Technical Debt is not Only about Code and We Need to be Aware about It

Clara Berenguer
Salvador University
Brazil
claraberenguerledo@gmail.com

Adriano Borges
Salvador University
Brazil
arborges.12@gmail.com

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

Nicolli Rios
Federal Univ. of Rio de Janeiro
Brazil
nicolli@cos.ufrj.br

Robert Ramač
University of Novi Sad
Serbia
ramac.robert@uns.ac.rs

Nebojša Taušan
Chamber of Commerce and Industry
Serbia
nebojsa.tausan@infora.rs

Boris Pérez
Univ. of Los Andes and Francisco de Paula S/der University
Colombia
br.perez41@uniandes.edu.co

Camilo Castellanos
University of Los Andes
Colombia
cc.castellanos87@uniandes.edu.co

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

Alexia Pacheco
University of Costa Rica
Costa Rica
alexia.pacheco@ucr.ac.cr

Gustavo López
University of Costa Rica
Costa Rica
gustavo.lopez_h@ucr.ac.cr

Davide Falessi
University of Rome Tor Vergata
Italy
d.falessi@gmail.com

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

Vladimir Mandić
University of Novi Sad
Serbia
vladman@uns.ac.rs

Clemente Izurieta
Montana State University and Idaho National Laboratories
United States
clemente.izurieta@montana.edu

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

ABSTRACT

Context: It is common for a software project to incur technical debt (TD) during its development. It can impact several artifacts produced throughout the software development process. Therefore, it is necessary to carry out management actions to find a balance between the benefits of incurring it and the effects of its presence. However, so far, much of the attention has been given only to discussions relating TD to coding issues. This is a worrying scenario because other types of debt can also have impactful, or even worse, consequences on projects. **Aims**: This study elaborates on the need to consider other issues of the development process and not just the source-code when investigating the TD phenomenon. **Method**: We analyze responses from 653 practitioners concerning TD causes, effects, prevention, reasons for non-prevention, repayment, and reasons for non-repayment and investigate whether these TD management elements are related to coding or to other software development issues. **Results**: Coding issues are commonly related to the investigated elements but, indeed, they are only part of the big picture we draw. Issues related to the project planning and management, human factors, knowledge, quality, process, requirements, verification, validation, and test, design, architecture, TD management, and the organization are also common. Lastly, we present a hump diagram that, in combination with the detailed results, provides guidance on what to expect from the presence of TD and how to react to it considering several issues of software development. **Conclusion**: The results shed light on other concerns beyond code that the research community and practitioners need to be aware of.

SBQS'21, November, 2021, Vitória, Brazil

https://doi.org/XXX

CCS CONCEPTS

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

KEYWORDS

Technical debt, Technical debt management, Causes, Effects

ACM Reference format:

Clara Berenguer, Adriano Borges, Sávio Freire, Nicolli Rios, Robert Ramač, Nebojša Taušan, Boris Pérez, Camilo Castellanos, Darío Correal, Alexia Pacheco, Gustavo Lopez, Davide Falessi, Carolyn Seaman, Vladimir Mandić, Clemente Izurieta, and Rodrigo Spínola. 2021. Technical Debt is not Only about Code and We Need to be Aware about It. In *Proceedings of 20th Brazilian Symposium on Software Quality (SBQS'21). ACM, New York, NY, USA, 10 pages.* https://doi.org/10.1145/1234567890

1 Introduction

Technical debt (TD) conceptualizes technical compromises that can bring short-term benefits (e.g., higher productivity and lower costs) but may negatively impact the long-term health of software projects [1]. Although initially TD was associated only with code level issues, it can also impact other artifacts of the software development process, such as documentation and requirement engineering [2]. If not managed, accumulated debt yields risks associated with, among others, unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project [3].

Successful TD management is about reaching a balance between the benefits of incurring it and later impacts of its presence [4, 5]. Managing TD involves making decisions related to whether a debt item should be repaid and the most appropriate time to do it [6]. It also includes preventive actions, as preventing debt items in earlier stages of software development can reduce the chances of those items impacting development activities later on [7].

Research on TD management related to understanding the causes that lead development teams to incur debt items in their projects and their effects have also been done [3, 8]. Knowing TD causes can support development teams in defining TD prevention actions. Having information on TD effects can aid in the prioritization of TD items to pay off, by supporting a more precise impact analysis and the identification of corrective actions to minimize possible negative consequences for the project.

Despite the growing number of studies on TD, there is a clear concentration of studies investigating TD from the source code and its related artifacts perspective, suggesting tools to support its management [2, 9] and identification [2, 13]. But, if debt items are present in requirements specifications or test cases, then how can they be identified? Or, until which point are those items tolerable for the project? Focusing from a code-related perspective alone can bring risks to software projects, because TD can affect other artifacts produced throughout the software development process.

This paper elaborates on growing need to expand TD research to other areas of software development. To this end, we analyze six elements related to TD management: causes, effects, preventive practices, reasons for non-prevention, repayment practices, and reasons for non-repayment. We use a subset of the data collected by the InsighTD project, a family of surveys globally distributed on causes, effects, and management of TD [3]. The subset consists of data from six replications, totaling 653 responses from software practitioners. By investigating how practitioners face TD in their projects, we gained insight into the state of practice regarding TD management, which allow us to identify existing gaps in TD management theory. The data are analyzed qualitatively and quantitatively to investigate whether those TD management elements are more related to coding or other issues (e.g., planning and management, requirements engineering, human factors) of the software development.

