Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations

Citation for the original published paper (version of record):
Martini, A., Besker, T., Bosch, J. (2018). Technical Debt tracking: Current state of practice: A survey and multiple case study in 15
large organizations. Science of Computer Programming, 163: 42-61.
http://dx.doi.org/10.1016/j.scico.2018.03.007

Science of Computer Programming 163 (2018) 42–61

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Technical Debt tracking: Current state of practice
A survey and multiple case study in 15 large organizations

Antonio Martini [a,b,*], Terese Besker [c], Jan Bosch [c]

[a] CA Technologies Strategic, Research Team Barcelona, Spain
[b] University of Oslo, Programming and Software Engineering, Oslo, Norway
[c] Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden

ARTICLE INFO

Article history:
Received 5 November 2017
Received in revised form 20 March 2018
Accepted 25 March 2018
Available online 29 March 2018

Keywords:
Technical Debt
Change management
Software process improvement
Survey
Multiple case study

ABSTRACT

Large software companies need to support continuous and fast delivery of customer value both in the short and long term. However, this can be hindered if both the evolution and maintenance of existing systems are hampered by Technical Debt. Although a lot of theoretical work on Technical Debt has been produced recently, its practical management lacks empirical studies. In this paper, we investigate the state of practice in several companies to understand what the cost of managing TD is, what tools are used to track TD, and how a tracking process is introduced in practice. We combined two phases: a survey involving 226 respondents from 15 organizations and an in-depth multiple case study in three organizations including 13 interviews and 79 Technical Debt issues. We selected the organizations where Technical Debt was better tracked in order to distill best practices. We found that the development time dedicated to managing Technical Debt is substantial (an average of 25% of the overall development), but mostly not systematic: only a few participants (26%) use a tool, and only 7.2% methodically track Technical Debt. We found that the most used and effective tools are currently backlogs and static analyzers. By studying the approaches in the companies participating in the case study, we report how companies start tracking Technical Debt and what the initial benefits and challenges are. Finally, we propose a Strategic Adoption Model for the introduction of tracking Technical Debt in software organizations.

1. Introduction

Large software companies need to support continuous and fast delivery of customer value both in the short and long terms. However, this can be hindered if both the evolution and maintenance of the systems are hampered by Technical Debt.

Technical Debt (TD) has been studied recently in the software engineering literature [1–4]. TD is composed of a *debt*, which is a sub-optimal technical solution that leads to short-term benefits as well as to the future payment of *interest*, which is the extra cost due to the presence of TD (for example, slow feature development or low quality) [5]. The *principal* is regarded as the cost of refactoring TD. Although accumulating Technical Debt might prove useful in some cases, in others, the interest might largely surpass the short-term gain, for example, by causing development crises in the long term [6].

* Corresponding author.
E-mail addresses: antonio.martini@ifi.uio.no (A. Martini), besker@chalmers.se (T. Besker), jan.bosch@chalmers.se (J. Bosch).

https://doi.org/10.1016/j.scico.2018.03.007

There are several kinds of TD, such as Architecture TD, Testing TD, Source Code TD, and so on [3], depending on what artifact and what level of abstraction the sub-optimality has occurred. TD can be measured only partially by static analysis tools; the rest of TD needs to be tracked; otherwise it will be invisible, as outlined in a quadrant by Kruchten et al. [7]. In 2011, Guo et al. proposed an initial portfolio approach, with the creation of TD *items* to be tracked and managed, which was empirically studied [8]. Seaman et al. have identified the theoretical importance of TD as a risk-assessment tool in decision-making [9]. However, current literature does not cover a number of aspects related to TD: how teams manage (and track) TD, what tools are used in practice, and how TD management is introduced in large organizations. Finally, current literature lacks any *estimation* of the effort spent by the practitioners for managing TD.

In this paper, we therefore aim at addressing the following RQs:

RQ1: How much of the software development time is estimated to be employed in managing TD?

It is also important to understand how a TD tracking process is *introduced* and *implemented* in large software companies:

RQ2: To what extent are software practitioners familiar with the term Technical Debt?
RQ3: To what extent are software practitioners aware of the TD present in their system?
RQ4: To what extent do software organizations track TD?
RQ5: Is there a difference between individual and collective management of TD?
RQ6: Does the background of the respondents influence the way in which TD is managed?
RQ7: What tools are used to track TD?
RQ8: How do software organizations introduce a TD tracking process?
RQ9: What are the initial benefits and challenges when large organizations start tracking TD?

To shed light on these questions, we have conducted a survey in 15 organizations with 226 participants, and we have carried out a multiple case study in three companies that have started tracking TD: In this context, we have interviewed 13 practitioners responsible for tracking TD and analyzed 79 TD items from a pool of 597 improvements. Our findings include the following contributions:

1. The cost of managing TD in large software organizations is substantial, and it is estimated to be, on average, 25% of the whole development time.
2. We list the tools that are currently used to track TD, and we provide a first assessment of which ones create less management overhead.
3. We report the state of practice related to the introduction of a TD management process in 15 Scandinavian organizations.
4. We report the lessons learned from three companies that have started tracking Technical Debt: their starting process, the perceived benefits, and the challenges.
5. We propose a Strategic Adoption Model for Tracking Technical Debt (SAMTTD), aimed at helping companies assess their Technical Debt management process and make decisions on its improvement. The model also defines the next research challenges to be addressed in theory and to be evaluated in practice.

This paper adds new and more in-depth results to the findings reported in a previous paper [10]. In particular, we address new research questions (RQ2, RQ3, RQ5, RQ6, RQ7), while we add new insights related to the relationship between RQ4 and RQ7 (or else, we study how the practitioners' perception of tracking Technical Debt is related to their usage of tools).

The remainder of the paper reports our methodology in section 2, the results in section 3, and then we discuss the results in section 4, concluding in section 5.

2. Methodology

For the execution of this study, we aimed at combining different sources of data (source triangulation) and different methodologies (methodology triangulation) to obtain reliable results [11]. To fulfill these triangulation strategies, we surveyed 226 participants. The different sources included 15 large organizations and different roles, that is, developers, architects, and managers. To complement such quantitative investigation, we followed up with a qualitative, in-depth multiple *case study* at three of the companies involved in the survey and that have started tracking TD. Here, we conducted *interviews* with 13 employees, and we analyzed *documents* including 79 *TD issues* out of a pool of 597 improvements present at the companies.

2.1. Survey

In this study, we have involved 15 software organizations belonging to eight distinct large software companies. We consider a *large software company* an organization with more than 250 employees. As shown in the descriptive statistics

Table 1
Kinds of Technical Debt recognized in [3,9,10].

Survey entries	Source and literature term
Lack or low quality of testing	Test Debt [3]
Low code quality	Source Code Debt [3]
Lack or low quality of requirement	Requirement Debt [3]
Lack or low quality of documentation	Documentation Debt [3]
Dependency violations	Architecture Debt [3,12]
Complex architectural design	Architecture Debt [3,12]
Too many different patterns and policies	Architecture Debt [3,12]
Dependencies on external resources/software	Architecture Debt [3,12]
Lack of reusability in design	Architecture Debt [3,12]
Uneasy/Tensed social interactions between different stakeholders	Social Debt [3,13]
Lack of adequate environment and infrastructure during development	Infrastructure Debt [3]

in Table 2, 91.6% of the respondents reported working for an organization bigger than 250 employees. The remaining 8.6% were consultants from small/medium organizations working on the same systems and projects developed by the large organizations participating in the survey. The latter can, therefore, be considered as working in the same context as the other 91.6% of the participants.

Seven out of eight companies developed embedded software, while another one developed software for optimization (company D). The companies are anonymized and named A-H, and the sub-organizations are called B1, B2, F1–F4, and G1–G4.

