Hearing the Voice of Software Practitioners on Causes, Effects, and Practices to Deal with Documentation Debt

Nicolli Rios[1], Leonardo Mendes[2], Cristina Cerdeiral[2], Ana Patrícia F. Magalhães[8], Boris Perez[3,4], Darío Correal[3], Hernán Astudillo[5], Carolyn Seaman[6], Clemente Izurieta[7], Gleison Santos[2], and Rodrigo Oliveira Spínola[8](✉)

[1] Federal University of Bahia, Salvador, BA, Brazil
nicollirioss@gmail.com
[2] Federal University of the State of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
{leonardo.cabral,gleison.santos}@uniriotec.br,
cerdeiral@gmail.com
[3] University of Los Andes, Bogota, Colombia
{br.perez41,dcorreal}@uniandes.edu.co
[4] University Francisco de Paula Santander, Cúcuta, Colombia
[5] Univ. Técnica Federico Santa María, Valparaíso, Chile
hernan@inf.utfsm.cl
[6] University of Maryland Baltimore County, Baltimore, MD, USA
cseaman@umbc.edu
[7] Montana State University, Bozeman, MT, USA
clemente.izurieta@montana.edu
[8] Salvador University, Salvador, BA, Brazil
{ana.fontes,rodrigo.spinola}@unifacs.br

Abstract. **[Context and Motivation]** It is common for teams to take shortcuts during software development that, in the future, will lead to maintainability issues and affect productivity and development cost. Different types of technical debt may affect software projects, including those associated with software documentation. Although there are many studies on technical debt, few focus on documentation debt in an industrial environment. **[Question/Problem]** We aimed to identify how software practitioners perceive the occurrence of documentation debt in their projects. We present a combined analysis of the results from two complementary studies: a survey (*InsighTD*) and an interview-based case study. **[Principal Ideas/Results]** We provide a list of causes and effects of documentation debt, along with practices that can be used to deal with it during software development projects. **[Contribution]** We find that documentation debt is strongly related to requirements issues. Moreover, we propose a theoretical framework, which provides a comprehensive depiction of the documentation debt phenomenon.

Keywords: Documentation debt · Causes of documentation debt · Effects of documentation debt · Technical debt · InsighTD

1 Introduction

The technical debt (TD) metaphor describes a daily challenge that software development teams face in their projects: balancing the costs for properly performing short-term product development activities with its long-term quality [1]. When the TD items incurred in a software project are identified, development teams will be able to understand their possible benefits or drawbacks to the project [3].

Different types of TD may affect software projects [4]. Some examples include design, architecture, testing, and documentation debt (DD). The latter is perceived as one of the four most important types in the embedded systems industry [5]. According to Seaman and Guo [7], DD refers to problems encountered in software project documentation looking for missing, inconsistent, outdated, or incomplete documentation.

Despite the growing number of studies in the area [8, 9], and particularly in the software industry [18, 19], little is still known about the impacts of TD [4]. Analyzing the TD phenomenon from the perspective of its causes, effects, and control practices deserves investigation because it is expected that TD prevention could sometimes be cheaper than TD repayment. Besides, when TD is prevented as much as possible, it also helps other TD management activities, and setting up TD prevention practices helps especially in catching inexperienced developers' not-so-good solutions [11]. Knowing the causes for TD can support development teams in defining actions that could be taken to prevent the occurrence of debt items. From the effects perspective, implications of TD can affect projects in different ways. Having this information could aid in prioritization of TD items to pay off, by supporting a more precise impact analysis and also the definition of corrective actions to minimize possible negative consequences for the project [12].

This paper contributes to this discussion from the perspective of DD. Defining, documenting and maintaining requirements is an important step in software engineering, and these critical activities form an integral part of producing high quality software. Thus, understanding DD will lead to decidedly better software. Although the lack of direct perceived benefits of a document to its producer is considered a base reason for many issues in software documentation [20, 21], it is also necessary to investigate other factors that can influence software documentation.

We investigate the causes, effects, and practices that can be employed to deal (prevent or pay the debt off) with this type of debt in the software industry. The research strategy adopted is based on the triangulation of the results of two complementary studies. The first study, *InsighTD*, is a globally distributed family of industrial surveys on TD [12]. For this article, we considered the data sets from Brazil, Chile, Colombia, and the United States. Although significant analysis has already been conducted over the available *InsighTD* data [12–14, 17], much still remains to be studied. The second study is an interview-based case study with practitioners from a large organization.

We provide a list of the top 10 DD causes (deadline being the most cited) and effects (low maintainability being the most cited) from *InsighTD* data. From the case study we identified 15 practices that can be used to deal with DD during a software development project, which corroborate with *InsightTD* participants' opinion that DD can be prevented. The case study provided six additional causes and one effect associated with DD. Results from both studies indicated a strong relationship between this type of debt and requirements issues in software projects. Moreover, based on the evidence we gathered from the data triangulation, we present a theoretical framework that depicts the DD phenomenon.

