Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items

Sávio Freire
Federal University of Bahia
Federal Institute of Ceará
Brazil
savio.freire@ifce.edu.br

Nicolli Rios
Federal University of Bahia
Brazil
nicollirioss@gmail.com

Boris Gutierrez
University of Los Andes
Francisco de Paula Stder. Univ.
Colombia
borisperezg@ufps.edu.co

Darío Torres
University of Los Andes
Colombia
dcorreal@uniandes.edu.co

Manoel Mendonça
Federal University of Bahia
Brazil
manoel.mendonca@ufba.br

Clemente Izurieta
Montana State University
United States
clemente.izurieta@montana.edu

Carolyn Seaman
University of Maryland Baltimore County
United States
cseaman@umbc.edu

Rodrigo O. Spínola
Salvador University and State University of Bahia
Brazil
rodrigo.spinola@unifacs.br

ABSTRACT

Background: Little is known about the practices used for technical debt (TD) payment. The study of payment practices, as well as the reasons for not applying them, can help practitioners to control and manage TD items. **Aims**: To investigate, from the point of view of software practitioners, if TD items have been paid off in software projects, the practices that have been used to pay off TD and the reasons that hamper the implementation of these practices. **Method**: We analyzed - both quantitatively and qualitatively - a corpus of responses from a survey of 432 practitioners, from four countries, about the possibility of TD payment. **Results**: We found that, for most of the cases, TD items have not been eliminated from software projects. The main reasons for not paying off TD are lack of organizational interest, low priority on the debt, focus on short-term goals, cost, and lack of time. On the other hand, we identified that code refactoring, design refactoring, and update system documentation are the most used practices for TD payment. Practitioners also cited practices related to the prevention, prioritization, and creation of a favorable setting as part of TD payment initiatives. **Conclusion**: This paper summarizes the identified practices and reasons for not paying off debt items in a map. Our map reveals that the majority of payment practices are of a technical nature while the majority of reasons for not paying off debts are associated with non-technical issues.

EASE 2020, April 15–17, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-7731-7/20/04…$15.00
https://doi.org/10.1145/3383219.3383241

CCS CONCEPTS

• General and reference~Surveys and overviews • Software and its engineering

KEYWORDS

Technical debt, Technical debt payment, Technical debt management, *InsighTD*

ACM Reference format:

Sávio Freire, Nicolli Rios, Boris Gutierrez, Darío Torres, Manoel Mendonça, Clemente Izurieta, Carolyn Seaman, and Rodrigo O. Spínola. 2020. Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items. In *Proceedings of Evaluation and Assessment in Software Engineering (EASE 2020). Trondheim, Norway, April 15-17, 2020, 10 pages.* DOI: https://doi.org/10.1145/3383219.3383241

1 Introduction

Technical debt (TD) describes the effects faced during software development due to immature artifacts [1, 2]. These effects can be either beneficial (e.g., higher productivity) or harmful (e.g., hindering system evolution). Through TD management, project teams can handle these effects, decreasing risks imposed by the presence of debt [3]. TD management is comprised of activities that include the identification, monitoring and payment of debt items [4]. The last one refers to the activities undertaken with the goal of eliminating debt items [5].

We define TD payment practices as ways to eliminate identified TD items. Several research articles have sought to identify practices, tools, and activities for TD management [5, 6]. However, little is known about practices used for paying off TD items [5]. If

known, these practices could support development teams in choosing the most suitable practices for paying off TD items [8].

There is a lack of empirical evidence from the software industry in this area [7]. Although there is some discussion on TD payment in the technical literature, there is little evidence on whether TD items have been eliminated from software projects, what TD payment practices are currently applied by software practitioners, and what reasons exist to the non-application of these practices. These issues are precisely the topics addressed in this paper.

The goal of this work is to investigate, from the point of view of software practitioners, if they have paid debt items off in their projects, the practices that have been used for payment, and the issues that hamper the implementation of these practices.

The paper uses a subset of the data collected by the *InsighTD* Project, http://td-survey.com/, which is a globally distributed family of industrial surveys on TD causes and implications [9]. In total, 432 professionals from Brazil, Chile, Colombia, and the United States responded the survey up to now. We analyzed these responses through qualitative and quantitative procedures.

The survey analysis was organized as follows. It initially characterizes the survey's participants. Then, it identifies the answers that indicates that the payment of the TD items described by the participants was possible, and qualitatively analyzes the cited practices. For those who indicated that the payment of their described debt items was not possible, it qualitatively analyzes the possible reasons for not eliminating those TD items.

Results show that debt items were not paid off in most cases. From a total of 28 identified reasons for not paying off a TD *low priority of TD*, *lack of organizational interest*, *focusing on short-term goals*, *costs*, and *lack of time* are the most cited ones. By grouping the identified reasons for not paying off a debt, we identified that *planning and management* and *organizational issues* are the categories that most hamper the application of TD payment practices. Further, 13 of these reasons are related to decision factors, while the other 15 are impediments that do not allow debt payment even if the development team wanted to pay it off.

On the other hand, from a total of eight identified payment practices, *code refactoring*, *design refactoring*, and *update system documentation* are the most common ones. In addition, we have also found that the use of practices for the prevention, prioritization and creation of a TD pay-setting is commonly cited in the context of TD payment.

To support the practical use of the results of this work, we organize the set of practices and reasons into a TD Payment Map. Software practitioners can consult this map to guide their decisions about eliminating debt items from their projects.