2.1.1. Survey data collection

In the first part of the survey, we asked about the participants' background information:

- Software development experience: "<2 years," "2–5 years," "5–10 years," ">10 years"
- Role: "Product Manager," "Project Manager," "Software Architect," "Developer," "Tester," "Expert," "Other (Specify)"
- Gender
- Education
- Team size
- Organization size
- Size of their current project in MLOC (Millions of Lines of Code)

In the second part of the survey, we asked for and analyzed the data related to the effort *caused by* several Technical Debt challenges. To make sure that the respondents did not misinterpret the question, the challenges were listed as reported in current literature and not as generic "Technical Debt." Table 1 reports the different kinds of TD together with their scientific names and the related academic source. This assured that a better construct validity of our survey was achieved, as we reduced the subjectivity of the respondents interpreting "Technical Debt."

It is important to notice that the details and the results from the questions in the second part of the survey are not included in this paper because the data has been used to cover a different scope and to answer different questions related to Technical Debt in another work [14]. Therefore, the only questions overlapping between the papers are the ones related to the background of the respondents.

In the third part of the survey, we asked the following questions, some of which can be mapped directly to the RQs. Some of the following questions are instead statements. In those cases, we have asked the agreement of the participants to such a proposition.

Q1. "How much of the overall development effort is usually spent on TD management activities?"
Q2. "How familiar are you with the term 'Technical Debt'?"
Q3. "I am aware of how much Technical Debt we have in our system."
Q4. "All team members are aware of the level of Technical Debt in our system."
Q5. "I track (using tools, documentation, etc.) Technical Debt in our system."
Q6. "All team members participate in tracking Technical Debt in our system."
Q7. "I have access to the output of the tracking of the Technical Debt in our system."
Q8. "All team members have access to the output of Technical Debt in our system."
Q9. "If you track Technical Debt in your project, what kind of tool(s) do you use?"

The formulation of Q1 was slightly different, as we did not mention "TD," but we referred to the challenges mentioned in the second part of the survey (see Table 1). However, we use the formulation in Q1 in the rest of the paper for the sake of readability.

After question Q2, the survey included the following definition of Technical Debt:

"*Technical Debt (TD) is constituted of non-optimal code or other artifacts related to software development that give short-term benefits, but cause a long-term extra cost during the software life-cycle*."

This definition has been operationalized based on the explanations and definitions given by Cunningham [1] and McConnell (presentation given at the workshop at ICSE 2013 [15]). We could not include the most recent one from the dedicated Dagstuhl seminar [16] because it was held after the survey was conducted. However, the difference between our definition and the one given in Dagstuhl does not seem very distant, as visible below:

"*In software-intensive systems, technical debt is a design or implementation construct that is expedient in the short term, but sets up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability*."

In our definition, we omitted the second part of the Dagstuhl definition. However, by enumerating the different kinds of TD in the first part of the survey (excluding external qualities from the questionnaire), we can be sure that the second part of the Dagstuhl definition was also covered, although not explicitly mentioned.

This assures that we provided the participants with a good means to understand what Technical Debt meant when we asked about its management. However, we cannot guarantee that the practitioners read and understood the definition.

For question Q1, since we wanted to quantify the amount of effort related to TD faced by the companies, we provided a scale including the following options: "<10%," "10–20%" ... "80–90%," and "I don't know." This question was aimed directly at answering **RQ1**.

For Q2, we provided the answers "Not at all familiar," "Slightly familiar," "Moderately familiar," "Very Familiar," and "Extremely Familiar." The answers were mapped on a 5-grade Likert scale, respectively 0–4. This question aimed directly at answering **RQ2**.

For Q5–Q8, we asked the respondents to report their agreement on a 6-grade Likert scale: "strongly disagree," "disagree," "somewhat disagree," and the symmetric scale for agreement. These statements were aimed at answering **RQ3**, **RQ4**, and **RQ5**. In particular, we wanted to understand if tracking Technical Debt was an individual activity (by asking the same questions for the individual and about the whole team) and if there was a discrepancy between the awareness of the practitioners and their tracking process.

As for question Q9, we asked the participants to report the tools used in a qualitative way (text-box). The input was then post-processed and compiled in the resulting word cloud. This question was used, together with the previous ones, to answer **RQ7**.

2.1.2. Survey data analysis

First, we analyzed the answers from Q1 to understand the magnitude of the estimated effort spent by the respondents on managing TD. We transformed the answers from categorical to numerical: for example, we parsed "<10%" to 5, "10–20%" to 15, and so on. After the calculations, we can reapply the tolerance interval of $+5/-5$, and the various means and so forth would not change. When calculating the means, we did not consider the "I don't know" answers. However, only a small portion of the answers was of this kind (11.5%).

To avoid the bias introduced by different roles answering the questionnaire, we ran a cross-tabulation chi-square test of independence to understand whether the role of the participants affected the answers.

The second step was to apply frequency analysis on questions Q5–Q8. To do so, we transformed the categorical data to a Likert scale (1–6), where "strongly disagree" was mapped to 1 and "strongly agree" to 6. As for Q5, we also reported the grouped answers in three main intervals, "No tracking" {1–2}, "Somewhat tracking" {3–4}, and "Tracking" {5–6}. We used these aggregated intervals only for the last results related to the adoption model SAMTTD.

For some of the results, we used a standard *boxplot*. The boxplot is a comprehensive way to visualize various descriptive statistics altogether at a glance. We used this method when we aimed at showing the difference about the distribution of the data with respect to two specific variables. For example, Fig. 4 shows the comparison, with respect to different companies, of the distribution of the management effort: We can compare the medians (the black lines in the middle), but we can also see different percentiles (where most of the answers were concentrated) and outliers.

In other cases, we compared the different variables using statistical tests. For example, it seemed interesting to compare how much the respondents were aware of TD with respect to how much they were tracking it. To do so, we performed a number of tests for linear correlation using the tool *R*. Most of the numerical variables did not have a strong linear correlation with each other, except the answers for Q5, Q6, Q7, and Q8. This is not surprising because, if TD is not tracked by an individual, it is probably not tracked by the team, and the output will not be visible to the individual or to the others as well. The Pearson tests for linear correlation gave results from 0.72 up to 0.89 with p-value vastly lower than 0.05. This can be considered a good test for the reliability of the answers. Since these variables all strongly correlate, in the remainder of the paper, when studying different variables, we will use only the "tracking" variable without considering whether the output was available or not.

We also wanted to understand whether the results depended on a specific variable. For example, we tested whether developers answered differently from architects or managers. Thus, to answer RQ6, we ran several chi-squared tests of independence between the background variables of the participants and their answers related to questions Q5–Q8. For example, we wanted to know if the familiarity, awareness, tracking, and so on of the respondents would depend on their background, such as by their affiliation with a company, their education, and so on. This analysis was done to answer **RQ6**.

We finally analyzed the qualitative answers from Q9 to understand better the results answering **RQ7**. We selected the answers in which the respondents reported that tools were explicitly used (61/226, 27% of the respondents), and we compared the respective levels of awareness, tracking, and familiarity. This was done to understand better what the respondents meant by "tracking." We also created a word-cloud representation of the qualitative answers for Q10. This, we found, could represent quite well which tools were the most used and in what way. To do so, we processed the qualitative data, removing terms that would appear in the word cloud but would not make sense from the tool point of view, for example, "code" and "Technical Debt." Finally, from the coding of the qualitative answers, we could also identify the frequencies of the tools used. To do so, we manually coded the 61 answers in the following six categories:

- *Comments:* These are usually "TODO" comments, left by the developers in the code or other artifacts. These are useful for the developers to know that something is left to do, but it does not imply a systematic monitoring of the TD reported in the comments.
- *Documentation:* From the qualitative answers, this represents a text or spreadsheet where issues are listed and explained in a semi-systematic format. Another example could be a wiki. However, such documentation is different from a backlog as it is more difficult to monitor, and it does not use a specific technology to manage and perform operations on the backlog.
- *Issues:* using the same ticket system for bug fixing, but usually down-prioritizing the issues related to Technical Debt.
- *Backlog:* This is either a dedicated backlog for TD issues or the usual feature backlog where TD items are mixed with features. This practice usually involves a technology such as project management tools.
- *Static analyzer:* These are tools such as SonarQube, SonarGraph, Klockwork, and so on used to analyze the source code in search for Technical Debt. In a few cases, respondents report that they built their own metrics tools. These tools usually check (language-specific) rules or patterns that can warn the developers of the presence of TD. These tools are used as trackers by the developers, with the limitation that they cover only part of the TD.
- *Lint:* They are also static analyzers but are used more to find potential bugs and security issues rather than technical debt.
- *Test coverage*: Some of the respondents measure test coverage, and they consider a low test coverage as presence of test debt.

