external quality") is not considered TD.

Only one issue associated with the TD concept was marked by less than 50% of the 22 answers, which is "Code smells" (42%). Therefore, although we could not find consensus between the industry and academia, we consider that there is some agreement among participants of what should be considered TD since 17 out of 22 participants identified that TD should be related to internal quality issues. However, 50% of the participants believe that TD should also be associated with external quality issues, which is worrisome and contradicts the definition asserted at the Dagstuhl Seminar (Avgeriou et al. 2016). It could indicate that there is a misconception of what should be considered TD, associating its definition with any issue occurring during the software development.

We could observe some alignments in the views on TD between the participants and academia since most of the issues associated by more than half of the participants are also

[1] https://www.sonarqube.org
[2] http://www.sqale.org/
[3] http://findbugs.sourceforge.net/
[4] https://br.atlassian.com/software/jira
[5] https://trello.com/

in agreement with the definition indicated in the technical literature. We believe that despite the reasonable TD definition understanding by some software practitioners, it is vital to disseminate better the distinction between issues related to internal quality (TD) and those related to external quality (defects).

6.1.2 RQ2: BSOs' practitioner's perception of TD

Only 43% of 40 participants claimed to perceive TD in their software projects, which could be considered low, given the importance of the topic. Moreover, only 25% of the 40 participants adopt TDM activities, possibly indicating the existence of a severe gap in the overall product quality perspective. However, we did not assess the adoption of other internal quality assurance methods to replace the low perception of the TD presence in the organizations.

Grouping the TD perception with the size of the organizations (**Table 5**), we could observe that a higher percentage of participants from larger organizations know about TD when compared to smaller-sized organizations. Most of the participants from these companies also answered that they perceive TD in their projects. It could indicate that more prominent organizations (generally being active for a more extended period, and having more solid processes to manage software development projects) could have a broader perspective on TD and TDM.

Unfortunately, due to the low number of responses, we could not analyze the correlation of the adoption of maturity models to evaluate software processes with any aspect of the study. Out of the nine participants that answered their organizations adopt any maturity models, six of them did not know what TD is, or they did not perceive it in their last projects. From the remaining three that did perceive in their projects, only one conducted any activities to manage it. This gap in the results can be used to develop the research on TD in BSOs further.

6.1.3 RQ2.1.1: Most relevant TDM activities

The results of our survey show that there was no consensus on which TDM activities are more relevant to surveyed software projects. However, almost half of the participants that answered this question mentioned that TD prevention is relevant to a project. It is a possible research gap for future works since most of the studies regarding TDM focus on TD identification, measurement, and prioritization. Regarding the main TDM activities conducted by the participants, our results are mostly in line with Yli-Huumo et al. (2016) in which indicates that TD communication is most commonly adopted by the development teams, followed by TD identification, documentation, prioritization, repayment, and prevention. The rarely managed TD activities described in Yli-Huumo et al.'s study (2016) are TD measurement and TD monitoring, as also observed in our study. Despite the number of participants indicating the importance of TD prevention, only two reported performing TD prevention activities.

6.1.4 RQ2.1.2: Technologies and strategies

As presented in section 5.3.6, a list of tools and technologies used to manage TD activities (see **Table 9**) can be used in further studies looking for evidence on their effectiveness and efficiency in managing the TD.

6.2 Comparison with results from related works

Most of the studies previously described in section 3 have distinct populations, being researches from other countries. However, we could observe that their results are coherent and complementary to the findings of our survey, as we discussed in sections 5.3.1 and 5.3.9. When analyzing the results concerning the TD understanding or TD perception in the projects, it is possible to observe that most of the surveyed software practitioners reported having a low level of knowledge about TD.

Regarding the management of TD, we could identify that it still seems to be incipient in the surveyed software organizations in such studies. Most of the studies reported that TDM activities are performed in an informal and ad-hoc way. Although some strategies and technologies identified by Holvitie et al. (2018) to support the TDM activities are coherent and complementary to those identified in our survey (see Section 5.3.9), the evidence on their effectiveness and efficiency in managing the TD must be further investigated. Besides, as previously mentioned, some distinction and similarities were identified regarding which roles should be responsible for each TDM activity (see Section 5.3.1).

