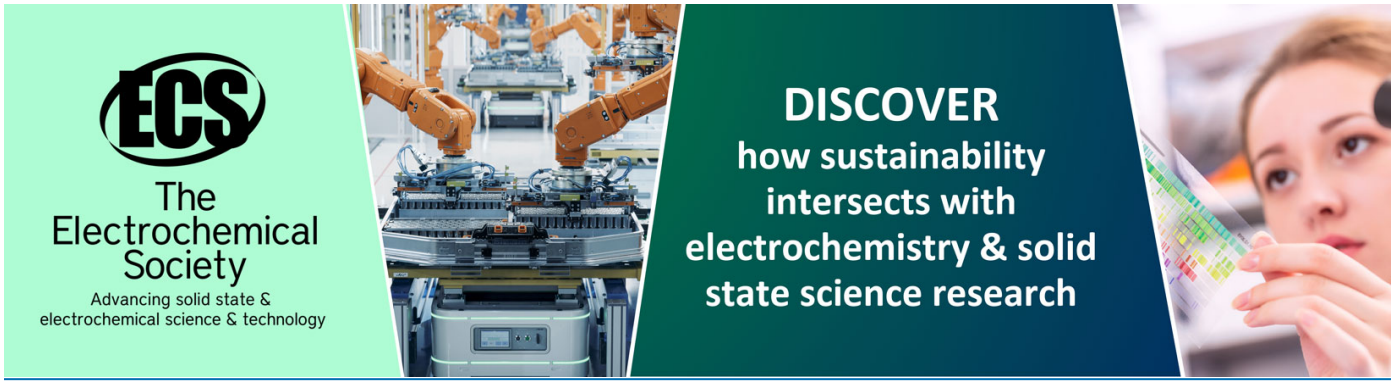
How do software architects perceive technical debt in Colombian industry? An analysis of technical debt causes



Journal of Physics: Conference Series **1513** (2020) 012003 doi:10.1088/1742-6596/1513/1/012003

How do software architects perceive technical debt in Colombian industry? An analysis of technical debt causes

B Pérez[1,2], D Correal[2], and F H Vera-Rivera[1,3]

[1] Grupo de Investigación en Inteligencia Artificial, Universidad Francisco de Paula Santander, San José de Cúcuta, Colombia
[2] Grupo de Investigación en Tecnologías de Información y Construcción de Software, Universidad de los Andes, Bogotá, Colombia
[3] Materials Science and Technology Research Group, Foundation of Researchers in Science and Technology of Materials, Colombia

E-mail: borisperezg@ufps.edu.co, fredyhumbertovera@ufps.edu.co

Abstract. Technical debt is a metaphor used to describe technical decisions that can give the company a benefit in the short term but possibly hurting the overall quality of the software in the long term. Architectural decisions are considered one of the most common sources of technical debt, therefore, it becomes relevant to understand what causes lead to technical debt from the point of view of software architects. To accomplish this task, we used a survey research method to collect and analyze a corpus of 28 software architects from Colombia, as a part of the InsighTD project. Results showed that inappropriate planning is the most cited technical debt cause by software architects. However, results differ when comparison against engineers and manager are performed. Innacurate time estimate and producing more without quality were the most selected causes of technical debt according to engineers and managers. To improve this comparison, the rank-biased overlapping technique was used. As more elements were compared, more similar were these lists of causes among all three roles.

1. Introduction

Software companies usually work under tight schedules and deadlines to release software to customers in faster cycles, therefore, increasing the pressure for the development teams to deliver working features to their customers [1]. Technical debt (TD) is a metaphor used in software development to describe technical decisions that can give the company a benefit in the short term [2, 3] but possibly hurting the overall quality of the software and the productivity of the development team in the long term. According to Ernst, *et al.* in [4], architectural decisions are the most common source of TD, therefore, it becomes crucial to understand how TD is perceived by software architects, the causes reported by them and the practices to deal with TD.

Despite the attention surrounding TD by both the industry and academia [5, 6], there is still a lack of empirical evidence about TD causes and payment-related practices used by software architects in real-life software development teams [7, 8]. This information could help software practitioners in selecting the best possible strategy to keep their software systems healthy, and therefore, to use that knowledge to improve the existing processes and tools.

