A Taste of the Software Industry Perception of Technical Debt and its Management in Uruguay

A survey in software industry

Cecilia Apa
ceapa@fing.edu.uy
Universidad de la República
Montevideo, Uruguay

Martin Solari
martin.solari@ort.edu.uy
Universidad ORT Uruguay
Montevideo, Uruguay

Diego Vallespir
dvallesp@fing.edu.uy
Universidad de la República
Montevideo, Uruguay

Guilherme Horta Travassos
ght@cos.ufrj.br
PESC/COPPE, Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

ABSTRACT

Background: Technical debt (TD) has been an important focus of attention in recent years by the scientific community and the software industry. TD is a concept for expressing the lack of internal software quality that directly affects its capacity to evolve. Some studies have focused on the TD industry perspective. **Aims:** To characterize how the software industry professionals in Uruguay understand, perceive, and adopt technical debt management (TDM) activities. **Method:** To replicate a Brazilian survey with the Uruguayan software industry and compare their findings. **Results:** From 259 respondents, many indicated any awareness of the TD concept due to the faced difficult to realize how to associate such a concept with actual software issues. Therefore, it is possible to observe a considerable variability in the importance of TDM among the respondents. However, a small part of the respondents declares to carry out TDM activities in their organizations. A list of software technologies declared as used by practitioners was produced and can be useful to support TDM activities. **Conclusions:** The TD concept and its management are not common yet in Uruguay. There are indications of TD unawareness and difficulties in the conduction of some TDM activities considered as very important by the practitioners. There is a need for more effort aiming to disseminate the TD knowledge and to provide software technologies to support the adoption of TDM in Uruguay. It is likely other software engineering communities face similar issues. Therefore, further investigations in these communities can be of interest.

CCS CONCEPTS

• **General and reference** → *Surveys and overviews*; • **Software and its engineering** → **Software post-development issues**.

ESEM '20, October 8–9, 2020, Bari, Italy

ACM ISBN 978-1-4503-7580-1/20/10…$15.00
https://doi.org/10.1145/3382494.3421463

KEYWORDS

Technical debt, survey, experimental software engineering, empirical software engineering

ACM Reference Format:
Cecilia Apa, Martin Solari, Diego Vallespir, and Guilherme Horta Travassos. 2020. A Taste of the Software Industry Perception of Technical Debt and its Management in Uruguay: A survey in software industry. In *ESEM '20: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20), October 8–9, 2020, Bari, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3382494.3421463

1 INTRODUCTION

The concept of technical debt (TD) has been widely used in the software engineering community since its introduction by Ward Cunningham in 1992 [7]. It represents a simple way to capture and communicate typical software project issues: balancing the short-term value against the internal software quality to support the software product's proper evolution.

Therefore, TD's understanding and management from an industrial perspective represent a valuable input to support software organizations in reducing the risks involved in the accumulation of TD and its improper management. Besides, it can reveal to software engineering researchers the need for future investigations to benefit the professional practice. Therefore, several studies conducted in the industry have focused on observing the understanding and perception of TD by practitioners, as well as the level of adoption of TD activities and the tools used to their support ([18] [28] [26] [31] [9] [15] [22]).

In 2017, some Brazilian researchers performed a survey to characterize the TD and its management under the perspective of Brazilian software organizations using their practitioners as proxies [8]. At that moment, the results have shown that TD was still unknown to a considerable fraction of the participants. Besides, only a small group of organizations informed to adopt TD management (TDM) in their projects. The results were reported as evidence briefings and sent back to the industry, aiming improving TD awareness and its importance when the Brazilian software industry is building software products.

Cultural development issues can jeopardize software building. Therefore, understanding how the Uruguayan software practitioners perceive and manage TD is highly significant because the software industry is the third-largest exporting sector in Uruguay. In South America, Uruguay is the largest software exporter per capita, and the third-largest worldwide, exporting software to 52 countries. Improving the software evolution is highly relevant to its software industry and the Uruguayan government.

Therefore, the IS.uy program[1] was created in 2019 to collaborate with the Uruguayan software industry through evidence-based software engineering research. The IS.uy program aims to promote collaborative and applied software engineering research among industry, government, and academia. The understanding and management of TD have an essential place in it. Within the framework of the IS.uy program, the replication of the Brazilian survey on TD intended to contribute to the improvement of software development in Uruguay regarding the understanding of TD and its management in the software projects.

The replicated survey supported the observation of different issues regarding TD and its management. However, for the sake of industrial interest, this paper offers just a selection of industry-focused results and its corresponding challenges.

In general, it has been observed that there is no common understanding of TD among the Uruguayan software practitioners, and there is a low level of TDM adoption. Some apparent contradictions between the perceived TD importance and the declared TDM activities adoption suggest that there is an actual difficulty in the systematic conduction of TDM activities, mainly regarding the measurement of TD.

The rest of this paper is structured as follows: section 2 provides some background on TD and related studies; section 3 presents the survey design and conduction; section 4 explains the survey results regarding the research questions; section 5 offers the results discussion, section 6 presents the limitations and threats to validity, and section 7 presents the conclusions with the implications for practitioners and researchers.

2 TECHNICAL DEBT BACKGROUND AND RELATED STUDIES

The technical literature offers some definitions of TD, such as [25] [19] [27]. However, a recent Dagstuhl Seminar of 2016 [2] defined TD as *"a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible."* This study uses it as "the TD definition".