Overall, the results indicate that coding issues are commonly affected by TD but, indeed, they are only a part of the big picture of TD management. Given all the investigated TD management elements, other software development issues are more commonly reported by practitioners. *Planning and management issues* and *human factors* stand out, but there are several other issues involved with the presence of debt items such as *process*, *knowledge*, *TD management*, and *requirement engineering issues*. Results are conveniently presented with a hump diagram that, in combination with the analyses of each of the investigated TD management elements, provides guidance on what to expect from the presence of TD and how to react to them considering several issues of the software development process.

In addition to this introduction, this paper has six additional sections. Section 2 presents background information on TD research and the InsighTD project. Section 3 describes the methodology used. Then, Section 4 presents the results, which are discussed in Section 5. Section 6 discusses the threats to validity. Finally, Section 7 presents final considerations.

2 Background

In this section we first discuss taxonomy of TD, followed by the introduction of the InsighTD project.

2.1 Taxonomy of TD

TD occurs in several artifacts over the software development process, having different characteristics depending on the time it is incurred and on the activities it is associated with [2]. However, even with the growth of research in the area, much effort is still concentrated on solely investigating TD at the source code level.

Li *et al.* [6] classified TD into ten types. Code debt was the most studied type among the primary articles analyzed. In another mapping study of the area, Alves *et al.* [2] reported artifacts that have been frequently used to identify debt items in software projects, pointing out a greater focus on strategies for identifying debt items from the source code. On the other hand, other artifacts such as requirements specification, documentation, and test reports, among others, were only mentioned occasionally. The authors suggested that the concentrated focus on source code may be related to the existence of several tools that perform static analysis of the code, supporting the detection of debt items.

More recently, Rios *et al.* [9] updated the taxonomy of types of debt to fifteen types. Again, the main focus of investigations has also been on types that are related to the source code. The authors reported that one of the possible explanations for this is the influence of the concept of TD coined by Ward Cunningham [14], which focuses specifically on development activities. Another reason for this concentration can be that the types related to code tend to cause effects that can be felt more quickly by development teams [9].

Such concentration of studies at the coding level is a worrying scenario because other types of debt can also have impactful, or even worse, consequences on projects. We claim that it is necessary to go beyond the source code and investigate other facets of TD. We do it under the perspective of TD causes, effects, prevention, and repayment, and use data collected from InsighTD project, presented in the next section.

2.2 The InsighTD Project

The InsighTD project is a family of surveys that have been carried out with industry practitioners in several countries to investigate the causes, effects, and how professionals deal with TD in their projects [3]. So far, several results from the project have been disseminated as shown at http://www.td-survey.com/publication-map/.

From the InsighTD data, we have reported the general list of (i) causes and effects of TD [3], (ii) preventive practices and reasons for not applying these practices [7], and (iii) repayment practices and reasons for not applying these practices [11], and the relationship between TD management and process models [12]. Although these results provide an initial view of TD management, we did not run specific analysis to investigate if the TD elements (causes, effects, prevention practices, reasons for non-prevention, repayment practices, and reasons for non-repayment) were related or not to coding issues. In this work, we fill that gap by investigating whether the TD management elements are more related to coding or other software development issues.

3 Method

This section presents our research questions and the data collection and analysis procedures.

3.1 Research Questions

The main research question (RQ) is "*Are the TD management elements (causes, effects, prevention, and repayment) more related to coding issues or to other software development issues?*". Through this question, we intend to shed light on the importance of other software development issues when dealing with TD, by reporting evidence from industry on the topic. Thus, we consider the following sub-questions:

RQ1: Are the causes of TD more related to coding issues or other software development issues?

RQ2: Are the effects of TD more felt in coding issues or other issues in the software development process?

RQ3: Is TD prevention more related to coding issues or other issues in the software development process?

RQ4: Are the reasons for not preventing TD more related to coding issues or other development issues?

RQ5: Is TD repayment more associated with coding issues or other issues in the software development process?

RQ6: Are the reasons for not paying TD more related to coding issues or other development issues?

3.2 Data Collection

This study uses a subset of available data from 18 questions of the *InsighTD* questionnaire. Table 1 presents these questions, reporting their type and the RQ they are related to.

Questions Q1 thru Q8 capture the characterization of the survey respondents. In Q13, they provide an example of a TD item that occurred in their projects. Participants discuss causes of TD in Q16 thru Q18 and effects in Q20. We use the answers given to these questions for answering RQ1 (Q16-Q18) and RQ2 (Q20). Concerning TD prevention, participants give their responses in Q22 and Q23, and address TD repayment in Q26 and Q27. The answers given in these questions are used for answering RQ3-4 (Q22 and Q23) and RQ5-6 (Q26 and Q27).

Table 1. Subset of the InsighTD survey's questions (adapted from [3]).

RQ	No.	Question (Q) Description	Type
-	**Q1**	What is the size of your company?	Closed
-	**Q2**	In which country are you 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
-	**Q10**	In your words, how would you define TD?	Open
-	**Q13**	Please give an example of TD that had a significant impact on the project that you have chosen to tell us about:	Open
RQ1	**Q16**	What was the immediate, or precipitating, cause of the example of TD you just described?	Open
RQ1	**Q17**	What other cause or factor contributed to the immediate cause you described above?	Open
RQ1	**Q18**	What other causes contributed either directly or indirectly to the occurrence of the TD example?	Open
RQ2	**Q20**	Considering the TD item you described in question 13, what were the impacts felt in the project?	Open
RQ3-4	**Q22**	Do you think it would be possible to prevent the type of debt you described in question 13?	Closed
RQ3-4	**Q23**	If yes, how? If not, why?	Open
RQ5-6	**Q26**	Has the debt item been repaid (eliminated) from the project?	Closed
RQ5-6	**Q27**	If yes, how? If not, why?	Open

We invite only software practitioners from the Brazilian, Chilean, Colombian, Costa Rican, North American, and Serbian software industries through LinkedIn, industry-affiliated member groups, and industry partners for answering the survey.