This paper is organized as follows: Sect. 2 presents a brief introduction to TD and the *InsighTD* project history, Sect. 3 presents the research strategy adopted, Sects. 4 and 5 describe and present our results, Sect. 6 discusses our main findings, Sect. 7 presents threats to the validity of the study, and finally, Sect. 8 presents our final considerations and next steps.

2 Background

2.1 Technical Debt

TD contextualizes the problem of pending software development tasks (for example, inexistent software documentation, tests not performed, non-adoption of good practices) as a type of debt that brings a short-term benefit to the project (usually in terms of higher productivity or shorter release times), but that may have to be paid later with interest in the development process (for example, the evolution of a poorly designed class tends to be more costly than if it was implemented considering good object-oriented design principles) [1, 7].

Alves *et al.* [9] identified that TD can occur in several artifacts throughout the life cycle of a software product. This paper focuses on the study of DD. The existing knowledge in the technical literature about this type of debt is still scarce, restricting itself to recognizing its existence and importance [4, 9]. In one of the few studies that specifically considered this type of debt, Spínola *et al.* [2] identified that DD cannot be automatically identified by current TD identification strategies based on the use of metrics. This paper sheds some light on this discussion by analyzing the causes and effects of DD, and practices that can be employed to prevent or address its existence.

2.2 The *InsighTD* Project

InsighTD is a globally distributed family of industrial surveys initiated in 2017. Planned cooperatively among TD researchers from around the world, the project aims to organize an open and generalizable set of empirical data on the state of practice and industry trends in the TD area. This data includes the causes that lead to TD occurrence, the effects of its existence, how these problems manifest themselves in the software development process, and how software development teams react when they are aware of the presence of debt items in their projects. Its design establishes the foundations for the survey to be continuously replicated in different countries. Up to date, researchers from 11 countries (Brazil, Chile, Colombia, Costa Rica, Finland, India, Italy, Norway, Saudi Arabia, Serbia, and the United States) have joined the project. At the moment, we have concluded data collections of the *InsighTD* replications in Brazil, Chile, Colombia, and the United States.

Rios *et al.* [12] discussed the basic survey design and the preliminary results of the first round of *InsighTD*. In that paper, the authors focused on the discussion on the top 10 causes and effects of TD, regardless the type of debt. *Rios et al.* [13] complemented the discussion of the previous work, focusing specifically on the causes and effects of TD in agile software projects. Rios *et al.* [14] proposed the use of cross-company probabilistic cause-effect diagrams to represent information about the TD causes and effects being analyzed. More Recently, Freire *et al.* [17] investigated preventive actions that can be used to curb the occurrence of TD and the impediments that hamper the use of those actions.

In this work, we go further into the analysis of *InsighTD* data by considering the point of view of the respondents about DD.

3 Research Strategy

3.1 Research Questions

We defined the following main Research Question (RQ) "How do software development teams perceive the occurrence of DD in their projects?" The goal of this RQ is to gather information on how practitioners face DD in their daily activities. To investigate it, we broke down this question into the following sub-questions:

RQ1: What are the main causes that lead development teams to incur DD in their projects? This question investigates the possible causes that contribute to the insertion of DD in software projects.
RQ2: What effects does DD have on software projects? This question is aimed at identifying the main effects felt by development teams due to the presence of DD.
RQ3: How often is the occurrence of DD items seen as preventable in software projects? Although DD can be incurred by choice –for example, to reduce costs or speed a release, it still has decidedly negative consequences on a project. In this question we explore our pre-conception that DD can be prevented, given a choice to do so, regardless of whether the DD item is intentional or unintentional. Through it, we explore practitioners' responses and have an indication on how often TD items could be prevented in their scenarios.
RQ4: What stage of a software development life cycle is most affected by the presence of DD? The documentation of a software project is a broad area, ranging from requirements specification to code comments. The purpose of this question is to investigate which stage of a software development life cycle has been more commonly seen as affected by DD.
RQ5: How can development teams react to the presence of DD? This question is aimed at identifying actions that can be used to deal (prevent or pay the debt off) with DD.

3.2 Method

The method is based on the combined analysis of the results from two complementary studies: a survey (*InsighTD*) and an interview-based case study, both with a population of software practitioners. While *InsighTD* allowed us to achieve a broad audience and collect answers to support the answering of RQ1, RQ2, RQ3, and RQ4, the interview study served to check and complement, through data triangulation, the findings from *InsighTD* and, also, to gather more contextual information to answer the RQ5. We chose to perform triangulation because it is an important tool for confirming the validity of conclusions [15].

Sections 4 and 5 describe the data collection and analysis procedures of each study as well as the obtained results. Then, Sect. 6 combines the results to answer the posed research questions. The empirical package of the survey and interview study containing their questions, answers/transcriptions, and codes are available at http://bit.ly/2uHv8x9.