Many studies have focused on presenting a consolidated state-of-the-art on TD, synthesizing the evidence presented in the technical literature. Some tertiary and secondary studies characterize TD in the software life cycle [25] [29] [23] and its management [25] [23] [11] [5] [4]. These studies define different types of TD from different perspectives and regarding its occurrence, its influence, the software artifact where it can be identified, and the phase or the activity in the software life cycle where it was introduced. Despite the valuable work these academic-based studies represent, the voice of the software industry needs to be heard to contrast, validate, strengthen, and extend such TD knowledge.

In this sense, some studies focus on the industrial perspective. The practitioners' perception of TD, its causes, impact, and occurrence in software projects has been investigated ([18] [28] [21]). For instance, the InsighTD[2] project represents a family of surveys on the perception, causes, and consequences of TD. Their surveying strategy relies on continuous and independent replications of a questionnaire in different countries (Brazil, Chile, Colombia, Costa Rica, Finland, India, Italy, Norway, Saudi Arabia, Serbia, and the United States). Some results were reported recently ([28] [24] [12]), which supported the researchers to produce a list of causes and effects organized in probabilistic cause-effect diagrams. These diagrams highlight the most contributing causes of TD, as well as the most common effects resulting from the debt. Despite its potential and coverage, the survey asks the participants about their familiarity with the TD definition. However, it does not look for the performed TDM activities (and which of them) and their corresponding support technologies.

Other works focus on the industry perception about TDM activities, strategies, and technologies supporting them ([31] [9] [15] [22]). Yli-Huumo et al. present an exploratory case study in a large software development organization regarding a framework of activities, practices/tools, stakeholders, and responsibilities of TDM [31]. Ernst et al. surveyed three large US organizations, followed by some interviews to understand how software engineers deal with TD in their software projects and about the used tools and techniques to manage it [9]. Martini et al. combined empirical methods (surveys, interviews, and case studies) to investigate the state of practice in different software companies in order to understand how they start tracking TD [22]. In another study, Komyakov et al. [16] undertook an SLR to investigate the available techniques for evaluating TD using automated tools.

In summary, several studies conducted in the industry have focused on observing the understanding and perception of TD by practitioners, as well as the level of adoption of TD activities and the tools used to support them. However, to our knowledge, no study focuses on the perspective of the Uruguayan software industry on TD. Conducting replications of studies in different contexts is very valuable in empirical software engineering [13] because they contribute to provide new evidence and strengthen the software engineering body of knowledge.

3 SURVEY DESIGN AND CONDUCTION

This study's objective is to understand and characterize how the software professionals in Uruguay understand, perceive, and manage TD, as well as the level of adoption of TDM technologies, using the engaged practitioners as proxies. The research questions are:

- **RQ1**: Is there a consensus on the perception of TD among software practitioners? The purpose of this question is to determine whether the understanding of TD is homogeneous among professionals. If so, this perception can be compared with the academic perspective looking for common ground.
- **RQ2**: Do practitioners perceive TD in their software projects? Before characterizing the TDM activities, it is essential to

[1] Ingeniería de Software del Uruguay, https://is.uy/

[2] http://www.td-survey.com/project-info/

confirm that the software organizations (through their practitioners) understand, i.e., observe the presence of TD in their projects.

- **RQ2.1**: Do the companies manage TD? If TD is perceived, it is essential to know whether companies manage it in their software projects.
  - **RQ2.1.1**: What TDM activities are most relevant to software projects? The goal of this question is to identify which TDM activities, among those proposed by Li et al. [20], are more relevant or at least more often considered during software projects.
  - **RQ2.1.2**: Which technologies and strategies are adopted for each TDM activity? For all eight TDM activities proposed by Li et al. [20], which approaches and techniques are used to their support.

The differences between this replication and the original [8] are the following:

- All survey materials were translated from Portuguese to Spanish.
- It adds a new question to characterize startup companies.

The survey is divided into fourteen sections: three sections to characterize the participant, the organization, and the software project; one part regards the TD perception; one section relates to TDM in general, and eight sections are concerned with specific TDM activities. The last section intends to collect information on TDM activities not eventually listed by Li et al. [20] but used in the participant's software organization. The complete laboratory package is available at https://doi.org/10.6084/m9.figshare.5923969.

A pilot trial was performed in January 2019 after reviewing all the materials by the Brazilian and Uruguayan researchers. It involved four software engineering researchers from IS.uy Program with a strong relation with software industry. Next, the survey invitation was distributed in February 2019 among personal contacts, to the more than 500 software practitioners subscribed to the IS.uy mailing list, social networks (Facebook, LinkedIn and Twitter), as well as through IT associations (CUTI[3] and Uruguay XXI[4]). The survey was conducted between February and March 2019. 397 participants answered the survey. After discarding incomplete and invalid responses, 259 valid responses were obtained.

4 RESULTS

The level of participation in the survey can be considered high. We assume that it was due to the effort invested in the dissemination of the study. The strong cooperation between industry and academy in Uruguay and the multiple personal contacts in the industry that the researchers have helped to strengthen this response rate. These results will be object of discussion in Section 5.

4.1 Participants' characterization

Out of the 259 valid answers, 57% of the participants work in companies with more than 100 employees. The participants' activity areas are quite diverse. Figure 1 shows that a large part is concentrated in Information Technology followed by Financing.

