Journal of Software Engineering Research and Development, 2023, 11:12, doi: doi.org/10.5753/jserd.2023.3191

Investigating the Point of View of Project Management Practitioners on Technical Debt - A Study on Stack Exchange

Felipe Gomes [**Federal University of Bahia** | ***felipe.gustavo@ufba.br***]
Eder Santos [**Federal University of Bahia** | ***edersantos@ufba.br***]
Sávio Freire [**Federal Institute of Ceará** | ***savio.freire@ifce.edu.br***]
Thiago Souto Mendes [**Federal Institute of Bahia** | ***thiagosouto@ifba.edu.br***]
Manoel Mendonça [**Federal University of Bahia** | ***manoel.mendonca@ufba.br***]
Rodrigo Spínola [**Virginia Commonwealth University** | ***spinolaro@vcu.edu***]

Context: Technical debt (TD) can bring short-term benefits to software projects, but its presence is associated with issues such as decreasing product quality. Recent literature has proposed various approaches for identifying and managing TD, but most of them focus on the software developers' point of view. Little is known about how project management practitioners perceive and manage TD items. **Goal:** This work aims to investigate how project management practitioners discuss and experience TD. **Method**: To achieve this goal, our work mines, curates, and selects a total of 108 TD-related discussions on the Stack Exchange Project Management (SEPM) Q&A site. These discussions amount to 547 posts and 882 text comments on the subject. We analyze this data set quantitatively and qualitatively, using open coding to derive TD types, indicators, and management practices. **Results**: We identified 74 indicators used for recognizing debt items and 126 TD management practices. The types of debt most discussed at SEPM are process and people debts. This contradicts studies done with developers where code and design debts are most discussed. **Conclusion**: The perspective considered by project management practitioners to analyze the TD phenomenon is different from the one considered by other roles in the software development process. Our work organizes the identified TD indicators and management practices into a Sankey diagram, which may assist TD management practitioners and serve as guidance for future research on the subject.

Keywords: *Technical debt, Project management practitioners, Stack Exchange*

1 Introduction

Technical debt (TD) contextualizes the effects of immature software artifacts that bring short-term benefits in terms of increased productivity and lower costs, but which may need to be adjusted later with interest during the software life cycle (Kruchten et al., 2012). TD is commonly incurred when development teams have to choose between carrying out their activities following quality standards or delivering products in the shortest possible time, using minimum resources (Rios et al., 2019). Even though TD mostly affects the later phases of software development, it can be incurred and spread in all phases of a project (Alves et al., 2016). Different types of debt may occur during this process, and the development practices used in an organization can affect their presence (Li et al., 2015).

While acquiring TD may offer strategic benefits to a software project, it is necessary to properly manage it to mitigate its risks and avoid significant losses in the long run (McConnell, 2008; Klinger et al., 2011; Kruchten et al., 2012). **TD management (TDM)** seeks to balance short-term and long-term goals, supporting development teams to decide on the need and the best time to eliminate the debt (Freire et al., 2023).

TDM has become a subject of interest to practitioners in software engineering forums and this work aims to investigate how practitioners experience and manage TD based on discussions they post in Question and Answer (Q&A) platforms.

Q&A platforms, such as Stack Overflow, are part of daily activities of modern software development and this has risen the interest of software researchers (Garousi et al., 2019). They are a rich source of information because countless issues are discussed in them, bringing to light the point of view of practitioners on possible solutions for those issues. Moreover, those platforms usually count on a self-moderating system for the discussions and users, making the available information more reliable for practitioners (and researchers).

Stack Overflow (SO) has emerged as the most prominent example of a repository for practitioners' knowledge of software development. Several studies have analyzed its discussions regarding software engineering topics such as code smells and antipatterns (Tahir et al., 2020), topics and trends (Barua et al., 2014), people soft skills (Montandon et al., 2021), and mobile software development (Rosen and Shihab, 2016).

Following this trend, researchers have also studied TD based on Stack Overflow discussions. Digkas et al. (2019) investigated the impact that code reused from SO had on TD. Gama et al. (2019, 2020) investigated TD indicators and types of debt based on SO discussions. Our work follows the same trend as this later work. However, we seek to take a project management point of view in our analyses.

Stack Overflow is a generic platform mostly accessed by software developers. There are other platforms focused on more specialized topics. One of these platforms is **Stack Exchange Project Management (SEPM)**, which encompasses discussions of practitioners interested in software project management.

Our work has been focusing on SEPM discussions as a counter point to SO discussions. We seek to investigate how project management practitioners commonly discuss, experience, and manage TD, identifying what TDM practices are used and which TD indicators are faced by this type of practitioner in their projects.

Our preliminary results were published in two conference papers (Gomes et al., 2022; dos Santos et al., 2022). Those results pointed out that project management practitioners are most concerned with people and process debt, contrasting with code, architecture, test and infrastructure debts identified by Gama et al. (2019, 2020).

This paper extends the previous conference papers by:

- analyzing a larger data set of SEPM discussions.
- producing a comprehensive list of TDM practices used to eliminate or mitigate TD items.
- analyzing the relationships between TD indicators and TDM practices.

Our work mines, curates, and selects a set of 108 discussions, composed of 547 posts and 882 comments from SEPM. Each discussion is analyzed quantitatively and qualitatively. For each, we seek to identify the types of TD being addressed, their indicators, and the recommended practices used to manage them. Besides that, we also identified, and related, roles, artifacts and practices affected by TD.

Our results show that project management practitioners are usually concerned with people and process debt. We found 74 TD indicators and 126 management practices. The most common TD indicators are related to planning and management (mentioned 87 times) and people (mentioned 36 times), for example, *"Low morale"* (mentioned 15 times), and *"The agile practices are not respected"* (mentioned 14 times).

The end of this paper provides a Sankey diagram that shows the most common relationships between TD indicators and TDM practices. To visualize those relationships is valuable for both practitioners and researchers. Managers can apply this information to identify and decide how to address TD based on the experience of others. Researchers can use those relations to understand how practitioners experience and address TD, helping to guide research efforts regarding this subject.

The remainder of this paper is organized into seven other sections. Section 2 describes background concepts related to TD and related work. Section 3 presents the research method. Section 4 presents the result and discussion of the quantitative and qualitative analyses. Section 5 shows the relations of TD indicators with TDM practices and TD types. Section 6 discuss the implications of our results to practitioners and researchers. Section 7 presents the threats to the study validity. Lastly, Section 8 presents our final remarks.

2 Literature Review

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 (Kruchten et al., 2012).

Alves et al. (2016), through a systematic mapping study, identified 15 different types of debt that could affect software development and their relation with their respective indicators. The focus of this work is on nine types of debt that were identified during the analysis. According to Alves et al. (2016), these types are among the most common ones in the area:

- **Process Debt**: Refers to inefficient processes, for example, the defined or used process may not be the most appropriate for the organization's current activities, which consequently may hinder the software development activities (Codabux and Williams, 2013);
- **People Debt:** Refers to people issues that, if present in the software organization, can delay or hinder some development activities. An example of this kind of debt is expertise concentrated in too few people, as an effect of delayed training or hiring (C. and R., 2013);
- **Documentation Debt:** Refers to problems found in the project documentation such as missing, inadequate, or incomplete documentation of any type (Guo and Seaman, 2011);
- **Requirement Debt:** Refers to trade-offs made concerning what requirements the development team needs to implement or how to implement them. Examples of this type of debt are partially implemented requirements or implementations that don't fully satisfy a nonfunctional requirement (Kruchten et al., 2012);
- **Testing Debt:** Refers to issues found in testing activities that can affect the quality of those activities. Examples of this type of debt are planned tests that were not run, or known deficiencies in the test suite (e.g. low code coverage) (Guo and Seaman, 2011).
- **Design Debt:** Refers to debt that can be discovered by analyzing the source code and identifying violations of the principles of good object-oriented design (e.g. very large or tightly coupled classes) (Guo and Seaman, 2011; Izurieta et al., 2012);
- **Code Debt:** Refers to problems found in the source code that can negatively affect the legibility. Usually, this debt can be identified by examining the source code for issues related to bad coding practices (Bohnet and Döllner, 2011);
- **Defect Debt:** Refers to known defects, usually identified by testing activities or by the user and reported on bug tracking systems, that the Configuration Control Board (CCB) agrees should be fixed but, due to competing priorities and limited resources, have to be addressed at a later time (Snipes et al., 2012);
- **Versioning Debt**: Refers to source code versioning issues, such as unnecessary use of forks, delays from merges in a software project, or not appropriate tools to maintain development history (Greening, 2013);

As project management activities are crucial in decision-making concerning TD management (Freire et al., 2020a,b, 2023), it is necessary to investigate the influence of project management practitioners' perception on the execution of TDM activities. Also, the inherent context resulting from the process model followed by managers can influence their decisions on TDM (Rios et al., 2021; Berenguer et al., 2023). The technical literature has investigated these topics.

Behutiye et al. (2017) performed a systematic literature review to analyze and synthesize the state of the art of TD in the agile software development (ASD) context. The results indicated five research areas of interest for TD in ASD. TD management in ASD received the highest attention, followed by architecture in ASD and its relationship with TD. Their findings indicate the need for more tools, models, and guidelines that support TD management in ASD. Moreover, there is a potential research gap for standardized approaches to managing TD. Examining the relationship between the causes and consequences of TD in ASD will assist in uncovering potential TDM strategies that address TD-related issues.