3.3 Data Analysis Procedures

The analysis procedures are divided into three steps: demographics, preparing data for analysis, and data classification and analysis.

3.3.1 Step 1 - Demographics. We calculated the number of respondents choosing an option available through the closed questions of the survey. Afterwards, we summarize the participants' characterization.

3.3.2 Step 2 - Preparing Data for Analysis. We applied a coding process for the open-ended questions [15]. In answers given to Q16 thru Q18 and Q20, we followed the coding process previously described in [3], resulting in a set of causes and effects and their respective number of occurrences. In answers given to Q23, we performed the coding process described in [7]. From this process, we identified practices for TD prevention when Q22 received a positive response, otherwise, we recognized reasons for TD non-prevention. Lastly, following the coding process described in [11], we coded the answers given to Q27. Similarly, when Q26 received a positive answer, we identified TD repayment practices, otherwise, we identified reasons for non-repayment. For both prevention and repayment, we also had a list of practices and reasons, and their corresponding number of occurrences.

The coding process was performed by at least two researchers from each replication. The first codified list of causes, effects, practices for prevention, reasons for not preventing, repayment practices, and reasons for not repaying was done by the Brazilian replication team and was sent to the other replication teams to standardize the used nomenclature. The consistency was verified by the Brazilian replication team.

3.3.3 Step 3 - Data Classification and Analysis. Initially, we analyzed the codes of each TD management element and defined if they are related to coding issues or other software development issues. For example, the repayment practices *bug fixing*, *code refactoring*, and *using code reuse* were classified as practices related to coding issues. On the other hand, the repayment practices *prioritizing TD items* and *update system documentation* were associated with other software development issues. This process was performed by the first and second authors independently. The consensus was done by the third (prevention and repayment) and forth (causes and effects) authors. Further, the final classification was reviewed by the last author.

Next, we grouped the TD management elements related to the other software development issues into categories following the grouping process defined by [15]. The categories reveal the relationship among issues of the software development process (e.g., requirement engineering issues, planning and management issues, human factors issues) and each TD management element. The categories' names arise from the continuous process of grouping the TD management elements around the central concern to which they are related. For example, the causes *deadline* and *inappropriate planning* are part of the category *planning and management issues*, while the effects *team demotivation* and *dissatisfaction of the parties involved* compose the category *human factors*. This process was conducted by the first and second authors independently. The consensus was done by the third (prevention and repayment) and forth (causes and effects) authors, and the final result was reviewed by the last author.

4 Results

Participants were required to provide a definition (Q10) and an example of a significant TD item (Q13). As detailed in [3], we used the responses for these questions as the inclusion criterion of the participants. In total, we consider responses from 653 practitioners from six countries (Brazil=107, Chile=89, Colombia=134, Costa Rica=145, Serbia=79, and the United States=99). The upcoming subsections detail the demographics and the responses for the posed research questions.

4.1 Demographics

Figure 1 presents demographic information. Overall, although it is not possible to guarantee that the participants represent all the professionals in the software industry of the surveyed countries, the sample encompasses a broad and diverse set of professionals.

4.2 The relation between TD management elements and software development issues

The results indicate that coding issues related to the causes, effects, prevention, non-prevention, repayment, and non-repayment of TD are only a small part of the concerns that practitioners face in the presence of TD. Indeed, TD has been more commonly found in other software development issues.

The radar graph presented in Figure 2 shows the percentages of the distribution of the participants' responses to each of the investigated elements concerning the categories *coding issues* and *other software development issues*. We calculated the percentages considering all citations of the participants for each TD management element. For example, 13% (8) of the citations of reasons for not preventing TD are related to coding, while 87% (55) are associated with other development issues. For every investigated element, most of the responses are related to other software development issues. The difference is quite bigger for the elements: causes, prevention, reasons for not preventing, and reasons for not repaying. The values for TD repayment are very close between the two groups (56% vs 44%). This is an indication that, although practitioners perceive that TD is ubiquitous in software development projects, they also see that its repayment is commonly related to coding issues.

We present the detailed results of each investigated TD management element in the following subsections. We use the same structure when describing the results. For example, for the element TD cause, initially we (i) present the overall result. Next, we (ii) discuss the causes related to coding issues. Then, we (iii) present the causes related to the other software development issues, and (iv) analyze which are the types of those issues (e.g., planning and management, human factors, knowledge issues).