4 Surveying Software Practitioners on Causes and Effects of Documentation Debt (*InsighTD*)

4.1 Data Collection

The data were collected in the context of the *InsighTD* project. The *InsighTD* questionnaire consists of 28 questions, previously described in [12]. Table 1 presents the subset of the survey's questions related to the context of this work. Q1 to Q8 capture the characterization questions, Q13 asks participants to provide an example of a TD item that occurred in their project and Q15 asks participants about the representativeness of that example. In Q16 to Q18 and Q20, the participants answer questions about causes and effects, respectively, considering the example provided in Q13. Finally, Q22 asks participants if the TD item (from Q13) could be prevented.

Table 1. Subset of the *InsighTD* survey questions considered in this paper.

No.	Question (Q)	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 describes the development process model you follow on this project?	Closed
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
Q16	What was the immediate, or precipitating, cause of the example of TD you just described?	Open
Q17	What other cause or factor contributed to the immediate cause you described above?	Open
Q18	What other causes contributed either directly or indirectly to the occurrence of the TD example?	Open
Q20	Considering the TD item you described in question 13, what were the impacts felt in the project?	Open
Q22	Do you think it would be possible to prevent the type of debt you described in question 13?	Closed

The questionnaire was sent to only practitioners, because the objective of *InsighTD* is to investigate the state of the practice of TD. Some keywords related to software development activities and roles were used in LinkedIn to identify the participants. Also, invitations were sent to industry-affiliated member groups, mailing lists, and industry partners. The same strategy was applied in Brazil, Chile, Colombia, and the United States.

4.2 Data Analysis

The survey questionnaire consists of closed and open-ended questions. Therefore, it is necessary to adopt a series of different procedures to analyze the data. For the answers to the closed questions, descriptive statistics were used to understand the data, and then mode and median statistics were used for the central tendency of ordinal and interval data. For nominal data, the number of participants' choices about each option was calculated.

Qualitative data analysis techniques [15, 16] were applied to open-ended questions about causes and effects of TD. As the answers were unrelated to any previous expectations, an inductive logical approach was adopted. Then, manual coding was applied to the open questions as follows: initially, two researchers from each country (BR: authors N.R. and R.S., CH: B.P. and H.A., CO: B.P. and D.C., and US: N.R. and C.S.) individually coded the set of all answers for two subsets of related questions (RQ1: Q16 + Q17 + Q18 and RQ2: Q20). This involves open coding as described in [16] and axial coding to derive higher level categories. They then discussed possible differences in their coding until they reached consensus. Thus, the answers were coded and the emerged concepts (causes/effects) were organized into a hierarchy of categories. This process was performed until the point where no new code or category was identified.

4.3 Results

Characterization of the Participants. In total, 39 participants from *InsighTD* replications in Brazil, Chile, Colombia, and the United States answered questions about DD. Participants are well distributed among small (28%), medium (41%), and large (31%) companies. Looking at the data in more detail, we can see that most (mode) participants work in organizations with more than 2,000 employees (9 participants) and 11–50 employees (9), closely followed by 51–250 companies (7), and 251–500 employees (7). The average size of organizations is 251 to 500 employees. Therefore, participants tend to work in larger companies, but there are representatives of companies of all sizes.

Responses to Q3 on the system size, most participants indicated that the systems had between 10–100 KLOC (36%), followed by smaller systems (<10 KLOC - 26%). The results also indicated a significant sample of responses for larger systems (1–10 MLOC - 18%) (>10 MLOC - 5%). Responses to Q4 on the size of development teams that participants tend to work, most (31%) reported working in teams 5–9 people, followed by less than five members of staff (26%). There is also a good sample in teams of 10–30 people (20%) and larger development teams with more than 30 professionals (23%). Responding to Q5 about the age of the system developed in the project, most indicated age 2 to 5 years (33%). There are also a significant number of systems represented from 1 to 2 years (26%), closely followed by less than one year (20%). About 18% of participants indicated working on projects for more than 10 years.

Regarding the role assumed by the participants in their projects (Q6), several types of project roles were indicated in the results. Most of them work as a developer (46%), followed by project leader/project manager (18%), architect software (7%), test manager/tester (7%), and requirements analyst (7%). In response to Q7, results show that a significant portion of the sample is Proficient, Competent or Expert (92% of the total), indicating that, in general, the questionnaire was answered by professionals with experience in their roles. On the other hand, answers from professionals with low experience level (8%) were also obtained.

Finally, the responses to Q8 indicate that of the 39 respondents, 15 participated in projects that adopted agile process models (38%) and 15 indicated the use of hybrid process models (38%). Less common is the use of a traditional process (24%).

Thus, in general, although it is not possible to guarantee that the participants represent all the professionals in the software industry from Brazil, Chile, Colombia, and the United States, respondents characterize a broad and diverse audience that spans different functions and participant experience levels, different sizes of organizations and projects of different ages, sizes, team sizes, and process models.

What are the main causes that lead development teams to incur DD in their projects (RQ1)? In total, as informed by the participants in Q16–18, there are 37 causes that lead development teams to incur DD in their projects. The ten most commonly cited causes are displayed in Table 2 (the complete list can be accessed at http://bit.ly/2BtHglx). *Deadline* is the most cited cause. This indicates that it is a factor that normally contributes to the occurrence of DD items. The *company does not give importance to documentation* and *non-adoption of good practices* are other causes cited by at least 13% of the participants.