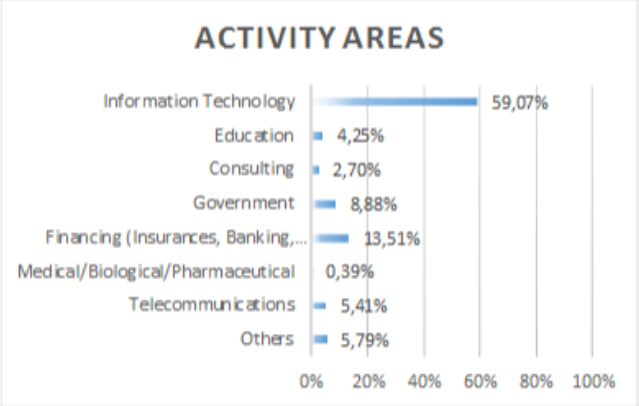

Figure 1: Participants' organization - Activity areas

The majority (85%) of the respondents state to use agile or incremental life cycles (62% and 23% respectively) in their software projects (to develop or evolve software products), leaving 15% for waterfall-type approaches, spiral and others (Figure 2).

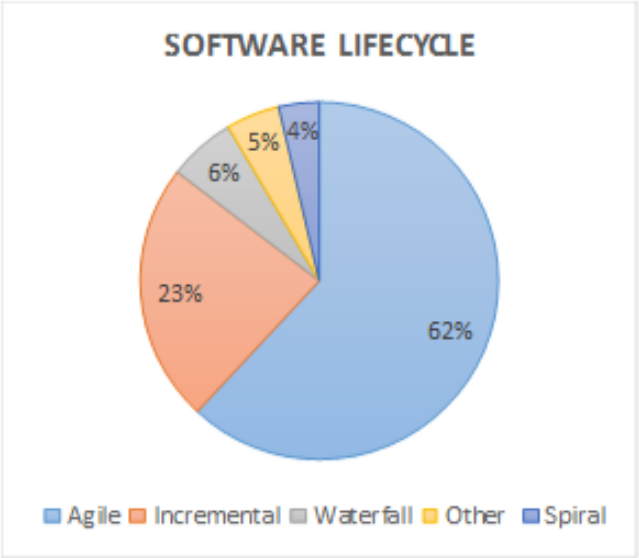

Figure 2: Participants' organization - Software lifecycle

Concerning the participants' roles (more than one role could be marked), 45% mentioned to play leadership roles, 41% informed to be programmers or software developers, 19% act as software architects, 14% reported doing requirements analysis, and 21% declared to perform other types of activities.

Thirty-three participants (13%) informed to work in startups. An in-depth analysis of this survey results, considering only the startup context is available in [1].

4.2 TD Awareness and TD Perception

Regarding the perception of TD, 109 respondents (42%) claimed to be unaware of the TD concept. In contrast, the remaining 150 (58%)

[3]Cámara Uruguaya de Tecnologías de la Información, https://www.cuti.org.uy/portada
[4]https://www.uruguayxxi.gub.uy/en/

Table 1: TD perception - related issues

Issue	% of participants
Architectural problems (like modularity violation)	**79%**
Low internal quality aspects, such as maintainability and reusability	**78%**
"Shortcuts" taken during design	**78%**
Poorly written code that violates code rules	**59%**
Presence of known defects that were not corrected	47%
Code smells	44%
Trivial code quality issues, that do not violate code rules	42%
(*)Low external quality aspects, such as usability and efficiency	40%
(*)Planned, but not performed, or unfinished, tasks	24%
(*)Lack of support processes to the project activities	24%
(*)Defects	24%
(*)Required, but unimplemented features	22%

declared to understand the concept and related it to issues that best match (from their point of view) to the TD metaphor. Table 1 shows these issues sorted by the number of votes. According to the TD definition [2], some of the problems (marked with '*' in Table 1) presented in the survey do not relate to TD.

Although all the participants who indicated the TD issues reported to be aware of the TD concept, there is no consensus about which items are associated with this concept (no option achieved 100% of answers). However, more than 78% of the respondents agreed that the concept of TD is associated with internal software quality problems, architectural problems, and design shortcuts (in adherence to the TD definition). Code smells were selected by less than half of the participants (42%) who declared to be aware of the TD concept. Code smells represent an excellent example of poor internal software quality that negatively affects the maintenance and evolution of software, and it is directly concerned with the TD definition. Contradicting the TD definition, 40% of the respondents associated it with external quality problems.

The degree of adherence to the TD definition relates to the responses of the participants. It is possible to observe that from the 150 (58%) participants who declared knowing the TD concept, only two of them selected all the TD issues concerned with the TD definition and did not select any of the issues that does not directly relate to it.

From this same 58% of TD aware participants, 126 of them (48% of all participants) perceived the occurrence of TD in their software projects, and 83 (32% of all participants) of these 126 participants stated that their organizations or their project managers have carried out TDM activities in their projects (Figure 3).

4.3 TD Management

Taking into account the total number of participants (256), only 32% (83) declared to carry out TDM activities. We asked these participants what TDM activities their organizations adopted and

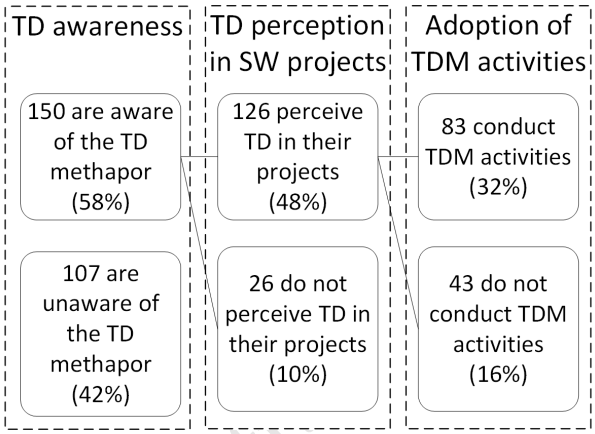

Figure 3: Summary of TD awareness, TD perception and TD management responses

which roles are responsible for them. In turn, we also asked them to rank the different TD activities according to the importance that they should have in their organization, in their perspective.