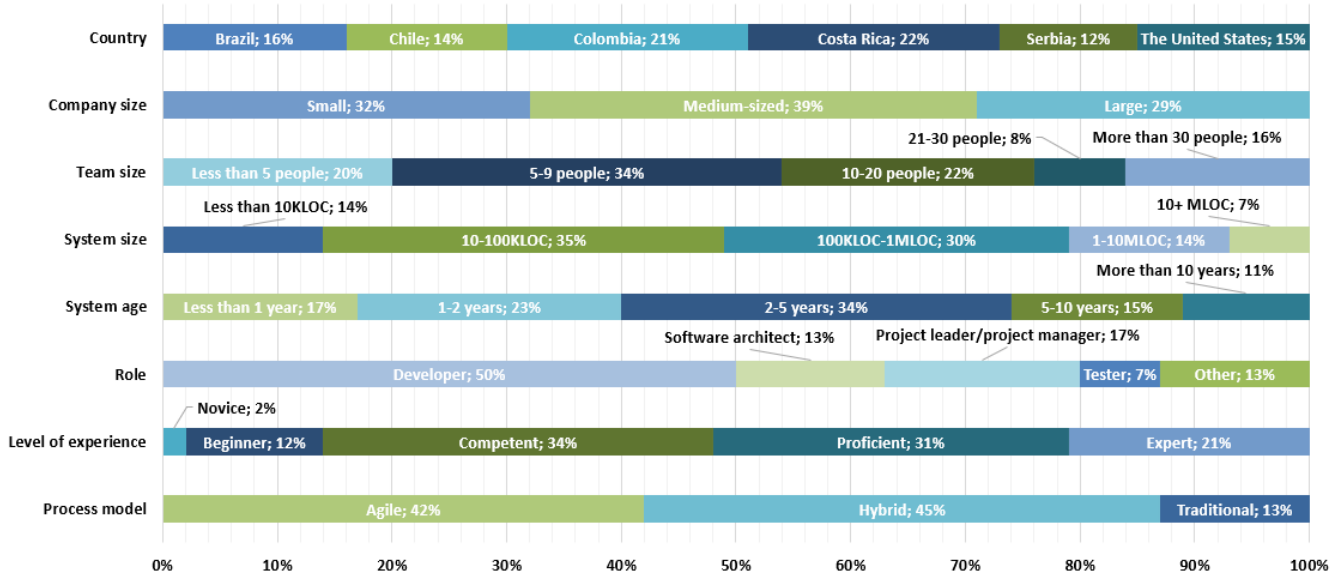

Figure 1: Participants' demographics.

4.2.1 RQ1 - Are the causes of TD more related to coding issues or other software development issues? In total, 96 causes that lead to the occurrence of TD were identified, totaling 1695 citations. Of this total, ~92% were related to other development issues, while only ~8% were related to code. This indicates a significant difference between the two subsets, representing a tendency of other software development issues to have a big influence on the occurrence of TD items.

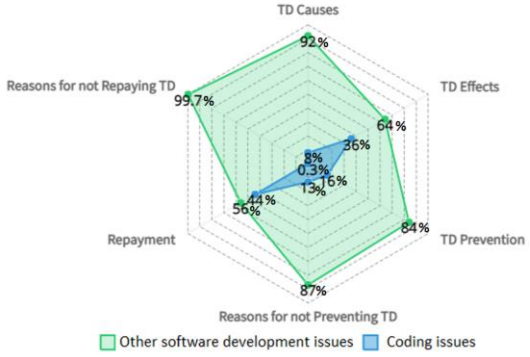

Figure 2: Participant response distribution

There are 13 causes related to coding issues. The five most commonly cited are presented in the second column of Table 2. The complete list is available at https://bit.ly/3zoAtFL. The causes *non-adoption of good practices*, *sloppy code*, and *lack of refactoring* stand out. All of them indicate issues that compromise the internal quality of the product.

Alternatively, we identified 83 causes related to other software development issues. The three most commonly (third column of Table 2) cited causes reflect concerns focused on project management and planning: *deadline*, *not effective project management*, and *inappropriate planning*. Other issues related to the team's lack of technical knowledge and processes were also commonly mentioned.

Table 2. The five most cited causes related to coding and other development issues

	Coding		Other development issues	
	Cause	**#**	**Cause**	**#**
1st	Non-adoption of good practices	54	Deadline	169
2nd	Sloppy code	21	Not effective project management	98
3rd	Lack of refactoring	17	Inappropriate planning	83
4th	External component dependency	12	Lack of technical knowledge	80
5th	Adoption of contour solutions as definitive	11	Producing more at the expense of quality	67

We observed that those causes were related to each other and grouped them, identifying 14 categories of causes that reflect the main concerns that practitioners have during the development of software projects:

- Planning and management: refers to causes related to the project's planning and management issues. Some examples are *deadline and inappropriate planning;*
- Human factors: groups causes related to people's participation in project issues. Some examples are *lack of experience* and *lack of commitment*;
- Knowledge issues: groups items originating from concerns around the knowledge of team members. Two examples are *lack of technical knowledge* and *lack of domain knowledge*;
- Requirements engineering: encompasses the causes related to requirements issues. Examples are: *change of requirements* and *requirements elicitation issues*;
- Verification, validation, and testing: encompasses the causes related to the execution of quality assurance issues. Two examples are *inappropriate/poorly planned/poorly executed test* and *lack of code review*;
- Architectural issues: groups causes related to decisions made regarding software architecture. Examples are: *inadequate technical decision* and *problems in architecture*;

- Process issues: refers to causes related to the definition or execution of the processes used in the development of the software. Two examples are *lack of a well-defined process* and *lack of traceability of bugs*;
- Design issues: encompasses causes related to the design of the software. There are two causes in this category: *poor design*, and *changes in design*;
- Documentation: groups causes related to documentation. Example of causes in this category are *nonexistent documentation* and *outdated/incomplete documentation*;
- External factors: refers to causes associated with external factors, such as *customer does not listen the project team* and *structural change in the involved organizations*;
- Infrastructure issues: encompasses causes related to problems in the software development infrastructure, such as *required infrastructure unavailable* and *updating existing tools*;
- Organizational issues: groups causes from the organizational context, such as *lack of awareness of the importance of testing and refactoring* and *organizational misalignment*;
- Quality issues: refers to causes (*lack of quality*) associated with lack of quality in software artifacts;
- TD Management: encompasses causes related to management of TD items. This category has only the cause *lack of perception of the importance of dealing with TD*.

Table 3 shows the categories together with the corresponding number of causes, number of citations, and percentage of the causes cited in relation to the other categories. The category planning and management stood out with ~47% of citations, representing more than three times the citations of the second ranked category. This is an indication that the causes of the occurrence of TD are strongly related to project management issues. The results also highlight the importance that human factors have, occupying the second position with ~13% of citations. This result is somehow aligned with previous work on social debt [16, 17]. Concerns related to requirements engineering and issues related to knowledge were also mentioned.

Table 3. Categories of causes not related to coding.