Table 2. Top 10 documentation debt causes cited.

Rank	Documentation debt cause	# of citations	Confirmed in interview based case study?
1st	Deadline	12	Yes
2nd	The company does not give importance to documentation	6	Yes
3rd	Non-adoption of good practices	5	Yes
4th	Inaccurate time estimate	4	Yes
5th	Inappropriate planning	4	Yes
6th	Outdated/incomplete documentation	4	Yes
7th	Team Overload	4	Yes
8th	Nonexistent documentation	3	Yes
9th	Not effective project management	3	Yes
10th	Poor allocation of resources	3	Yes

We can also observe in Table 2 that practitioners also commonly cited other causes like *inaccurate time estimate*, *inappropriate planning*, *outdated/incomplete documentation*, and *team overload*. The results of the data triangulation with the interview-based case study, partially presented in the fourth column, is discussed in Sect. 5.3.

What effects does DD have on software projects (RQ2)? In total, as informed by the participants in Q20, there are 24 effects of DD in software projects. Table 3 shows the ten most commonly cited effects (the complete list can be accessed at http://bit.ly/2BtHglx). Two effects stood out: *low maintainability* and *delivery delay*. *Low maintainability* encompasses problems that occur during software maintenance activities, such as an increased effort to fix bugs as well as limitations in system evolution. *Delivery delay* refers to the non-fulfillment of the deadlines agreed upon with the customer.

We can also observe in Table 3 that, in the point of view of the practitioners, *rework*, *low external quality* and *inadequate/nonexistent/outdated documentation* are issues that commonly affect software projects in the presence of DD. Also, there are three effects related to relations among people: *developer dependency* and *stress with stakeholders*. Thus, practitioners see that the presence of DD can harm the work environment.

How often is the occurrence of DD items seen as preventable in software projects (RQ3)? Answers to Q22 (yes/no question) of *InsighTD* indicated that DD could be prevented for most of the cases (95%). The two participants that reported that their DD items could not be prevented indicated cost as the main reason: "*the development team does not have the documentation updated because it needs to be more productive to do not lose the contract*" and that "*the effort needed to be invested to maintain the documentation updated is too high.*"

Table 3. Top 10 documentation debt effects cited.

Rank	Documentation debt effects	# of citations	Confirmed in interview based case study?
1st	Low maintainability	9	Yes
2nd	Delivery delay	8	Yes
3rd	Rework	5	Yes
4th	Low external quality	4	Yes
5th	Inadequate/nonexistent/outdated documentation	3	Yes
6th	Developer Dependency	2	Yes
7th	Difficulty conducting tests	2	Yes
8th	Increased effort	2	Yes
9th	Need of refactoring	2	No
10th	Stress with stakeholders	2	Yes

Answers to RQ5 in the interview-based case study, discussed in Sect. 5.3, complement this result by indicating some practices that can be used to prevent DD.

What stage of a software development life cycle is most affected by the presence of DD (RQ4)? Results from *InsighTD* indicate a strong relationship between DD and requirements issues. By analyzing the answers of participants to Q13, about 53% of them reported requirements issues in their examples of DD. Some examples are: "*Little clarity and specificity in the definition of requirements,*" "*Having to create code that* *was not stipulated, since the requirement was not considered in the documentation,*" and "*The lack of documentation and understanding of the requirements in the analysis and design activities caused rework in the construction of the prototypes.*" Besides, all participants reported in Q15 that those instances of debt occur often or very often in their projects.

Other commonly cited software development areas (in Q13) affected by the presence of DD are design (10%), code comment (7%), testing (5%), and architecture (2%). We could not identify the specific area for 23% of the responses because they were about documentation in general (e.g.: "*hard maintenance and future change due to poor documentation from the development team*" and "*do not keep the documentation updated*").

5 Interview-Based Case Study

This study complements the results obtained with *InsighTD* by (i) identifying new evidence to confirm causes and effects, and (ii) identifying practices that can be employed to prevent or deal with DD in software projects.

5.1 Data Collection

Data collection was performed through face-to-face interviews considering four open-ended questions. All interviews were recorded, with previous authorization of the participants, and then transcribed. Next, the transcripts were validated by their corresponding interviewees through peer reviews. The interviews were performed in Portuguese as well as their data analysis. Only the results were translated to English. The author L.M. translated the results (and also the text fragments used in this article), which were reviewed by the author G.S.

The first three questions address the characterization of the organization's development process: (P1) "*Does the organization adopt any project management methodology (traditional/agile)? If so, which one?*"; (P2) "*Did the organization define processes for software development? How are they performed?*"; (P3) "*How is the documentation process carried out in the organization and how are the documents prepared for each phase of the software life cycle?*" Finally, we characterized the issues found in the execution of processes in the organization: (P4) "*Are those involved in the software development process aware of the problems that may arise from not adopting adequate documentation? If so, could you cite possible causes and effects?*"

5.2 Data Analysis

The data analysis began with transcription of interviews and approval by participants. Then, we coded the content of the transcripts [16]. The coding considered excerpts of the transcriptions containing evidence on causes, effects and practices to deal with DD. Then, we grouped the identified causes, effects and practices.