This study focuses on the acknowledgment of causes leading to TD occurrence and the practices used on TD payment from the point of view of software architects in real-life software systems projects. To achieve this, we performed an industrial survey with 28 software architects from Colombia. These answers were compared against answers from management roles (project manager, business analyst, etc) and engineering roles (developer, tester, etc). The contributions of this work are two-fold. First, this study presents a list of the top seven causes (inappropriate planning being the most cited) leading to TD occurrence in software projects. And second, a numerical comparison of similarity between the list of causes cited by software architects against management and engineer groups is provided in this study.

Accordingly, the paper is organized as follow, in section 2 we present a description of the InsighTD project history. In section 3, we present the survey design, whose results are presented in section 4. Implications for researchers and practitioners are presented in section 5. Finally, in section 6, we present threats to validity, and in section 7 we conclude the paper.

2. InsighTD project

InsighTD is a globally distributed family of industrial surveys initiated in 2017 and planned cooperatively among TD researchers from around the world. 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. 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.

Rios, *et al.* [9] discussed the basic survey design and the preliminary results of the first round of InsighTD, and complemented this discussion, focusing specifically on the causes and effects of TD in agile software projects. In that paper, the authors focused on the discussion on the top 10 causes and effects of TD. Pérez, *et al.* [10] focused on how practitioners react to the presence of debt in the Chilean software industry. More recently, Freire, *et al.* [11] investigated preventive actions that can be used to curb the occurrence of TD and the impediments that hamper the use of those actions.

Thus, although significant analysis has already been conducted over the available InsighTD data, much still remains to be studied. In particular, a noticeably absent and important perspective is the one from the architect's point of view.

3. Study design

This research was designed with the goal of characterizing comprehensively the current knowledge on causes leading to TD occurrence and the current state of practices related to TD payment. Based on our research goal, we derived the following two research questions (RQ) guiding our study and the reporting of the results:

> **RQ1:** From a software architect's point of view, what causes lead software development teams to incur TD?
>
> **RQ2:** Is there any difference of causes leading to TD injection among software architects, engineers and managers?

Data gathering was done using an online questionnaire (Google Forms) to increase the number of possible participants. Invitations were sent online to software practitioners and the survey was anonymous. Survey questions were defined within the InsighTD replication package and was made up of 28 questions, previously described in [9]. Table 1 presents the subset of the survey's questions related to the context of this work.

Demographics questions (Q1 to Q8) ask participants about, for example, the size of his/her company, size of the system (in terms of LOC) he/she is working on, number of people involved

in that project, participant's role, and her/his level of experience in that role. Questions Q9 to Q15 seek information about how familiar the respondent is with the TD concept. Questions Q16 to Q19 support the identification of the causes that lead development teams to insert debt items into their projects. Questions Q20 and Q21 look to identify effects of the presence of TD in software projects. Finally, Questions Q22 to Q28, were used to provide an understanding on how TD has been managed in practice, in particular with respect to prevention, repayment, and monitoring. The full questionnaire was previously presented in [9]. In the context of this work, we considered for analysis the characterization (Q1-Q8) and causes of TD (Q16-19).

Table 1. Subset of the survey questions.

No.	Question	Type
Q1	What is the size of your company?	Closed (SC)
Q2	In which country you are currently working?	Closed (SC)
Q3	What is the size of the system being developed in that project? (LOC)	Closed (SC)
Q4	What is the total number of people of this project?	Closed (SC)
Q5	What is the age of this system up to now or to when your involvement ended?	Closed (SC)
Q6	To which project role are you assigned in this project?	Closed (SC)
Q7	How do you rate your experience in this role (at the time)?	Closed (SC)
Q8	Which of the following most closely describes the development process model you follow on this project?	Closed (SC)
Q9	How familiar you are with the concept of Technical Debt?	Closed (SC)
Q10	In your words, how would you define TD?	Open
Q11	How close to the above TD definition is your understanding about TD?	Closed (SC)
Q12	Are there any parts of the definition above from McConnell that you disagree with?	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
Q14	Why did you select this example?	Open
Q15	About this example, how representative it is?	Closed (SC)
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 motives or reasons or causes contributed either directly or indirectly to the occurrence of the TD example?	Open

The validation of the questionnaire comprised an internal validation, an external validation, and a pilot study [9]. To reach the target population (software practitioners) we utilized the social media platform LinkedIn along with industry-affiliated member groups, mailing lists, and industry partners, as invitation channels. LinkedIn gave us direct access to a large number of professionals with whom we did not have previous contact.

