How do Technical Debt Payment Practices Relate to the Effects of the Presence of Debt Items in Software Projects?

***Abstract*—Context: Knowing the effects of technical debt (TD) can support software development teams in the prioritization of TD items to pay off. However, little is known about the relations between the effects of TD and TD payment practices. Having this knowledge can provide valuable information for decision making about which payment practice can be applied given the presence of specific effects of TD. Aims: To investigate, from the point of view of software practitioners, (i) which TD payment practices have been used when certain effects of the presence of debt are felt in software projects and (ii) the reasons for not paying debt items despite the effects they are causing to the project. Method: We analyze quantitatively and qualitatively data collected from a survey with 432 practitioners across four countries. Results: Among the identified relations, the practice "code refactoring" is commonly used to pay debt items off when the effects "delivery delay" and "rework" are felt in software projects. On the other hand, when practitioners face the TD effects "low external quality" and "delivery delay", they commonly justify the non-payment of the debt items indicating the need of "focusing on short term goals". Conclusion: We organize the relationship between TD effects, and payment practices and reasons for not eliminating debt items. All this information is structured in an alluvial diagram, which can facilitate the visualization of the identified relations.**

Keywords—technical debt, technical debt effects, technical debt management, technical debt payment

I. INTRODUCTION

Technical debt (TD) refers to the problem of pending or incomplete tasks and artifacts that bring short-term gains to a software project but may have to be paid with interest later on in its life cycle [1, 2]. Knowing the possible effects of TD and performing TD management activities are necessary to deal with the drawbacks of the presence of debt items [3,4]. Having information about TD effects can support software development teams in prioritizing debt items for payment and identifying payment practices to eliminate those items [4,5]. Moreover, knowing the relationship between TD effects and payment practices can provide valuable information for decision making on which payment practice can be applied given the presence of specific effects of TD.

Several works have investigated TD effects [4-8] and TD payment practices [7,9-11]. For example, Martini and Bosh [5] investigated the effects of architecture debt conducting a case study in five large companies, and Apa et al. [10] identified the practices for eliminating TD by applying a survey in software startups. Although these studies presented findings on TD effects and payment practices, they did not focus on the possible relationship between them.

In this work, we bridge this knowledge gap by investigating, from the point of view of software practitioners, (i) which TD payment practices have been used when certain effects of the presence of debt are felt in software projects and (ii) the reasons for not paying debt items despite the effects they are causing to the project. We use data collected in the *InsighTD Project*' context, a globally distributed family of industrial surveys to investigate causes, effects, and management of TD. In total, 432 practitioners answered the survey conducted in Brazil, Colombia, Chile, and the United States. We analyzed these answers, both quantitatively and qualitatively.

We ground this work in two others from the *InsighTD* Project. In [4], we investigated the effects of TD considering only the Brazilian replication of *InsighTD*. Thus, we revisited its results to include the data from the other three replications. In [11], we approached the TD payment practices and the reasons for the non-payment of debt items. This last work considers all the four replications; thus, we used its findings (lists of practices and reasons) to investigate the relationship between them and the effects of TD.

Results show that *delivery delay*, *low maintainability*, and *rework* are the most common effects felt by practitioners. Concerning the relationship of effects with TD payment practices and reasons for not paying TD items, we discuss the identified relations between practitioner's top 10 (most commonly found in software projects) lists. Among others, the practice *code refactoring* is commonly used to pay debt items when the effects *delivery delay* and *rework* are felt in software projects. On the other hand, when practitioners face the TD effects *low external quality* and *delivery delay*, they commonly justify the non-payment of debt items indicating a need for *focusing on short term goals*. We represented the identified relationships using an alluvial diagram.

This paper is organized as follows. Section II presents some background information. Section III presents the research method. Section IV presents the results. Section V presents the alluvial diagram for supporting the visualization of the relationships between effects, TD payment practices, and reasons. Section VI discusses the implications of the study for researchers and practitioners. Section VII discusses the threats to validity. Finally, Section VIII presents the future work.