Figure 4 shows the adoption (in %) of each TDM activity (blue bar). In turn, on a secondary axis, the green bars show the degree of importance given by the participants (measured through a weighted vote count) to each TDM activity.

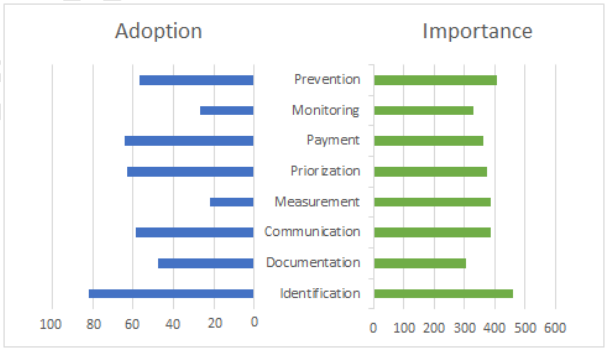

Figure 4: TD Importance vs. TD Adoption

Following the blue bars, the most adopted TDM activity is TD identification (82%), followed by TD payment (64%) and TD prioritization (63%). The less taken TDM activity is TD measurement (18%), followed by TD monitoring (27%). Following the green bars, the activities voted as most important are TD identification (461 points), TD prevention (408 points), TD measurement, and TD communication (388 points each). The activities voted as least necessary are TD documentation (306 points), TD monitoring (327 points), and TD payment (362 points).

As shown in Figure 4, for TD measurement and TD monitoring activities, there is a shallow adoption compared with the relative importance that is given to these activities. TD measurement was ranked as one of the essential activities but is the least adopted.

Although TD identification is the activity with the highest degree of adoption, it does not reach 100%. There are 15 participants (18%)

Table 2: TD Results about responsibilities of each TDM activity

TDM Activity	PM	TL	SA	DT	None	TDM-F
Identification	15%	57%	54%	**87%**	0%	DT, SA
Documentation	13%	50%	45%	**65%**	0%	DT, SA
Communication	45%	**73%**	35%	47%	0%	DT, SA, TL/PM
Measurement	22%	33%	50%	**56%**	0%	SA, TL/PM
Prioritization	**65%**	60%	33%	33%	0%	SA, TL/PM
Payment	17%	42%	45%	**87%**	4%	DT, SA
Monitoring	36%	**55%**	36%	36%	5%	SA, TL/PM
Prevention	34%	72%	62%	**77%**	2%	DT, SA

PM = Project Management, TL = Team Leader, SA= Software Architect, DT = Development Team, None = None of the previous roles, TDM-F = TDM Framework [18]

who do not carry out TD identification but carry out other activities such as TD documentation, TD prioritization, and TD monitoring.

Regarding the responsibilities, there were differences among participants on which roles should be responsible for each TDM activity. Table 2 shows the percentage of participants who marked each role as responsible for each TDM activity. The results show some agreement between the participants' responses and the TDM framework proposed by Yli-Huumo et al. study [31].

Most of the TDM activities show an agreement with the TDM framework. However, in the case of the TD measurement, most of the participants indicated that it was the responsibility of the development team. In contrast, the framework suggests that the primary responsibility should be with the architect or project manager/team leader. At least 33% of the participants marked the development team as responsible for each TDM activity.

Table 3 presents a list of practices, techniques, and tools used in each TDM activity, as reported by the participants. The numbers in parentheses represent the number of participants answering that specific section (column "TDM activity") and the number of participants that claimed using that technology or strategy (column "Technologies and strategies"). We can observe that different technologies support TDM, and it is not possible to observe any common pattern about which one to use.

5 DISCUSSION

RQ1: Consensus on the awareness of TD. We did not observe consensus among participants on the awareness of TD. Each participant was asked to select which of the 12 suggested issues should be associated with the TD concept, as presented in Table 1. Out of those options, only three issues were selected by 75% or more of the 150 respondents who claimed to be aware of the TD concept.

None of the suggested issues received indications from all of the TD aware participants. In addition, only two participants demonstrated a 100% adherence to the TD definition.

The lack of adherence to the TD definition by many participants, together with a lack of TD perception in the software projects by a minor percentage of participants, can indicate the existence of misconceptions regarding TD. It leads the participants to associate the TD definition with any issue occurring during the software development. However, despite this apparent misconception, it is also possible to observe some alignment with the TD definition since most of the indicated TD issues are concerned with its definition.

It can suggest a smooth and not a consistent spread of TD concept among the software practitioners in Uruguay. Therefore, it will be imperative to promote better dissemination on the distinction between issues related to internal quality, which impact the software evolution (in this particular case, due to TD) and those related to external quality or defects, which have no impact on software evolution.