6.3 Threats to validity

This research has some threats to validity as any other empirical study. Next, we report them together with some of the adopted mitigation actions, relying on the classification as proposed by Wohlin et al. (2012) and Linåker et al. (2015).

A potential internal threat comes from the participants that might have misunderstood some terms and concepts of the questionnaire. There is also a construct threat of a biased survey, from the researchers' perspectives and the collected information from the technical literature such as the TDM activities organized in Li et al. (2015). To reduce the level of this menace, we conducted three revision cycles during the survey development with two researchers. Furthermore, two pilot trials were executed, followed by a final revision by all the pilot survey participants aiming to ensure the modifications were aligned with their perspectives. We also observed a potential threat in the way that the main topic of the survey was disclosed to the potential participants in the invitations. If the participants did not have previous knowledge regarding the topic, this could have driven them away from the survey, which could have biased the results. We recognize that this effect of the invitations on the participants could have affected the study in some way.

We observed an external validity threat concerned with the representativeness and high mortality of surveys' respondents. As part of our disclosure strategy involved presenting the research in software engineering research events, some of our results might present some bias. There is a high rate of

mortality of respondents since a substantial number of responses were discarded. Only 65% of the 62 responses were valid to the point we could obtain some information. These discarded responses refer to 22 incomplete questionnaires, in which its respondents did not reach the questionnaire's section 4 (TD perception) so they could not be included in the analysis. Perhaps the reason for incomplete questionnaires might be associated with the survey length and the response time. Overall, the survey has 52 questions (no participant had to answer the complete survey, though), distributed over 23 pages. Studies report that every additional question can reduce the response rate by 0.5%, and every additional page, by 5% (Linåker et al. 2015). However, since we do not have data on this possibility, we cannot formulate any elaborate conclusions.

Another possible reason for the low number of responses is that the concept of TD is still incipient in BSOs and, since the topic was explicitly mentioned in the invitations, it could have kept away some practitioners that are not familiar with the term. If this case is indeed real, the results would be even more worrisome, as the percentage of practitioners that know what TD is could drastically drop. Considering the initial number of survey's participants, only ten them reported adopting TDM activities in their projects. However, it is essential to highlight that the participants might not be so representative among those who manage TD in the BSOs. On the other hand, the practitioners surveyed may be a good sample of how the TD concept is perceived in the BSOs, in which it is also an interesting result. This result may indicate that TD concept still needs to be further disseminated for the software industry. In this sense, maybe the dissemination about TD concept and aspects concerned with its management can occur at the university courses level or even at the professional training level. Even among those practitioners who responded to the complete survey, we could observe a level of misconception point out that the TD perception is not in line with the Dagstuhl's definition.

Finally, the main threat to validity is the generalization of the results. Since the target sampling is non-probabilistic, it is not possible to determine a priori the population size and the expected total number of participants. Therefore, the results confidence level might be low, making it hard to generalize the results to the entire population (BSOs). As argued by Mello et al. (2015), the establishment of representative samples for SE surveys is considered a challenge, and the specialized literature often presents some limitations regarding the interpreting surveys' results, mainly due to the use of sampling frames established by convenience and non-probabilistic criteria for sampling from them. As previously, methodological procedures were used since the planning stage of our study until its execution, aiming to reduce the level of such menace.

Despite that, the inevitable conclusions can suggest the TD research with initial indications of the level of knowledge of BSOs regarding the TD concept and TDM activities.

7 Concluding remarks

This paper presented background about the TD definition and the results of a survey conducted with practitioners in BSOs. The results provide initial observations regarding how BSOs (represented by their software practitioners) perceive and manage TD in their projects.

Before the analysis of the survey results, some observations can be made. First, we obtained a considerable low number of responses and an even lower number of complete responses. Notwithstanding, the results were enough to provide an initial and representative picture of the perception of TD and its management in the scenario of the BSOs.

Regarding the TD perception, our results indicate no unanimity concerning how Brazilian software practitioners perceive TD. Regarding TDM, it was observed that only a few BSOs report the TD management in their software projects, indicating that TDM seems to be still incipient in BSOs. Four out of nine practitioners reporting TDM activities claimed that TD prevention is the most critical activity in their projects, despite only two participants indicated to perform it.