This work has implications for both practitioners and researchers. Practitioners can use the list of TD payment practices to decide which ones better fit their needs. The list of reasons that hamper the application of these practices can support practitioners to foresee difficulties and better plan for TD payment in their work environment. For researchers, the findings provide a grounded view of the software industry needs with respect to TD payment. Results can guide new research efforts aligned with the demands and current context of TD payment experienced by practitioners.

This paper is organized in seven other sections. Section 2 presents background about the *InsighTD* project, and TD management and its payment. Section 3 presents the research method. Then, Section 4 presents the results of the survey concerning TD payment. Section 5 discusses the main findings, presenting the map for supporting TD payment. Section 6 discusses the implications of the study for researchers and practitioners. Section 7 discusses the threats to the study validity. And lastly, Section 8 presents our final remarks and discusses the next steps of this work.

2 Background

This section introduces the *InsighTD* project and TD payment concepts related to this work.

2.1 The *InsighTD* Project

InsighTD is a globally distributed family of industrial surveys on TD. The project aims at establishing an open and generalizable set of empirical data on practical problems of TD [9]. Its design supports replications of the survey in different countries. At the time of this writing, researchers from 11 countries – namely, Brazil, Chile, Colombia, Costa Rica, Finland, India, Italy, Norway, Saudi Arabia, Serbia, and the United States – have joined the project. At this point, the project has concluded data collections for the *InsighTD* replications in Brazil, Chile, Colombia, and the United States.

Significant analyses of *InsighTD* data have already been conducted, regarding causes, effects, and preventive actions about TD. These include: preliminary results concerning TD causes and effects (along with the detailed design of the study) [9], an analysis of TD causes and effects in agile projects [10], the use of cross-company probabilistic cause-effort diagrams to support TD management initiatives [11], and TD prevention [12].

Although the aforementioned analyses have supported conclusions about the state of the practice on TD, much still remains to be investigated. As previously discussed, this paper analyzes the *InsighTD* data with respect to TD payment from the point of view of the survey respondents.

2.2 Technical Debt and its Payment

Technical debt management is a decisive factor for increasing the success of software projects [14]. It is composed of activities supporting decisions about the need and the appropriate time to eliminate TD items from a software project [13]. These decisions can cause negative and positive impacts on a project. A positive impact can be achieving project goals sooner, while negative impacts can be increasing cost and technical problems.

There are several activities associated with TD management: identification, measurement, prioritization, prevention, monitoring, payment, documentation, communication, visualization, time-to-market analysis, and scenario analysis [5, 7]. A TD payment activity refers to practices for resolving TD items using approaches such as reengineering and refactoring [5]. Li *et al.* [5] conducted a systematic mapping study on the subject.

They identified seven categories of practices for paying off TD items, namely, refactoring, rewriting, automation, reengineering, bug fixing, repackaging, and fault tolerance. However, these categories have not been investigated in industrial settings [5]. More recently, Rios *et al*. [7] conducted a tertiary study about TD, evidencing that few practices for TD payment have been described from studies conducted in industry.

Our study seeks to better understand (i) the practices that can be used to pay off TD items and (ii) the reasons that hamper the utilization of these practices in the software industry.

3 Research Method

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

3.1 Research Questions

Our main research question (RQ) is ***"How have software practitioners paid off technical debt in their projects?"*** The purpose of this RQ is to identify the main practices that software practitioners have used to eliminate debt items. We divide this question into the following sub-questions:

RQ1: *Have software practitioners paid off TD in their projects?* Through this question, we will explore practitioners' responses from the *InsighTD* dataset and calculate how often their TD items have been paid off.

RQ2: *What are the main practices used by software practitioners to pay off TD?* This question aims to investigate the most cited practices used to pay TD items.

RQ3: *What are the main reasons cited by software practitioners for not paying off TD items?* The purpose of this question is to investigate the possible reasons that hinder the implementation of TD payment practices.

We also have a sub-question that aims at verifying whether the role of software practitioners influences in the perception of the reasons that curb the application of TD payment practices.

RQ3.1: *Does the role of practitioners influence in the perception of the reasons (decision factors and impediments) for not paying TD?* This question aims to investigate if there is support for the idea that technical folks always want to pay off the debt (so they see the reasons for not paying it off as *impediments*) and management folks never want to pay it off (so they see the reasons for not paying it off as *decision factors*). Thus, to approach this question, we consider two main groups of practitioners: technical and non-technical. The first is related to roles involved in the technical development of a software project, including hands-on activities such as requirement analysis, design, coding, and testing. The second is related to non-technical roles, encompassing practitioners with managerial responsibilities in their projects.

3.2 Data Collection

As said, this work uses data collected by survey in the context of the *InsighTD* project. The survey is composed of 28 questions, as previously described in [9]. The work described here is concerned with a subset of these questions. This subset is shown in Table 1 along with the question identifier number (No), description, and type. Q1 to Q8 characterize the survey participants and their work context. Q9 and Q10 identify the participants' knowledge level of TD. Q13 and Q15 ask participants to describe an example of a TD item that occurred in their software project (this example is used as a basis for answering TD payment questions) and indicate the representativeness level of this example, respectively. Finally, questions Q26 and Q27 focus on the discussion of the participants' experiences in TD payment.

As the *InsighTD* project intends to investigate the state of the practice of TD, we sent e-mail invitations to answer the survey to software practitioners from Brazil, Chile, Colombia, and United States. In all cases, we followed the same strategy, using LinkedIn, industry-affiliated member groups, mailing lists, and industry partners, as invitation channels.