2.2. Multiple case study

To understand better to what extent companies tracked TD (**RQ4**) and how the tracking process was introduced (**RQ8–9**), we conducted a multiple case study, investigating some of the companies involved in the survey. We interviewed 13 employees from cases B1 (project manager, system architect, and two developers), F1 (three software architects responsible for TD management in three different teams), and F4 (two system architects, two project managers, and two developers). In particular, to understand what was considered "good tracking," we had the opportunities to interview the participants, belonging to company F1, who answered "strongly agree" (the highest level of tracking) to question Q5. This gave us an idea of what was considered as current best practices for tracking TD. To support the interviews, we also analyzed 79 out of 597 TD backlog items used for tracking improvements (and thus including TD items) in companies B1, F1, and F4.

2.2.1. Interviews

2.2.1.1. Data collection The interview questions were designed to cover taxonomies we found in the pre-study concerning the *reason for initiation*, the *activities* within the TD management process, and the *process implementation*. All interviews were audio-recorded, and the results of the interviews were organized by different questions and activities for later analysis.

We formulated the interview questions in three sections.

- The *first* section contains questions about the profile of the interviewees and their companies.
- The *second* section focused the questions on the initialization of the process for managing TD. "What was the main reason for implementing a TD management process?" "Who decided that the process should be implemented?" "What negative effects did you experience in your system due to TD?" (**RQ8**).
- In the *third* section, we asked about the outcome of the implemented process (**RQ4**) and how the companies experienced the implementation of the process in terms of the most obvious benefits and challenges (**RQ9**).

2.2.1.2. Data analysis The data analysis used an inductive approach based on open coding [17]. We were looking for activities related to the introduction of a TD management process in the company. For this purpose, we followed the points in [18], which is a well-known study on change management in software engineering. The data were coded using a Qualitative Data Analysis (QDA) software tool called Atlas.ti. Such a tool supports keeping track of the links between taxonomies, codes, and quotations. Based on the taxonomies, we developed a coding scheme that contains a corresponding set of codes and sub-codes. Fig. 1 shows an example of our code hierarchy and how the codes were mapped to the taxonomy. The graph is part of the overall data collection model (not completely displayed here for space limitations).

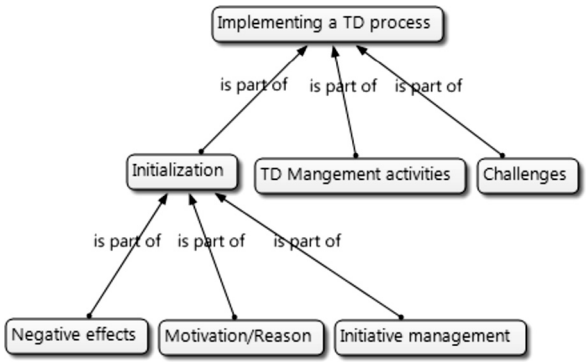

Fig. 1. The coding process.

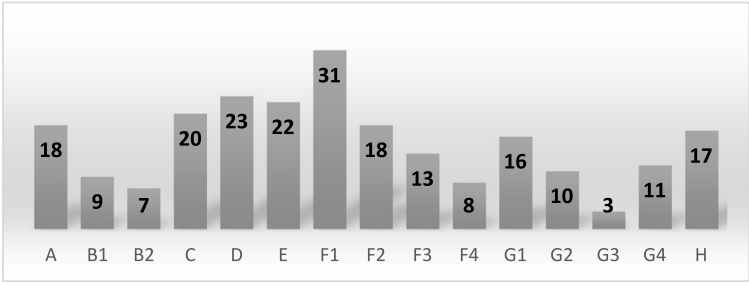

Fig. 2. Number of participants per organization.

As an example of how the coding was conducted, we present a quotation from one of the interviewees which was mapped to the *Motivation* sub-code. *"We realized that for each and every release it took much time correcting or fixing problems with additional patches and it took more and more time adding new features on top of the system*."

2.2.2. Document analysis

To gain more evidence on how the companies were tracking TD (**RQ4**), we investigated the existing documentation. Also, we had access to the TD backlogs of the studied teams: 26 items in the organization B1, 451 items in F1, and 20 items in F4. We analyzed the TD items' fields, values, and how they were ranked. We did not analyze all items in company F1, as 451 items also included improvements that were not TD. We randomly selected 30 items that corresponded to the definition of TD; we analyzed them, and then we tested our assumptions by randomly looking at other items in the backlog. We used the backlogs in the interviews (see previous section) to ask follow-up questions of the participants. Also, we analyzed the documentation that was created by the organizations to explain TD to the users of the tracking process.

3. Results

3.1. Demographics and background of the respondents

In total, we obtained 226 complete answers. The total respondents were 259, which gives us a completion rate of 87%. We aimed at having a similar number of respondents from each organization (Fig. 2). The participants were almost all experienced practitioners, since 156 respondents (69%) had more than 10 years of experience, while only 8 (3.5%) had less than two years of experience (the remaining 62, 27.5%, had between two and 10 years of experience). Several roles participated in the survey: 37 managers (16%), 52 software architects (23%), 105 developers (46%), seven testers (2.65%), 14 experts (5.75%), and nine system engineers (4%) completed the survey.

As shown in Table 2, we can infer the following characteristics of the studied sample:

- *Experience*: Most of the respondents had more than two years of experience, while 69% of them had more than 10 years of experience. The estimations can, therefore, be considered reasonably reliable, as they are made by expert practitioners used to estimating their work (more discussion in the threats to validity section).
- *Education*: Most of the respondents had a bachelor's or master's degree. The level of education is therefore quite high. However, the sample does not include many practitioners involved in research projects.
- *Team size*: Although many of the teams are small (1–10 members), the sample includes a substantial number of respondents working in large teams as well.

Table 2
Background data related to the respondents, with the percentage, the number of respondents, and the relative distribution.

Option	*%*	*Resp.*	*Distribution*
Practitioners' Experience			
< 2 years	3.50%	8	
2 - 5 years	8.80%	20	
5 - 10 years	18.60%	42	
> 10 years	69.00%	156	
Gender			
Female	7.10%	16	
Male	92.00%	208	
Other / Don't want to	0.90%	2	
Education			
No University education	7.10%	16	
Bachelor degree	24.80%	56	
Master degree	58.00%	131	
Ph.D. degree	4.40%	10	
Other:	5.80%	13	
Team Size			
1-5 team members	21.70%	49	
6-10 team members	38.10%	86	
11-20 team members	15.50%	35	
21-40 team members	5.80%	13	
> 40 team members	19.00%	43	
Organization Size			
< 50 employees	1.30%	3	
51-250 employees	7.10%	16	
251-1000 employees	15.00%	34	
1001-5000 employees	14.60%	33	
>5000 employees	61.90%	140	
Age of current system			
< 2 years	10.20%	23	
2-5 years	20.80%	47	
5-10 years	33.20%	75	
10-20 years	29.60%	67	
>20 years	6.20%	14	

- *Organization size*: As mentioned in the analysis made in section 2.1, the organization of the respondents is large. This was chosen by design. We wanted to restrict our results to large organizations. This imposes a limitation on our study: we cannot generalize these results to small organizations.
- *Age of the current system*: The distribution of the different systems is quite even, as the sample covers almost equally all the different phases of the system. This raises the degree of generalizability of our results, as it assures that our data cover both "young" and "old" systems.