Holvitie et al. (2017) carried out a multi-national survey questionnaire to classify the effects of the adoption of agile methodologies in TD management. By analyzing 184 responses from practitioners in Brazil, Finland, and New Zealand, the study indicated that: 1) Practitioners are aware of TD, although, there was underutilization of the concept, 2) TD commonly resides in legacy systems, however, concrete instances of TD are hard to conceptualize which makes it problematic to manage, 3) Queried agile practices and processes help to reduce TD; in particular, techniques that verify and maintain the structure and clarity of implemented artifacts (e.g., Coding standards and Refactoring) positively affect TD management. The fact that TD instances tend to have characteristics in common means that a systematic approach to its management is feasible. However, notwithstanding the positive effects of some agile practices on TD management, competing stakeholders' interests remain a concern.

Rios et al. (2018) identified several TDM strategies, most of which still require further investigation and empirical evaluation, as few empirical studies have been performed in real settings. This is an indicator that, for some areas, we still do not fully understand the costs or benefits of the proposed TDM strategies. In addition, the authors identified a list of situations in which debt items can be found in software projects and organized a map representing the state of the art of activities, strategies, and tools to support TDM. The authors also reported several gaps that need to be addressed when dealing with TDM. The existing limitations, such as the lack of tools and strategies to support some activities and the lack of comprehensive solutions that consider a management process for TD as a whole, can make TDM difficult to perform.

In another work, Rios et al. (2019) presented the most common causes and effects of TD in ASD, i.e., the reasons that lead software teams to incur debt and the pain that developers suffer because of its presence in agile software projects, respectively.

Recently, researchers have been using (Q&A) platforms to empirically understand how practitioners have been managing TD. Digkas et al. (2019) conducted an empirical study on the relation between the existence of reusing code retrieved from Stack Overflow on the TD of the target system. The results provide insights into the potential impact of small-scale code reuse on TD and highlight the benefits of assessing code quality before committing changes to a repository.

In another work, Gama et al. (2020) investigated the point of view of practitioners on Stack Overflow on how developers commonly identify TD items in their projects. They reported that developers normally discuss TD identification, revealing 29 different low-level indicators for recognizing code, infrastructure, architecture, and test debt items. Our study follows a similar approach, but it analyzes a forum focused on project management. Besides that, we also analyzed: the recommended TDM practices, the agile artifacts, roles, and events affected by TD, and the relationship between the TD indicators and TDM practices.

Conjecturing that the perspective of project management practitioners to analyze the TD phenomenon is different from the other roles in the software development process, we performed a study to investigate, from the point of view of project management practitioners, how they commonly discuss, experience, and manage TD on SEPM (Gomes et al., 2022). The results show that the most commonly discussed types of debt are process and people, revealing 47 indicators for recognizing debt items and 72 practices related to TD management. This previous work was limited in scope and did not consider the specifics inherent to each software development process.

In another study, we focused on how software practitioners (mostly software managers) involved in agile software development discuss TDM on SEPM (dos Santos et al., 2022). The results once again show that the most commonly discussed types of TD are process and people debt. In addition, the Product Owner and Development Team are the most important roles concerning ASD-TD. Sprint Backlog and Sprint Planning are the agile elements most affected by ASD-TD.

The results reported in this article are an extension of the works presented in Gomes et al. (2022) and dos Santos et al. (2022). Compared to our last work (dos Santos et al., 2022), we increased the data set from 79 to 108 discussions. Our work now considers agile and non-agile discussions. More importantly, we now include the use of TDM practices in the investigation. Our analyzes now relate the TD indicators to the practices used to manage technical debt in software engineering projects.

3 Research Method

This section presents the research questions along with the data collection and analysis procedures we performed in our investigation.

3.1 Research Questions

As previously stated, this work aims to investigate how project management practitioners experience and manage TD. To this end, we seek answers to the following main research questions (RQs):

- **RQ1:** *What TD indicators do project management practitioners face during their projects?* This question seeks to identify which TD indicators are being discussed by SEPM users.
- **RQ2:** *What types of TD most concern project management practitioners?* This question aims to identify the different types of TD discussed in SEPM according to the classification presented by Rios et al. (2018).

- **RQ3:** *What TDM practices have been applied to mitigate or eliminate debt items?* The purpose of this question is to identify the TD management activities discussed by SEPM users to mitigate or eliminate TD.

On top of those RQs, we did realize that most of the TD discussions in SEPM were grounded on agile software development (ASD) practices. We then decide to expand the original set of questions with the following secondary research questions:

- **RQ4:** *Which agile artifacts are most affected by TD?* This question intends to identify which ASD artifacts suffer consequences from the accumulation of TD based on SEPM discussions.
- **RQ5** *Which agile events are harmed by TD?* This question aims to identify which events are negatively impacted by TD based on SEPM discussions.
- **RQ6:** *What agile roles are involved with the consequences of TD?* The purpose of this question is to identify which agile roles suffer complications from TD according to SEPM discussions.

As expected, research questions **RQ4**, **RQ5**, and **RQ6** are only answered by the discussions related to ASD-TD. The following sections explain how we considered a discussion as being related to ASD-TD.

3.2 Data Collection

This study used the Stack Exchange data dump[1] (version 09/07/2021), which is the raw data dump of all Stack Exchange websites. From this repository, one can access data from any Stack Exchange website, including posts (both questions and answers), comments, and metadata.

The discussions on Stack Exchange websites are composed of three main components: the question around which the discussion is centered, the answers to the question, and the comments on both questions and answers. Comments are presented below the post they are related to. Thus, they are part of the discussions. Our work included the comments in its analyses because they provide helpful information relevant to identifying TD indicators and management practices, such as scenario context. Examining comments is particularly relevant when considering discussions on Stack Exchange websites because comments go beyond the discussion, clarifying and enriching the content conveyed through questions and answers (Sengupta and Haythornthwaite, 2020).

Our study searched the Stack Exchange Project Management (SEPM) content for the strings "debt" or "shortcut" in the questions (title, body, and tags), answers' body, and comments' body. The term "debt" catches the different forms of reference to TD, for example, "tech debt" and "code debt". By performing pilot studies we, also detected that the term "shortcut" was used to refer to TD in the discussions. The term "shortcut" catches other references to sub-optimal solutions.

[1]`https://archive.org/details/stackexchange`

We choose such general terms because of the free nature of discussion forums. The forum user can refer to TD in several ways and, unlike formal literature, there is no standard. Some users refer to TD as "tech debt", "development shortcut", "delivery shortcut", or just "debt". To decide which strings to use, we tested some strings like "technical debt", "code debt" and other terms used in the literature, but the number of returned discussions was very low. Therefore, after the tests, we decided to use the terms "debt" and "shortcut" because they encompassed the discussions found using the other terms and returned a higher number of discussions.

The string-matching phase yielded a total of 263 discussions. Next, we filtered the data as shown in Figure 1, using the following criteria:

- **Step 1:** Eliminate incomplete discussions from the data set. We considered a discussion complete when a question was followed by one or more answers, where there was at least one answer whose author differed from the question's author. After applying this criterion, 224 discussions remained out of the initial 263.
- **Step 2:** Eliminate untrustworthy discussions. Other studies have found that data from Q&A forums can be affected by noise (Ahasanuzzaman et al., 2016; Kavaler et al., 2013), requiring mitigating this noise using different proxies (Gama et al., 2020). We decided to use the discussion score as a filtering proxy. A post score is a Stack Exchange popularity metric in which users, other than the post author, can give an up-vote to the post if they find it useful or a down-vote if they find it not useful. A discussion score is the difference between up-votes and down-votes of all its posts. We decided to filter out discussions with negative scores, since overall they are not considered useful. While we did not discard any discussion in this step, we considered it important since can help to ensure the quality of the discussions.
- **Step 3: Qualitative data analysis.** In the third step, we conducted a qualitative data analysis of the data set. Each of the 224 discussions went through a thorough analysis process, as described in the next section.

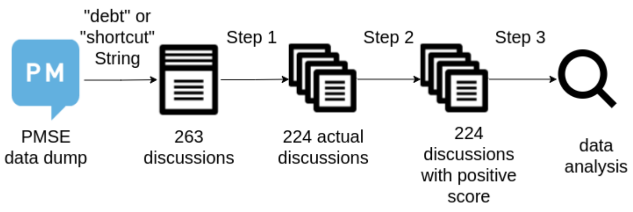


Figure 1. Data extraction and filtering process.

3.3 Data Analysis

We performed manual qualitative analysis (Seaman, 1999) over the data set. The analysis was composed of three steps shown in Figure 2. In **step 1**, the first two authors of this paper analyzed each discussion independently. For each discussion, individually, they filled in the set of questions presented in Table 1, reporting the types of debt, TD indicators, TDM practices, agile roles, artifacts, and events affected by the TD presented in each discussion. Alongside each question in Table 1, there are also the RQs that it helps to answer.

Table 1. Questions to gather TD information in SEPM discussions.

Questions (Q)
Q1. What types of TD are discussed? **(RQ2)**
Q2. What are the discussed indicators of TD? **(RQ1)**
Q3. What are the activities and strategies that were discussed to support TD management and identification? **(RQ3)**
Q4. Which agile artifacts are most affected by TD? **(RQ4)**
Q5. Which agile events are harmed by TD? **(RQ5)**
Q6. What roles are involved with the consequences of TD? **(RQ6)**

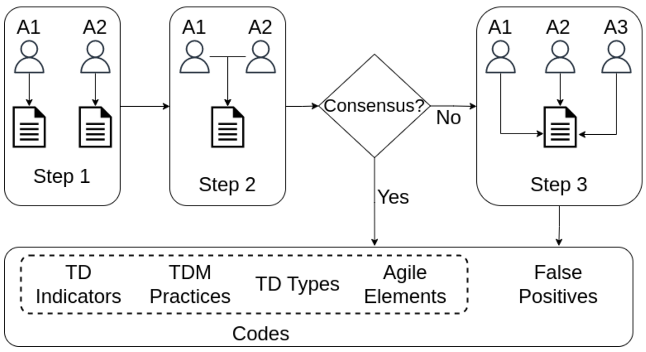


Figure 2. Discussions analysis procedure.