Table 1: Subset of the *InsighTD* Survey's Questions Related to TD Repayment (Adapted from [9])

No.	Question (Q) Description	Type
Q1	What is the size of your company?	Closed
Q2	In which country you are currently working?	Closed
Q3	What is the size of the system being developed in that project? (LOC)	Closed
Q4	What is the total number of people of this project?	Closed
Q5	What is the age of this system up to now or to when your involvement ended?	Closed
Q6	To which project role are you assigned in this project?	Closed
Q7	How do you rate your experience in this role?	Closed
Q8	Which of the following most closely describes the development process model you follow on this project?	Closed
Q9	How familiar you are with the concept of Technical Debt?	Closed
Q10	In your words, how would you define TD?	Open
Q13	Give an example of TD that had a significant impact on the project that you have chosen to tell us about:	Open
Q15	About this example, how representative it is?	Closed
Q26	Has the debt item been paid off (eliminated) from the project?	Closed
Q27	If yes, how? If not, why?	Open

3.3 Data Analysis

As shown in Table 1, the questionnaire is composed of closed and open-ended questions. To analyze closed questions, we used descriptive statistics to verify the central tendency of the ordinal and interval data (mode and median statistics) and to calculate the share of participants choosing each option (nominal data). These data analysis procedures were applied for the characterization questions and also for Q26, supporting the response to *RQ1*.

We applied qualitative data analysis techniques to the open-ended questions [15, 16]. In answers given to Q27, we initially applied manual open coding resulting in a set of codes. We divided those codes in two subsets based on the answers to Q26. If the answer was positive, the code was related to the TD payment practices (*RQ2*). If the answer was negative the code was related to reasons for not paying off the TD (*RQ3*). The process was performed iteratively revising and unifying codes at each cycle of analysis until reaching the state of saturation, i.e., a point

where no new codes were identified. In the end of the analysis, we obtained a stable list of codes along with their citation frequency.

At least three researchers conducted the coding in each of the four *InsighTD* replications. Each researcher could assume one of the following roles: (i) ***code identifier*** – the person responsible for extracting the existing codes in the answers, (ii) ***code reviewer*** – responsible for reviewing all extracted codes, and (iii) ***referee*** – responsible for resolving disagreements in codes identified by the code identifier and code reviewer. More specifically, we had the 1st and 2nd authors performing the roles of code identifier and reviewer, respectively, in the Brazilian and North American datasets; and the 3rd and 4th authors performing the roles of code identifier and reviewer, respectively, in the Chilean and Colombian datasets. The 8th author acted as the referee in all replications, pursuing consistency among all analyses.

Code unification was especially laborious. For example, participants cited the following practices to pay off TD: "by implementing required technology" and "(...) updating the solution to the latest technology changes". The originally extracted codes were implementing required technology and technology/tool/platform change, respectively. Then, as these codes had different nomenclature but shared a common meaning, we unified them as solving technical issues.

By perceiving that many of the codes in the same subset (practices or reasons for do not eliminate the debt) were related to each other, we followed a grouping process to organize them into categories reflecting the main concern of each subset. This process followed axial coding [16]. The 1st author analyzed each code comparing its meaning with that of the other codes. When the researcher identified a relation between them, he grouped them into a category. To name the categories, we used the list proposed by Rios *et al*. [11]. For example, we used the category *internal quality issues* to group TD payment practices like *code refactoring* and *design refactoring*. For consistency, once again, the entire process was reviewed by the last author of this paper.

Lastly, to respond *RQ3.1*, we divided the dataset of reasons into two groups (role: technical or non-technical) along two dimensions (reasons: decision factors or impediments). This partitioning allows us to investigate if there are significant differences between the groups using the Pearson's Chi-squared statistical test. This test is indicated when the variables are categorical and paired. The role and reason are defined as test's variables. These variables (i) are non-numerical and describe data fitted into categories, and thus are categorical and (ii) are related to each other as long as each survey participant specifies their role and the reasons identified in their projects, in other words, they are paired.

4 Results

In total, we obtained 432 valid answers: 107 from Brazil, 92 from Chile, 133 from Colombia, and 100 from the United States.

4.1 Characterization of the Participants

Table 2 details the roles of the participants, reporting the role name (role), the frequency of occurrence of each role (#FR) and the percentage of FR in relation to the total of all cited roles (%FR). The column *level of experience* shows the number of participants by their level of experience in their role among the following options: Novice (Minimal or "textbook" knowledge without connecting it to practice), Beginner (Working knowledge of key aspects of practice), Competent (Good working and background knowledge of area of practice), Proficient (Depth of understanding of discipline and area of practice), and Expert (Authoritative knowledge of discipline and deep tacit understanding across area of practice). For each level of experience, we presented, in parentheses, its percentage in relation to the whole dataset.

The majority of respondents identified themselves as developers (42.4%). The roles of project manager, software architect, tester, requirement analyst, and process analyst are also common. Most respondents identified themselves as proficient (33%), followed by competent (28%), expert (27%), beginners (11%), and novice (1%), indicating that, in general, the questionnaire was answered by professionals with experience in their functions.