3.2. Estimation of management cost of TD (RQ1)

First, we report the answers to Q1 from the survey. In Fig. 3, we show the distribution of the respondents with respect to the different levels of estimated effort that were reported. By picking the middle values, as explained in the methodology section (e.g., 10–20% was transformed into 15), we calculated that the average cost of managing the TD was estimated by 215 respondents to be 25.9% with a median of 25% of the whole development time.

From the results, we can see how most of the respondents answered between 0 and 40%, while half of them are between 10 and 30%. However, some respondents report spending more than 40% of their time managing TD.

Looking at the comparison of medians (bold lines) and percentiles among the companies (boxplot in Fig. 4), we cannot see a big difference in how the respondents answered, apart for the slight difference for E, F1, and F3. This means that the amount of time spent managing TD is quite not dependent on the organization.

A chi-square test of independence, aggregating the intervals over 50% in the same category (the lack of values would have invalidated the chi-square test) yielded a p-value of 0.144, so we could not reject the hypothesis that the role of the

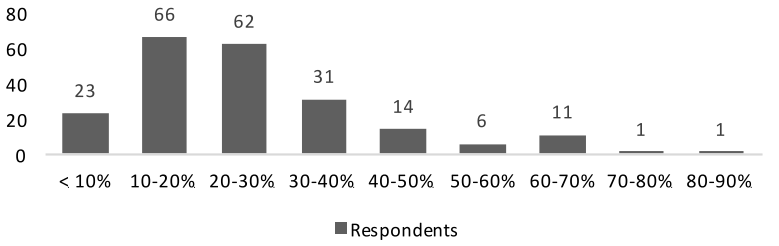

Fig. 3. Distribution of respondents for Q1: "How much of the overall development effort is usually spent on TD management activities?"

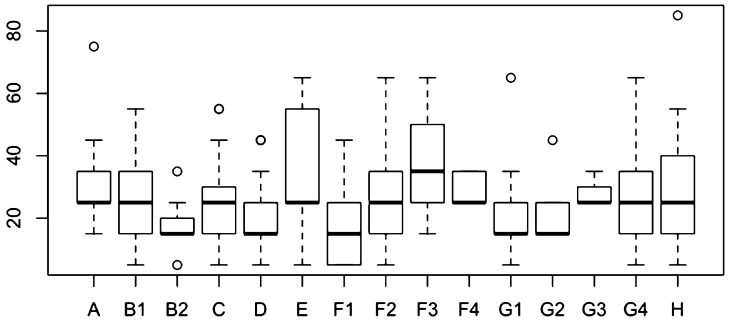

Fig. 4. Comparison of companies with respect to Q1: "How much of the overall development effort is usually spent on TD management activities?"

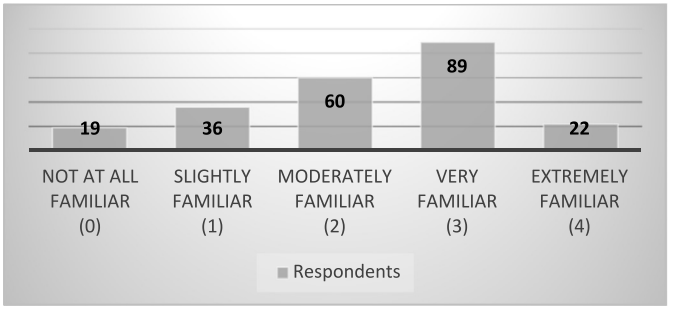

Fig. 5. Distribution of respondents according to their answers to Q2: "How familiar are you with the term "Technical Debt"?"

respondents would influence their answer. This means that the answers did not vary significantly across the roles, contrary to what one might expect, considering different views and experiences of different roles in the organizations.

3.3. Familiarity with the term "Technical Debt" (RQ2)

The respondents seem to be, in total, moderately familiar with the term Technical Debt (Fig. 5). The mean is 2.26, while the median is 2. From the graph below, we can see that there are more respondents who are *very familiar* with respect to the other ones.

From the comparison among the companies, we can see how they are mostly on the same level: F4 is above all the rest, while the organizations B2 and G4 are not very familiar with the TD concept. However, since we did not have access to the practitioners working in these two organizations, we cannot tell what the cause of this lack of familiarity was. We omit the test of independence, as the results are clearly visible in Fig. 6.

3.4. Awareness of Technical Debt present in the system (RQ3 and RQ5)

When assessing the level of awareness of the TD present in their system, the respondents, on average, somewhat agree that they are aware of how much TD they have in their system (mean = 3.69, median = 4). Almost half of them (45%) *somewhat agree*, while only 21% feel more confident (they *agree* or *strongly agree*) and the remaining 32% *disagree* or *somewhat disagree*. Only 3% of the respondents were not aware of TD.

On the other hand, the practitioners seemed less convinced that the whole team would be aware of how much TD is present in the system. Here, the mean is 2.8, while the median is 3, both close to a mild disagreement. The comparison of the answers is reported in Fig. 7. The chi-square test of independence confirmed that the distributions are not dependent, with a p-value $<2.2e{-}16$.

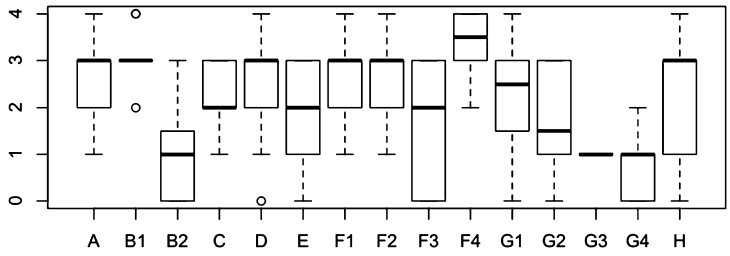

Fig. 6. Level of familiarity with the term Technical Debt for each organization (answering Q2: "How familiar are you with the term 'Technical Debt'?").

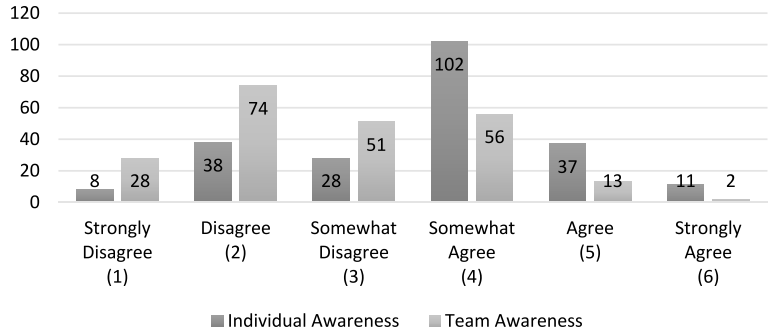

Fig. 7. Comparison of answers for Q3: "I am aware of how much Technical Debt we have in our system" (Individual Awareness) and Q4: "All team members are aware of the level of Technical Debt in our system" (Team Awareness).

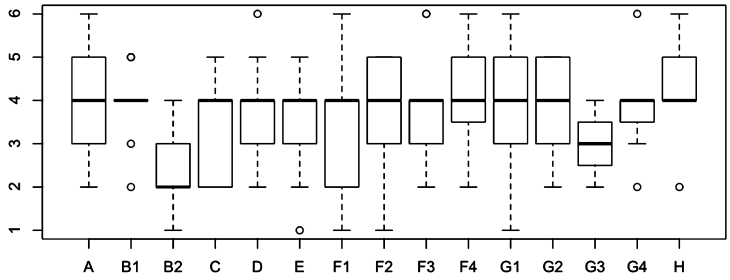

Fig. 8. Distribution of answers with respect to Q3: "I am aware of how much Technical Debt we have in our system." 1–6 correspond to "strongly disagree" to "strongly agree."