II. BACKGROUND

Several works have approached the topics of TD effects and payment practices. Concerning the effects of TD, Yli-Huumo et al. [6] conducted interviews with 17 practitioners from two software companies and reported that workarounds, hours, cost, and poor quality are effects of TD. The mapping study performed by Li et al. [7] identified that TD affects software quality attributes, such as maintainability, reliability, security, portability, and performance efficiency.

Martini and Bosch [5] conducted a case study in six large software companies for investigating the effects caused by the

presence of architecture debt. As a result, the authors defined a model of effects that can be used to prioritize architecture debt items. In another work on the area, Besker et al. [8] performed a systematic review on the effects of architecture debt and identified the following items: flexibility, maintenance and evolvability, innovation and system growth, performance degradation, and reliability.

Finally, in [4] we report the effects of TD as cited by 107 practitioners from the Brazilian software industry. The top 10 most commonly cited were: low quality, delivery delay, low maintainability, rework, financial loss, team demotivation, stakeholder dissatisfaction, inadequate documentation, low performance, and bad code.

Concerning TD payment, Li et al. [7] identified the following categories of TD payment practices: refactoring, rewriting, automation, reengineering, repackaging, bug fixing, and fault tolerance. In another systematic literature review, Behutiye et al. [9] investigated TD payment practices used in agile software development. The authors indicated that refactoring was the most used practice. Lastly, Apa et al. [10] applied a survey for investigating TD in startups. The authors identified that refactoring, redesign, and rewrite of code are TD payment practices used in startups.

In our previous work [11], based on 432 answers from Brazilian, Chilean, Colombian, and North American practitioners, we identified a set of 34 TD payment practices and 28 reasons for the non-application of those practices. We reviewed these results, resulting in a slight update on the number of occurrences of each practice and reason. Table I summarizes the 10 most commonly cited practices, reporting the practice name and the total number (i.e., count) of citations (**#CP**). The column **%PP** presents the percentage of #CP in relation to the total of all projects, revealing how frequently each practice was used in software projects.

TABLE I. TOP 10 TD PAYMENT-RELATED PRACTICES [11].

NO	Practice	#CP	%PP
1st	Code refactoring	59	39.1%
2nd	Investing effort on TD payment activities	23	15.2%
3rd	Design refactoring	16	10.6%
4th	Investing effort on testing activities	15	9.9%
5th	Prioritizing TD items	12	7.9%
6th	Monitoring and controlling project activities	10	6.6%
7th	Negotiating deadline extension	9	6.0%
8th	Increasing the project budget	8	5.3%
9th	Solving technical issues	8	5.3%
10th	Update system documentation	8	5.3%

Table II summarizes the 10 most commonly cited reasons for non-payment. This table reports the reason name and the total number (i.e., count) of citations (**#CR**). The column **%PR** presents the percentage of #CR in relation to the total of all projects.

TABLE II. TOP 10 REASONS FOR NON-PAYMENT OF TD ITEMS [11].

NO	Reason	#CR	%PR
1st	Focusing on short term goals	52	30.8%
2nd	Lack of organizational interest	35	20.7%
3rd	Cost	24	14.2%
4th	Lack of time	19	11.2%
5th	Customer decision	11	6.5%
6th	Lack of resources	8	4.7%
7th	Complexity of the TD item	7	4.1%
8th	Complexity of the project	5	3.0%
9th	Insufficient management view about TD payment	5	3.0%
10th	The project was discontinued	5	3.0%

Although these studies reported several findings on TD effects and payment, none of them investigated the relationship between effects and TD payment practices and reasons for TD non-payment. We address this knowledge gap in this paper. For that, we revisit the list of effects from [4], by considering data from three new replications of *InsighTD*, and use the results on TD payment practices and reasons from [11] as is.