The respondents worked in organizations of different sizes. Most of them worked in medium-sized companies (39%, organizations with 51 to 1000 employees), followed by large (34%, more than 1000 employees) and small (27%, up to 50

Table 2: Participant Roles

Role	#FR	%FR	Level of Experience				
			Novice (1%)*	Beginner (11%)	Competent (28%)	Proficient (33%)	Expert (27%)
Developer	183	42.4%	1	20	52	65	45
Project Leader / Project Manager	84	19.4%	0	7	24	28	25
Software Architect	73	16.9%	1	5	17	28	22
Test Manager / Tester	36	8.3%	0	3	12	10	11
Requirements Analyst	17	3.9%	0	4	6	5	2
Process Analyst	12	2.8%	0	4	5	1	2
Business Analyst	7	1.6%	0	2	4	0	1
Database Administrator	7	1.6%	0	2	0	3	2
Performs multiple functions	5	1.2%	0	1	2	1	1
Infrastructure analyst	3	0.7%	1	0	1	0	1
Configuration Manager	2	0.5%	0	0	0	0	2
Quality Analyst	2	0.5%	0	0	0	0	2
Data Scientist	1	0.2%	0	1	0	0	0

* The number in parentheses represents the percentage of total participants with that level of experience.

employees). The most common team size that respondents worked in was 5-9 people (31%), followed by team sizes with 10-20 people (25%), more than 30 people (19%), less than 5 people (17%), and 21-30 people (8%).

The system age mentioned in TD examples was typically between 2 and 5 years old (35%), but we found systems with 1 to 2 years of age (23%), 5 to 10 years old (16%), less than 1-year-old (16%) and more than ten years old (10%). The most common system size was between 10 KLOC and 1 million LOC (61%), followed systems with size between 1 to 10 million LOC (17%), less than 10 KLOC (14%), and more than 10 million LOC (7%).

Finally, concerning the process used by respondents, 44% was hybrid (a combination of agile and traditional practices), 41% was agile, and 14% was traditional.

Thus, in general, although it is not possible to guarantee that the participants represent all the professionals in the software industry of the surveyed countries, the sample set encompasses a broad and diverse set of professionals who perform different roles and have experience levels, including different sizes of organizations and projects of different ages, sizes, team sizes and process models.

4.2 Have software practitioners paid off TD in their projects? (RQ1)

To answer RQ1, we analyzed the survey's responses for Q26. In this question, we verify whether the TD item described by the participant in Q13 was paid off or not. In total, 60% (258) of the participants indicated that the TD item mentioned was not paid off. This is a worrying scenario because 87% of the participants indicated in Q15 that their example represents a situation that occurs very often or happens time to time in the project.

To better understand the reasons that lead development teams to not eliminate the debt from their projects, we go further into this RQ and investigate the reasons that hamper the payment of TD items in RQ3. The same is performed in RQ2 to understand the practices used for TD payment.

4.3 What are the main TD payment practices used by software practitioners to pay off TD? (RQ2)

In Q26, 174 participants indicated that the TD item was paid off and 165 of them explained how (in Q27). From Q27, we have identified 34 practices related to the payment of TD items. These are all listed in Figure 1 (Section 5.1). Table 3 summarizes the 10 most common ones. The table reports the practice name, the total number citations for the practice (#CP), and the percentage of CP in relation to the total of cited practices (%CP). In total, the top 10 practices related to the payment of TD items correspond to ~70% of the overall frequency of citations.

4.3.1 Technical and non-Technical Practices. One can classify the identified practices in technical and non-technical aspects of a software development. Technical aspects refer to activities performed during software development that involve technical issues. On the other side, non-technical aspects are related to managerial issues. The technical subset is composed of *code refactoring*, *design refactoring*, *updating system documentation*, *adoption of good practices*, and *solving technical issues*, representing 42.6% of all the identified practices. The non-technical subset is composed of *investing effort on TD payment activities*, *monitoring and controlling project activities*, *investing effort on testing activities*, *prioritizing TD items*, and *negotiating deadline extension*, representing 26.9% of all the identified practices. This is an indication that technical aspects are decisive when eliminating debt items, but we still need to consider managerial issues.

4.3.2 Types of TD Payment related Practices. By going further into the analysis of the whole set of identified practices related to the payment of TD items, we realize that some practices do not directly allow the elimination of TD items. For instance, the practice *investing effort on TD payment activities* contributes to create a favorable scenario for eliminating TD items, but does not eliminate the item by itself. As a result of this analysis, we identified four types of practices related to TD payment:

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

Table 4 shows the identified types of practice, reporting the type's name, the number of unique practices cited (#P) and the total number (i.e., count) of practices (#CP) cited in each type. Column %CP indicates the percentage of CP in relation to the total of all cited practices. We can observe that the most cited type by survey participants (*payment practice*) represent 45.5% of the total citations. The types *defining a favorable setting for TD payment* and *TD prevention* were also commonly cited by the practitioners, revealing their concerning about having a favorable scenario to improve the use of the payment practices and to minimize the occurrence of debt.

Table 3: Top 10 Cited Practices related to TD Payment

Practice related to the TD payment	#CP	%CP
Code refactoring	64	26.4%
Investing effort on TD payment activities	21	8.7%
Design refactoring	16	6.6%
Investing effort on testing activities	12	5%
Monitoring and controlling project activities	12	5%
Prioritizing TD items	11	4.5%
Updating system documentation	9	3.7%
Negotiating deadline extension	9	3.7%
Increasing the project budget	8	3.3%
Using short feedback iterations	8	3.3%

Table 4: Types of Practices related to TD Payment

Type of Practices related to TD Payment	#P	#CP	%CP
Payment practice	8	110	45.5%
Defining a favorable setting for TD payment	12	67	27.7%
TD prevention	13	54	22.3%
TD prioritization	1	11	4.5%

4.3.3 Categories of TD Payment Related Practices. We also organized the set of practices related to the payment of TD items into eight categories:

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

Table 5 shows the identified categories of practices related to TD payment, reporting the category's name, the number of unique practices cited (#P), and the total number of practices cited in each category (#CP). Column %CP indicates the percentage of CP in relation to the total of all cited practices. We can observe that the most cited category by survey participants (*internal quality issues*) represents 33.1% of the total citations, indicating that this category plays a central role in practices related to the payment of TD items. The categories *methodology*, *development issues*, and *planning and management* were also commonly remembered as relevant when paying off TD.