The survey instrument is composed of a mix of closed and open questions. For closed-ended questions, we used descriptive statistics to get a better understanding of the data. Answers for open-ended questions were codified using a code schema provided with the InsighTD replication package. We initially applied manual open coding resulting in a set of codes. The process was performed iteratively revising and unifying codes at each cycle of analysis until reaching the state of saturation, i.e., a point where no new codes were identified.

Data analysis was done focusing on architects and comparing its results against management and engineer groups. We are aware that software architects could be part of the engineer group, however, considering the focus of this study, it was decided to have software architects as a distinct group.

4. Results

In total, 132 practitioners answered the survey. After filtering answers according to their role, we found 28 (21.2%) participants classified as software architects, 37 (28%) as managers and 67 (50.8%), as presented in Figure 1.

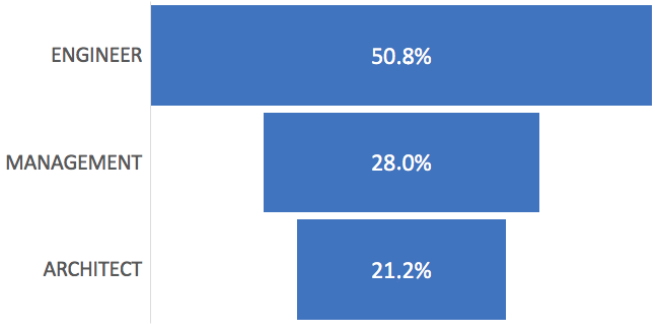


Figure 1. Practicitioners distribution by role.

Participants are well distributed among small (28.6%), medium (46.4.1%), and large (25%) companies. Related to the size of development teams, most (28.6%) reported working in teams of 5-9 people and teams of 10-20 people (28.6%). Regarding the age of the system developed in the project, most indicated age 1 to 2 years (46.4%). There are also a significant number of systems represented from 2 to 5 years (21.4%). Most respondents identified themselves as proficient (39.3%), followed by expert (28.6%), and competent (21.4%). In general, the questionnaire was answered by professionals with experience in their functions.

4.1. Main causes of technical debt occurrence (RQ1)

Figure 2 presents the seven most commonly cited causes that lead development teams to incur debt in their projects from the point of view of software architects, as informed by Q16-18. These seven causes correspond to 43.2% of all cited causes. From this Figure, we can observe that "innappropriate planning" is the most cited cause (9.9%), followed by "producing more without quality" and "inaccurate time estimate" and "inappropriate test" all with 6.2%. From this list of causes, it is possible to say that management issues are the main cause of technical debt. Only "producing more without quality" and "lack of knowledge on development tools" can be linked to technical issues.

Other causes not listed in Figure 2 are also linked to technical issues, such as "lack of experience" and "bad design". In general, software architects relies the responsibility of TD injection on bad management practices. Other cause is related to the development process and lack of a well defined process. One important thing to note is that only one software architect stated "problems in architecture" as a cause of technical debt. This could have an impact on how these causes are perceive by software architects. Another causes are linked to the human aspect of the software projects, such as "lack of knowledge, customer not listening to the project team" and "lack of awareness of customer needs". Other causes could be associated to problems in how documentation is handle, for example, "outdated/incomplete documentation", "discontinued component" and "normative changes". These problems in documentation usually are a consequence of having time pressure or deadlines [12].

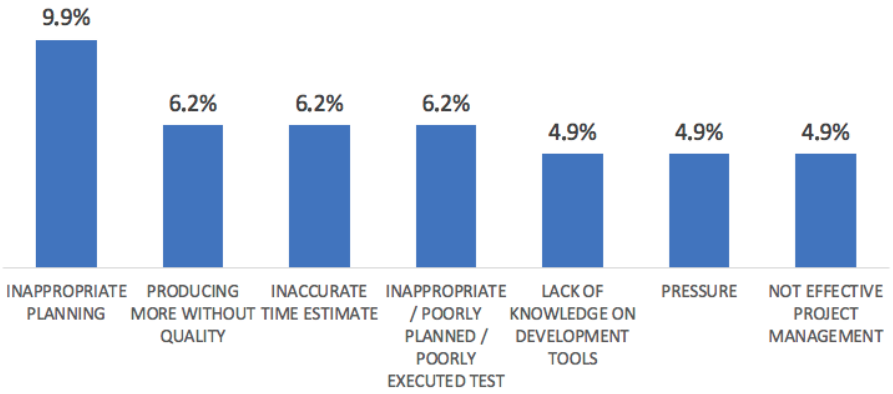


Figure 2. Top seven causes leading to TD occurrence by software architects.