For what concerns the different companies, they are quite aligned on the awareness among each other. Once again, B2 seems to have a somewhat lower level of awareness. The results suggest that belonging to one or the other organization would not have an impact on the level of awareness of their employees (Fig. 8).

3.5. Tracking Technical Debt (RQ4)

In this section, we report the results from Q5: "I track (using tools, documentation, etc.) Technical Debt in our system." The average tracking level, reported by 219 respondents, is 2.3 with a median of 2. On the team level, it seemed to be just slightly worse, as shown in Fig. 9 and discussed below.

Based on the results of a chi-square test of independence between the role and the tracking level, we could not reject the hypothesis (p-value 0.63) that the role of the respondents would influence their answer with respect to tracking. In Fig. 10, we show the comparison among different companies. We can see how the different companies answered similarly, apart from company F4 and partly company D. However, the test for independence did not show any significant relationship between the variable company and the answer given in the survey with respect to Q5 (tracking TD).

Finally, there is very little difference between tracking on an individual (Q5) or team level (Q6). Only some of the individuals track TD more than the rest of their team. This is strongly confirmed by a Wilcoxon test, which rejected the null hypothesis (p-value $=$ 2.008e–05) that the difference in the two paired distributions is given by chance. In other words, the same participant answered very similarly when asked Q5 and Q6, and this is not because of randomness, which means that if someone in the team tracks TD, it is very probable that the whole team is involved in the tracking.

Finally, as observed in the methodology section, the results from Q7 and Q8 (related to who in the team has access to the outcome of TD tracking) very strongly correlated with the answers to Q5, so we do not report the exact results here. In other words, this means that the respondents who track TD also have access to its output (e.g., backlogs, dashboards, etc.).

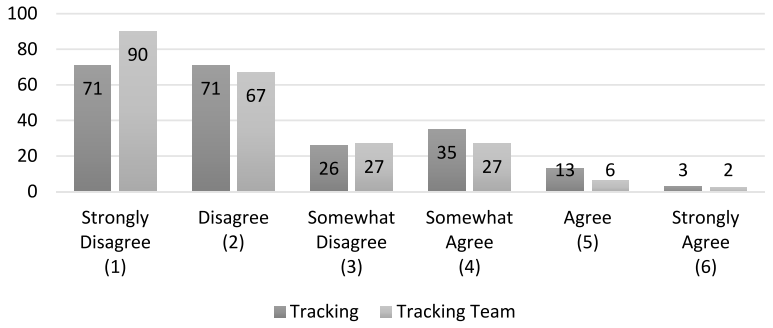

Fig. 9. Distribution of answers related to Q5: "I track (using tools, documentation, etc.) Technical Debt in our system" and Q6: "All team members participate in tracking Technical Debt in our system."

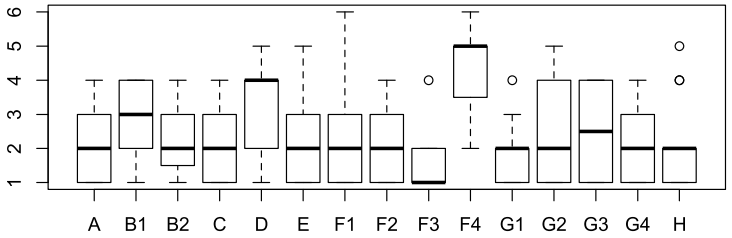

Fig. 10. Distribution of answers for Q5: "I track (using tools, documentation, etc.) Technical Debt in our system."

3.6. Influence of the background of respondents on the management of TD (RQ6)

We have partly answered RQ6 ("Does the background of the respondents influence the way in which TD is managed?") in the previous sections, especially with respect to the variables *roles* and *organizations*. However, we had several other variables in the background section, and we investigated whether any of those variables would help in understanding what causes a more or less mature TD tracking. To answer this question, we ran several statistical chi-squared tests of independence between the background variables (education, team size, etc.) and the variables of interest (familiarity, awareness, and tracking of TD). However, none of the statistical tests yielded a significant answer. Technically, we could not reject any hypotheses for which the answers were dependent on the background of the respondents. Since the results would include several combinations of p-values that would not add any meaning to the manuscript, we decided to omit such a table.

In conclusion, the management of TD depends on some factors that have not been captured by the surveyed background variables. However, in the next sections, we provide some answers that could not be found in the quantitative data but seem to be related to the historical and social context where the participants work. More information is given in sections 3.8 and 3.9.

3.7. Tools used to track Technical Debt (RQ7)

In this section, we analyzed whether the respondents who used some tools to track TD were also more aware of TD or tracked it more than the others. To do so, we considered only the answers from the 61 participants (27%) who answered the qualitative question Q9 (specifying what tools they used). We also report the boxplot for the questions Q5 and Q3: we compared the answers of the participants who used a tool with the ones who did not. Fig. 11 illustrates the results we found: "*Awareness*" is the awareness of the respondents who did not use a tool, while "*Awareness_Tool*" is the one for the ones using a tool (same for "*Tracking*"). It seems that, indeed, if the participants used a tool to track TD, then they would report a high perception of tracking TD. A chi-squared test of independence confirms a strong difference in the distribution of the answers (p-value $<$2.2e–16), strongly confirming this claim. However, more surprisingly, their perception of the level of awareness of how much TD is present in the system would only slightly change. This is confirmed by a chi-squared test of independence (p-value of 0.59), which did not show any difference in the distribution of the answers between the participants using a tool or not. Very similar results were found at the team tracking level, so we do not report them in the boxplot below.

Given the high difference in tracking between the respondents who claimed to use a tool and the ones who did not, we can safely claim that the respondents tracking TD also use a tool. This result confirms that we captured most of the respondents' answers related to the tool that they used. The respondents who did not input an answer for the tool also most probably don't use any tool, since they have in general a much lower level of tracking. Therefore, we can further validate the result that only 26% of the participants used a tool to track TD. This is also important for the reliability of our results related to the SAMTTD model explained in the next sections.

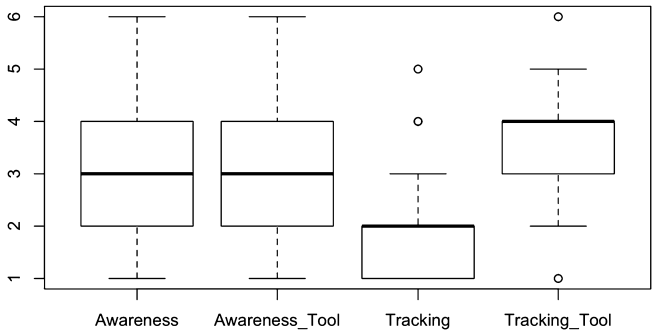

Fig. 11. Distribution of answers for Awareness and Tracking and comparison if a tool is used or not.

Fig. 12. Word cloud of the tools used by the participants to track TD.

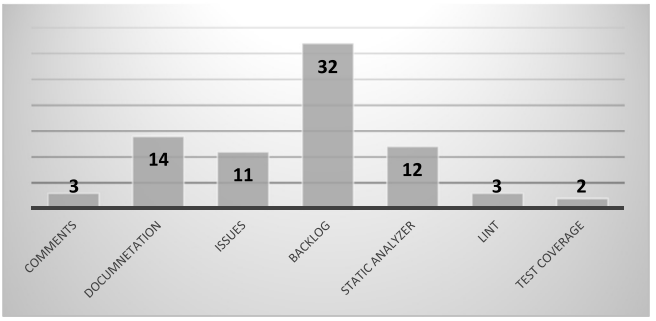

Fig. 13. Number of participants using a specific kind of tool.

From the qualitative data, we could also report what tools were used in practice. After removing some of the words that would just create noise (such as "Technical Debt," see methodology section for more details), we obtained the following word cloud, which shows the distribution of tools used among the respondents (Fig. 12).

By codifying the qualitative answers in *comments, documentation, issues, backlog, static analyzer, lint* and *test coverage*, we can also analyze the frequencies. We can see how the tool that was mostly used is a backlog (dedicated to TD or the same used for feature development), followed by documentation, static analyzers and issue trackers (Fig. 13).