The following example illustrates how we performed the coding of causes: "*We need to go fast with the development process, so we went directly to the testing phase.*"

As consequence, the documentation of the project was inappropriate." The underlined fragments support the codes: *inappropriate documentation* and *focus on producing more at the expense of quality*.

The coding was done by the author L.M. The author G.S. reviewed all citations and codes. Participants were also asked to validate the results obtained. In the end, the codes identified for causes and effects were standardized considering the nomenclature obtained from the *InsighTD* results. This standardization was performed by the authors L.M. and R.S., who were involved in each of the studies.

5.3 Results

Execution. Participants were advised to feel free to talk about work processes and documentation issues. Participants allowed the interviews to be recorded and signed a Consent and Participation document. Each participant was interviewed individually, and the interviews took about 30 min.

The defined questions (P1–4) were performed sequentially. The researcher responsible for conducting the interviews complemented the questions with brief comments to adapt them to the working reality of the interviewees. During the interviews, new questions were formulated for gathering more details as needed (e.g.: "*Is there any practical situation that generated a problem with the documentation?*").

Characterization of the Participants. Four software practitioners who have worked as project manager, systems analyst, developer, and tester for 7, 2, 1 and 14 years, respectively, participated in the study. The selection of participants was made by convenience. Our main selection criterion was having answers from practitioners with different responsibilities in the development process and with different levels of experience. The four participants worked on the same development team and reported their experiences with both traditional and agile methods. All participants also indicated that the existing development processes are not followed.

The software organization where the study was conducted operates in the public health area. The study was conducted specifically in one of the software development areas. We selected the organization by convenience. The second author works in the organization, but in a different unit from the interviewees. In addition, we chose a public organization because the staff had little turnover so that we could better observe the influence of organizational culture on development processes.

The product used as a reference by the participants was an academic management system of the organization. The organization started the development of the product in 2009 and it has been maintained by the organization since 2014. The system is currently in production, but some of its modules are being refactored.

What are the main causes that lead development teams to incur DD in their projects (RQ1)? The participants reported 23 causes that contribute to the occurrence of DD. Participants reported, for example, the situation of a specific project that had three consecutive management changes during its implementation. The changes were mainly characterized by unsuccessful attempts to implement different development methods (agile and traditional). These consecutive management changes were characterized as one possible cause and were coded as *changes in management during the project*. The interviewed project manager, systems analyst, and developer stated that the organization did not have well-defined processes (*lack of a well-defined process*). *Deadline,* one of the causes reported by all participants, is considered responsible for most of the problems faced by the teams.

Participants were unanimous in stating that all team members were aware of documentation problems in their projects. The systems analyst said that, while practitioners are aware of the problems, there were negligence in not trying to solve them (*the company does not give importance to documentation*).

Table 2 presents ten of the causes identified in this study that are among those most commonly cited by *InsighTD* participants. The complete list of all identified causes can be accessed at http://bit.ly/2BtHglx. Seventeen common causes were identified between the two studies, and six were uniquely identified in the interviews: *inappropriate documentation, unknown legal requirements that affect the existing documentation, delays in the project, tacit knowledge not documented, changes in management during the project,* and *political issues.*

What effects does DD have on software projects (RQ2)? Participants reported 15 effects of DD. Nine of them are among those most cited by *InsighTD* participants as can be observed in Table 3. Only one of the reported effects had not been previously identified in *InsighTD*: *communication issues among team members*.

Some of the most critical effects reported by participants in this study were: *project does not serve customer*, *developer dependency*, *difficulty in project development*, *lack of understanding,* and *increased effort to maintain the product*.

How can development teams react to the presence of DD (RQ3 and RQ5)? We identified 15 practices (Table 4) that have been used by the organization to address software documentation issues and could be employed to deal with (prevention or payment) DD.

Most of the identified practices refer to preventive actions, as can be seen in the second column of Table 4. This result complements the indication provided by *InsighTD* participants that DD can be prevented. We can also observe that, from the point of view of the participants, the preventive actions usually have a well-defined documentation process. We also highlight the need of having the commitment from people responsible for documenting activities. To improve the commitment, one possible solution would be to increase the incentives to produce the documentation, and further, developers must feel how these improvements are a benefit to themselves [20, 21]. From Table 4, we see that only three practices are focused on debt payment.

What stage of a software development life cycle is most affected by the presence of DD (RQ4)? The interview-based case study confirmed the results from *InsighTD* for this question. Thus, participants also indicated during the interviews that DD is related to several areas of software development (requirements, design, coding, test, and maintenance). Particularly, the identified causes (~91% of them), effects (~80%), and practices (~87%) are almost all related to requirement issues as confirmed by the participants after we reported the results to them.