III. RESEARCH METHOD

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

A. Research Questions

Our main research question (RQ) is "*How do technical debt payment practices, and the reasons for not applying them, relate to the effects of the presence of debt items in software projects?"*. This question aims to identify the practices used to pay TD items off when practitioners are aware of TD effects. Besides, the RQ also seeks to recognize the reasons for not paying TD items off when practitioners feel these items' effects on their projects. To investigate this RQ, we derived the following questions:

- **RQ1**: *What are the primary TD effects felt by software practitioners in their projects?*
- **RQ2**: *What are the leading payment practices used for software practitioners when they felt TD effects in their projects?*
- **RQ3**: *What are the leading reasons considered by software practitioners to justify the non-payment of TD when they face TD effects in their projects?*

B. Data Collection

This work uses data collected from the *InsighTD* survey. Although the survey is composed of 28 questions [4], we use a subset of its questions, as presented in Table III. The characterization of the participants and their workspace are captured in Q1 to Q8. In Q13, the participants described an example of TD. Based on the example, the participants discussed TD effects in Q20 and TD payment in Q26 and Q27.

The survey invitation was sent by e-mail using LinkedIn, industry-affiliated member groups, mailing lists, and industry partners. Only practitioners were invited to participate.

TABLE III. SUBSET OF THE *INSIGHTD* SURVEY'S QUESTIONS RELATED TO TD EFFECTS AND TD PAYMENT (ADAPTED FROM [4])

No.	Question (Q) Description	Type
Q1	What is the size of your company?	Closed
Q2	In which country you are currently working?	Closed
Q3	What is the size of the system being developed in that project?	Closed
Q4	What is the total number of people of this project?	Closed
Q5	What is the age of this system up to now?	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
Q13	Give an example of TD that had a significant impact on the project that you have chosen to tell us about:	Open
Q20	Considering the TD item you described in question 13, what were the impacts felt in the project?	Open
Q26	Has the debt item been paid off (eliminated) from the project?	Closed
Q27	If yes, how? If not, why?	Open

C. Data Analysis

Due to the questionnaire being composed of closed and open-ended questions, we performed different data analysis procedures. For closed questions, we used descriptive statistics to calculate the mode and median statistics and the share of participants choosing each option. These procedures were applied for the characterization questions.

For the open-ended questions, we applied qualitative data analysis techniques [12]. In answers given to Q20 and Q27, following the process previously described in [4] and [11], we applied manual open coding resulting in a set of codes. These codes were divided into three subsets. The first subset was composed of the effects identified in Q20 (*RQ1*). The second and third subsets was formed based on the answers to Q26 (yes/no question). If the answer was positive, the code was associated with TD payment practices (*RQ2*), otherwise, the code was associated to reasons for not paying TD items (*RQ3*). This analysis was performed iteratively until no new codes were identified, resulting in three subsets with their list of codes and their respective frequency. All the analyses were performed by at least three researchers, two of them acting as coders and one of them as reviewer. For example, the effects *schedules are impacted significantly*, *not meeting deadlines*, and *missed deadlines* were cited by three survey participants. As these answers are associated with problems in meeting deadlines, they were unified under the code *delivery delay*.

As the answers given to Q20 and Q27 were related to the example of TD item described in Q13, we can relate the effects with TD payment practices (*RQ2*) and the reasons for not paying TD items off (*RQ3*). For instance, in the response of a participant, we found the effects *delivery delay* and *team demotivation* in answers given to Q20. The same participant indicated the practice *code refactoring* in her/his answer to Q27. Then, we identified two relationships between effects and practices: *delivery delay* and *code refactoring*, and *team demotivation* and *code refactoring*. This relationship indicates the existence of a co-occurrence among them, i.e., the participant used *code refactoring* for paying a TD item off, whose effects caused in the project were *delivery delay* and *team demotivation*. This analysis procedure resulted in two lists containing the relationships between effects and practices, and effects and reasons for not paying TD items off along with their respective number of occurrences.