Categories of causes	#causes	#cited causes	~%cited causes
Planning and Management	22	733	47%
Human Factors	10	206	13%
Knowledge Issues	7	128	9%
Requirement Engineering	7	120	8%
VV&T	6	91	6%
Architectural Issues	6	63	5%
Process Issues	6	54	4%
Design Issues	2	45	3%
Documentation	4	37	2%
External Factors	4	25	2%
Organizational Issues	3	25	2%
Infrastructure Issues	4	15	1%
Quality Issues	1	12	1%
TD Management	1	1	0.1%

4.2.2 RQ2 - Are the effects of TD more felt in coding issues or other issues in the software development process? The participants reported a total of 73 TD effects, totaling 980 citations. Among them, ~64% are related to other development issues and ~36% are related to coding.

There are 18 coding-related effects experienced by the participants. The five most commonly cited are presented in Table 4 (second column). The full list is available at https://bit.ly/3zoAtFL. Concerns about the capacity of the team to evolve the code, rework, and the need of employing refactoring practices to improve the internal quality of the software are common. Other common effects are: *bad code* and *low performance*.

Table 4. The five most cited effects related to coding and other development issues

	Coding		Other development issues	
	Effects	**#**	**Effects**	**#**
1st	Low maintainability	97	Delivery delay	141
2nd	Rework	86	Low external quality	78
3rd	Need of refactoring	35	Financial loss	55
4th	Bad code	31	Increased effort	41
5th	Low performance	28	Stakeholder dissatisfaction	34

We identified 55 effects related to other development issues. The five most commonly (third column of Table 4) cited reflect concerns on the project management and planning (*delivery delay*, *increased effort*, *financial loss*), external quality of the product (*low external quality*), and human factors (*stakeholder dissatisfaction)*.

Table 5 shows the categories of effects not related to coding. The category *planning and management* has ~47% of citations, revealing that managerial aspects of software development are commonly affected by the presence of debt items. Next is the *human factors* category, with ~18% of the effects cited, showing that TD also impacts human aspects of software development. *Quality issues* are also a common concern. The other categories are less commonly cited.

Table 5. Categories of effects not related to coding.

Categories of effects	#effects	#cited effects	~%cited effects
Planning and Management	15	297	47%
Human Factors	7	110	18%
Quality issues	6	110	18%
VV&T	3	23	4%
Design Issues	2	21	3%
Knowledge issues	8	21	3%
Architectural Issues	4	18	3%
Organizational issues	3	10	2%
Documentation	1	6	1%
Process Issues	2	4	1%
Requirement Engineering	2	4	1%
Infrastructure Issues	1	3	0.5%
TD Management	1	2	0.3%

4.2.3 RQ3 - Is TD prevention more related to coding issues or other issues in the software development process? The data shows a total of 89 practices to support the prevention of TD items,

resulting in 819 citations. From this, ~84% are items related to other development issues, while only ~16% are associated with code. This result indicates a tendency for other development issues to play a key role in the prevention of TD.

We identified a total of 13 TD prevention practices related to coding. Table 6, second column, presents the five most cited items. The complete list is available at https://bit.ly/3zoAtFL. *Adopting good practices* and *using good design practices* reflect concerns that practitioners should have when carrying out their coding and design activities. The practices *refactoring* and *code review* are related to the continuous improvement of the code under development. Lastly, *increasing time for analysis and design* is related to concerns that teams must have around an adequate analysis of the functionalities.

Table 6. Top five most commonly cited TD prevention practices related to coding or other development issues.

	Coding		Other development issues	
	Prevention Practices	**#**	**Prevention Practices**	**#**
1st	Adoption of good practices	49	Well-defined requirements	57
2nd	Using good design practices	26	Better Project Management	43
3rd	Refactoring	12	Providing training	36
4th	Code review	10	Follow the proj. planning	34
5th	Increasing time for analysis and design	7	Improving software development process	33

On the other hand, we found 76 prevention practices related to other development issues. Table 6 (third column) shows the five most cited. Interestingly, they reflect different concerns through the software development process, such as management (*following the project planning* and *better project management*), the process itself (*improving software development process*), the documentation (*well-defined requirements*), and the qualification of the team (*providing training*).

We see in Table 7 that TD prevention practices are commonly related to project management issues (~34%). The results also highlight the importance that the process followed by the team have, ranking second (~12%) among the most cited categories. Concerns related to requirements, validation, TD management, and human factor were also commonly mentioned.

Table 7. Categories of prevention practices not related to code

Categories of prevention practices	#practices	#cited practices	~%cited practices
Planning and Management	21	232	34%
Process Issues	8	80	12%
Requirement Engineering	5	69	11%
VV&T	11	67	10%
TD Management	7	64	10%
Human Factors	11	61	9%
Knowledge Issues	4	51	8%
Documentation Issues	2	28	4%
Architectural Issues	3	27	4%
Organizational Issues	2	4	1%
Infrastructure Issues	2	3	1%

4.2.4 RQ4 - Are the reasons for not preventing TD more related to coding or other development issues? Participants reported 25 reasons that lead to the non-prevention of TD items, resulting in 63 citations. Of them, ~87% are related to other development issues, while only eight ~13% are related to coding. Again, other development issues have an important role in preventing TD.

There are only four reasons related to code leading teams not to prevent the occurrence of debt items: *lack of technical knowledge*, *lack of good technical solutions*, *lack of concern about maintainability*, and *continuous change of coding standards*. On the other hand, we found 21 reasons (the five most cited are presented in Table 8) not related to code. *Short deadline* was the most commonly cited.

Table 8. Top five most cited reasons for not preventing TD related to other development issues.

	Other development issues	
	Reason	**#**
1st	Short deadline	14
2nd	Ineffective management	7
3rd	Lack of predictability in the software development	5
4th	Requirements change	5
5th	Pressure for results	4

Table 9 shows the categories identified. *Planning and management* once again stands out with ~38% of citations. The other categories were less commonly cited, with less than seven citations. Although not too mentioned, the result suggests that other issues related to the software development can also negatively influence teams in TD prevention.