RQ2: practitioners' perception of TD Eighty-four percent of 150 participants claimed to perceive TD in their software projects. Although this number could be considered high, 16% declared not to perceive any TD. In general, it is unlikely not to have any TD at some point during software development. Even when there is an exceptional architecture quality, clean code, among others, TD often arises from external factors: technological obsolescence, change of environment, rapid commercial success, and the advent of new and better technologies. Kruchten mentioned, *"Even an architectural design that makes the system more flexible and adaptable than it really should be, can be a form of TD, if this additional flexibility hinders future development without actually being exploited"* [19]. Why some participants did not perceive TD is a question that we think can reveal some misunderstandings about the TD concept or possibly indicating a lack of quality perception in the overall product internal quality perspective.

RQ2.1: Do the companies manage their TD? Considering the total of valid answers (259), the declared adoption of TDM activities is low (32%). The participants stated to adopt at least one of the eight TDM activities presented. Therefore, some of these adoptions are, in fact, a weak adoption of TDM activities. Because we only asked about TDM adoption to the participants who declared knowing the TD concept, there is a small chance of a high TDM adoption. Possibly, participants who were unaware of the TD concept, but were familiar with internal quality issues (e.g. low internal quality, design erosion, among others), actually managed TD but were unable to respond to that question.

The responsibilities declared by participants show some agreement with the TDM framework [31]. At least 33% of the participants marked the development team as responsible for all TDM activities. Maybe the "agile culture" influenced it because more than 80% of the participants declared to follow agile software life cycles.

Figure 5 shows a timeline in which our study, the original study conducted in Brazil [8], and the Rios et al. study [26] are placed temporarily according to their period of execution. Comparing the results of the three studies (Figure 6), we can observe a very similar distribution of TD awareness. Regarding the perception of TD, it became difficult to compare with Rios' study [26] because it does not have a specific question of TD perception without taking into account a low level of TD management. Compared with Da Silva's study [8], we observe a higher level of TD perception in Uruguay. Regarding TD management, Uruguay presented a slightly

Table 3: TDM activities in software companies in Uruguay- technologies and strategies

TDM Activity	Technologies and strategies
TD identification (68)	Manual code inspection (59), dependency analysis (33), checklist (23), SonarQube/SCALE (15), FindBugs(9), Code Climate (2)
TD documentation / representation (40)	TD backlog (21), specific artifacts for TD documentation (1), JIRA (24), Wiki (9), Trello (2), Excel (1), Youtrack (1)
TD communication (49)	Discussion forums (14), TD topic in project meetings (29), specific TD meetings (22)
TD measurement (18)	Manual measurement (7), SonarQube (4), JIRA (8), Wiki (2), Proprietary tool (1)
TD prioritization (52)	Cost/benefit analysis (15), specific technology for decision-making (6), classification of issues (35)
TD payment (53)	Refactoring (47), Re-writing code (44), Re-design (35) TD monitoring (22), Manual (15), SonarQube (2), JIRA (8), Wiki (3), Definition of Done (2)
TD prevention (47)	Guidelines (32), coding standards (40), code revisions (44), retrospective meetings (36), Definition of Done (24)

higher level than Da Silva's study [8] and more elevated level than Rios' study [26]. It represents an unexpected observation whether comparing the sampling sizes and results of both Da Silva's and Rio's studies.

Figure 5: Timeline

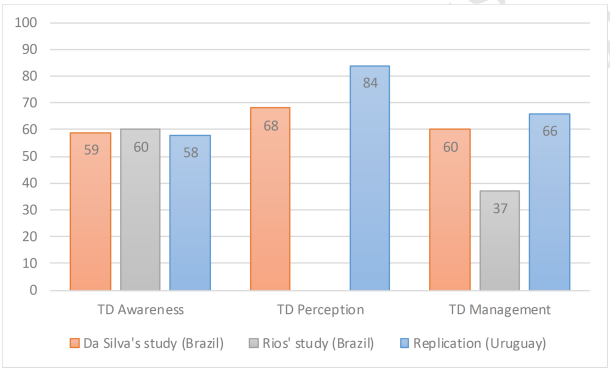

Figure 6: Studies' results comparison

Despite the fact that the time difference in conducting studies is small (one and a half years), it is possible that the industry has deepened its TD knowledge. Also, it is possible that the performing of Da Silva's study [8] improved to some extent the TD awareness when compared with the Rios' study [26]. Regarding our study, we carried out a TD workshop in October 2018 (with the participation of 30 Uruguayan software practitioners), which could have a positive effect on Uruguay's results. Finally, the sample size in each study is different (Da Silva's study = 37, Replication in Uruguay = 259, Rios' study = 112) and therefore affects the results' confidence. Perhaps this could explain the differences found between the studies, especially between the two Brazilian ones.

RQ2.1.1: What TDM activities are most relevant to software projects? The activities declared as most important for the participants are TD identification, TD prevention, and TD measurement. Most studies regarding TDM focus on TD identification, measurement, and prioritization, which shows some agreement with the declared importance. However, TD prevention was declared as very relevant to TDM. Maybe it is worth paying more attention to it.

None of the activities are declared as adopted by all participants. TD identification and TD prevention have a high rate of adoption. Still, the participants declare TD measurement as the least adopted activity, although it has been ranked as one of the three most important activities. This could indicate the existence of difficulties in the adoption of activities that are considered essential to carry out and should be explored further in future works.