In this information-gathering effort, **Q1** captures the type of debt discussed by SEPM's users, according to the concepts presented by Rios et al. (2018). **Q2** captures the TD indicator reported by SEPM users to recognize TD items. **Q3** catches TDM practices proposed by SEPM users to address the identified TD items. In case a discussion was related to ASD-TD, **Q4**, **Q5**, and **Q6** capture, respectively, the agile artifacts, events, and roles present in the discussion. These elements were derived from the Scrum (Schwaber and Sutherland, 2011) and eXtreme Programming (XP) practices (Agile_Alliance, 1999), and the terminology adopted in that methodologies (Partogi, 2018). We chose to follow the agile implementation defined by these methodologies because they are the most widely used by practitioners (Ai, 2021). We did not find mentions of other agile methodologies in the analyzed discussions.

While going through the information-gathering questions (**Step 1**), we also looked for false positives, taking into consideration the following rules:

- **Rule 1:** *The discussion must be related to TD.* Discussions that, in spite of having the terms *"debt"* or *"shortcut"*, did not discuss TD were marked as false positives.
- **Rule 2:** *The discussed situation must be real.* This work intends to map real problems faced by practitioners, so questions asking for advice without bringing any actual situation from the present or past were flagged as false positives.
- **Rule 3:** *The TD indicators must come from the question's author.* Since the question's author is the one with actual knowledge about the situation, we only considered the author's words concerning TD indicators. TD indicators inferred by other users were considered only if sustained by the question's author in a comment or a post in the same discussion.
- **Rule 4:** *The recommended TDM practices needed to be backed by the community*. We intend to map practices viewed as good by the community, thus, we only considered practices recommended in answers or comments that were backed by other users than the recommendation's author. Therefore, practices needed to be accepted by at least one other user quantitatively (giving a positive score) or qualitatively (giving a favorable opinion) to be considered.

The combination of the rules defined above and the questions presented in Table 1 allowed us to verify whether a discussion was within the scope of information gathering or was a false positive. After removing all false positives, our final dataset was reduced to 108 discussions containing 547 posts and 882 comments. Of those, 81 discussions were related to ASD-TD (436 posts and 691 comments), and 27 were unrelated to agile (111 posts and 191 comments).

During **step 1**, the discussions were examined, seeking information that would make it possible to answer the questions presented in Table 1. This information, when found, was recorded according to the perception of each researcher, without any previously established codes. The researchers were encouraged to freely record information regarding TD, adding as much information as they deemed necessary to answer the questions in Table 1. We decided to use this approach because the analysis required a fair amount of understanding and interpretation, and we wanted to keep the initial data acquisition as simple as possible. This approach was considered most applicable given the lack of previous analysis of the discussions and the nature of the discussions themselves.

The exception to the above-described procedure was using predefined codes for *TD types*. To identify the type of debt addressed in the discussions, we employed the definitions reported by Rios et al. (2018). As an example, let us consider the following discussion excerpt: *"I suspect the team is either interviewing or ramping up their skills to prepare for their next job. How do I motivate them to restore their velocity?"*. This discussion occurs in the context of problems that might happen due to overloaded individuals, characterizing a discussion regarding people debt, as defined in (Rios et al., 2018).

We used **step 2** to examine the responses recorded individually in the first step and reach a consensus between the researchers. Together, the two researchers checked the information extracted from the discussions and in the case of analysis agreement, they also performed the coding process. Discussions that yielded divergent information were subjected to a second examination in **step 3**, this time with a third researcher. A majority vote among the researchers would define which information should remain for the following analysis step. This process was repeated until all responses were consolidated.

For each discussion, we created terms to represent and group the TD indicators that were addressed in various ways. For example, the following TD indicator *"I've been appointed as a Scrum Master to a new team and senior management (CTO) expectation is that our team should deliver more than the SP capacity we can currently realistically deliver."* was coded to *"The agile practices are not respected."*. Whenever we found TD indicators related to more than one TD type, we considered them associated with all TD types. For example, the indicator *"The agile practices are not respected."* is at the same time related to process and people debt. The relation to process debt is due to the process that lets the problem occur, and the relation to people debt is due to the practitioners deciding not to follow the agile practices.

Once again, disagreements in the coding were resolved with the help of the third researcher. This process was performed until no new codes were identified (point of saturation). Then, this encoded data set was added to the results. Besides that, we also performed a grouping of the TDM practices following the same approach, by a consensus of two researchers and using a third researcher to help with disagreements.

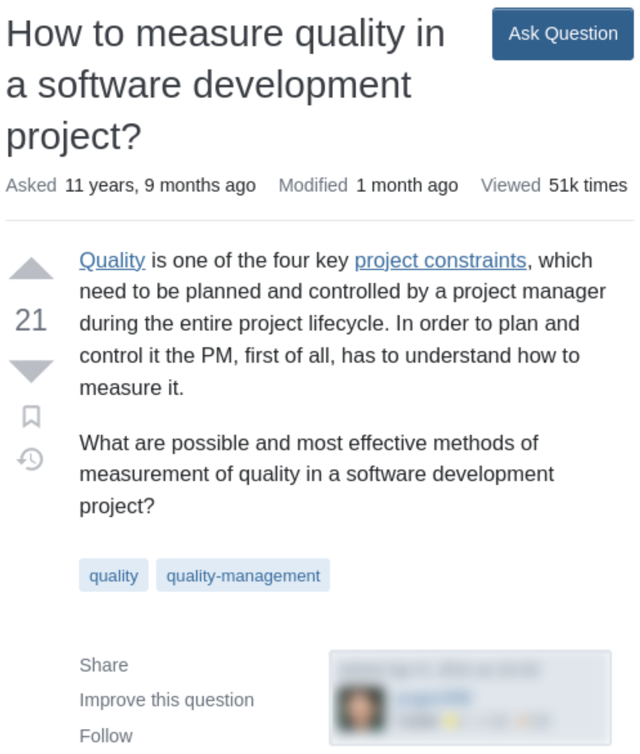
Next, we present five examples, one from a discussion marked as a false positive, Figure 3, and four from TD discussions and comments of interest, Figures 4–7.

As discussed before, we use the questions in Table 1 and the previously listed rules to verify whether a discussion was a false positive or not. Discussions with the terms "debt" or "shortcut" but unrelated to TD or did not bring a real-world problem were considered false positives. Figure 3 illustrates this situation. The author is asking for advice about measuring quality in a software project, due to his/her inexperience in software engineering, without bringing any real-world situation where TD indicators could be detected. Since this work is trying to map TD indicators and TDM practices to mitigate them, any discussion that did not fit the structure of "TD-related problems and practices to mitigate these problems" was considered a false positive.

An example of an analysis of a discussion of interest is shown in Figure 4. Parts of the discussion have been removed to preserve space. In this example, we can see the structure of a discussion: the question that has a title, a body, and a score by the side; the answers that have a body and a score; and the comments that have a body, and also a score, which follow the same principle as the post score, but comments can only be up-voted. It is shown only comments attached to one answer due to space limitations. Comments can be attached to the question as well.

Based on the question's title, we marked the discussion in Figure 4 as process and defect debt. Corroborating with the marked TD types, the question's body gives more context to the problem, helping us to identify two TD indicators related to process and defect debt, *"Bugs occurrence."* and *"The agile principles are not respected."*. From the question's content, we can assume that what is harming the development productivity is the management decisions.

The answers in Figure 4 indicate three TDM practices that can be applied to mitigate the problem under discussion. The TDM practices revolve around changes in the development process to address the TD. The recommended TDM practices in the discussion were: *"Educate team members about agile*


How to measure quality in a software development project?

Asked 11 years, 9 months ago Modified 1 month ago Viewed 51k times

21

Quality is one of the four key project constraints, which need to be planned and controlled by a project manager during the entire project lifecycle. In order to plan and control it the PM, first of all, has to understand how to measure it.

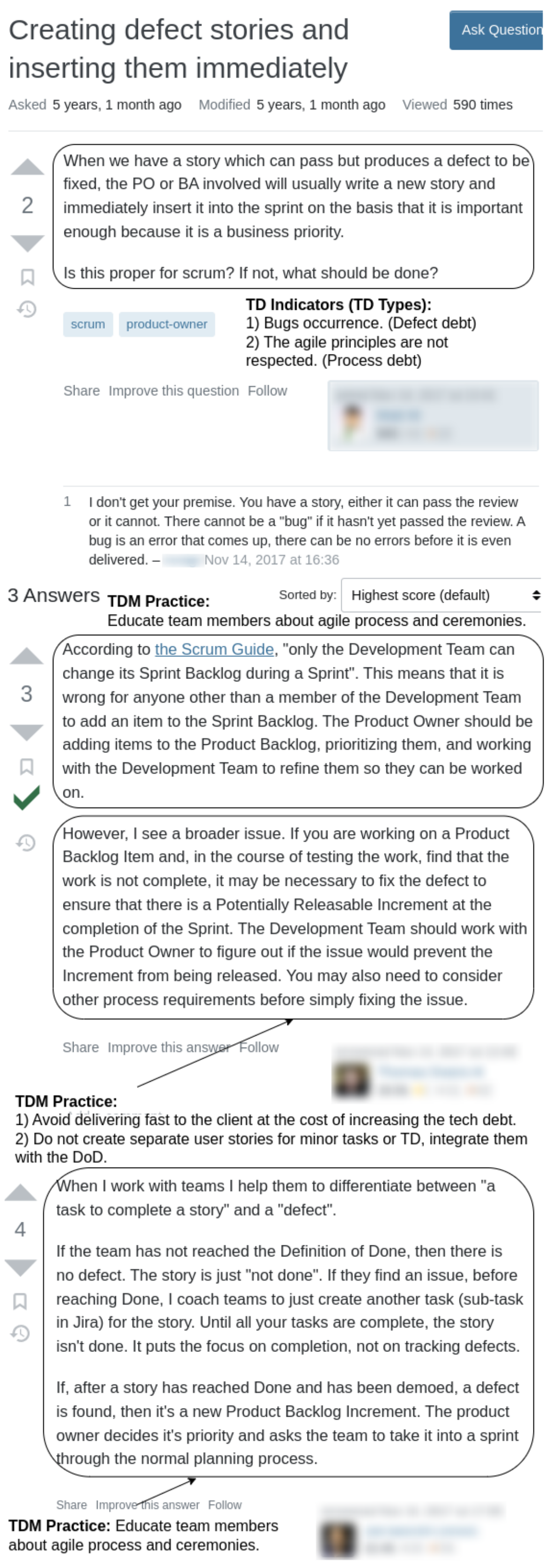
What are possible and most effective methods of measurement of quality in a software development project?