Table 9. Categories of reasons for TD non-prevention not related to coding.

Categories of reasons	#reason	#cited reasons	~%cited reasons
Planning and Management	2	21	38%
Requirement Engineering	2	6	11%
Coding	1	5	9%
External Factors	2	5	9%
Human Factors	4	4	8%
Process Issues	2	3	6%
Design Issues	1	2	4%
Documentation Issues	1	2	4%
Knowledge Issues	1	2	4%
TD Management	2	2	4%
Architectural Issues	1	1	2%
Infrastructure Issues	1	1	2%
Organizational Issues	1	1	2%

4.2.5 RQ5 - Is TD repayment more associated with coding issues or other issues in the software development process? We identified 32 TD repayment practices, resulting in 315 citations. Of them, ~56% are related to other development issues, while ~44% are associated with code. Unlike the other variables, these percentages differ slightly, indicating that coding issues play a key role in TD repayment initiatives.

We identified eight TD repayment practices related to coding, and the five most cited are presented in Table 10. *Code* and *design*

refactoring are the most cited practices. Both are associated with changes in the internal structure of the system without changing its external behavior. Lastly, the practices *solving technical issues* and *bug fixing* are focused on fix open issues in the code.

Table 10. Top five most commonly cited TD repayment practices related to coding or other development issues.

	Coding		Other Development Issues	
	Repayment practices	**#**	**Repayment practices**	**#**
1st	Code refactoring	80	Investing effort on TD repayment activities	33
2nd	Design refactoring	25	Investing effort on testing activities	22
3rd	Adoption of good practices	10	Prioritizing TD items	15
4th	Solving technical issues	9	Negotiating deadline extension	14
5th	Bug fixing	6	Update system documentation	9

The remain 24 repayment practices are related to other development issues. Table 10 (third column) shows the five most cited ones. These practices evidence several concerns in software development processes: documentation (*update system documentation*), project management (*negotiating deadline extension*, *investing effort on TD repayment*, and *prioritizing TD items*), and software quality (*investing effort on testing activities*).

Table 11 presents the categories of repayment practices. *TD management* and *planning and management* stand out with ~32% and ~27% of the total of citations. The categories *verification, validation and test*, and *process issues* were both cited by ~12% of participants, while the others were less commonly cited.

Table 11. Categories of repayment practices not related to code

Categories of repayment practices	#practices	#cited practices	~%cited practices
TD Management	4	56	32%
Planning and Management	8	47	27%
VV&T	1	22	13%
Process Issues	5	21	12%
Documentation	1	9	6%
Organizational issues	1	8	5%
Human Factors	1	6	4%
Requirement Engineering	1	3	2%
Infrastructure Issues	1	3	2%
Design Issues	1	2	1%

4.2.6 RQ6 - Are the reasons for not repaying TD more related to coding issues or other development issues? We identified 27 reasons for not repaying TD items, totaling 319 citations. From these, 99.7% are related to other development issues and only *lack of access to the component code* (0.3%) is associated with code. The reasons for TD non-repayment arise from development issues other than coding.

Table 12 shows the five best-positioned reasons for not repaying TD. The complete list is available at https://bit.ly/3zoAtFL. We notice that the majority of the reasons (*focusing on short-term goals*, *lack of time*, *cost*, *lack of resources*) are associated with project planning and management.

We also grouped the reasons into categories. As shown in Table 13, *planning and management* stands out with ~58% of the citations, indicating that the reasons from this category are decisive for TD non-repayment. The categories *organizational issues* and *TD management* were also commonly cited by ~16% and ~11% of the participants.

Table 12. Top five most cited reasons for not paying off TD related to other development issues.

	Other Development Issues	
	Reason	**#**
1st	Focusing on short term goals	69
2nd	Lack of org. interest	48
3rd	Lack of time	41
4th	Cost	34
5th	Lack of resources	19

Table 13. Categories of reasons for TD non-repayment not related to coding.

Categories of reasons	#reason	#cited reasons	~%cited reasons
Planning and Management	7	185	58%
Organizational issues	2	50	16%
TD Management	7	34	11%
External Factor	1	13	5%
Knowledge issues	3	12	4%
Human Factors	3	11	4%
Architectural Issues	2	11	4%
VV&T	1	2	1%

5 Discussion

In the technical literature [6,9], the taxonomy of types of debt, composed of fifteen types, indicates that TD items can be related to different software development issues, such as requirements, code, and test. Even though, the current number of studies that investigate the presence of TD items in coding issues is much bigger than in other software development issues [2,6]. Although we understand that source code is one of the main artifacts generated during software development projects, investigating the relation between TD items and other software development issues can shed light on the needs of software practitioners to increase their capability to appropriately manage TD items.

Our results reveal that TD management elements are more related to other software development issues than coding issues. From software practitioners' point of view, the causes that lead to the occurrence of TD items and the effects of their presence are more commonly related to non-coding issues. It means that there is a risk for software projects' healthy when software development teams consider only the source code to identify TD items or applies strategies for reducing their effects. Also, strategies used for managing TD items can add risks to the project when preventive and repayment practices are only applied in the source code.

For making our results more feasible to be digest by software practitioners, we represented them using a hump diagram (see Fig. 3). The diagram represents the relationship between the

investigated TD management elements (causes, effects, prevention practices, reasons for non-prevention, repayment practices, and reasons for non-repayment) and software development issues. In order to plot results for coding and for other issues in the same hump diagram, we normalized the number of citations for an element of a specific software development issue with the total number of citations for that element. For example, prevention practices have in total 819 citations, but 232 citations for the issue *planning and management*. Thus, hump value for *planning and management issues* of prevention practices is 28% (232/819*100). This count is slightly different from the ones we used in Tables 3, 5, 7, 9, 11, and 13 because now we consider *coding* as another software development issue.