We believe that the results of this study provide the following contributions to both industry and academia:

- To the BSOs (industry) - the initial results indicate that software practitioners and their organizations need to understand better the concept of TD. It is necessary to achieve better results in their projects since the perception of TD and its management in this scenario is still incipient. The findings also present a list of technologies that can be used to support TDM activities, as long as software engineers evaluate their usage based on the organizations' needs at the time. Moreover, the findings indicate that TDM activities usually involve distinct roles throughout the projects. In general, we consider that the BSOs need to systematize some actions (e.g., training) to enable their teams to perceive and manage existing TD.
- To the researchers - our results indicate that there is a need for more investigations aiming to disseminate the TD knowledge to practitioners on BSOs, as well as provide strategies and software technologies to support the TDM on these organizations. Besides, we believe the sharing of this study package can contribute to support the development of investigations on TD and its management more connected to the software organization needs in Brazil and other regions.

Following this study, we conducted two other works that compose a research framework on TD and its management. Both were reported in Silva et al. (2018a). The first work consisted of a quasi-systematic literature review to gather available technologies that support TDM in the technical literature. The second work organized Evidence Briefings (Cartaxo et al., 2016) in both English and Portuguese, combining the survey and the literature review results. The evidence briefings intend to address critical points observed in the survey research, primarily regarding the practitioners' general lack of knowledge or misconceptions concerning

TD. They are available online at https://doi.org/10.6084/m9.figshare.7011281.

Overall, we believe this study offered a new perspective on the TD research in BSOs. To the best of our knowledge, only one other survey analyzed the TD specifically in BSOs (Rocha et al. 2017), but the authors focused mainly on the TD located at the code level, not using a broader software engineering perspective like our study. Moreover, our survey package provides the materials that can be used by other software engineering researchers to study this topic in other organizations and software communities, facilitating a better understanding, future comparisons, and providing indications to evolve TDM activities.

Acknowledgements

The authors thank all the professionals that took part in this survey and the researchers that have collaborated with their feedback on the pilot trials.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Prof. Travassos is a CNPq researcher and an ISERN member.

References

Alves NSR., Mendes TS. de, De Mendonça MG., et al. (2016) Identification and management of technical debt: A systematic mapping study. Inf Softw Technol 70:100–121. doi: 10.1016/j.infsof.2015.10.008.

Ampatzoglou A, Ampatzoglou A, Chatzigeorgiou A, et al. (2016) The Perception of Technical Debt in the Embedded Systems Domain: An Industrial Case Study. Proc - 2016 IEEE 8th Int Work Manag Tech Debt, MTD 2016 9–16. doi: 10.1109/MTD.2016.8.

Assuncao TR de, Rodrigues I, Venson E, et al. (2015) Technical Debt Management in the Brazilian Federal Administration. 2015 6th Brazilian Work Agil Methods 6–9. doi: 10.1109/WBMA.2015.11.

Avgeriou P, Kruchten P, Ozkaya I, et al. (2016) Managing Technical Debt in Software Engineering Edited by. Dagstuhl Reports 6:110–138. doi: 10.4230/DagRep.6.4.110.

Becker C, Chitchyan R, Betz S, McCord C (2018) Trade-off decisions across time in technical debt management. 85–94. doi: 10.1145/3194164.3194171.

Boehm B (2008) Making a Software Century.

Brown N., Cai Y., Guo Y., et al. (2010) Managing technical debt in software-reliant systems. In: Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research, FoSER 2010. pp 47–51.

Cartaxo B, Pinto G, Vieira E, Soares S (2016) Evidence Briefings: Towards a Medium to Transfer Knowledge from Systematic Reviews to Practitioners. pp 1–10.

Cunningham W (1993) The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger 4:29–30. doi: 10.1145/157710.157715.

de França BBN, Jeronimo H, Travassos GH (2016) Characterizing DevOps by Hearing Multiple Voices. Proc 30th Brazilian Symp Softw Eng - SBES '16 53–62. doi: 10.1145/2973839.2973845.

Ernst NA., Bellomo S., Ozkaya I., et al. (2015) Measure it? Manage it? Ignore it? Software practitioners and technical debt. In: 2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings. pp 50–60.