4.2. Comparison of technical debt causes among software architects, engineers and managers (RQ2)

As part of this research, it is also important to measure quantitatively how similar the cited causes of TD among the three groups are (engineers, management and architects). We used a similarity measure for indefinite rankings called Rank-Biased Overlap (RBO) [13]. RBO is defined by Equation (1).

$$RBO(S,T,p) = (1-p)\sum_{d=1}^{\infty} p^{d-1} \cdot A_d, \tag{1}$$

where S and T are the ranked lists; p is the probability of looking for overlap at depth $d + 1$ after having examined element at d. The smaller the p value, the more top-weighted the metric. A_d is the agreement between S and T at depth d, *i.e.* the proportion of S and T that are overlapped. Figure 3 depicts the RBO comparison among the list of causes of the three groups (one line for each pair of groups). By increasing the p value, this comparison aims to explore how similar these causes are per group at the top of their rankings and at the bottom of their rankings. Comparison went from $p = 0.5$ (top 2 elements approx.) to $p = 0.97$ (top 33 elements approx.).

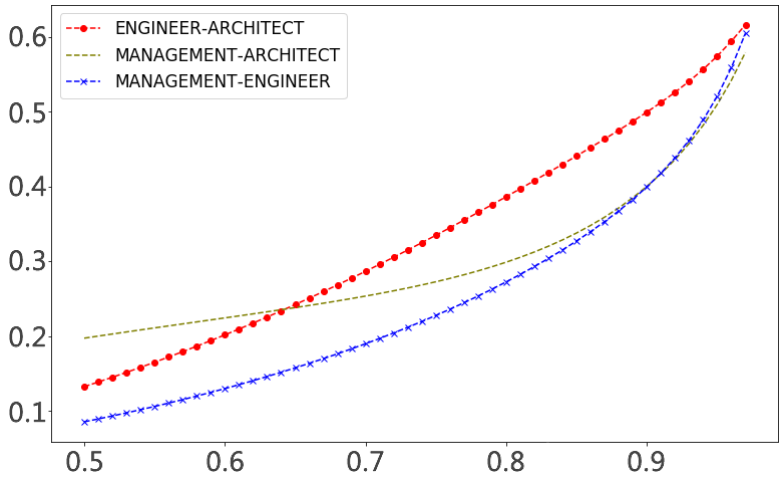


Figure 3. RBO of TD Causes.

From Figure 3 it is possible to establish that the three lists begin by being very different, and as more elements are compared, the lists begin to show similarities. For $p = 0.5$, the Management-Architect pair showed the highest similarity (RBO = 0.21), whilst Management-Engineer exhibited the lowest RBO (0.09). With increasing depth (p), the similarity between the pairs of groups tended to increase. All three list ended having the same similarity. This is particularly interesting about these lists. Also, engineer-architect pair have some special behavior, considering its increasing almost like a linear trend. It is important to note that only the first ten or fifteen elements need to be compared, the rest of the causes listed are one-time cited. Considering this, it is possible to establish that the engineer-architect pair is the most similar of all.

5. Implications to researchers and practitioners

As presented in previous section, "inappropriate planning" was the most cited cause leading to TD occurrence. However, it is important to note that "inappropriate planning" could be considered as a consequence of something else, such as: lack of qualified professionals or even deadlines imposed by the client. As presented in Figure 2, most of the causes were non-technical, meaning that problems are caused mostly by management or external aspects of software development.

Software practitioners can benefit from the results of this study by using the list of the most cited TD causes presented in industry in Colombia as a guide to support initial efforts to understand their debt and to pay it off from their software projects. Also, software practitioners could review the causes associated with the presence of TD and map their situation to these causes, allowing to define a roadmap in order to support the payment of the debt in the context of their software projects [14].