We can read the hump diagram horizontally and vertically. Horizontally, we have a broad view on the impact of each software development issue through the TD management elements. For example, we can notice that *coding* plays an important role for all the analyzed TD elements, but mainly for TD repayment. There is a high concentration of practices related to TD repayment and, at the same time, almost none of reasons for the non-repayment of debt items is due to coding issues.

We also perceive that there are many other issues we need to be aware of when dealing with TD in software projects, mainly, *planning and management*. Indeed, this is even stronger when combined with *TD management* concerns. Much about the non-repayment of TD can be understood by looking at these issues.

Human factors also call our attention, clearly indicating that TD, more than technical aspects of the software development, is also about team morale, satisfaction, motivation, communication, and commitment. Other commonly found issues in several elements of the TD management are *architectural issues*, *design issues*, *documentation*, *knowledge issues*, *process issues*, *requirement engineering*, and *VV&T*.

By reading the diagram vertically, we can observe the impact of all identified software development issues on each TD management element. For example, *planning and management*, *organizational*, and *TD management issues* are decisive for the non-repayment of debt items. We also notice that the presence of debt items mainly impacts (effect) *planning and management*, *quality issues*, *maintenance issues*, *human factors*, and *coding*.

Practitioners can use the hump diagram to have a comprehensive view on how TD relates to several issues of their software projects, ranging from organizational to coding level issues. Moreover, for each TD management element, they can go through the detailed results presented in Section 4 and the auxiliary material to understand how to deal with them. For example, by looking at Fig. 3, a practitioner can see that the effects of TD are commonly related to *coding*, *human factors*, *maintenance*, *quality*, and *planning and management* issues. If (s)he is interested in discovering more about the *human factors* issues, then (s)he can observe in the results and auxiliary material that *team demotivation*, *dissatisfaction of the parties involved*, and s*tress with stakeholders* are the main concerns to be mitigated. Thus, the findings can be used as guide to support decision making on TD management

6 Threats to Validity

As in any empirical study, there are threats to validity in this work [10]. We attempt to remove them when possible, and mitigate their effects when removal is not possible.

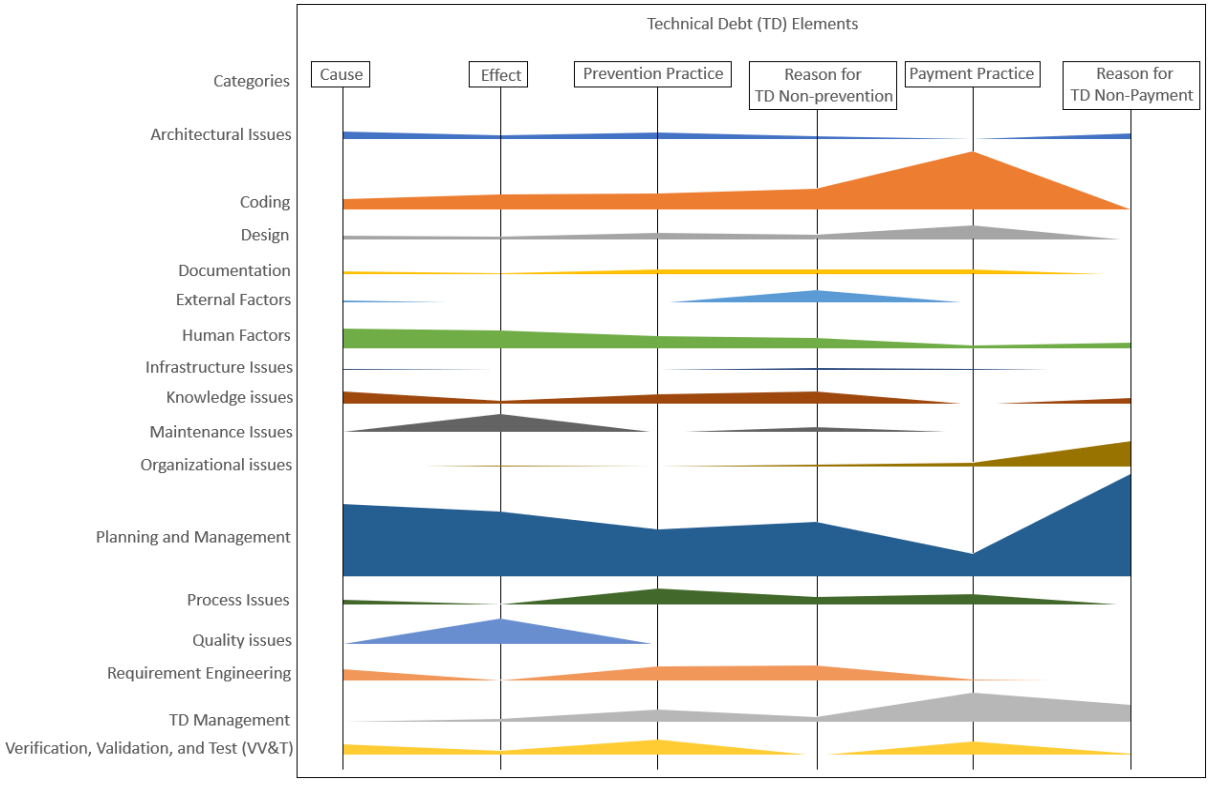

Figure 3: The hump diagram for TD management elements and their software development issues.

Regarding external validity, the study focused on industry professionals and sought to obtain a variety of participants in terms of the level of experience and work environment. Even so, we cannot generalize the results of this study. To strengthen external validity, future steps in this research include expanding the data collected through other replications of InsighTD project.