Despite TD identification being declared as the most adopted activity, there are some responses where other TD activities are declared as adopted without including TD identification. It draws our attention as it goes against the natural sequence between some TDM activities, which can be derived from the activities description presented in several studies [20] [14]. In the Li et al. study [20] the definition of TD prioritization is *"ranks identified TD according to certain predefined rules to support deciding which TD items should be repaid first and which TD items can be tolerated until later releases."* We observed this lack of "order" in many responses about TDM adoption.

Although this apparent contradiction (paying a TD that has not been previously identified), we conjecture that the participants answered about the systematic adoption of formal TDM activities. The adoption of TDM activities can be recognized as immersed in other software management or quality assurance activities. For example, TD identification can be performed when a programmer is implementing a change in existing code and realizes that the old code has intricate and duplicated parts that can be improved by reusing built-in software components. That is, to ad-hoc identify TD. Also, that identified TD can be documented, e.g., in the form of code comments like *fixme*, *hack*, or *TODO*, without following any

agreed practice or standard that, e.g., making challenging to track TD in the software project. Thus, many times, there is no systematic execution/conduction of TDM activities. Developers simply "*know*" about the existence of TD and pay it. The consequences of not carrying out TDM activities in a systematic way can be experienced like underlying maintainability issues in software systems, which sometimes, developers perform ad-hoc identification. Other times, developers make their identification so late that the cost and effort to address them are incredibly high, making the evolution of the system quite hard or unfeasible.

These findings are in agreement with other study findings. In Martini's study [22], the observed TDM Adoption was as much in ad-hoc or manual level, according to their "Strategic Adoption Model for Tracking Technical Debt". In Ernst's survey [9], more than 65% of the participants declared not to have a standard approach to manage TD.

RQ2.1.2: Which technologies and strategies are adopted for each TDM activity?

Table 3 shows a list of technologies and tools the participants reported to manage TD activities. Some technologies are used for several activities. An example is *Sonarqube*[5], which is used for identification, measurement, and monitoring of TD. In the case of TD measurement from the 18 participants who declared to adopt it, seven participants (38%) declared to carry out the measurement activity manually, without any tool support. TD measurement has been presented as one of the most complex TD management activities [17], which justifies the research efforts concentrated on it. Khomayakov et al. [16] reported 20 available TDM tools for evaluating TD. From these, the participants in Uruguay declared only to use two (*SonarQube* and *FindBugs*[6]).

Overall, the adoption of tools is rather low. It is not necessarily due to the unavailability of TDM tools. Perhaps, it may be due to the lack of knowledge on the available tools by the participants, or even a lack of empirical evidence on tools' potential usefulness. For instance, Khomayakov et al. [16] indicated that only *SonarQube* and *Findbugs* have been empirically evaluated. Again, the same two tools reported by the participants in Uruguayan replication.

The technologies presented can be used in further studies looking for evidence on their effectiveness and efficiency in managing the TD, helping to increase their adoption rate. Despite that, we believe that the use of tools per se does not solve the problems caused by the accumulation of TD if they are not supported by the adoption of TDM activities in the software development process.

6 LIMITATIONS AND THREATS TO VALIDITY

This replication has limitations and validity threats, as it regards any empirical study [30]. Regarding the generalization of the results that affect **the external validity**, we had a relatively high response rate, according to the Uruguayan population (3.5 million), when compared with the original study [8] and with other related works [26] [31]. The participants' characterization presented in section 4.1 revealed a high level of diversity concerning the participants' role, the size, type, and activity area of the participants' organization. The level of awareness, perception, and management of TD was similar to the first results from the Brazilian study. These results cannot be generalized to the entire Uruguayan software industry or even to other software engineering communities. However, the high response rate can suggest a certain level of confidence in the results.

Regarding the participants' bias and the instrumentation that affect **the internal and the construct validity**; the participants might have misunderstood some terms and concepts presented in the questionnaire, based on their different experiences and knowledge. The issues that can be related to the TD definition and the TDM activities organized in [20] were obtained from the technical literature. To minimize the impact of this threat, we conducted a pilot trial, which contributed to evolve all the materials before performing the replication.

To mitigate the research bias in the data analysis that affects the **conclusion validity**, we minimized the amount of open questions in the questionnaire to avoid the subjective data interpretation.

7 CONCLUSIONS AND FUTURE WORKS

In this work, we presented the results of a survey replication conducted with the software industry in Uruguay. Its results were discussed and compared with the Brazilian study (original) and other related works. It was possible to observe that there is no common ground yet on how software practitioners in Uruguay perceive the technical debt, and there is a low level of performing of TDM activities' and tools' adoption. The perceived importance of TDM activities seems to contradict their adoption since some activities such as TD measurement were declared as very important, but in practice little adopted. It could indicate the existence of difficulties in adopting TDM activities. In addition, the observed "lack of order" in the adoption of such management activities can indicate that there is non-systematic conduction of TDM activities, which can bring some difficulties in the management of technical debt in software projects.

It is expected that the results of this study provide the following contributions:

- **For software practitioners**: they suggest that there is a need for a better understanding of the TD concept and to improve the adoption of TDM activities and tools in software projects. The findings in Uruguay present a list of technologies that can be used to support TDM activities, which need to be evaluated according to the context of each software project. The results also indicate the roles involved in each TDM activity, as informed by the practitioners who conduct TDM activities in their software projects. It can be used as an initial guideline to assign TDM responsibilities into a software project. However, the results also indicate a lack of activities systematization in software organizations, which reduces their capability of perceiving and managing TD.
- **For researchers**: our results strengthen the findings of the original study in Brazil and confirm the results of other related studies. It provides higher confidence and increases their level of generalization. The results indicate that there is a need to improve the dissemination of the TD knowledge to practitioners, as well as to provide more precise instructions and directions on how to adopt TDM activities. Besides, there

[5]https://www.sonarqube.org/
[6]http://findbugs.sourceforge.net/

is a need for further evaluation and proposals of software technologies to support the management of technical debt. The replication of this survey can represent an opportunity to support the investigation of TD and its management in other software engineering communities.