IV. Results

The survey received 432 valid answers: 107 from Brazil, 92 from Chile, 134 from Colombia, and 99 from the United States.

A. Demographics

The majority of the participants worked in medium-sized organizations (38%, organizations with 51 to 1000 employees), followed by large (33%, more than 1000 employees) and small (29%, up to 50 employees). The participants followed mainly hybrid process model (44%), followed by agile (41%), and traditional (15%). Besides, they worked in teams composed of 5-9 people (31%), followed by teams with 10-20 people (25%), more than 30 people (19%), less than 5 people (17%), and 21-30 people (8%).

Concerning the role performed by the participants, 42% of are developers, but we found project leaders or managers (19%), software architects (17%), testers (8%), requirement analysts (4%), process analysts (3%), and others (7%). Most respondents identified themselves as having high-level of experience (59%), followed by middle (29%) and low-level of experience (12%).

Systems mentioned by participants were mainly age between 2 and 5 years (35%), followed by ones with 1 to 2 years of age (23%), 5 to 10 years old (16%), less than 1-year-old (16%) and more than ten years old (10%). Besides, the system size was generally between 10 KLOC and 1 million LOC (63%), followed by systems with size between 1 to 10 million LOC (16%), less than 10 KLOC (14%), and more than 10 million LOC (7%).

B. TD Effects felt in Software Projects (RQ1)

We identified 80 TD effects, available at https://bit.ly/2IGfKZB. Table IV summarizes the 10 most commonly cited ones. This table reports the effect name and the total number (i.e., count) of citations (**#CE**). #CE also indicates the number of projects that felt a TD effect. The column **%PE** presents the percentage of #CE in relation to the total of all projects (432), revealing how frequently each effect was felt in software projects.

The most cited effect is *delivery delay*, which impacts 25% of the software projects, followed by *low maintainability*, *rework*, and *low external quality*, impacting about 18% of the projects.

TABLE IV. Top 10 TD Effects

NO	Effect	#CE	%PE
1st	Delivery delay	99	25.0%
2nd	Low maintainability	78	19.7%
3rd	Rework	70	17.7%
4th	Low external quality	68	17.2%
5th	Increased effort	31	7.8%
6th	Increased cost	28	7.1%
7th	Low performance	25	6.3%
8th	Need of refactoring	23	5.8%
9th	Stress with stakeholders	22	5.6%
10th	Team demotivation	20	5.1%

C. Relationship between TD Effects and Payment Practices (RQ2)

Table V presents the relationship between the top 10 effects and the top 10 payment practices, indicating the number of times that each relationship occurred. The table containing all relationships is available at https://bit.ly/2IGfKZB. We can observe that the practices *code refactoring* and *investing effort on TD payment activities* stand out, both are used for eliminating TD items when a project faces all the top 10 effects. Besides, these practices along with *design refactoring*, *monitoring and controlling project activities*, and *increase the project budge* were most commonly used when a project faced the effect *delivery delay*.

When software teams feel the effect *low external quality*, they commonly use the practices *investing effort on testing activities* and *monitoring and controlling project activities*. The practice *prioritizing TD items* is commonly used when the effect *low maintainability* is felt in a project. Lastly, when practitioners feel the effect *rework*, they commonly use the practices *increasing the project budget*, *negotiating deadline extension*, *solving technical issues*, and *update system documentation*.