Table 4. List of practices.

Practices	Prevention/payment
Adopt TD payment prioritization criteria	Payment
Comment the code	Prevention
Create tutorials on how to fill in the documentation	Prevention
Define process and good practices for documentation	Prevention
Define roles concerning the documentation process	Prevention
Document the project since its begin	Prevention
Have a documentation repository	Prevention
Improve commitment of the team concerning documentation	Prevention
Involve several roles in documenting the project	Prevention
Keep the documentation updated	Payment
Penalties if not follow the documentation process	Prevention
Review outdated documentation	Payment
Training on the problems by don't document	Prevention
Use of Peer review	Prevention
Use of UML to document and share information	Prevention

6 Discussion

There is a clear need for research that consolidates data collected from empirical studies in the software industry. Results from both studies presented in this paper indicate that, from the point of view of software practitioners, DD can be prevented. Further, although practitioners are particularly concerned about requirements issues, we also found that DD can affect other areas of software development projects.

The aforementioned results stimulated us to organize the data (causes, effects, and practices) collected from both studies into a theoretical framework of DD, which is presented in the next subsection.

6.1 Theoretical Framework of Documentation Debt

Figure 1 summarizes the theoretical framework developed from this research. The framework aims to provide a comprehensive depiction of the DD phenomenon. It consists of causes that can lead development teams to incur DD in their projects, effects that can be felt in its presence, and, also, practices that can be employed to prevent or eliminate items of debt present in projects. The organization of causes and effects into groups (e.g.: development issues, methodology, people issues, external quality issues) followed the categories proposed by Rios *et al.* [12].

As a conceptual device, the framework can be employed to inform action in response to perceived DD, and as a comprehensive guide when assessing software development

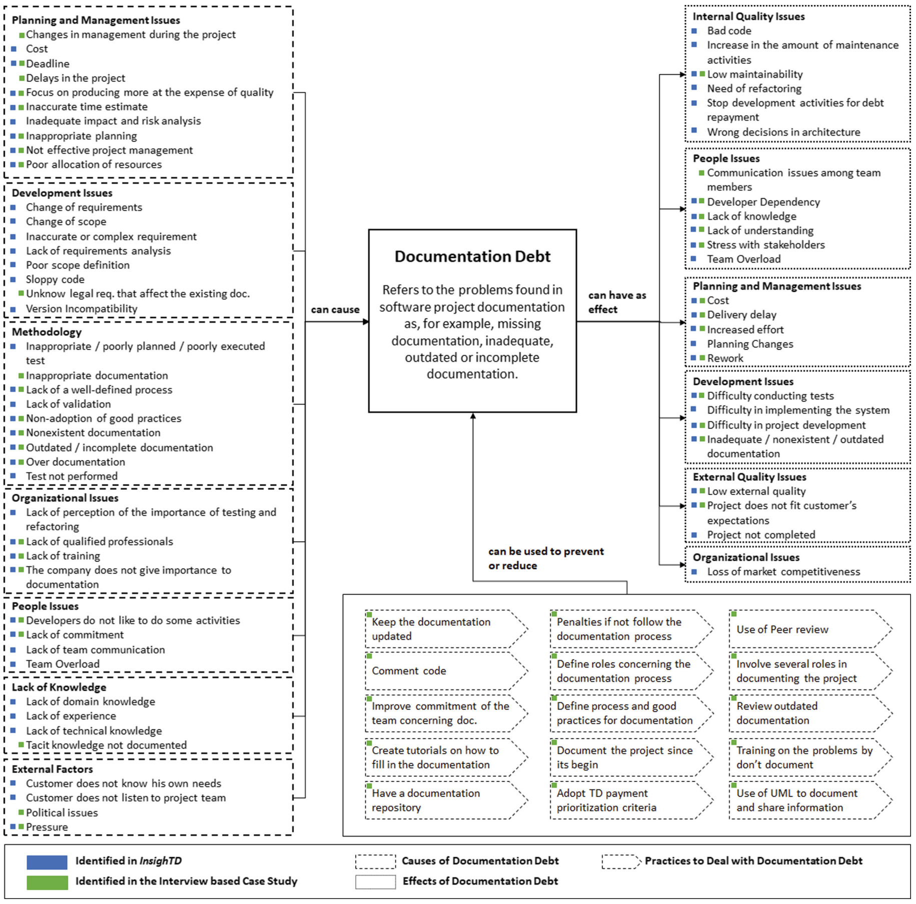
Planning and Management Issues
🟩 Changes in management during the project
🟦 Cost
🟦🟩 Deadline
🟩 Delays in the project
🟦🟩 Focus on producing more at the expense of quality
🟦🟩 Inaccurate time estimate
🟦 Inadequate impact and risk analysis
🟦🟩 Inappropriate planning
🟦🟩 Not effective project management
🟦🟩 Poor allocation of resources
Development Issues
🟦 Change of requirements
🟦 Change of scope
🟦 Inaccurate or complex requirement
🟦 Lack of requirements analysis
🟦 Poor scope definition
🟦 Sloppy code
🟩 Unknown legal req. that affect the existing doc.
🟦 Version Incompatibility
Methodology
🟦 Inappropriate / poorly planned / poorly executed test
🟩 Inappropriate documentation
🟦🟩 Lack of a well-defined process
🟦 Lack of validation
🟦🟩 Non-adoption of good practices
🟦🟩 Nonexistent documentation
🟦🟩 Outdated / incomplete documentation
🟦🟩 Over documentation
🟦 Test not performed
Organizational Issues
🟦 Lack of perception of the importance of testing and refactoring
🟦🟩 Lack of qualified professionals
🟦🟩 Lack of training
🟦🟩 The company does not give importance to documentation
People Issues
🟦🟩 Developers do not like to do some activities
🟦🟩 Lack of commitment
🟦 Lack of team communication
🟦 Team Overload
Lack of Knowledge
🟦 Lack of domain knowledge
🟦 Lack of experience
🟦 Lack of technical knowledge
🟩 Tacit knowledge not documented
External Factors
🟦 Customer does not know his own needs
🟦 Customer does not listen to project team
🟩 Political issues
🟦🟩 Pressure
can cause
Documentation Debt
Refers to the problems found in software project documentation as, for example, missing documentation, inadequate, outdated or incomplete documentation.
can have as effect
Internal Quality Issues
🟦 Bad code
🟦 Increase in the amount of maintenance activities
🟦🟩 Low maintainability
🟦 Need of refactoring
🟦 Stop development activities for debt repayment
🟦 Wrong decisions in architecture
People Issues
🟩 Communication issues among team members
🟦🟩 Developer Dependency
🟦🟩 Lack of knowledge
🟦🟩 Lack of understanding
🟦🟩 Stress with stakeholders
🟦 Team Overload
Planning and Management Issues
🟦🟩 Cost
🟦🟩 Delivery delay
🟦🟩 Increased effort
🟦 Planning Changes
🟦🟩 Rework
Development Issues
🟦🟩 Difficulty conducting tests
🟦 Difficulty in implementing the system
🟦🟩 Difficulty in project development
🟦🟩 Inadequate / nonexistent / outdated documentation
External Quality Issues
🟦🟩 Low external quality
🟦🟩 Project does not fit customer's expectations
🟦 Project not completed
Organizational Issues
🟦 Loss of market competitiveness
can be used to prevent or reduce
Keep the documentation updated
Penalties if not follow the documentation process
Use of Peer review
Comment code
Define roles concerning the documentation process
Involve several roles in documenting the project
Improve commitment of the team concerning doc.
Define process and good practices for documentation
Review outdated documentation
Create tutorials on how to fill in the documentation
Document the project since its begin
Training on the problems by don't document
Have a documentation repository
Adopt TD payment prioritization criteria
Use of UML to document and share information
🟦 Identified in *InsighTD*
🟩 Identified in the Interview based Case Study
Causes of Documentation Debt
Effects of Documentation Debt
Practices to Deal with Documentation Debt