As has been raised, the results indicate the existence of difficulties in adopting some TDM activities and systematizing them. Why TDM is not adopted in a more systematic manner and what understanding do companies have of the costs and benefits of managing TD are questions that can not be answered with this research, but they will guide our future research. Therefore, the next steps of this research include investigations together with the Uruguayan software industry aiming to propose tailored strategies to support them in a viable way for dealing with technical debt and its management in their software projects. In this sense, we started to conduct focus group-based studies in different contexts to acquire the challenges and needs of TD. Also, a more in-depth analysis of the survey's data is necessary to provide further insights from different and complementary perspectives. The Uruguayan software industry received the results of this study through different mechanisms:

- An evidence briefing distributed throw the IS.uy mailing list, which is available at https://www.fing.edu.uy/sites/default/files/biblio/38129/technical%20debt%20-%20uruguay.pdf.
- Realization of a meeting with software practitioners from different organizations invited through the MIS.uy group (which is part of the IS.uy program) at the *meetup.com*[7] platform (which is part of the IS.uy program) at the https://www.meetup.com/es-ES/MIS-uy/events/262071850/.
- An industry talk at the *29th GeneXus Meetings*, which is available at https://meetings.genexus.com/2019/speakers/ENG#S, technical-debt-and-how-are-we-doing-at-home.
- A short talk recorded to the *#StayAtHome* series of videos of the IS.uy program, created due to the context generated by COVID-19. The video is available at https://www.youtube.com/watch?v=DM3C0cTcdc0&t=5s.

We believe in the need for more collaborative research between academia and industry, focusing on solving real problems, as highlighted by the software engineering community [6] [3] [10]. That is one of the main objectives of the IS.uy program and the study presented in this paper. Furthermore, this is one of the biggest motivations to keep going with our research.

ACKNOWLEDGMENTS

To the participants that answer the questionnaire and the IS.uy program to support this investigation. Prof. Travassos is a CNPq Research in Brazil and an ISERN member.

REFERENCES

[1] Cecilia Apa, Helvio Jeronimo, Luciana M Nascimento, Diego Vallespir, and Guilherme Horta Travassos. 2020. The Perception and Management of Technical Debt in Software Startups. In *Fundamentals of Software Startups: Essential Engineering and Business Aspects*, Anh Nguyen-Duc, Jürgen Münch, Rafael Prikladnicki, Xiaofeng Wang, and Pekka Abrahamsson (Eds.). Springer International Publishing, Cham, 61–78. https://doi.org/10.1007/978-3-030-35983-6{_}4

[2] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. *Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162).* Technical Report 4. 110–138 pages. https://doi.org/10.4230/DagRep.6.4.110

[3] Victor Basili, Lionel Briand, Domenico Bianculli, Shiva Nejati, Fabrizio Pastore, and Mehrdad Sabetzadeh. 2018. Software Engineering Research and Industry: A Symbiotic Relationship to Foster Impact. *IEEE Software* 35, 5 (2018), 44–49. https://doi.org/10.1109/MS.2018.290110216

[4] Woubshet Nema Behutiye, Pilar Rodríguez, Markku Oivo, and Ayşe Tosun. 2017. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology* 82 (2 2017), 139–158. https://doi.org/10.1016/j.infsof.2016.10.004

[5] Terese Besker, Antonio Martini, and Jan Bosch. 2016. A Systematic Literature Review and a Unified Model of ATD. In *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016.* Institute of Electrical and Electronics Engineers Inc., 189–197. https://doi.org/10.1109/SEAA.2016.42

[6] Lionel Briand, Domenico Bianculli, Shiva Nejati, Fabrizio Pastore, and Mehrdad Sabetzadeh. 2017. The Case for Context-Driven Software Engineering Research: Generalizability Is Overrated. *IEEE Software* 34, 5 (2017), 72–75. https://doi.org/10.1109/MS.2017.3571562

[7] Ward Cunningham. 1992. The WyCash portfolio management system. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, Vol. Part F1296. 29–30. https://doi.org/10.1145/157710.157715

[8] Victor Da Silva, Helvio Junior, and Guilherme Travassos. 2019. A Taste of the Software Industry Perception of Technical Debt and its Management in Brazil. *Journal of Software Engineering Research and Development* 7 (2019), 11–116. https://doi.org/10.5753jserd.2019.19

[9] Neil A Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L Nord, and Ian Gorton. 2015. 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).* Association for Computing Machinery, New York, NY, USA, 50–60. https://doi.org/10.1145/2786805.2786848

[10] Michael Felderer and Vahid Garousi. 2020. Together We Are Stronger: Evidence-Based Reflections on Industry-Academia Collaboration in Software Testing BT - Software Quality: Quality Intelligence in Software and Systems Engineering, Dietmar Winkler, Stefan Biffl, Daniel Mendez, and Johannes Bergsmann (Eds.). Springer International Publishing, Cham, 3–12. https://doi.org/10.1007/978-3-030-35510-4{_}1