We then analyzed the distributions of the respondents for *Awareness* and *Tracking* levels (Fig. 14) with respect to the different kinds of tools. On the other hand, by analyzing the kind of used tool with respect to the mean amount of effort spent on management activities (Fig. 15), we can see a quite clear difference. Although this difference could not be statistically tested (the chi-square tests did not report significant difference, but this could be due to the small sample), backlog and static analyzers are the ones that seem to create less overhead.

In conclusion, the following considerations on the tools can be made:

- Comments in the code help awareness, but they are not considered tracking, and they are used by just 1% of the respondents. This is probably because they are not used in a document that can be monitored by the team outside the code.

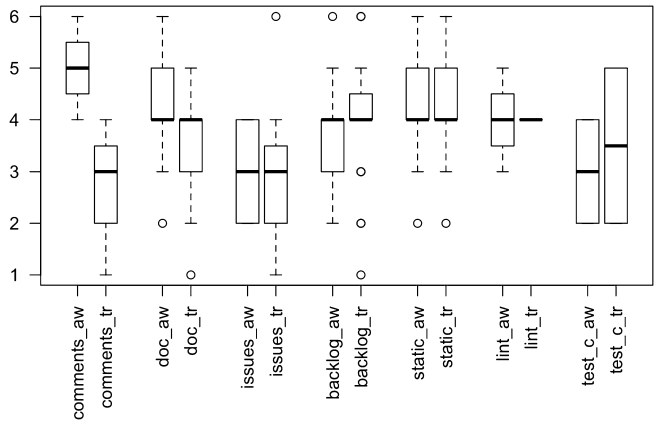

Fig. 14. Distributions of levels of TD tracking ("_tr") and awareness ("_aw") reported to the user of each kind of tool.

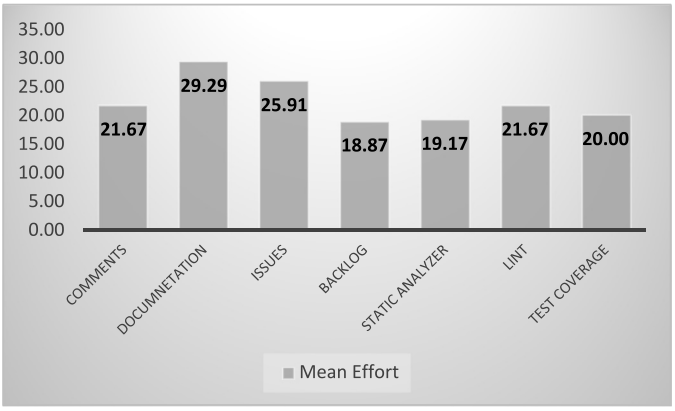

Fig. 15. Mean of management effort for each kind of tool.

- Documentation increases TD awareness, but it is not considered as a high level of tracking, and it has the highest overhead. The main tools used here were Microsoft *Excel* or *Word*. We can infer that this practice is not recommendable in comparison with the other ones.
- Using a bug system for tracking TD is not considered as contributing to a better level of awareness or tracking compared to the other techniques, and it has a slightly higher overhead. We would infer that this is also not the best way of tracking TD.
- Backlogs, static analyzers and "lint" programs all increase the tracking level, but we cannot see a big difference (although static code analyzers seem to contribute better to the participants' awareness). They are also the ones with the least overhead. They therefore seem to be considered the best practices at the moment to track TD.
- Backlogs are the most used tool among the participants. In particular, the most used backlog tools are Jira, Hansoft, and Excel.
- Test coverage does not seem to contribute too much to the awareness and tracking level, although it does not involve much overhead. This might be because test coverage is related to only a small part of TD.

3.8. Why and how do companies start tracking TD? (RQ3)

First, we report *why* the companies decided to start tracking TD, or else their *motivation*. Then, we found that the *preparation* activity was critical to start tracking TD, and we, therefore, report the main steps involved in this practice.

3.8.1. Motivation for start of TD tracking

The main reasons behind the start of tracking TD were related to experiencing the *interest* of TD, or else there were *too many bugs* to fix, *decreased feature development*, *performance issues*:

"*Because we realized that for each and every release it took much time correcting or fixing problem with additional patches and it took more and more time adding new features on top of the system.* [...] *The system became more and more inefficient.*" These statements confirm our previous results [6], as one of the architects also mentioned: "*After some time the TD was increasing and we had a crisis situation.*"

In other words, the main motivation was related to the negative impact experienced by the practitioners, or else the perception of the interest associated to the TD.

3.8.2. Preparation of the tracking process

From the cases investigated, it was clear that adopting a TD tracking process requires some initial activities and time to implement the process. From B1, we understood that they "*Have done this for 1.5 years more or less, switching from reactive to more proactive. It's a better information about the status of the system*." The *preparation* includes the following aspects. Although we used [11] to code these results, we prefer to report them in a way that is more readable in the context of Technical Debt management:

- Initiative—In all the three cases, the tracking process started from an individual initiative. A manager, a system architect, an experienced developer, and so forth. In other words, tracking TD requires a *champion* in the sub-organization who is aware of TD and is willing to promote the adoption of the practices.
- Budget—Tracking TD needs both an initial effort and a continuous effort. Company B1 started with 150 hours, in the beginning, for a development unit (i.e., a sub-organization responsible for a sub-system, which includes a few teams). However, this was "*ok just to start the backlog, but not to go in depth investigation*." The continuous time allocated to tracking TD varied across our cases: it ranged from 10% (company F) to 30% (company A). The cases also show how the continuous allocation of resources to manage TD could be dynamic, and varying according to newly identified items, as suggested for Architectural TD in [19].
- Management involvement—Although the initiative can start from anyone in the organization, tracking TD requires an initial and a continuous investment (budget). This entails the need of involving a manager who understands the importance of TD and who can grant a budget for this activity.
- Benefits—As the previous point entails, there is a need, for the management to understand the benefits of tracking TD given the initial and continuous budget allocation. Such benefits need to be communicated and continuously evaluated to justify such investment.
- Measurement set up—According to company B1, an amount of time is needed to set up measurements (e.g., complexity) and TD identification (static code analyzers). In other companies, such as F1, we found that a developer set up a specific analysis tool to measure complexity and bug density: this activity was supported by a team dedicated to the measurements in the organization.
- Explanation and alignment—The *Champion* for the TD tracking activity needs to communicate well to the teams what TD is and what needs to be reported (to avoid overhead). The interviewees mentioned that they conducted a first workshop for explaining TD and its tracking, and they also produced some documentation. It is also important to have a *validation* workshop in which the teams bring up some TD issues to align their understanding with the main TD concepts such as *Principal* and *Interest*.
- Appointing of a Sub-System TD Responsible (SSTR)—TD tracking needs someone responsible across the organization who can take the initiative to support the tracking process. In all the studied cases, the people responsible for collecting and maintaining a list of TD issues were chosen as experienced developers on a given sub-system. The sub-system TD responsible, however, needs to be supported by the knowledge of the teams when tracking the issues because different practitioners have better and more detailed views of different parts of the system.
- Breaking down and distributing TD items—The SSRT needs to allocate the TD items to the teams according to their competences and their responsibilities with respect to the system. Architecture items were explicitly appointed to an experienced developer to be analyzed and estimated.
- Communication of TD to management—Once the first TD backlog was prepared, it was communicated to a manager connected to the evaluated (sub-)system. This was supposed to show management the risk associated with such a system due to TD.

In summary, quite a few activities are necessary to set up a TD tracking process; this requires the organizations to take the initial decision of allocating some budget to TD tracking.

3.9. What are the benefits and challenges of tracking TD? (RQ4)

3.9.1. Benefits

When we evaluated the tracking process together with the teams, they mentioned several benefits of tracking TD. The backlog gave them a long-term perspective, not only the short-term one given by the feature backlog. The respondents did not think that the TD backlog was hard to maintain. This is supported by the lower management overhead reported in the survey with respect to the other practices.