Figure 3. Example of false positive discussion.

process and ceremonies.", *"Avoid delivering fast to the client at the cost of increasing the tech debt."* and *"Do not create separate user stories for minor tasks or TD, integrate them with the DoD."*. This discussion is an example of the "TD-related problems and practices to mitigate these problems" structure we were looking for during the analysis. Through it, we can trace the relationship between the TD indicators and TDM practices.

Regarding the usefulness of comments in the analysis. Figure 5 shows a comment attached to a question. Based on the comment provided by the question author (highlighted in the figure), we can see that one of the TD indicators faced by the practitioner is *"Rushed development."*. Besides that, this question is an example of a TD-related discussion, despite not having the term 'debt' in its body. It shows that it is relevant to consider various terms related to TD when searching in Q&A forums. Usually, practitioners can bring the subject using the terms they know, which can vary from the ones used in the literature.

Figure 6 shows an example of an ASD-TD discussion. From the analysis of the question's title, we marked the discussion as belonging to the ASD context. The *"agile"* and *"scrum"* tags attached to the question confirm the ASD context. The question's body helped us to understand better the situation experienced and identify the problem under discussion. We detect the following TD indicators: *"Poor performance."*, *"The TD cause is neglected."* and *"Rushed development."*. These indicators pointed out a process debt. We also observed the presence of the terms *"story points"* and *"sprint process"*, which is common to the ASD domain, and reaffirms the ASD-TD context. Regarding the affected agile artifact and event, we considered, respectively, *Sprint Backlog* and *Sprint Planning* because wrong estimations can harm them both and lead to misplanning. As for roles, we considered *Development Team* and *Product owner* because mis-


Creating defect stories and inserting them immediately

Ask Question

Asked 5 years, 1 month ago Modified 5 years, 1 month ago Viewed 590 times

2

When we have a story which can pass but produces a defect to be fixed, the PO or BA involved will usually write a new story and immediately insert it into the sprint on the basis that it is important enough because it is a business priority.

Is this proper for scrum? If not, what should be done?

scrum product-owner

TD Indicators (TD Types):
1) Bugs occurrence. (Defect debt)
2) The agile principles are not respected. (Process debt)

Share Improve this question Follow [redacted]

1 I don't get your premise. You have a story, either it can pass the review or it cannot. There cannot be a "bug" if it hasn't yet passed the review. A bug is an error that comes up, there can be no errors before it is even delivered. – [redacted] Nov 14, 2017 at 16:36

3 Answers Sorted by: Highest score (default)

TDM Practice:
Educate team members about agile process and ceremonies.

3

According to the Scrum Guide, "only the Development Team can change its Sprint Backlog during a Sprint". This means that it is wrong for anyone other than a member of the Development Team to add an item to the Sprint Backlog. The Product Owner should be adding items to the Product Backlog, prioritizing them, and working with the Development Team to refine them so they can be worked on.

However, I see a broader issue. If you are working on a Product Backlog Item and, in the course of testing the work, find that the work is not complete, it may be necessary to fix the defect to ensure that there is a Potentially Releasable Increment at the completion of the Sprint. The Development Team should work with the Product Owner to figure out if the issue would prevent the Increment from being released. You may also need to consider other process requirements before simply fixing the issue.

Share Improve this answer Follow [redacted]

TDM Practice:
1) Avoid delivering fast to the client at the cost of increasing the tech debt.
2) Do not create separate user stories for minor tasks or TD, integrate them with the DoD.

4

When I work with teams I help them to differentiate between "a task to complete a story" and a "defect".

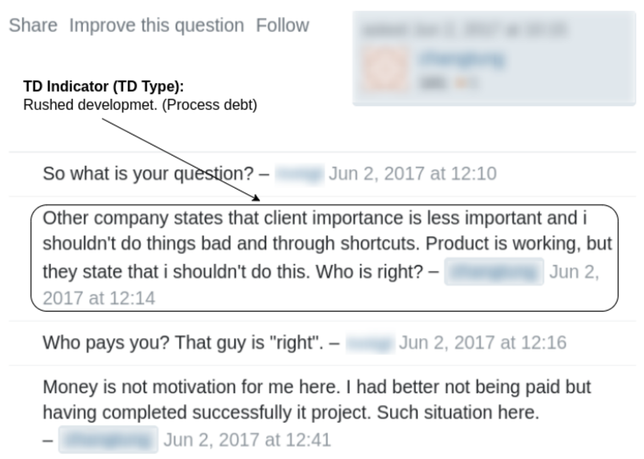
If the team has not reached the Definition of Done, then there is no defect. The story is just "not done". If they find an issue, before reaching Done, I coach teams to just create another task (sub-task in Jira) for the story. Until all your tasks are complete, the story isn't done. It puts the focus on completion, not on tracking defects.

If, after a story has reached Done and has been demoed, a defect is found, then it's a new Product Backlog Increment. The product owner decides it's priority and asks the team to take it into a sprint through the normal planning process.

Share Improve this answer Follow [redacted]

TDM Practice: Educate team members about agile process and ceremonies.

Figure 4. Example of a valid discussion.


Share Improve this question Follow [redacted]

TD Indicator (TD Type):
Rushed developmet. (Process debt)

So what is your question? – [redacted] Jun 2, 2017 at 12:10

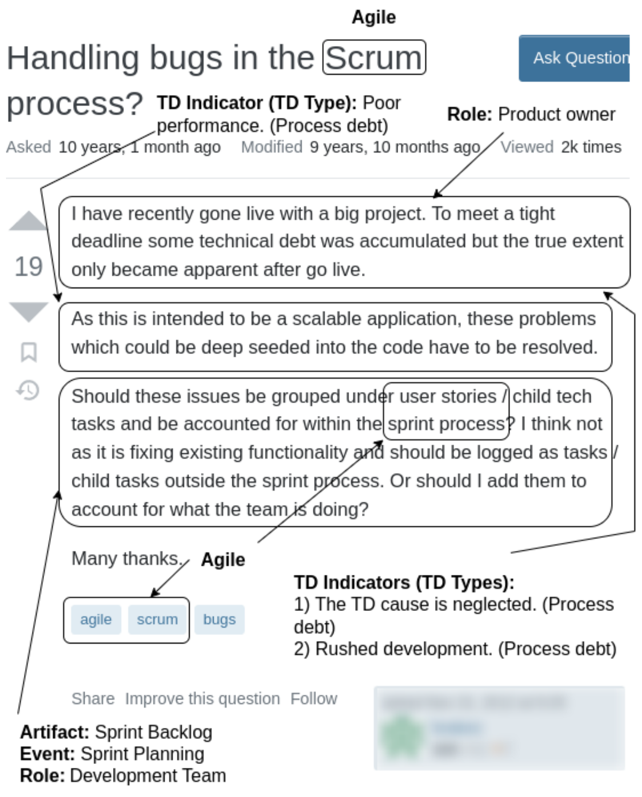
Other company states that client importance is less important and i shouldn't do things bad and through shortcuts. Product is working, but they state that i shouldn't do this. Who is right? – [redacted] Jun 2, 2017 at 12:14

Who pays you? That guy is "right". – [redacted] Jun 2, 2017 at 12:16

Money is not motivation for me here. I had better not being paid but having completed successfully it project. Such situation here. – [redacted] Jun 2, 2017 at 12:41

Figure 5. Example of comment analysis.

planning can lead to developer's overwork to meet what was planned on the due dates, which may harm the quality of the product delivered to the client, with bugs and unfinished stories, and reduce the developers' morale.


Agile

Handling bugs in the Scrum process?

Ask Question

TD Indicator (TD Type): Poor performance. (Process debt)

Role: Product owner

Asked 10 years, 1 month ago Modified 9 years, 10 months ago Viewed 2k times

19

I have recently gone live with a big project. To meet a tight deadline some technical debt was accumulated but the true extent only became apparent after go live.

As this is intended to be a scalable application, these problems which could be deep seeded into the code have to be resolved.

Should these issues be grouped under user stories / child tech tasks and be accounted for within the sprint process? I think not as it is fixing existing functionality and should be logged as tasks / child tasks outside the sprint process. Or should I add them to account for what the team is doing?