For researchers, our results support future research by providing insights into software practitioners' perspectives on causes leading to TD occurrence. Finally, the global family of surveys not only allows researchers to reproduce the results and their interpretation, but also allows practitioners to evaluate their own TD situation against overall industrial trends.

6. Threats to validity

There are threats to validity in this work that we attempt to mitigate and remove entirely when possible. First, regarding construct validity, to prevent hypothesis guessing and evaluation apprehension [15], we explain in the invitation to the survey the goal of the study and request that interviewees reply to questions by relying on their own background. Second, regarding conclusion validity, to avoid potential coding process dependencies on the researcher's subjective criteria, the coding activity was performed individually by two researchers, and then, discussed until an agreement was reached.

Regarding external validity, although the results can not be generalized, the population provides representative results from the perspective of the software industry. Finally, the lack of control over the participants is another threat to validity of this study, since it could happen that only developers interested in the TD area answer the survey. This might bias the results towards a more positive view of TD knowledge. However, a small number of the participants indicated that they were not familiar with the concept of TD, which means that this wouldn't be a significant bias.

7. Conclusions

The contributions of this work are two-fold. First, we presented a list of the top seven causes (inappropriate planning being the most cited) leading to TD occurrence in software projects. And second, we compared the lists of td causes of the roles (engineering and management) to

understand how similar or different are. This contribution was done from the point of view of 28 software architects from Colombia.

We found that "nappropriate planning" is the most cited causes of TD occurrence in software projects. From the seven most cited causes of TD, we can conclude that non technical issues are important causes of TD for software architects. We also found similarities between causes described by software architects and engineers.

The next steps of this research include: (i) a deeper analysis (including demographics variables) to identify possible patterns of TD payment related practices, (ii) investigation of how or if types of debt influence them, (iii) running other possible analyses including others reactions to TD, such as monitoring practices and preventative actions.

References

[1] Yli-Huumo J, Maglyas A and Smolander K 2016 *Journal of Systems and Software* **120** 195
[2] Kruchten P, Nord R L and Ozkaya I 2012 *IEEE Software* **29(6)** 18
[3] Verdecchia R 2018 Architectural technical debt identification: Moving forward *IEEE International Conference on Software Architecture Companion (ICSA-C)* (Seattle: IEEE) p 43
[4] Ernst N A, Bellomo S, Ozkaya I, Nord R L and Gorton I 2015 Measure it? manage it? ignore it? software practitioners and technical debt *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)* (New York: Association for Computing Machinery) p 50
[5] Seaman C and Guo Y 2011 Measuring and monitoring technical debt *Advances in Computers* vol 82 (San Diego: Elsevier) p 25
[6] Power K 2013 Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options *4th International Workshop on Managing Technical Debt (MTD)* (San Francisco: IEEE) p 28
[7] Li Z, Avgeriou P and Liang P 2015 *Journal of Systems and Software* **101** 193
[8] Rios N, de Mendonça Neto M G and Spínola R O 2018 *Information and Software Technology* **102** 117
[9] Rios N, Spínola R O, Mendonça M and Seaman C 2018 The most common causes and effects of technical debt: First results from a global family of industrial surveys *12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* 39 (Oulu: Association for Computing Machinery)
[10] Pérez B, Brito J P, Astudillo H, Correal D, Rios N, Spínola R O, Mendonça M and Seaman C 2019 Familiarity, causes and reactions of software practitioners to the presence of technical debt: A replicated study in the chilean software industry *38th International Conference of the Chilean Computer Science Society* (Concepcion: IEEE)
[11] Freire S, Mendonça M, Falessi D, Seaman C, Izurieta C and Spínola R O 2020 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* (Brno: Association for Computing Machinery)
[12] Pérez B, Correal D and Astudillo H 2019 A proposed model-driven approach to manage architectural technical debt life cycle *IEEE/ACM International Conference on Technical Debt (TechDebt)* (Montreal: IEEE) p 73
[13] Webber W, Moffat A and Zobel J 2010 *ACM Trans. Inf. Syst.* **28(4)**
[14] Martini A, Sikander E and Madlani N 2018 *Information and Software Technology* **93** 264
[15] Wohlin C, Runeson P, Hst M, Ohlsson M C, Regnell B and Wessln A 2012 *Experimentation in Software Engineering* (Berlin: Springer-Verlag)