One of the architects in organization F4 mentioned that, after an important architectural TD item was refactored, "*The evidence was visible in the next release with positive impact when adding new features on top of the one we fixed. Easier to add and no side effect, cleaner architecture*." According to the project manager interviewed in company B1, the initiative was overall successful, but it needed to be continuously supported, to be really effective. "*Yes it was worth it, but it is important to follow it up now and to make sure that parts of the list are done* [*refactored*]."

Another benefit reported by the architect in company B1 was that the initiative and the weekly meeting promoted a discussion with teams working on other systems, for example, when an issue on the interfaces was revealed. The same architect reported that the TD backlog was a great way to receive feedback from the developers, as it made clear what, according to them, was important to refactor.

One of the interviewed SSTRs and a developer mentioned that focusing on TD was important in order to "*zoom out*" from their daily work, and it was an opportunity to check the system with a broader perspective. Finally, all the architects mentioned that, by using the tracking process, they learned issues that were not known before.

3.9.2. Challenges

Although the respondents mentioned several benefits, some issues with the current approaches were also reported. The most important one was the acceptance, from the managers, of the need for refactoring. Even with the list updated, the information about the risk and benefits of performing a refactoring was not always clear to the managers. This meant that, especially for large TD items, it was difficult to receive the needed budget for TD repayment.

One of the major problems in starting to track TD was that the first step needed a substantial amount of effort to collect all the existing items. Although this would be only a one-time effort, in some teams the managers would not concede the necessary budget. A challenge mentioned by all the participants was that the refactoring became more difficult to be prioritized and completely repaid when several items and several teams were involved. It required "*double*" the effort to prioritize the item with different managers (who could disagree on the necessity of refactoring) and the coordination of the refactoring was considering quite risky and as a dangerous overhead. For example, TD issues involving interfaces were more time-consuming to estimate and prioritize, because they required more discussions involving more stakeholders from more teams.

Another challenge in the prioritization activity was the difficulty of prioritizing *among* TD items, especially where an explicit risk/impact value was not calculated. The participants reported that it was generally difficult to show an actual gain from the cost/benefit analysis to the managers, even with a field explicitly represented in the backlog. In general, the intuitive values used for the risk/interest (but usually not including a systematic calculation) were working only sometimes, and more explanations and indicators were required by the managers to accept a costly refactoring.

The respondents mentioned the difficulty of coordinating the different teams in using a standardized process for tracking TD. In some cases, it was difficult to "*make them care*" about reporting TD, while for other teams the TD list was created with enthusiasm.

Finally, the participants mentioned that in some cases the TD backlog itself did not make the TD more convincing for the management to be refactored, but it served for the teams to remember to take care of TD, which would otherwise remain invisible and overlooked.

3.10. Strategic Adoption strategy

As a final result from the combination of the various analyses performed so far, we aggregated the results and combined them with the roadmap related to the current literature on TD. This led to the Strategic Adoption Model for Tracking Technical Debt (SAMTTD, Fig. 16). The first four steps in the model represent the results from the survey on the current state of practice in the companies.

We used the results from Q4 to create the first step: If the respondents were not familiar with the TD concept, they could be on a higher level. Then, we defined three more levels of TD tracking maturity. To discern between the different levels, we mapped practices that we found used or not and that correlated with different levels of tracking (e.g., the usage of a tool). We additionally used the results from the interviews where it was clear what different practices were introduced to track TD.

- *Unaware*: There is no awareness of what Technical Debt is and therefore how to manage it. According to our survey data, only 8.4% of the participants are in this stage. This datum is related to the respondents that answered "Not familiar at all" with the term Technical Debt, as visible in Fig. 5.
- *No tracking*: In this stage, the software engineers are aware of the TD metaphor, and there is a general understanding of the negative effects brought by having TD in the system, but there is no initiative to track TD, which remains invisible. Around 65.6% of the respondents report being on this level, by (strongly) disagreeing about tracking TD. The percent was calculated by counting the total answers minus the answers from Q4, counted previously as the *unaware* respondents, and the ones who use tools, counted in the next levels (26%). Therefore, this yielded $100–8.4–26 = 65.6$.
- *Ad-hoc*: In this stage, the software engineers are aware of what TD is, and some of the individuals have started tracking TD on their own. This makes the TD management process ad-hoc, since, without a dedicated budget, such individuals use what is available, in terms of tools and processes, for other activities. For example, according to the qualitative answers related to Q3, the sprint or product backlog, a common issue tracker or a simple excel spreadsheet can be used for tracking TD. Static analysis tools might be in use but are limited to the individual usage. According to the survey, approximately 26% of the respondents are at least on this stage (61 participants, 26%, were using tools, see section 3.6). However, from these ones, we need to take away around 7% that we place on the next level (see point). In total, we therefore report around 19% of respondents on this level.

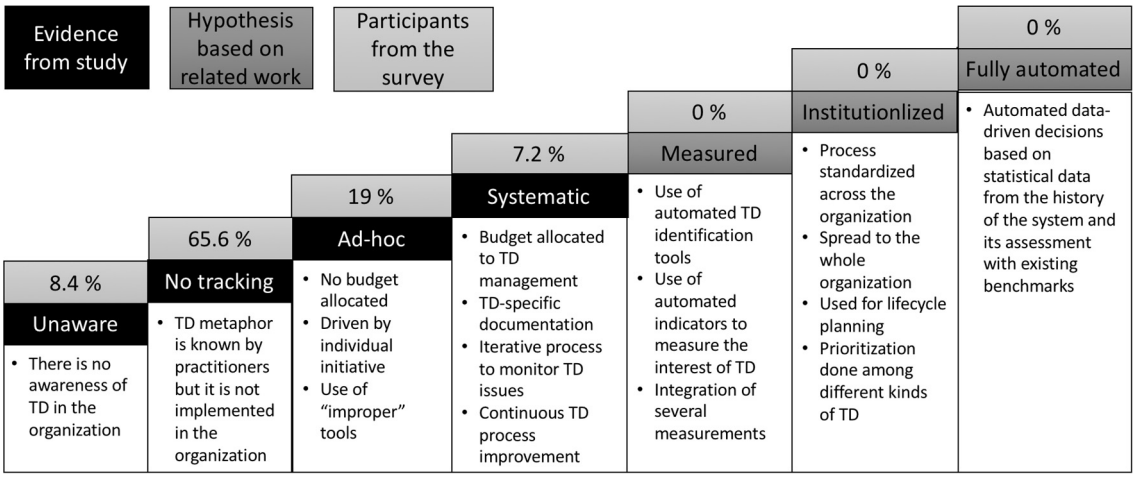

Fig. 16. The Strategic Adoption Model for Tracking Technical Debt: the main milestones and the state of practice (% of respondents per category).

- *Systematic tracking*: The company in this level has acknowledged the importance of tracking TD also on a management level (see *Preparation* section). Therefore, there is a budget generically associated with the management of TD. This amount usually ranges between 10% and 30%. According to the document analysis of the TD items from the case study, a specific backlog and documentation related to TD is necessary, with TD-specific values useful to analyze the principal and the risk/interest. The TD is understood by the participants, who have been instructed by a person responsible for the process (see *Preparation*). There is an iterative process in place to monitor TD (identify, estimate, prioritize, and repay it), and such process is subjected to *continuous improvement*. 7.2% of the respondents are on this stage, actively tracking TD. This is the maximum level achieved by the companies, as confirmed by the interviewees. This amount can be obtained when taking into consideration the respondents who answered "Agree" or "Strongly Agree" to Q5 (see Fig. 9).