The categories *people and organizational*, each corresponding to 2%, indicate that few participants have the improvement of the technical knowledge of the team and organizational environment as part of their TD payment initiatives. Finally, practices from the categories *infrastructure* and *external quality issues* are cited by only 1.7%. This result is not surprising due to the fact that TD is mostly related to internal quality issues in software products [18].

Table 5: Categories of Practices related to TD Repayment

Category of Practices related to TD Repayment	#P	#CP	%CP
Internal quality issues	2	80	33.1%
Methodology	13	70	28.9%
Planning and management	5	42	17.4%
Development issues	6	32	13.2%
People	3	5	2%
Organizational	2	5	2%
Infrastructure	2	4	1.7%
External quality issues	1	4	1.7%

4.4 What are the main reasons cited by software practitioners for not paying off TD items? (RQ3 and RQ3.1)

In Q26, 258 participants indicated that the TD item was not paid off and 213 of them explained how in Q27. Table 6 shows the 10 most commonly cited reasons for not paying off TD (out of a total of 28) identified in Q27. The complete list of reasons is presented in Figure 1 (Section 5.1). Table 6 reports the reason name, the total number (i.e., count) of reasons (#CR), and the percentage of CR in relation to the total of all cited reasons (%CR). In total, the top 10 reasons correspond to ~83% of the overall frequency of citations.

Table 6: Top 10 cited Reasons for TD Non-Payment

Reason for TD Non-Payment	#CR	%CR
Low priority	45	19.7%
Lack of organizational interest	37	16.2%
Focusing on short term goals	35	15.3%
Cost	28	12.2%
Lack of time	18	7.9%
Customer decision	10	4.4%
Lack of resources	8	3.5%
Complexity of the TD item	5	2.2%
Insufficient management view about TD repayment	5	2.2%
The project was discontinued	5	2.2%

4.4.1 Technical and non-Technical Reasons. The reasons *low priority*, *lack of organizational interest*, *focusing on short term goals*, *cost*, *lack of time*, *lack of resources*, *customer decision*, *the project was discontinued*, and *insufficient management view about TD repayment* are related to non-technical aspects of the software development, and represent almost 81.2% of the all identified reasons. On the other side, the reason *complexity of the TD item* is related to technical aspects, representing only 2.2% of all the identified reasons. This result clearly suggests that dealing with managerial issues is quite decisive if we are interested in improving the use of TD payment practices.

4.4.2 Types of Reasons for not Paying off TD. By going further into the analysis of the whole set of identified reasons for not paying TD items, we organized them into the following types:

- *Decision factor*: refers to reasons for deciding not to pay off the TD. Among them, we have *low priority*, *lack of organizational interest*, and *focusing on short-term goals*;
- *Impediment*: points to situations in which the development team wanted to pay off the TD, but they couldn't pay it off

for some reason. Examples are *cost*, *lack of time*, and *customer decision*.

Table 7 presents the types of reason, reporting the type's name, the number of unique reasons cited (#R), and the total number (i.e., count) of reasons (#CR) cited in each type. As before, the column %CR indicates the percentage of CR in relation to the total of all cited reasons. *Decision factors* represent 59% of the total citations, while the type *impediment* is 41%. This result indicates that impediments are common, but most of times, TD items are not eliminated from projects due to decisions of the team.

Table 7: Types of Reason for TD Non-Payment

Type of Reason for TD Non-Payment	#R	#CR	%CR
Decision factor	13	135	59%
Impediment	15	94	41%

4.4.3 Categories of Reasons for not Paying off TD Items. Regardless of their types (decision factor or impediment), we also classified the identified reasons into eight categories:

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

Table 8 presents the categories of reason, reporting the category's name, the number of unique reasons cited (#R) and the total number (i.e., count) of reasons (#CR) cited in each category. Once again, the column %CR indicates the percentage of CR in relation to the total of all cited reasons. One can notice that the most cited category (*planning and management*) represents 59% of the total citations, indicating that this category plays a central role in reasons for not eliminating TD items.

The category *organizational* was also commonly found (20.1%). The categories *external factors* and *people* correspond to only 7.4% and 3.9%, respectively, indicating that few participants have perceived those factors as decisive for not paying off debt items. The other categories are even less commonly cited.

Table 8: Categories of Reason for TD Non-Payment

Category of Reason for TD Non- Payment	#R	#CR	%CR
Planning and management	7	135	59%
Organizational	3	46	20.1%
External factors	4	17	7.4%
People	3	9	3.9%
Methodology	5	6	2.6%
Internal quality issue	2	6	2.6%
Development issues	2	5	2.2%
Lack of knowledge	2	5	2.2%

4.4.4 Relationship between Technical/non-Technical Role of Software Practitioners and Reasons that Curb the Application of TD Payment Practices (RQ3.1). For this analysis, we considered only the participants who cited at least one reason (decision factor or impediment) for not paying the TD, resulting in the selection of 178 participants. After, we divided the participants into two subgroups. The non-technical subgroup (54 participants) is composed of the role project leader / manager while the technical subgroup (173) includes all other roles.