In addition, the questionnaire was designed to eliminate threats to internal validity. As discussed in [3], the questionnaire went through a series of validations (three internal and one external) and a pilot study to identify any problems before its execution. It is also worth mentioning that the participants could act differently from what they usually do because they are part of a study. To avoid this, we clearly explain the purpose of the study and ask participants to answer the questions based on their own experience. We also state explicitly that the questionnaire is anonymous, and that the data collected is analyzed without considering the identity of the participants. Also, participants may have misinterpreted the use of the terms prevention and repayment of TD. To investigate whether this threat manifested, all responses on how participants avoided and repaid the debt item were analyzed (Q23 and Q27) to analyze if there were invalid answers. A high proportion of invalid responses would mean that the questions could be misinterpreted. In the end, we did not identify any invalid response, indicating that this threat did not appear in the study.

The main threat to the validity of the conclusion is related to the qualitative analysis carried out. To mitigate it, the analyses were carried out separately by two researchers, and the consensus was carried out by a third, more experienced one. Also, additional procedures were considered for seeking consistency in the nomenclature used by each replication team during their coding activities, as described in Section 3.3.2. Lastly, the classification of the coded TD management elements into code/non-code, as well as the definition of their categories, are essentially subjective tasks. To mitigate them, we followed a rigorous analysis procedure previously described in Section 3.3.3. The classification process was always performed individually in pairs, being reviewed by at least one experienced researcher.

7 Conclusion

In this paper, we conjecture that TD is not only about the code. We need to be aware about the other important concerns of the software development process that can be impacted by its presence. The combined use of the hump diagram and the detailed results provides a comprehensive guidance for software development teams about what to expect from the presence of TD and how to react to them considering several software development issues.

Furthermore, our results point out the need of investing more effort on other issues of the software development. For example, complementary to understanding TD at the code level, it is also necessary to investigate strategies to mitigate the managerial reasons that lead software teams to not repay debt items. Another promising topic for investigation would be the relationship between human factors of the software development and TD.

The next steps of this work include an investigation into whether the type of debt impacts how practitioners see TD management elements. We also intend to investigate the main human factors associated with TD. Lastly, the next steps of this research also includes the development of an TD management instrument encompassing the hump diagram and the detailed results, and investigate how to position it into a strategy to support TD management.

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 by the David A. Wilson Award for Excellence in Teaching and Learning.

REFERENCES

[1] Clemente Izurieta, Antonio Vetrò, Nico Zazworka, Yuanfang Cai, Carolyn Seaman, and Forrest Shull, 2012. Organizing the technical debt landscape. in *Proc. of the 3rd Int. Work. on Managing Technical Debt (MTD)*. Zurich, 23-26.

[2] Nicolli N. S. Alves, Thiago S. Mendes, Manoel G. de Mendonça, Rodrigo O. Spínola, Forrest Shull, and Carolyn Seaman, 2016. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70, 100-121.

[3] Nicolli Rios, Rodrigo O. Spínola, Manoel Mendonça, and Carolyn Seaman, 2020. The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. *Empirical Softw. Eng.* 25, 3216-3287.

[4] Yuepu Guo, Rodrigo O. Spínola, and Carolyn Seaman. 2016. Exploring the costs of technical debt management --- a case study. *Empirical Softw. Engg*. 21, 1 (February 2016), 159–182.

[5] Erin Lim, Nitin Taksande, and Carolyn Seaman, 2012. A balancing act: What software practitioners have to say about technical debt. *IEEE Softw*. 29, 6 (November 2012), 22–27. DOI: https://doi.org/10.1109/MS.2012.130.

[6] Zengyang Li, Paris Avgeriou, and Peng Liang, 2015. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101, 193‑220. DOI: https://doi.org/10.1016/j.jss.2014.12.027.

[7] Sávio Freire, Nicolli Rios, Manoel Mendonça, Davide Falessi, Carolyn Seaman, Clemente Izurieta, and Rodrigo O. Spínola, 2020. Actions and impediments for technical debt prevention: results from a global family of industrial surveys. In *Proc. of the 35th ACM SAC*. Brno, 1548–1555.

[8] Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch, 2020. The influence of technical debt on software developer morale. *Journal of Systems and Software*,167. DOI: https://doi.org/10.1016/j.jss.2020.110586.

[9] Nicolli Rios, Manoel Mendonça, and Rodrigo Spínola, 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Inf. and Soft. Technology*, 102, 117-145.

[10] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering: An introduction*. Springer

[11] 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 *Proc. of the Evaluation and Assessment in Software Engineering*. Trondheim, 210–219.

[12] Nicolli Rios, Sávio Freire,Boris Pérez, Camilo Castellanos, Darío Correal, Manoel Mendonça, Davide Falessi, Clemente Izurieta, Carolyn B. Seaman, and Rodrigo O. Spínola, 2021. On the Relationship Between Technical Debt Management and Process Models. *IEEE Softw*.

[13] Nico Zazworka, Antonio Vetro', Clemente Izurieta, Sunny Wong, Yuanfang Cai, Carolyn Seaman, and Forrest Shull, 2014. Comparing four approaches for technical debt identification. *Software Quality Journal*, 22, 403–426.

[14] Ward Cunningham, 1992. The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger, 4, 2 (April 1993), 29-30.

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

[16] Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet (2015). Social debt in software engineering: insights from industry. Journal of Internet Services and Applications, 6(1), 1-17.

[17] Antonio Martini, Viktoria Stray, and & Nils Brede Moe, 2019. Technical-, social-and process debt in large-scale agile: an exploratory case-study. In International Conference on Agile Software Development (pp. 112-119). Springer, Cham.