TABLE V. RELATIONSHIP BETWEEN TOP 10 EFFECTS AND TOP 10 TD PAYMENT-RELATED PRACTICES

Payment Practice	Effect									
	Delivery delay	Low maintainability	Rework	Low external quality	Increased effort	Increased cost	Low performance	Need of refactoring	Stress with stakeholders	Team demotivation
Code refactoring	**16**	7	12	7	2	3	6	7	2	1
Investing effort on TD payment activities	**7**	4	5	5	4	2	3	1	2	1
Design refactoring	**7**	2	2	3	1	1	1	2	0	1
Investing effort on testing	3	0	3	**4**	2	0	0	2	0	0
Prioritizing TD items	1	**3**	2	1	0	0	1	1	1	1
Monitoring and controlling project activities	**2**	1	0	**2**	0	0	0	**2**	0	0
Negotiating deadline extension	3	0	**4**	0	0	2	0	0	1	0
Increasing the project budget	**4**	0	**4**	0	0	4	0	0	2	0
Solving technical issues	2	1	**4**	0	0	0	0	0	0	1
Update system doc.	1	1	**2**	1	1	0	0	0	0	0

D. Relationship between TD Effects and Reasons for not paying TD off (RQ3)

Table VI presents the relationship between the top 10 effects and the top 10 reasons for not paying TD items off, indicating the number of times that each relationship occurred. The complete table is available at https://bit.ly/2IGfKZB. We notice that the reasons *focusing on short term goals* and *lack of organizational interest* are used for explaining the non-payment of TD items when a project faced all top 10 effects.

When software teams feel the effect *delivery delay*, they commonly justify the non-payment of TD using the reasons *lack of time*, *insufficient management view about TD payment*, and *the project was discontinued*. The reasons *lack of organizational interest*, *cost*, *lack of time*, *customer decision*, and *lack of resources* are used for explaining the non-payment of TD items when a software project faced the effect *low maintainability*. When a project feels the effect *low external quality*, the team justify the non-elimination of TD using the reasons *focusing on short term goals* and *complexity of the TD item*. Lastly, the reasons *lack of time*, *lack of resources*, *complexity of the project*, and *the project was discontinued* are used for explaining the non-payment of TD when software teams feel the effects *rework*, *low external quality*, *increased cost*, and *low performance*, respectively.

TABLE VI. RELATIONSHIP BETWEEN TOP 10 EFFECTS AND TOP 10 REASONS FOR NON-PAYMENT OF TD ITEMS

Reason	Effect									
	Delivery delay	Low maintainability	Rework	Low external quality	Increased effort	Increased cost	Low performance	Need of refactoring	Stress with stakeholders	Team demotivation
Focusing on short term goals	11	8	7	**12**	5	4	4	2	1	1
Lack of organizational interest	1	**9**	5	4	5	4	1	1	3	2
Cost	2	**7**	4	3	4	2	2	1	0	2
Lack of time	**4**	**4**	**4**	3	2	2	2	1	0	0
Customer decision	1	**4**	1	2	1	0	0	0	0	1
Lack of resources	1	**2**	0	1	1	**2**	0	0	0	1
Complexity of the TD item	1	0	1	**3**	1	0	0	1	0	1
Complexity of the project	0	1	0	0	0	0	**2**	0	0	0
Insufficient management view about TD payment	**3**	1	0	0	1	0	0	0	0	1
The project was discontinued	**2**	1	1	1	0	0	0	0	**2**	0

V. DISCUSSION

We organized the relationships between TD effects, TD payment-related practices, and reasons for not paying TD items in an alluvial diagram. This diagram is composed of nodes and links. A node represents a source, while a link shows a relation between nodes. Further, a link can have a magnitude. The greater this magnitude, the wider the link.

The effects, payment practices, and reasons were represented as nodes and the relationships between them as links. We also defined the magnitude of each link. To calculate it, we identified all practices related to each effect. Considering each effect, we summed the number of occurrences of each relationship between the effect and each practice (*frequency of relationship with a practice*). Also, we summed the number of relationships between the effect and all their practices (*frequency of relationship with all practices*). To calculate each relationship's value, we divided the *frequency of relationship with a practice* by the *frequency of relationship with all practices*. The obtained result is multiplied by 100. For example, the effect *delivery delay* was related to the practice *design refactoring* seven times; then, the *frequency of relationship with a practice* is seven. As that effect has 64 relationships in total with practices, its *frequency of relationship with all practices* is 64. The percentage of the relationship between *need of refactoring* and *code refactoring* is 7/64*100 = ~11%. We also followed this procedure for computing the magnitude of each relationship between effects and their reasons for not paying TD items.