Many thanks, **Agile**

agile scrum bugs

TD Indicators (TD Types):
1) The TD cause is neglected. (Process debt)
2) Rushed development. (Process debt)

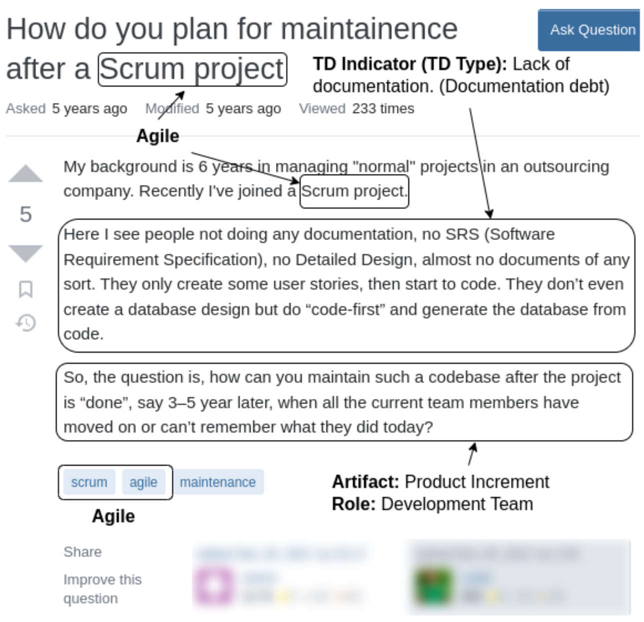
Share Improve this question Follow [redacted]

Artifact: Sprint Backlog
Event: Sprint Planning
Role: Development Team

Figure 6. Example of an ASD-TD valid discussion.

Figure 7 shows yet another example of an ASD-TD discussion. The question's body gives context to the problem, helping us to identify one TD indicator related to documentation debt (*"Lack of documentation"*). The title and the tags (*"scrum"* and *"agile"*) are a clear indication of an ASD-TD. Regarding the affected agile artifact, we considered *"Product Increment"* because the accumulation of TD can harm the internal quality and reduce the project's maintainability. Therefore for the role, we considered *"Development Team"* because more effort is required to deliver tasks due to the TD, leading to overwork and low morale. As for events, we could not find any events affected by ASD-TD in this discussion.

It is worth mentioning that the extraction of the TD-related



How do you plan for maintainence after a Scrum project

Asked 5 years ago Modified 5 years ago Viewed 233 times

My background is 6 years in managing "normal" projects in an outsourcing company. Recently I've joined a Scrum project.

Here I see people not doing any documentation, no SRS (Software Requirement Specification), no Detailed Design, almost no documents of any sort. They only create some user stories, then start to code. They don't even create a database design but do "code-first" and generate the database from code.

So, the question is, how can you maintain such a codebase after the project is "done", say 3–5 year later, when all the current team members have moved on or can't remember what they did today?

Ask Question
TD Indicator (TD Type): Lack of documentation. (Documentation debt)
Agile
5
scrum
agile
maintenance
Agile
Artifact: Product Increment
Role: Development Team
Share
Improve this question

Figure 7. Yet Another Example of an ASD-TD valid discussion.

data was not a straightforward task. We needed to interpret the posts to perform the data extraction and coding. This required us to reread posts several times to clear any doubts about their content. We also consulted the links to other web pages, such as articles and tutorials. Take the following discussion excerpt as an example: *"Here is a good article I wrote on my experience with the topic: I solved the Agile testing bottleneck problem! link[2]"*. There was also the necessity to understand the jargon used by the practitioners, such as *"low hanging fruit"*[3], *"gold plating"*[4], and others, that required further reading from external sources.

4 Results and Discussion

This section presents the main results obtained from the discussions analysis. These results are used to answer the proposed research questions.

Throughout the analysis of the 108 discussions, nine types of TD were found, which were found through 74 indicators that were mentioned 254 times in total. In addition, 126 TDM practices were also found to support the mitigation of the TD experienced.

Our replication package is available at `https://doi.org/10.5281/zenodo.7604307`. There the reader can find our complete data set, the list of TD indicators, and TDM practices.

4.1 RQ1: What TD indicators do project management practitioners face during their projects?

We identified 74 indicators of the presence of TD. These indicators were mentioned 254 times. Table 2 shows the top 10 indicators, which correspond to 114 citations in total (∼46% of the overall number of citations). The table also presents the number of mentions of each indicator (**TDIM**) and the percentage of the total of mentions. *"Low morale"*, *"The agile practices are not respected"*, and *"Poorly written code"* are the most commonly discussed indicators, these indicators are, respectively, related to people, process and code debt.

Alves et al. (2016) analyzed TD indicators that have been discussed in the literature, such as *"code smells"* and *"violation of modularity"*. These indicators were most focused on code, design, and architecture debt. It is also worth mentioning that their work did not detect any indicator regarding people and process debt, while our study detected 41 and 13 indicators related to people and process debt, respectively.

Gama et al. (2020) reported 29 TD indicators, which they later grouped into high-level indicators such as *"presence of bad coding"* and *"poor testing practices"*. These indicators referenced various types of debt, including code and architecture. However, there were no indicators related to people and process debt.

Table 2. Top 10 ASD-TD indicators.

TD indicator	TDIM (%)
Low morale	15 (6%)
The agile practices are not respected	14(6%)
Poorly written code	14 (6%)
Design problems	13 (5%)
More effort required to delivery tasks due the TD	13 (5%)
Lack of approach on how to create user stories	11 (4%)
Rushed development	10 (4%)
Bugs occurrence	9 (4%)
Sprint delivery/velocity not properly estimated	8 (3%)
Not well defined requirements	7 (3%)

4.2 RQ2: What types of TD most concern project management practitioners?

Based on the TD indicators gathered from the discussions, we identified nine TD types have been discussed. Table 3 shows the found TD types along with the number of indicators related to it (**TDTI**) and the percentage of the total of indicators. Overall, we found 74 indicators. Some indicators were related to more than one type of debt; for example, the indicator *"Duplicated code"* is related to design and code debt, so we counted a TDTI instance for each related type. We can see that the TD types with the most indicators are process and people debt, representing 59% and 20% of the total percentage of indicators, respectively.

Process debt refers to inefficient processes (Alves et al., 2016), as described in the discussion excerpt, *"Due to external pressures, I have twice pushed for deployment with the outlook that this first iteration of the product doesn't need to be perfect, and both times deployment has failed, and we*

[2]`https://medium.com/@salibsamer/i-solved-scrum-sprint-end-testing-bottleneck-problem-bfd6222284a1`
[3]`https://aiko.dev/visualising-and-prioritizing-technical-debt/`
[4]`https://www.linkedin.com/pulse/gold-plating-software-engineering-project-management-chirag-pmp`

Table 3. Types of debt with the amount of indicators.

TD Type	TDTI (%)
Process	44 (59%)
People	15 (20%)
Documentation	6 (8%)
Requirements	4 (5%)
Testing	4 (5%)
Design	3 (4%)
Code	3 (4%)
Defect	2 (3%)
Versioning	1 (1%)

had to rollback due to bugs and not being ready for the actual launch," while people debt refers to people issues that, if present in the software organization, can delay or hinder some development activities (Alves et al., 2016), as can see in the discussion excerpt *"Developer passion has gone. Nobody wants to spend so much time in meetings, and all the decision-making power has evaporated, leaving the devs demotivated."*

Table 4 reports each TD type along with the number of times the type has been discussed/mentioned in the discussions (**#TDTM**) and the percentage of the total of mentions (**%TDTM**). The table also shows the percentage of mentions related to the total amount of discussions (**%TDTD**). The 9 TD types found were discussed 188 times over 108 discussions. A discussion can refer to multiple types of TD. For example, a discussion can be related to people, process, and testing debt. In the given example, we counted one mention for each TD type. Besides, although a discussion can have multiple TD indicators related to the same TD type, we counted only one mention of the type. In other words, a discussion can be associated only once with a TD type. Once again, we point out that most discussions are related to processes and people debt. Process debt represents 46% of the total mentions and is present in 81% of the discussions, and people debt represents 19% of the mentions and is present in 33% of the discussions. This result was already expected, as our previous works also found that these types of debt are the most discussed by SEPM users (Gomes et al., 2022; dos Santos et al., 2022).

Table 4. Types of debt discussed by SEPM users.

TD Type	#TDTM	%TDTM	%TDTD
Process	87	46%	81%
People	36	19%	33%
Code	17	9%	16%
Design	15	8%	14%
Defect	10	5%	9%
Requirements	9	5%	8%
Testing	8	4%	7%
Documentation	6	3%	6%
Versioning	1	1%	1%

It is also worth mentioning that from the 87 times it occurred, process debt appeared 43 times with at least one of the other seven TD types: people (26 times), code (10), design (10), defect (8), testing (6), requirements (5), documentation (1). Moreover, most of the time, the other seven TD types are also related to process debt. This indicates that other types of debts are also related to process debt.

Table 5 reports each TD type along with the sum of their indicators occurrences (**TDIO**) and the percentage in relation to the total of indicator occurrences. We can see that indicators related to process debt and people debt are the most discussed, with 58% and 18% of all indicators occurrences, respectively.

Table 5. Amount of TD indicators occurrences per type.

TD Type	TDIO (%)
Process	147 (58%)
People	45 (18%)
Code	17 (7%)
Design	16 (6%)
Documentation	11 (4%)
Defect	10 (4%)
Requirements	10 (4%)
Testing	9 (4%)
Versioning	1 (0%)

Compared to other studies, one of the main contributions of this work is that we bring a new perspective on practitioners' experiences regarding TD. While other studies get analysis more focused on developers' point of view, we focus our efforts on management practitioners.

Gama et al. (2020) analyzed TD-related discussions on the Stack Overflow website, a general-purpose website mostly centered towards software developers of various technologies (Stack_Overflow, 2021). Therefore, their analysis was focused on the developers' perspective. When analyzing their results, most types of debt detected were related to coding, such as code, architecture, infrastructure, and testing. There were no occurrences of debt items related to people or process, the most common types of debt found in our study.

The authors of the tertiary study performed by Rios et al. (2018) show that most TD literature is centered on architecture, design, code, defect, test, and documentation debt. Again, these are types of debt that revolve around software developers. And they also show that only a few works discussed people and process debt, which are the most common in our study. This indicates there are still gaps regarding TD from a management perspective in the technical literature.