Table 9 shows the relationship between role group and type of reason, reporting, for each group, the quantity of decision factors and impediments. Non-technical participants cited 32 (23.9%) and 22 (23.6%) decision factors and impediments, respectively. On the other hand, technical participants cited 102 (76.4%) and 71 (76.1%) decision factors and impediments, respectively. To verify whether this difference is significant between the groups, we performed the Pearson's Chi-squared statistical test with 95% of confidential level. The resulting p-value was 0.9688, indicating that there is not significant difference between the groups. This result suggests that participant's role (technical or non-technical) does not influence the type of reason considered by the participant for not paying the debt.

Table 9: Relationship Between Role Type and Reason Type

Role Group	Type of Reason	
	Decision Factor	Impediment
Non-Technical	32	22
Technical	102	71

5 Discussion

This section presents a map that can be used by software development teams in TD payment activities. It then compares the categories of practices and reasons proposed in this article with the categories of TD payment strategies proposed by Li *et al.* [7]. Lastly, it summarizes the main findings of this work.

5.1 Map for TD Payment

Figure 1 shows our map for supporting development teams in TD payment activities. The map organizes practices and reasons grouped by category. The categories are represented by dashed-line rectangles. Small rectangles indicate when a practice or reason is related to technical aspects. A black rectangle means that a practice or reason has a technical characteristic while a white

rectangle means it is non-technical. For example, in *reasons*, the category *external factors* has three non-technical reasons (*the project was discontinued*, *customer decision*, and *TD items do not affect the user*) and only one technical reason (*lack of access on component code*).

Small circles represent the type of practice or reason. We used a specific color for each type. For instance, in *practices*, the category *planning and management issues* has three green practices (*negotiating deadline extension*, *increasing the project budget*, and *conducting risk analysis*) of the type *defining a favorable setting for TD payment*, one yellow practice (*monitoring and controlling project activities*) of the type *TD prevention*, and one orange practice (*prioritizing TD items*) of the type *TD prioritization*.

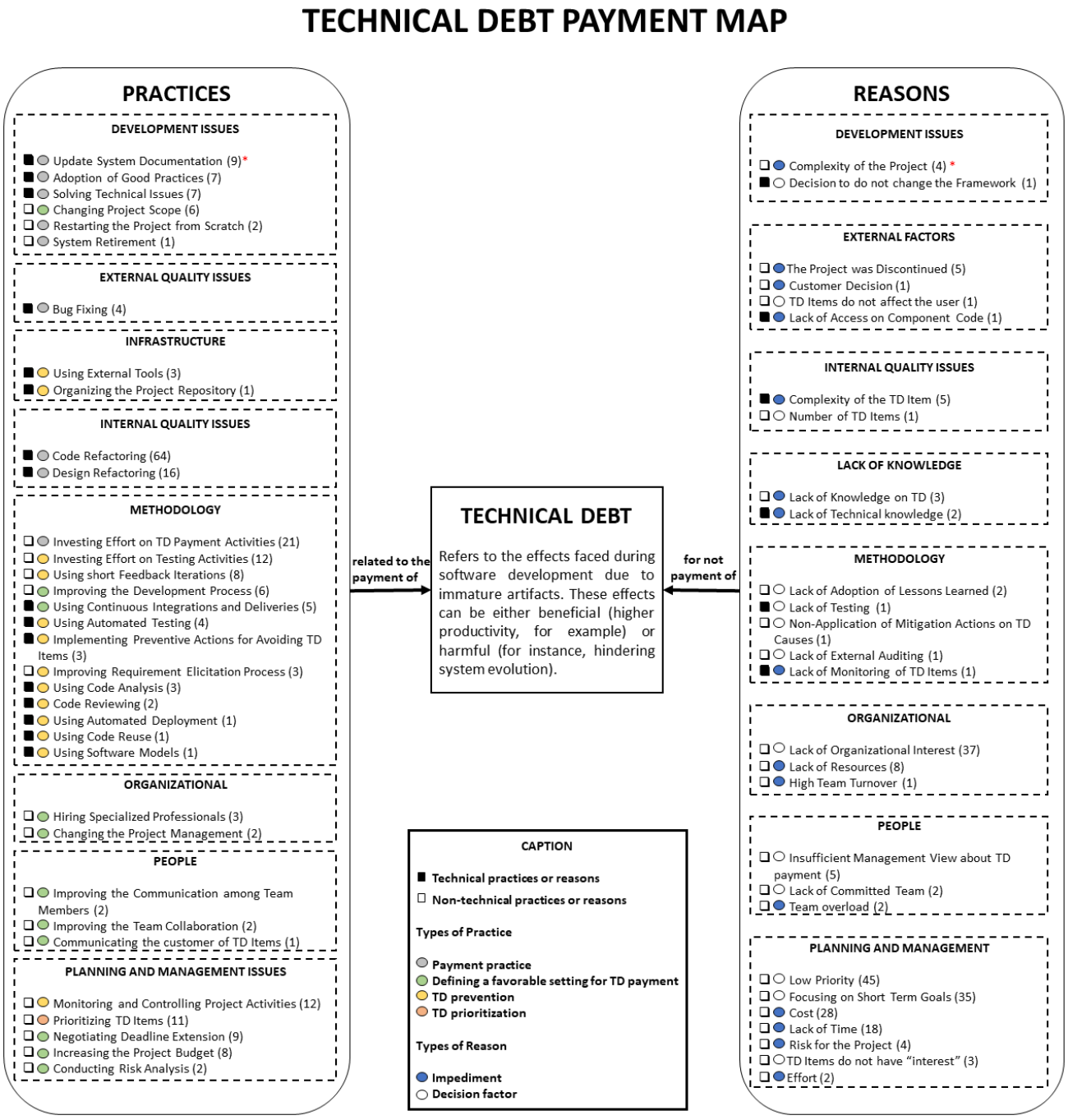

Figure 1: Map for supporting TD payment.