Fig. 1. Theoretical framework of documentation debt.

practices. The framework facilitates more effective identification and acknowledgement of DD by highlighting aspects of software development that impact or is impacted by the presence of the debt.

By assisting in making the DD visible, as a communication device, the framework can be used to support development teams to more effectively communicate technical problems to management, and for managers to make better-informed decisions concerning DD.

7 Threats to Validity

As in any empirical study, there are threats to validity [6] in this work. We attempted to remove them when possible and mitigate their effect when removal was not possible. In this work, the primary threat to conclusion validity arises from the coding process as coding is mainly a creative task. To mitigate this threat, in *InsighTD*, the coding process was performed individually by two researchers and reviewed by one experienced researcher. In the interview-based case study, the coding process was performed by one researcher and reviewed by one experienced researcher. The recording/transcription process could raise threats too. We reduced them by validating the transcriptions with a peer review process involving the corresponding interviewees. Lastly, the data triangulation activities were performed by one researcher from each study, who also discussed their results until consensus was reached.

Concerning the internal validity, the questionnaire represents the main threat that could affect *InsighTD*. As indicated in [12], the questionnaire has direct questions, avoiding misunderstanding that could lead to meaningless answers. Besides, the questionnaire has passed through successive validation tasks (three internal and one external) and a pilot study to detect any inconsistencies or misunderstandings before executing the survey.

Finally, we reduced the external validity threats by targeting industry professionals and seeking to achieve participant diversity among the respondents. In *InsighTD*, we approached 39 practitioners from replications of the questionnaire in Brazil, Chile, Colombia, and the United States. The interview-based case study had the participation of four practitioners with different roles and levels of experience. Although the population provides interesting results on DD, we still cannot generalize the results. In search of more generalizable results, the *InsighTD* is now being replicated in Finland and Costa Rica.