4.3 RQ3: What TDM practices have been applied to mitigate or eliminate debt items?

We identified 126 TDM practices. In total, these practices were mentioned 375 times. Table 6 shows the top 10 practices, which correspond to 41% of the overall number of citations. The table also presents the number of mentions of each practice (**TDMM**) and the percentage of the total mentions. *"Make the cost of TD a visible part of your process"*, *"Create a prioritized list of TD items/tasks"*, and *"Invest time* *fixing/minimize the root cause of the TD"* are the most commonly discussed practices.

Table 6. Top 10 TDM practices addressed by PM's practitioners.

TDM Practice	**TDMM (%)**
Make the cost of TD a visible part of your process.	22 (6%)
Create a prioritized list of TD items/tasks.	20 (5%)
Invest time fixing/minimize the root cause of the TD.	19 (5%)
Educate team members about agile process and ceremonies.	16 (4%)
Improve the communication between management and developers.	14 (4%)
Decompose the user story/task into smaller ones based on the INVEST approach.	14 (4%)
Educate the higher management about the short/long term impact of not dealing with TD and other non-functional aspects.	14 (4%)
Encourage and help self improvement of the team members.	13 (3%)
Do not create separate user stories for minor tasks or TD, integrate them with the DoD.	11 (3%)
Improve the development process approach or change to a more suitable approach.	10 (3%)

Considering the set of categories of TDM activities reported by Rios et al. (2018), we grouped the 126 TDM practices into these categories. This grouping is presented in Table 7, where we show the number of TDM practices related to each type of activity (**TDMP**) alongside the percentage concerning the 126 found practices. The table also presents the sum of the citations from the TDM practices related to each type of activity (**Citations**) alongside the percentage in relation to the total number of occurrences (375). The most cited activity in the discussions was *prevention*, with 63 (50%) TDM practices and 146 citations (39%). Next, it is *communication* with 23 (18%) TDM practices and 85 (23%) citations. The fact that management practitioners are often discussing *prevention* and *communication* shows that practitioners are trying to avoid increasing TD, and better communication between managers and developers could improve software quality in their projects.

Gama et al. (2019) also presented TDM practices. They are mostly related to coding activities such as *"refactoring"* and *"TDD development approach"*. However, in a few instances, the authors also found practices related to management activities such as *"sprints dedicated to TD"* and *"bugs correction prioritization"*. Those practices can be related to practices found in our study, namely, *"Focus all project resources during a timeframe on TD solving"* and *"Create a prioritized list of TD items/tasks"*, respectively. Nonetheless, once again, it should be noticed that in our study, most TDM practices are related to management, indicating that managers and software teams perceive TDM differently.

Table 7. TDM practices grouped by category of TDM activity.

Category	**TDMP (%)**	**Citations (%)**
Prevention	63 (50%)	146 (39%)
Communication	23 (18%)	85 (23%)
Monitoring	12 (10%)	51 (14%)
Repayment	8 (6%)	35 (9%)
Measurement	8 (6%)	21 (6%)
Prioritization	6 (5%)	27 (7%)
Documentation	6 (5%)	10 (3%)

4.4 RQ4: Which agile artifacts are most affected by TD?

Table 8 presents the artifacts affected by ASD-TD, reporting the number of mentions of each artifact (**TDAC**) and the percentage concerning the total citations. One can see a total of 98 occurrences of artifacts compromised due to contact with any of the ASD-TD indicators.

Table 8. Consequence of ASD-TD for artifacts.

Artifact	**TDAC (%)**
Sprint backlog	49 (50%)
Product increment	27 (28%)
None	10 (10%)
Product backlog	9 (9%)
Definition of done	3 (3%)
Total	98 (100%)

The artifacts most compromised due to the presence of ASD-TD indicators mentioned in the discussions were: the *sprint backlog* with 49 (50%) mentions; the *product increment* with 27 (28%) mentions; the *product backlog* with 9 (9%) mentions, and the *definition of done* with 3 (3%) mentions. It was impossible to identify an artifact's consequences in 10% of cases. These results may help managers identify compromised artifacts and indicate a path to decision-making on mitigating the problems caused by the accumulation of TD.

4.5 RQ5: Which agile events are harmed by TD?

Table 9 informs the ASD events compromised by the TD with the number of times that the event was discussed (**TDET**) as well as the percentage concerning the total of occurrences. On 82 occasions (posts and comments), project management practitioners addressed the presence of some ASD-TD consequences for one of these events. Of these, *sprint planning* was mentioned 59 (73%) times, and the *daily scrum* event was mentioned only two (1%) times in the discussions. It was impossible to identify the event involved in 21 (26%) of the times.

Table 9. Consequence of ASD-TD for events.

Event	**TDET (%)**
Sprint planning	59 (73%)
None	21 (26%)
Daily Scrum	2 (1%)
Total	82 (100%)

4.6 RQ6: What agile roles are involved with the consequences of TD?

Table 10 presents the agile roles involved with TD consequences, along with the number of mentions of each role (**TDRL**) and the percentage of the total of mentions. We can notice that roles *development team* and *product owner* were the most compromised roles due to ASD-TD indicators in SEPM, with 44% and 25% of occurrences in discussions, respectively. These results indicate that professionals tend to discuss several aspects related to some consequence resulting from the accumulation of ASD-TD that falls on one of those roles.

Table 10. Consequence of ASD-TD for roles.

Role	**TDRL (%)**
Development team	46 (44%)
Product owner	26 (25%)
None	21 (20%)
Scrum Master	11 (11%)
Total	104 (100%)

5 Relating TD indicators with TDM practices and TD types

RQ1 and RQ3 allowed us to identify TD indicators and TDM practices, but the discussions also allowed us to identify relationships between them further. For this, we identified the co-occurrence of indicators and practices and analyzed them to check if the practices were being used to deal with the indicators. For example, in one of the discussions, the indicator *"The agile practices are not respected"* and the practices *"Educate team members about agile process and ceremonies"* and *"Improve the communication between management and developers"* are mentioned to deal with it.

We recorded the pairs *indicator–TDM practice* as relationships and then grouped and counted all relationships found in the analyzed discussions. We then organized the relations between the TD indicators and TDM practices using a Sankey diagram (Lupton and Allwood, 2017). This diagram comprises bars representing the source (TD indicators) and destination (TDM Practices) of information and links showing the magnitude of information flowing between those bars. From the practitioners' point of view, the flows summarize which TDM practices are recommended by the community to address a specific TD indicator.

Figure 8 shows a Sankey diagram considering the most common relationships we found on SEPM. On the left, it has the TD indicators; on the right, the TDM practices are recommended for dealing with each indicator. The numeric values at the side of the TD indicators (left side) show the total relations of that indicator with TDM practices. On the other hand, the numeric values at the side of the TDM practices (right side) show the total of relations of that practice with the TD indicators on the left. The colored flow lines show the relation of a TD indicator with a TDM practice, where the thickness of each line represents the number of relations. In other words, a flow line indicates that practitioners recommended a TDM practice as a solution for a TD indicator, where the line thickness represents the number of times the relationship was identified in different discussions. For example, concerning the TD indicator *"Poorly written code"*, the following practices were recommended: *"Invest time fixing/minimize the root cause of the TD"* (7 times), *"Make the cost of TD a clearly visible part of your process"* (8 times) and *"Educate the higher management about the short/long term impact of not dealing with TD and other non-functional aspects"* (6 times).

We built a mind map to provide a more structured view of the types and indicators of TD identified. Figure 9 shows a mind map considering the indicators with the most citations of each type. The map brings the number of citations of each indicator between parenthesis. To create the map, we considered the types and indicators of technical debt perceived by the participants in questions RQ1 and RQ2.

Analyzing Figure 9, we can see that the most cited TD indicators of each type are: *"Low morale."* (15 citations, people), *"Poorly written code."* (14, code), *"Lack of documentation."* (4, documentation), *"Bugs occurrence."* (9, defect), *"Lack of automated tests."* (4, testing), *"Design problems."* (13, design), *"Not well defined requirements"* (7, requirements), *"Lack of VCS (Version Control System)."* (1, versioning) and *"The agile practices are not respected"* (14, process). Some indicators like *"Duplicated code."* appeared twice because it is considered a code debt and a design debt. By applying this view, we can see the most common indicators of each TD type faced by practitioners.

Our replication package is available at `https://doi.org/10.5281/zenodo.7604307`. There the reader can find our data set, the complete list of TD indicators and TDM practices, and the complete version of the Sankey diagram for this study.

6 Implications for Practitioners and Researchers

Practitioners may employ our findings in the identification of TD in their projects. Since we present a set of indicators, it can be used as a starting point to identify TD items. For example, when a software team recognizes a problem in the project, they can verify whether the problem is a TD indicator. Next, the team can use the related TDM practices and the type of debt they are facing to define actions to repay the debt or to prevent it to happen in the future.