As a conceptual guide, the map can be employed to inform practices in response to the need for TD payment, and to identify reasons that explain the non-application of these practices. The map facilitates the identification of practices related to TD payment and reasons by highlighting aspects of the software development process and the nature of the activity (technical or non-technical). Besides, the type of practice or reason can support the identification of practices that facilitate, eliminate, prevent or prioritize the TD payment and the main impediments or decisions that hamper the application of those practices. For example, when a software team decides to pay TD items off, it can use the practices of the type *payment practice*. Project managers can analyze the impediments and practices of the type *defining a favorable setting for TD payment* to understand the TD payment context and support their software teams in using payment practices. Even more specifically, the category *planning and management issues* existing in *practices* and *reasons* shed light about the main issues that can be improved by managers in the context of TD payment.

By assisting in making the practices and the reasons visible, as a communication device, the map can be used to support development teams to more effectively communicate technical problems to management, and for managers to make better-informed decisions concerning TD payment. We believe that the map can help to create a more favorable environment for eliminating debt items.

5.2 Comparison to Related Work

Table 10 shows the categories of TD payment practices we define in this work, their relationship with the categories reported by Li *et al.* [7], and the overlapping degree of this relation. The overlapping degree between the categories can be *total*, indicating that all practices from a category identified in our work are considered in the work from Li *et al*. [7] and vice-versa, and *partial*, indicating that our work has practices that are not considered in the related work.

Table 10: Comparison to Related Work

Our Categories	Categories from Li *et al.* [7]	Overlapping Degree
Internal quality issues	Refactoring	Total
	Rewriting	Total
Methodology	Automation	Partial
	Repackaging	Partial
Development issues	Reengineering	Partial
	Fault tolerance	Partial
External quality issues	Bug fixing	Total
Planning and management	-	-
People	-	-
Organizational	-	-
Infrastructure	-	-

The comparison of the results reveals that the point of view of software practitioners (from *InsighTD*) confirms the results from the technical literature (from Li *et al*. [7]) that practices related to *internal quality issues*, *methodology*, *development issues*, and *external quality issues* can be employed to eliminate debt items from software projects. In addition, eliciting the opinions of software practitioners allowed us to find that aspects related to *planning and management*, *people*, *organizational issues*, and *infrastructure* are also necessary to support TD payment initiatives.

In summary, our findings reveal new categories and practices related to TD payment and complement those defined by *Li et al*. [7]. Reasons for TD non-payment were not approached in the previous study.

5.3 Summary of Findings

The main takeaways of this paper can be summarized in the following points:

- The payment of TD items seems not to be a common practice yet in software organizations;
- *Code refactoring*, *design refactoring*, and *update system documentation* are the most cited TD payment practices;
- Technical practices are more commonly used for TD payment than non-technical practices;
- Besides payment practices, practitioners also have concerns about prevention, prioritization, and definition of a favorable setting for TD payment;
- *Low priority*, *lack of organizational interest*, *focusing on short term goals*, and *cost* are the most cited reasons for TD non-payment;
- The non-payment of TD occurs due to two main types of reasons (*decision factors* and *impediments*), regardless the participant's roles (technical and non-technical);
- Non-technical aspects are more commonly seen as reasons for not eliminating debt items. Therefore, resolving issues originated from managerial aspects should be the main concern in creating a favorable scenario to apply practices for the payment of TD items;
- A map for supporting TD payment was defined by organizing all practices and reasons grouped in their respective types and categories.

6 Implications for Practitioners and Researchers

Software practitioners can use the identified practices and reasons for not paying TD off (Tables 3 and 5) considering their frequency of occurrence as TD management criteria. In addition to the frequency, software practitioners can identify the types and categories that group practices or reasons by using the map presented in Figure 1. These types and categories help to recognize practices or reasons that are related to each other, shedding light in possible combinations of practices for paying off TD items, or drawing conclusions about reasons that hinder the application of TD payment in the workplace.

The map can also help software practitioners to envision whether practices and reasons are technical or not. The map shows that most practices are technical, implying that improving technical knowledge can be a good way to help eliminate TD items. The map also indicates that most reasons are non-technical, indicating that the organization and its work force must understand the impact of TD items and teamwork in providing a good environment for paying off TD.

For researchers, our results help to support the development of new research agendas on practices for TD payment and reasons for TD non-payment. The presented top 10 list of practices, the types and categories of practices, the top 10 list of reasons, the types and categories of reasons, and the map for TD payment can guide new investigations in a problem-driven way. For example, the development of new approaches for TD payment could consider the combination of practices from different categories.

7 Threats to Validity

We identified some threats to validity in our work [19]. For each one, we sought to remove it when possible or mitigate its effect when removal was not possible.

The coding process represents the main threat that could affect this study with respect to conclusion validity. Coding is subjective and subject to inconsistency. To mitigate this negative effect, the coding process was performed by three researchers playing different roles: code identifier, code reviewer, and referee.