Fig. 1 shows the alluvial diagram considering the top 10 effects and their top 3 practices and reasons. The complete diagram is available at https://bit.ly/3kALUm0. Analyzing the diagram, we notice that the practices *code refactoring* and *investing effort on TD payment activities* are the most used for eliminating TD items when a software project faces the top 10 effects. On the other hand, the reasons for not paying TD items off *focusing on short term goals*, *lack of organizational interest*, *lack of time*, and *cost* are the most used for explaining the TD non-payment when a software project faces the top 10 effects. Visualizing the relationships, the diagram also presents the percentage of each relationship. Let us consider the effect *need of refactoring*. Comparing the link's width of its practices, we can notice that *code refactoring* is the most common practice used by software teams when this effect is felt, and the magnitude of this relationship is 32%.

This information can be useful in two scenarios. Firstly, if a team is starting to manage TD, it can learn about the effects of TD and its relationship to (i) practices for eliminating them and (ii) reasons for not eliminating them. Secondly, if a team has experience in TD management, the diagram can reveal new effects, practices, or reasons for not paying TD items off. In both situations, this knowledge can be used to improve the team's workspace, facilitating the inclusion or continuity of TD management activities in its daily activities.

VI. IMPLICATIONS FOR PRACTITIONERS AND RESEARCHERS

Practitioners can use the list of TD effects to support the prioritization of TD items to pay off. The relationship between effects and practices give directions on which practices can be applied given the presence of specific effects. Also, the relationship between effects and reasons can shed some light on why TD items have not been paid due to the presence of some effects.

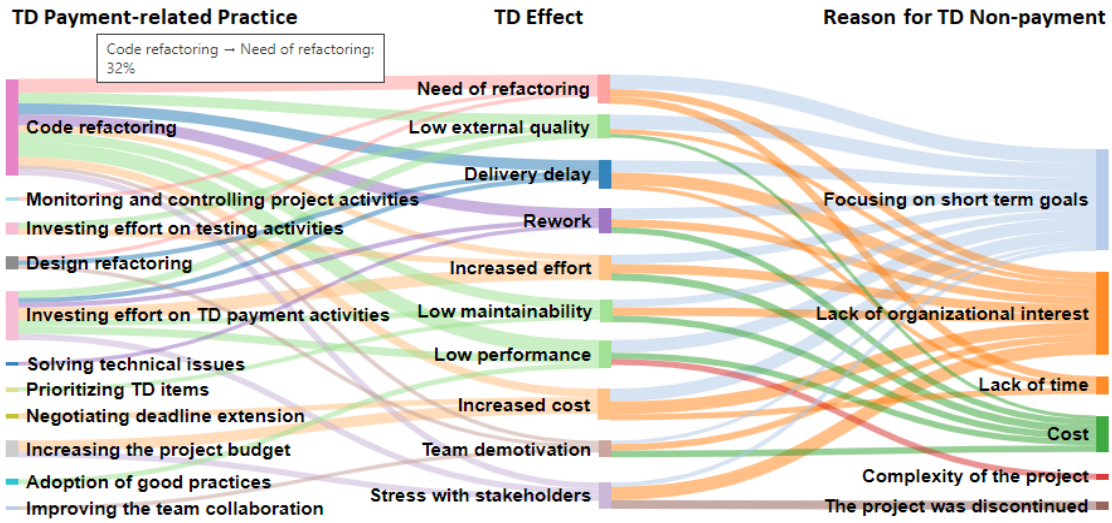

Fig. 1. Alluvial diagram for the top 10 TD effects and their top 3 TD payment-related practices and top 3 reasons for TD non-payment.