[11] Carlos Fernández-Sánchez, Juan Garbajosa, Agustín Yagüe, and Jennifer Perez. 2017. Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. *Journal of Systems and Software* 124 (2 2017), 22–38. https://doi.org/10.1016/j.jss.2016.10.018

[12] Sávio Freire, Nicolli Rios, Manoel Mendonça, Davide Falessi, Carolyn Seaman, Clemente Izurieta, and Rodrigo O. Spínola. 2020. Actions and impediments for technical debt prevention: Results from a global family of industrial surveys. In *Proceedings of the ACM Symposium on Applied Computing.* Association for Computing Machinery, New York, NY, USA, 1548–1555. https://doi.org/10.1145/3341105.3373912

[13] Omar S Gómez, Natalia Juristo, and Sira Vegas. 2010. Replications types in experimental disciplines. *Empirical Software Engineering and Measurement* 38, 23 (2010), 9772–9782. https://doi.org/10.1021/ma051479o

[14] Yuepu Guo and Carolyn Seaman. 2011. A portfolio approach to technical debt management. In *Proceeding of the 2nd working on Managing technical debt - MTD '11.* ACM Press, New York, New York, USA, 31. https://doi.org/10.1145/1985362.1985370

[15] Johannes Holvitie, Ville Leppänen, and Sami Hyrynsalmi. 2014. Technical debt and the effect of agile software development practices on it - An industry practitioner survey. *Proceedings - 2014 6th IEEE International Workshop on Managing Technical Debt, MTD 2014* (2014), 35–42. https://doi.org/10.1109/MTD.2014.8

[16] Ilya Khomyakov, Zufar Makhmutov, Ruzilya Mirgalimova, and Alberto Sillitti. 2020. An Analysis of Automated Technical Debt Measurement. In *Lecture Notes in Business Information Processing*, Vol. 378 LNBIP. Springer, 250–273. https://doi.org/10.1007/978-3-030-40783-4{_}12

[17] Pawel Klimczyk and Lech Madeyski. 2020. Technical debt aware estimations in software engineering: A systematic mapping study. *E-Informatica Software Engineering Journal* 14, 1 (2020), 61–76. https://doi.org/10.37190/e-Inf200102

[18] Tim Klinger, Peri Tarr, Patrick Wagstrom, and Clay Williams. 2011. An enterprise perspective on technical debt. In *Proceedings - International Conference on Software Engineering.* ACM Press, New York, New York, USA, 35–38. https://doi.org/10.1145/1985362.1985371

[19] P Kruchten, R L Nord, and I Ozkaya. 2012. Technical Debt: From Metaphor to Theory and Practice. *IEEE Software* 29, 6 (11 2012), 18–21. https://doi.org/10.1109/MS.2012.167

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

[21] Erin Lim, Nitin Taksande, and Carolyn Seaman. 2012. A Balancing Act: What Software Practitioners Have to Say about Technical Debt. *IEEE Software* 29, 6 (2012), 22–27. https://doi.org/10.1109/MS.2012.130

[7] https://www.meetup.com/

[22] Antonio Martini, Terese Besker, and Jan Bosch. 2016. The Introduction of Technical Debt Tracking in Large Companies. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 161–168. https://doi.org/10.1109/APSEC.2016.032

[23] Santiago Matalonga and Alberto Villar. 2013. Definiciones y tendencia de deuda técnica : Un mapeo sistemático de la literatura. In *16th Congreso Iberoamericano en Software Engineering*. Montevideo, Uruguay, 33–46.

[24] Boris Pérez, Rodrigo Spínola, and Carolyn Seaman. 2020. What are the Practices used by Software Practitioners on Technical Debt Payment ? Results From an International Family of Surveys. In *International Conference on Technical Debt*. Seoul. https://doi.org/10.1145/3387906.3388632

[25] Nicolli Rios, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology* 102, February (2018), 117–145. https://doi.org/10.1016/j.infsof.2018.05.010

[26] Nicolli Rios, Rodrigo Oliveira Spínola, Manoel Mendonça, and Carolyn Seaman. 2020. The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. *Empirical Software Engineering* (6 2020), 1–72. https://doi.org/10.1007/s10664-020-09832-9

[27] Forrest Shull, Davide Falessi, Carolyn Seaman, Madeline Diep, and Lucas Layman. 2013. Technical debt: Showing the way for better transfer of empirical results. In *Perspectives on the Future of Software Engineering: Essays in Honor of Dieter Rombach*. Vol. 9783642373. Springer-Verlag Berlin Heidelberg, 179–190. https://doi.org/10.1007/978-3-642-37395-4{_}12

[28] Rodrigo O. Spínola, Antonio Vetrò, Nico Zazworka, Carolyn Seaman, and Forrest Shull. 2013. Investigating technical debt folklore: Shedding some light on technical debt opinion. In *2013 4th International Workshop on Managing Technical Debt, MTD 2013 - Proceedings*. IEEE Computer Society, 1–7. https://doi.org/10.1109/MTD.2013.6608671

[29] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (6 2013), 1498–1516. https://doi.org/10.1016/J.JSS.2012.12.052

[30] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Vol. 9783642290. Springer-Verlag Berlin Heidelberg. 1–236 pages. https://doi.org/10.1007/978-3-642-29044-2

[31] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. 2016. How do software development teams manage technical debt? – An empirical study. *Journal of Systems and Software* (2016). https://doi.org/10.1016/j.jss.2016.05.018