8 Final Remarks

Documentation debt is a type of debt that still suffers from a lack of empirical evidence from software industry. This article approached this gap by triangulating results from two complementary studies with software practitioners. Results include the indication that we can prevent DD and that it affects several software development areas but specially requirements. Moreover, we defined a theoretical framework of DD, which presents the DD phenomenon in a more complete and comprehensive form.

For the practitioner community, the framework helps to realize the utility of technical debt as a tool for conceptualization, communication, and management. It can be used as tool to understand the reasons that lead development teams to incur in debt, which are the possible effects of its presence, and what actions can be taken to prevent or pay the debt off.

The next steps of this research include the analyses of *InsighTD* data collected from replications in Costa Rica and Finland. We also intend to run a follow up study in one of our industry partners based on the results reported in this article. Lastly, based on the conceptual framework presented in Fig. 1, we are also planning to look into relating causes, effects, and practices more directly to each other.

Acknowledgements. This work was partially supported by the Coordination for the Improvement of Higher Education Personnel - Brazil (Capes), under the Capes/IIASA Sandwich Doctoral Program, process nº 88881.189667/2018-01. 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. Kruchten, P., Nord, R., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Softw. **29**(6), 18–21 (2012). https://doi.org/10.1109/MS.2012.167
2. Spínola, R.O., Zazworka, N., Vetro, A., Shull, F., Seaman, C.: Understanding automated and human-based technical debt identification approaches-a two-phase study. J. Braz. Comput. Soc. **25** (2019). https://doi.org/10.1186/s13173-019-0087-5
3. Ernst, N.A., Bellomo, S., Ozkaya, I., Nord, R.L., Gorton, I.: Measure it? Manage it? Ignore it? Software practitioners and technical debt. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pp. 50–60. ACM, New York (2015). https://doi.org/10.1145/2786805.2786848
4. Rios, N., Mendonça, M.G., Spínola, R.O.: A tertiary study on technical debt: types, management strategies, research trends, and base information for practitioners. Inf. Softw. Technol. **102**, 117–145 (2018). https://doi.org/10.1016/j.infsof.2018.05.010. ISSN 0950-5849
5. Ampatzoglou, A., et al.: The perception of technical debt in the embedded systems domain: an industrial case study. In: 8th International Workshop on Managing Technical Debt. IEEE (2016)
6. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29044-2
7. Seaman, C., Guo, Y.: Measuring and monitoring technical debt. Adv. Comput. **82**, 22 (2011)
8. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**, 193–220 (2015)
9. Alves, N.S.R., Mendes, T.S., de Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C.: Identification and management of technical debt: a systematic mapping study. Inf. Softw. Technol. **70**, 100–121 (2016). https://doi.org/10.1016/j.infsof.2015.10.008
10. Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C.: Managing technical debt in software engineering (dagstuhl seminar 16162). In: Dagstuhl Reports, vol. 6, no. 4. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
11. Yli-Huumo, J., Maglyas, A., Smolander, K.: How do software development teams manage technical debt? An empirical study. J. Syst. Soft. **120**, 195–218 (2016)
12. Rios, N., Spínola, R.O., Mendonça, M.G., Seaman, C.: The most common causes and effects of technical debt: first results from a global family of industrial surveys. In: The Proceedings of the 12th International Symposium on Empirical Software Engineering and Measurement, Oulu, p. 10. ACM, New York (2018). https://doi.org/10.1145/3239235.3268917. Article no. 39
13. Rios, N., Mendonça, M., Seaman, C., Spínola, R.O.: Causes and effects of the presence of technical debt in agile software projects. In: The Americas Conference on Information Systems (AMCIS), Cancun (2019)
14. Rios, N., Spínola, R.O., Mendonça, M.G., Seaman, C.: Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. In: Proceedings of the Second International Conference on Technical Debt (TechDebt 2019), pp. 3–12. IEEE Press, Piscataway (2019). https://doi.org/10.1109/techdebt.2019.00009
15. Seaman, C.: Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. **25**(4), 557–572 (1999). https://doi.org/10.1109/32.799955

16. Strauss, A., Corbin, J.M.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Thousand Oaks (1998)
17. Freire, S., et al.: Actions and impediments for technical debt prevention: results from a global family of industrial surveys. To appear in the Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing
18. Klotins, E., et al.: Exploration of technical debt in start-ups. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP 2018), pp. 75–84. ACM, New York (2018)
19. Nayebi, M., et al.: A longitudinal study of identifying and paying down architecture debt. In: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, pp. 171–180. IEEE Press (2019). https://doi.org/10.1109/ICSE-SEIP.2019.00026
20. Arkley, P., Riddle, S.: Overcoming the traceability benefit problem. In: The Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE 2005), Paris, France (2005). https://doi.org/10.1109/re.2005.49
21. Berry, D.M., Czarnecki, K., Antkiewicz, M., Abdelrazik, M.: The problem of the lack of benefit of a document to its producer. In: Proceedings of the IEEE International Conference on Software Science, Technology and Engineering, Beer-Sheva, Israel (2016). https://doi.org/10.1109/swste.2016.14