The main threat to internal validity arises from the *InsighTD's* questionnaire because its questions are answered remotely, allowing misunderstanding that can lead to meaningless answers. To reduce this threat, Rios *et al*. [9] indicated that three internal and one external reviewer assessed the questionnaire, and a pilot study was performed before the execution of the questionnaire.

Lastly, there is the threat to external validity. For this, we targeted industry practitioners and sought to achieve respondent diversity from several countries.

8 Concluding Remarks

This work identifies the point of view of practitioners on TD payment. We also report the most common practices related to TD payment and the most commonly cited reasons for not applying these practices in software projects. Those practices and reasons are organized into types and categories that summarize the main concerns of practitioners on TD payment. All this information was organized in a map that can be used to inform practices in response to the need to eliminate debt items and to identify reasons that could hamper the application of these practices in software organizations.

The next steps of this research include: (i) to use data from future *InsighTD* replications to increase the external validity of our results, (ii) to run deeper analyses to investigate whether TD practices and impediments are impacted by TD type, used process model, participant experience, and organization/project size, and (iii) to assess the proposed map empirically with respect to its effectiveness for supporting TD payment.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Clemente Izurieta, Antonio Vetrò, Nico Zazworka, Yuanfang Cai, Carolyn Seaman, and Forrest Shull, 2012. Organizing the technical debt landscape. In *Proceedings of Third International Workshop on Managing Technical Debt* (MTD), Zurich, p. 23-26. DOI: 10.1109/MTD.2012.6225995

[2] Rodrigo O. Spínola, Nico Zazworka, Antonio Vetrò, Forrest Shull, and Carolyn Seaman, 2019. Understanding Automated and Human Based Technical Debt Identification Approaches ? A Two-Phase Study. *Journal of the Brazilian Society* (Online), v. 1, p. 1. DOI: https://doi.org/10.1186/s13173-019-0087-5

[3] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya, 2012. Technical debt: from metaphor to theory and practice. *IEEE Software*, vol. 29, no. 6, pp. 18-21, Nov.-Dec. 2012. DOI: 10.1109/MS.2012.167

[4] Isaac Griffith, Hanane Taffahi, Clemente Izurieta, and David Claudio, 2014. A simulation study of practical methods for technical debt management in agile software development. In *Proceedings of the 2014 winter simulation conference*, Savanah, GA, p. 1014-1025. DOI: 10.1109/WSC.2014.7019961

[5] Nicolli Rios, Manoel Mendonça Neto, and Rodrigo O. Spínola, 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, vol. 102, June, p. 117-145, ISSN 0950-5849. DOI: 10.1016/j.infsof.2018.05.010

[6] Nanette Brown, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCoemack, Robert Nord, Ipek Ozkaya, Raghvinder Sangwan, Carolyn Seaman, Kevin Sullivan, and Nico Zazworka. 2010. Managing technical debt in software-reliant systems. In *Proc. of the FSE/SDP workshop on Future of software engineering research*. ACM, New York, NY, USA, 47-52. DOI: http://doi.acm.org/10.1145/1882362.1882373

[7] Zengyang Li, Paris Avgeriou, and Peng Liang. 2015. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, v. 101, p. 193–220.

[8] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. 2016. How do software development teams manage technical debt? - An empirical study. *Journal of System and Software*, 120, 195-218, October.

[9] Nicolli Rios, Rodrigo O. Spínola, Manoel Mendonça, and Carolyn Seaman. 2018. The most common causes and effects of technical debt: first results from a global family of industrial surveys. In *Proceedings of ACM/IEEE 12th International Symposium on Empirical Software Engineering and Measurement* (ESEM), Oulu. ACM, New York, NY, USA, Article 39, 10 pages. DOI: 10.1145/3239235.3268917

[10] Nicolli Rios, Manoel Mendonça, Carolyn Seaman., and Rodrigo O. Spínola. 2019. Causes and effects of the presence of technical debt in agile software projects. In *Proceedings of the Americas Conference on Information Systems* (AMCIS), Cancun, Article 3, 10 pages.

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

[12] Sávio Freire, Nicolli Rios, Manoel Mendonça, Davide Falessi, Carolyn Seaman, Clemente Izurieta, and Rodrigo O. Spínola. Actions and impediments for technical debt prevention: results from a global family of industrial surveys. To appear in *Proceedings of The 35th ACM/SIGAPP Symposium On Applied Computing* (ACM/SAC), Brno. ACM, New York, NY, USA, 8 pages. DOI: https://doi.org/10.1145/3341105.3373912

[13] Yuepu Guo, Rodrigo O. Spínola, and Carolyn Seaman. 2014. Exploring the costs of technical debt management– a case study. *Empirical Software Engineering*, J. 1, p. 1–24. DOI: https://doi.org/10.1007/s10664-014-9351-7

[14] Carolyn Seaman and Yuepu Guo. 2011. Measuring and monitoring technical debt. 1. ed. [s.l.] Elsevier Inc., v. 82.

[15] Carolyn Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557-572.

[16] Anselm Strauss and Juliet M. Corbin. 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*. Sage Publications.

[17] Stevem McConnell. 1007. *Technical debt 10x Software Development Blog. Construx Conversations*. URL= http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx, 2007.

[18] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. Managing Technical Debt in Software Engineering. Dagstuhl Seminar 16162. In *Dagstuhl Reports*. vol. 6, n. 4. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

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