For researchers, our findings support new research on the relationship between TD effects and TD payment practices in a problem-driven way. For example, can the prioritization of TD items consider the impact of both their effects and the payment practices related to these effects?

VII. Threats to Validity

We identified some threats to validity following the categorization of Wohlin et al. [13]. About the conclusion validity, a threat arises from the coding process as it is subjective. To mitigate this threat, the coding process was performed by two researchers and disagreements were resolved by a third researcher. We reduced the external validity threats by inviting practitioners from different workspaces and countries. Although the survey was answered by a good number of participants (#432), we are not be able to estimate the representativeness of our sample given the lack of empirical data characterizing the population. We intend to use more data from other *InsighTD* replications to reach more reliable and empirically founded results. Other threats to validity that affect the *InsighTD* project are further discussed in [4].

VIII. Final Remarks

This work identifies the effects felt by software practitioners due to the presence of TD items in their projects. Further, we investigated the co-occurrence between TD payment practices and reasons for not paying TD off, and TD effects. Our results can support practitioners and researchers to understand why some effects are not paid off and identify the practices commonly used for eliminating each of these effects.

The next steps of this work include: (i) to improve the external validity considering more data from other *InsighTD* replications, and (ii) to run other analysis to investigate if the relationships are impacted by other variables, such as type of debt, used process model, participant experience and role, and organization/project size.

References

[1] C. Izurieta, A. Vetrò, N. Zazworka, Y. Cai, C. Seaman and F. Shull, "Organizing the technical debt landscape," *2012 Third Int. Workshop on Managing Technical Debt* (MTD), Zurich, 2012, pp. 23-26.

[2] R.O. Spínola, N. Zazworka, A. Vetrò, F. Shull and C. Seaman, "Understanding automated and human based technical debt identification approaches-a two-phase study,". Journal of the Brazilian Computer Society, 25 (5), 2019, doi: 10.1186/s13173-019-0087-5.

[3] P. Kruchten, R.L. Nord and I. Ozkaya, "Technical debt: from metaphor to theory and practice," IEEE Software, vol. 29, no. 6, pp. 18-21, Nov.-Dec. 2012, doi: 10.1109/MS.2012.167.

[4] Not presented due to double blind review.

[5] A. Martini and L. Bosch, "On the interest of architectural technical debt: Uncovering the contagious debt phenomenon," Journal of Software: Evolution and Process, 29:e1877, 2017.

[6] J. Yli-Huumo, A. Maglyas and K. Smolander, "The benefits and consequences of workarounds in software development projects," *2015 6th Int. Conference of Software Business* (ICSBO), p. 1–16, 2015.

[7] Z. Li, P. Avgeriou and P. Liang, "A systematic mapping study on technical debt and its management," Journal of Systems and Software, v. 101, p. 193–220, 2015, doi: 10.1016/j.jss.2014.12.027.

[8] T. Besker, A. Martini and J. Bosch, "Managing architectural technical debt: A unified model and systematic literature review," Journal of Systems and Software, v. 135, p. 1–16, 2018.

[9] W.N. Behutiye, P. Rodríguez, M. Oivo and A. Tosun, "Analyzing the concept of technical debt in the context of agile software development: A systematic literature review," Information and Software Technology, v. 82, p. 139-158, 2017.

[10] C. Apa, H. Jeronimo, L.M. Nascimento, D. Vallespir and G.H. Travassos, "The Perception and Management of Technical Debt in Software Startups," Nguyen-Duc A., Münch J., Prikladnicki R., Wang X., Abrahamsson P. (eds) Fundamentals of Software Startups. Springer, Cham, 2020, doi: 10.1007/978-3-030-35983-6_4.

[11] Not presented due to double blind review.

[12] C. Seaman, "Qualitative methods in empirical studies of software engineering," IEEE Trans. on Soft. Engineering, 25(4):557-572, 1999.

[13] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell and A. Wesslén, "Experimentation in Software Engineering: An Introduction," Springer, 2012.