For researchers, our findings support new research efforts in TD identification and management. Our list of TD indicators can guide further investigations on approaches to de-

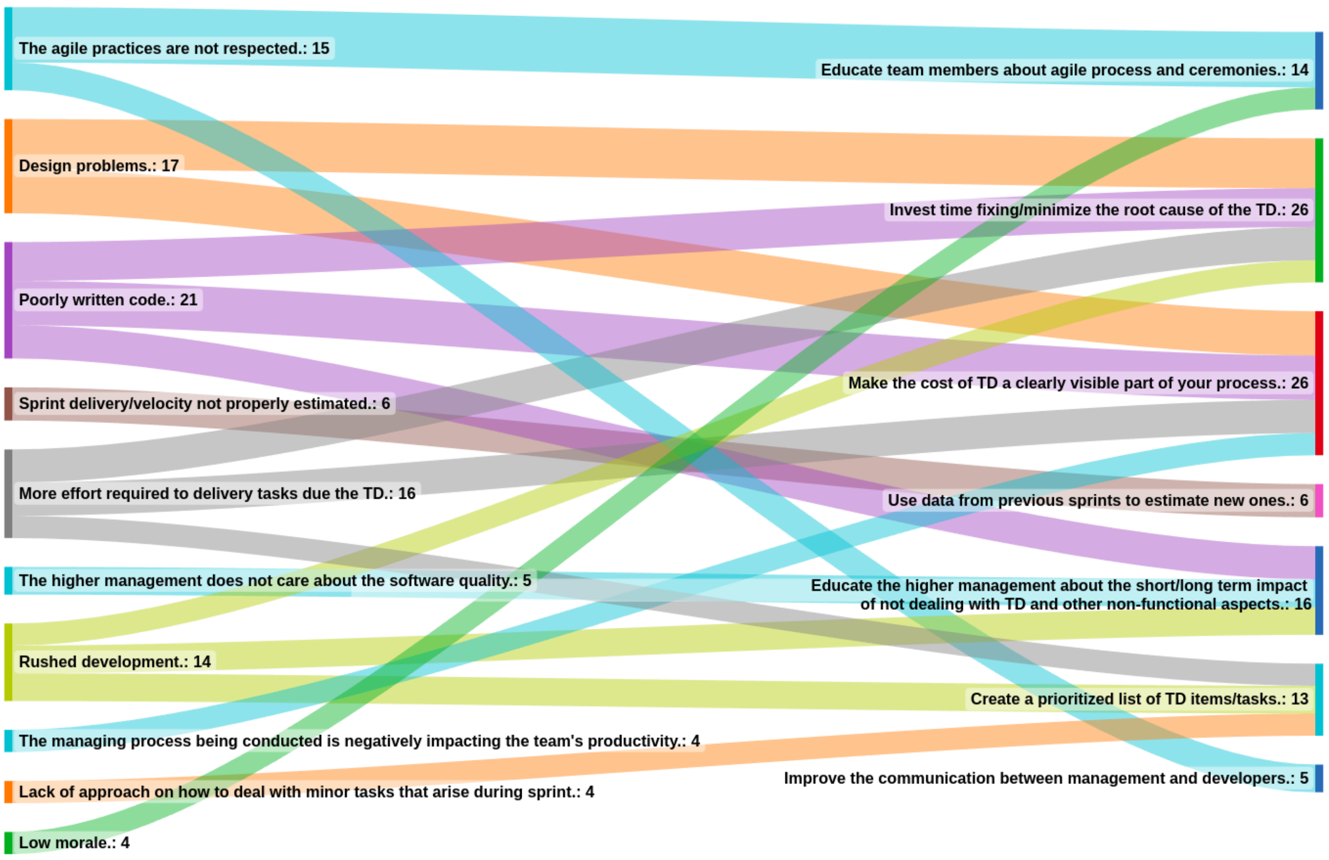


Figure 8. Sankey diagram showing the relationship between TD indicators and TDM practices.

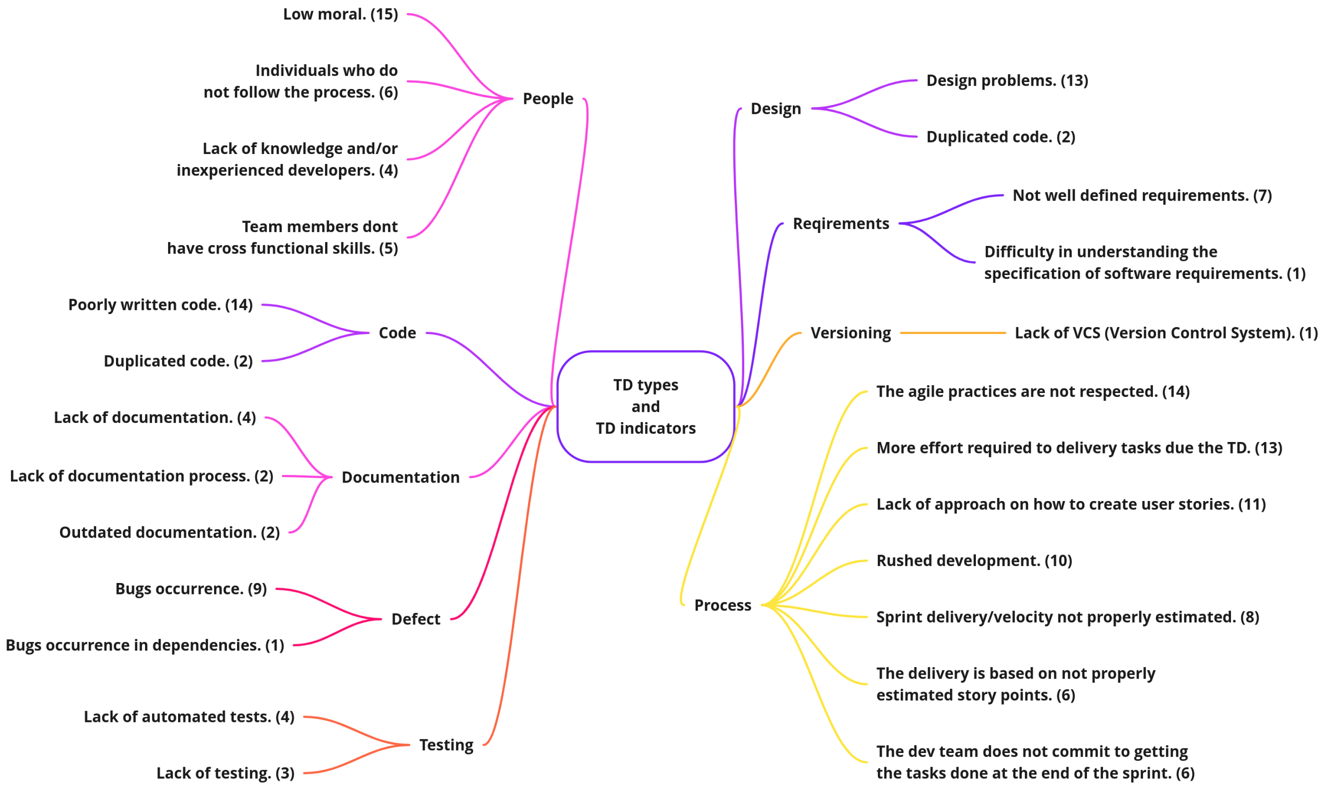

TD types and TD indicators
People
Low moral. (15)
Individuals who do not follow the process. (6)
Lack of knowledge and/or inexperienced developers. (4)
Team members dont have cross functional skills. (5)
Code
Poorly written code. (14)
Duplicated code. (2)
Documentation
Lack of documentation. (4)
Lack of documentation process. (2)
Outdated documentation. (2)
Defect
Bugs occurrence. (9)
Bugs occurrence in dependencies. (1)
Testing
Lack of automated tests. (4)
Lack of testing. (3)
Design
Design problems. (13)
Duplicated code. (2)
Reqirements
Not well defined requirements. (7)
Difficulty in understanding the specification of software requirements. (1)
Versioning
Lack of VCS (Version Control System). (1)
Process
The agile practices are not respected. (14)
More effort required to delivery tasks due the TD. (13)
Lack of approach on how to create user stories. (11)
Rushed development. (10)
Sprint delivery/velocity not properly estimated. (8)
The delivery is based on not properly estimated story points. (6)
The dev team does not commit to getting the tasks done at the end of the sprint. (6)

Figure 9. Relationship between TD types and TD indicators.

tect TD. For example, one of the most common indicators in the discussions, *"Low morale"*, with 15 occurrences, it could be worth investigating why developers have their morale reduced in projects, which can lead to other types of problems in the software project. Regarding agile, one of the most common indicators in the discussions, *"The agile practices are not respected"*, with 14 occurrences, indicates that it could be worth it to investigate the reason why the development teams do not follow agile practices and why the management cannot solve the issue.

Our list of TDM practices can guide investigations on how practitioners deal with TD and help bring to light new TDM approaches since we analyzed it from the perspective of project management, which brought a new range of possibilities to be investigated. For example, two of the most common TDM practices, *"Make the cost of TD a clearly visible part of your process"* and *"Create a prioritized list of TD items/tasks"*, with 22 and 20 occurrences, respectively, indicating that making TD visible and prioritizing it can help to reduce debt items. Regarding agile, one the most recommended TDM practices are *"Educate team members about agile process and ceremonies"*, with 16 occurrences indicating that training on processes and ceremonies can help to mitigate most problems related to agile development.

Lastly, the identified relationships between TD indicators and management practices deserve further investigation to evaluate the extent to which those practices support development teams in dealing with the identified TD indicators.

7 Threats to Validity

Below we present the threats to the validity of our research, grouping them according to the concepts specified in Wohlin et al. (2012).

Construct: One threat stems from the fact that the questions in Table 1 could not be enough to gather all necessary information from the discussion. This risk was mitigated using a theoretical framework based on a recent tertiary study on TD (Rios et al., 2018). Another threat is selecting and analyzing the discussions directly from SEPM. We did not survey the discussion authors and used generic search terms to find the discussions. We mitigate this threat by performing pilot studies with various terms related to TD before deciding which terms would be used. We also minimize the noise in the analyses by applying quantitative and qualitative filters.

Internal: The process used for analyzing and coding the TD indicators and TDM practices and grouping TDM practices can represent a threat in our study. To reduce this thread, the procedures portrayed in Figure 2 were performed by two researchers individually. Besides, a third researcher was inserted to resolve the divergences identified in the consensus phase.