Fowler M (2009) Technical Debt Quadrant. In: Martin-Fowler.com. https://martinfowler.com/bliki/TechnicalDebtQuadrant.html.

Guo Y., Spínola RO. c, Seaman C. (2016) Exploring the costs of technical debt management – a case study. Empir Softw Eng 21:159–182. doi: 10.1007/s10664-014-9351-7.

Holvitie J, Licorish S, Spinola R, et al. (2018) Technical Debt and Agile Software Development Practices and Processes: An Industry Practitioner Survey. Inf Softw Technol 96:141–160.

Klinger T, Tarr P, Wagstrom P, Williams C (2011) An enterprise perspective on technical debt. In: Proceedings - International Conference on Software Engineering. pp 35–38.

Ktata O, Lévesque G (2010) Designing and implementing a measurement program for scrum teams: What do agile developers really need and want? In: ACM International Conference Proceeding Series. pp 101–107.

Li Z, Avgeriou P, Liang P (2015) A systematic mapping study on technical debt and its management. J Syst Softw 101:193–220. doi: 10.1016/j.jss.2014.12.027.

Lim E., Taksande N., Seaman C. (2012) A balancing act: What software practitioners have to say about technical debt. IEEE Softw 29:22–27. doi: 10.1109/MS.2012.130.

Linåker J, Sulaman S, Maiani R, Höst M (2015) Guidelines for Conducting Surveys in Software Engineering Engineering.

McConnell S (2007) Technical Debt - 10x Software Development.

Oliveira F., Goldman A., Santos V. (2015) Managing Technical Debt in Software Projects Using Scrum: An Action Research. In: Proceedings - 2015 Agile Conference, Agile 2015. pp 50–59.

Prikladnicki R, Audy JLN, Damian D, De Oliveira TC (2007) Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring. Proc - Int Conf Glob Softw Eng ICGSE 2007 262–274. doi: 10.1109/ICGSE.2007.19.

Ribeiro LF. b, De Farias MAF. d, Mendonça M., Spínola RO. e (2016) Decision criteria for the payment of technical debt in software projects: A systematic mapping study. In: ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems. pp 572–579.

Rocha JC, Zapalowski V, Nunes I (2017) Understanding Technical Debt at the Code Level from the Perspective of Software Developers. Proc 31st Brazilian Symp Softw Eng - SBES'17 64–73. doi: 10.1145/3131151.3131164.

Santos PSM, Varella A, Dantas C (2013) Visualizing and Managing Technical Debt in Agile Development: An Experience Report.

Seaman C., Guo Y. (2011) Measuring and Monitoring Technical Debt. Adv Comput 82:25–46. doi: 10.1016/B978-0-12-385512-1.00002-5.

Silva VM, Junior HJ, Travassos GH (2018a) Technical Debt Management in Brazilian Software Organizations: A Need, an Expectation, or a Fact? In: Brazilian Symposium on Software Quality (SBQS). Curitiba.

Silva VM, Junior HJ, Travassos GH (2018b) A taste of the software industry perception of technical debt and its management in Brazil. Av en Ing Softw a Niv Iberoam CIbSE 2018 1–14.

Spínola RO., Vetrò A., Zazworka N., et al. (2013) Investigating technical debt folklore: Shedding some light on technical debt opinion. In: 2013 4th International Workshop on Managing Technical Debt, MTD 2013 - Proceedings. pp 1–7.

Tom E, Aurum A, Vidgen R (2013) An exploration of technical debt. J Syst Softw 86 1498–1516. doi: 10.1016/j.jss.2012.12.052.

van Solingen R, Basili V, Caldiera G, Rombach HD (2002) Goal Question Metric (GQM) Approach. Encycl. Softw. Eng.

Wohlin C, Runeson P, Höst M, et al. (2012) Experimentation in Software Engineering. Springer Science & Business Media, Heidelberg, Berlin.

Yli-Huumo J., Maglyas A., Smolander K. (2016) How do software development teams manage technical debt? - An empirical study. J Syst Softw. doi: 10.1016/j.jss.2016.05.018.

Zazworka N. e, Spínola RO., Vetro A., et al. (2013) A case study on effectively identifying technical debt. In: ACM International Conference Proceeding Series. pp 42–47.