External: Regarding the generalization of the conclusions, the study was based on a representative sample of SEPM, a well-known Q&A platform focused on project management discussions, where practitioners discuss day-to-day issues. Although we used SEPM data, we cannot guarantee that all professionals are project management practitioners. This is mitigated by the fact that SEPM discussions are contextualized in the software management area. Another threat arises from the risk that the selected discussions are unrelated to the ASD, TD, or project management domains. To mitigate this threat, we carefully checked if the authors tagged the question with a direct reference to ASD, TD, and TDM or if they made meaningful references to those in the discussion title, body, or comments. Due to these mitigation actions, we believe that the results provide a good perspective for analyzing TD in the context of project management.

Conclusion: Lastly, there is a risk that, even if the same approach is applied to the analysis and interpretation of the discussions, there is the possibility of producing different results. This risk stems from the subjectivity inherent in the coding process. It was mitigated by the consensus procedures used during the analysis process presented in Section 3.

8 Conclusion

This work investigates how project management practitioners experience and manage TD. Besides that, we also identified roles, artifacts, and practices affected by ASD-TD. By performing mining and qualitative analyses in the SEPM forum, we found 74 TD indicators, categorized into nine types, and 126 TDM practices. The relationships between TD indicators and management practices were organized into a Sankey diagram, which can be used as a supporting artifact to identify and manage TD.

This work shows that analyzing TD from the perspective of management practitioners brought new results. These results indicate these practitioners tend to perceive TD differently from other roles involved in software development. Process and people debt were the most discussed types of TD. As project planning and management play a decisive role in the management of TD (Rios et al., 2018), new studies in the area must be carried out from this point of view.

The results reveal that SEPM practitioners perceive that the agile elements most affected by debt are the *"sprint backlog"*, the *"development team"*, and the *"sprint planning"*.

As future work, we suggest (1) analyzing the discussion population in detail, expanding the sample by exploring related discussions, (2) investigating the relationship between the most compromised artifacts due to the ASD-TD, (3) investigating whether the effects of TD pointed out in the study by Rios et al. (2019) can be related to the elements compromised by the TD pointed out in our study, and (4) consider the use of a snowballing-like technique in the SEPM analysis process.

Acknowledgements

This study was financed in part by the following agencies: Brazil's Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Finance Code 001 and by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

References

Agile_Alliance (1999). Extreme programming (xp). *Available at: https://www.agilealliance.org/glossary/xp/ [Last accessed: April 29, 2022]*.

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C. K., and Schneider, K. A. (2016). Mining duplicate questions of stack overflow. In *2016 IEEE/ACM 13th Working MSR*, pages 402–412. IEEE.

Ai, D. (2021). 15th annual state of agile report.

Alves, N. S., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., and Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Inf. Softw. Technol.*, 70:100–121.

Barua, A., Thomas, S. W., and Hassan, A. E. (2014). What are developers talking about? an analysis of topics and trends in stack overflow. *Empir. Softw. Eng.*, 19(3):619–654.

Behutiye, W. N., Rodríguez, P., Oivo, M., and Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology*, 82:139–158.

Berenguer, C., Borges, A., Freire, S., Rios, N., Ramač, R., Taušan, N., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., López, G., Mendonça, M., Falessi, D., Seaman, C., Mandić, V., Izurieta, C., and Spínola, R. (2023). Investigating the relationship between technical debt management and software development issues. *Journal of Software Engineering Research and Development*, 11(1):3:1 – 3:21.

Bohnet, J. and Döllner, J. (2011). Monitoring code quality and development activity by software maps. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, MTD '11, pages 9–16, New York, NY, USA. ACM.

C., S. and R., S. (2013). Managing technical debt. In *(Short Course) XVII Brazilian Symposium on Software Quality*, Salvador, Brazil.

Codabux, Z. and Williams, B. (2013). Managing technical debt: An industrial case study. In *Proceedings of the 4th International Workshop on Managing Technical Debt*, MTD '13, pages 8–15, Piscataway, NJ, USA. IEEE Press.

Digkas, G., Nikolaidis, N., Ampatzoglou, A., and Chatzigeorgiou, A. (2019). Reusing code from stackoverflow: the effect on technical debt. In *45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 87–91.

dos Santos, E. P., Gomes, F., Freire, S., Mendonca, M., Mendes, T., and Spínola, R. (2022). Technical debt on agile projects: Managers point of view at stack exchange. In *SBQS 2022 - Research Track*.

Freire, S., Rios, N., Gutierrez, B., Torres, D., Mendonça, M., Izurieta, C., Seaman, C., and Spínola, R. O. (2020a). Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items. EASE '20, page 210–219. ACM.

Freire, S., Rios, N., Mendonça, M., Falessi, D., Seaman, C., Izurieta, C., and Spínola, R. O. (2020b). *Actions and Impediments for Technical Debt Prevention: Results from a Global Family of Industrial Surveys*, page 1548–1555. ACM.

Freire, S., Rios, N., Pérez, B., Castellanos, C., Correal, D., Ramač, R., Mandić, V., Taušan, N., López, G., Pacheco, A., Mendonça, M., Falessi, D., Izurieta, C., Seaman, C., and Spínola, R. (2023). Software practitioners' point of view on technical debt payment. *Journal of Systems and Software*, 196:111554.

Gama, E., Freire, S., Mendonça, M., Spínola, R. O., Paixao, M., and Cortés, M. I. (2020). Using stack overflow to assess technical debt identification on software projects. In *34th Brazilian Symposium on Software Engineering (SBES)*, pages 730–739.

Gama, E., Paixao, M., Freire, E. S. S., and Cortés, M. I. (2019). Technical debt's state of practice on stack overflow: a preliminary study. In *Proceedings of the XVIII SBQS*, pages 228–233.

Garousi, V., Felderer, M., and Mäntylä, M. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106(2):101–121.

Gomes, F., dos Santos, E. P., Freire, S., Mendonça, M., Mendes, T. S., and Spínola, R. (2022). Investigating the point of view of project management practitioners on technical debt - a preliminary study on stack exchange. In *2022 IEEE/ACM International Conference on Technical Debt (TechDebt)*, pages 31–40.

Greening, D. R. (2013). Release duration and enterprise agility. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4835–4841.

Guo, Y. and Seaman, C. (2011). A portfolio approach to technical debt management. In *Proceedings of the 2Nd Workshop on Managing Technical Debt*, MTD '11, pages 31–34, New York, NY, USA. ACM.

Holvitie, J., Licorish, S., Spínola, R., Hyrynsalmi, S., MacDonell, S., Mendes, T., Buchan, J., and Leppänen, V. (2017). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, in press.

Izurieta, C., Vetró, A., Zazworka, N., Cai, Y., Seaman, C., and Shull, F. (2012). Organizing the technical debt landscape. In *Proceedings of the Third International Workshop on Managing Technical Debt*, MTD '12, pages 23–26, Piscataway, NJ, USA. IEEE Press.

Kavaler, D., Posnett, D., Gibler, C., Chen, H., Devanbu, P., and Filkov, V. (2013). Using and asking: Apis used in the android market and asked about in stackoverflow. In *SocInfo*, pages 405–418. Springer.

Klinger, T., Tarr, P., Wagstrom, P., and Williams, C. (2011). An enterprise perspective on technical debt. In *Proceedings of the 2nd Workshop on Managing Technical Debt (MTD)*, pages 35–38.

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

Li, Z., Avgeriou, P., and Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220.

Lupton, R. C. and Allwood, J. M. (2017). Hybrid sankey diagrams: Visual analysis of multidimensional data for understanding resource use. *Resources, Conservation and Recycling*, 124:141–151.

McConnell, S. (2008). Productivity variations among software developers and teams: The origin of 10x. *10x Software Development*.

Montandon, J. E., Politowski, C., Silva, L. L., Valente, M. T., Petrillo, F., and Guéhéneuc, Y.-G. (2021). What skills do it companies look for in new developers? a study with stack overflow jobs. *Inf. Softw. Technol.*, 129:106429.

Partogi, J. (2018). Scrum and extreme programming (xp). *Available: https://www.scrum.org/resources/blog/scrum-and-extreme-programming-xp [Last accessed: April 26, 2022]*.

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

Rios, N., Freire, S., Perez, B., Castellanos, C., Correal, D., Mendonca, M., Falessi, D., Izurieta, C., Seaman, C. B., and Spinola, R. O. (2021). On the relationship between technical debt management and process models. *IEEE Software*, 38(5):56–64.

Rios, N., Mendonça, M. G., Seaman, C. B., and Spínola, R. O. (2019). Causes and effects of the presence of technical debt in agile software projects. In *25th Americas Conference on Information Systems, AMCIS 2019, Cancún, Mexico, August 15-17, 2019*. Association for Information Systems.

Rosen, C. and Shihab, E. (2016). What are mobile developers asking about? a large scale study using stack overflow. *Empir. Softw. Eng.*, 21(3):1192–1223.

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21(1).

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE TSE*, 25(4):557–572.

Sengupta, S. and Haythornthwaite, C. (2020). Learning with comments: An analysis of comments and community on stack overflow. In *53rd HICSS*.

Snipes, W., Robinson, B., Guo, Y., and Seaman, C. (2012). Defining the decision factors for managing defects: A technical debt perspective. In *Proceedings of the Third International Workshop on Managing Technical Debt*, MTD '12, pages 54–60, Piscataway, NJ, USA. IEEE Press.

Stack_Overflow (2021). 2021 developer survey. *Available at: https://insights.stackoverflow.com/survey/2021 [Last accessed: january 2, 2022]*.

Tahir, A., Dietrich, J., Counsell, S., Licorish, S., and Yamashita, A. (2020). A large scale study on how developers discuss code smells and anti-pattern in stack exchange sites. *Inf. Softw. Technol.*, 125:106333.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.