We do not have evidence that companies have better processes and tools in place. However, based on current literature on TD [3] and related work on change management [18], we hypothesize future maturity steps that can be reached by the companies when the results of research would be put in place. We identify the following three steps:

- *Measured*: In this stage, identification tools for TD are in place, for example, the use of the tool SonarQube for source code TD (such as McCabe complexity) or, for example, dependency checkers on the architecture level (as reported in company F1). The measurement of the interest is also in place, for example, there are indicators that show the amount of interest paid or predicted if the refactoring is not conducted. Such tools are not employed in practice yet and should be integrated to provide overall indicators to provide help to the stakeholders to estimate and prioritize TD. The authors of this paper are actively working on introducing such tools and indicators, as explained in our recent work [20].
- *Institutionalized*: According to change management [18], a process is mature when it is spread and standardized across the whole organization. This would allow an aligned prioritization of TD across the system. This would also allow the practitioners to plan the allocation of resources according to the quality of the (sub-)systems in order to plan for the life-cycle of the product. As an example, the reader can consider a team who needs to build a feature on a sub-system developed by other teams: knowing how much TD is present in such sub-system would allow the team to estimate whether refactoring is needed or the lead time for the features to reach the customer.
- *Fully automated*: In this stage, the decisions on the refactoring are completely data-driven, making use of statistics collected on historical data or by benchmarking the system against a collection of reference systems. For this purpose, however, the previous steps are necessary.

4. Discussion

The combination of data from 226 participants in 15 large software organizations with the in-depth case study provided an overall picture of the current state of practice with respect to TD tracking. In this section, we discuss the contributions in this manuscript, with respect to practitioners and researchers, we compare our results with existing literature, and we report limitations and threats to validity related to our study.

4.1. Current state of practice of tracking TD and implications for practitioners and researchers

The results related to RQ1 tell us that software companies spend, on average, around 25% of their development time on TD management activities. The boxplots (Fig. 4 and Fig. 10) show some consistency in the companies: The medians range

between 15 and 25% of the effort. The 50-percentile also shows some consistency in the answers, but there we can find some variance as well: Different organizations and individuals dedicate divergent amounts of time to TD management. We could find some differences in the approaches used by the participants, which seems promising. For example, the usage of backlog and code static analyzers appear to be related to less management overhead. However, considering the quantitative analysis, we cannot infer that any background variable related to the respondents would have a significant impact on the overall management overhead.

The results related to RQ2, RQ3, and RQ4 tell us that only a few employees do not know what TD is (around 8%). However, despite the familiarity with TD, more than half of the participants still do not track TD (approximately 65%), and almost one out of five do it in an ad-hoc way (19%), that is, by using tools that are not made for TD tracking and therefore are not effective. Finally, only 7% of the participants tracks TD in a more dedicated way.

An interesting observation is that the results are not significantly affected by the background and the role of the respondents. This datum increases the reliability of the results: Independent of the organization and the background of the participants, we found very similar results across the respondents, which can be considered also more general. In other words, the means and the variance across different practitioners are similar in different organizations.

However, this also led us to consider the following: Different roles with different priorities and views (e.g., managers and developers) agreed on the estimated amount of effort done to keep TD at bay, as well as on the fact that such effort is not systematic (TD is mostly not tracked). Then, an unanswered question is: If TD is so painful, why do organizations not track TD more systematically? One possible answer is that employees do not know how to track TD effectively. This is supported by the fact that most of those who track TD do not use proper tools or documentation, while the few who systematically track TD still do so manually and rarely use basic measurements. For this reason, we found it important to propose the SAMTTD model, to help practitioners understand what it means to track TD and what is necessary to implement a tracking process in practice.

Another answer to the current lack of TD tracking, despite the management effort, might be found in the results related to RQ8 and RQ9 concerning the necessity of a *Preparation* phase and its cost, which is critical for the introduction of a TD tracking process in the companies. At the outset, the initiative needs to be conducted by one or more *champions* in the organization. An initial budget should be allocated to allow the first activities related to the TD inventory, and this entails a need for a commitment by management, which is achieved by communicating how a systematic TD management process would bring benefits to the organization. Unfortunately, this is one of the challenges reported by the practitioners, who claim that there is a lack of good instruments and publicly available results to advocate for the need of systematic TD management. Other activities include the communication and alignment of what should be collected as TD, the set-up of measurement systems, the appointment of a Sub-System TD Responsible (SSTR), and the breakdown and distribution of the TD items to the teams. Unfortunately, the first investment can be burdensome. For example, a trial of 150 initial hours for a unit with three teams was barely enough to identify preliminarily the initial TD list. It also did not leave time for the company to set up measurement systems and accurately estimate and prioritize the TD items, although updating the TD backlog becomes lightweight in the following iterations.

For tools to track TD, we found that many participants use backlogs, implemented in project management tools such as Jira and Hansoft, and static analyzers. The results also suggest that these approaches require less management effort, and they seem to give slightly more awareness of the TD in the system. However, it seems that, for most of the respondents, the awareness of the amount of TD present in the system is not affected by the tool in use, if not slightly. This means that TD tools are not only used by the teams to be aware of the TD, but also for communication, monitoring, and management purposes. The usefulness of these tools is shown by the fact that the participants using backlogs and static analyzers spent less than the average time (18–19% compared to 25.9%) on TD management. However, the tools seem not to help raise the awareness of the respondents: The mean awareness remains between “somewhat disagree” and “somewhat agree.” Many qualitative answers, both from the survey and from the case study, also report the fact that many TD items cannot be automatically revealed because they are too context-specific and they cannot be represented by generic patterns. This leads to the conclusion that better and more specific tools for managing TD need to be developed.

In summary, managing TD requires a few investments that are not well known by the practitioners and are difficult to be motivated by a precise cost/benefits ratio. Consequently, without an investment in processes and tools to track TD, it is difficult to make TD visible, as well as to advocate for refactoring “invisible” TD. This represents a *vicious cycle*: companies suffer the negative effects of TD and try to contain it, but at the same time they do not find enough motivations to invest in a more systematic management process. By looking at the motivations for starting to track TD, the results show that organizations do so when they experience the *interest* of TD: *slow feature development*, *quality issues*, and *performance degradation*. However, at such a point, the interest associated with TD is already high and, as explained in other recent papers—[6], [12]—from the authors of this manuscript, it is hard to refactor, as the cost has also increased and has become too expensive. In conclusion, the only way out the vicious cycle seems to be, for the practitioners, to *proactively start tracking TD*. Using backlogs and static analyzers help reduce the management overhead and increase (even if slightly) the awareness of TD. New tools need to be developed, in two main directions: allowing the developers to communicate the urgency of refactoring TD to the management, and better (semi-)automatic tools to identify and track TD to increase the awareness of the respondents.

4.2. Related work

There are two survey-based studies regarding the familiarity and tool usage related to TD. In [21], the authors concluded that 50% of respondents said that no tools were used and only 16% said that tools gave enough details. Their study also shows that 27% of the respondents do not identify TD. Furthermore, Holvitie et al. [22] show that over 20% of the respondents (in Finland) indicated poor or no TD knowledge. However, in these studies, we cannot find an estimate of the effort spent on TD management, and there is no explanation of how a TD tracking process can be started or implemented. As a comparison with these studies, the results from our survey show that familiarity with TD and its tracking seems to be higher among the respondents who answered our survey. This may be related to the different size, culture, or domain of the organizations, but given that our study is more recent, we could speculate that the familiarity with TD is growing. In our results, only 8.4% of our respondents were not familiar with TD, and 27% of the respondents used tools. Both findings are higher than in the other surveys.

There are a few articles about industrial practices concerning Technical Debt, for example [8,23], and [24], but they are single case studies, and, in two cases, they were performed in small companies. Also, such work does not focus on the current state of practice of Technical Debt tracking, an estimation of the TD management effort, the motivations for starting to track TD, or the maturity evolution of tracking. This makes it difficult to compare the results with our survey, but we will take the topics one by one and discuss similarities and differences. As for the cost of tracking TD, [25] reports detailed results from a single case study. Some results are in line with the broad results reported here including, for example, that the effort might vary greatly, reaching even 70% of the development time, and starting the TD tracking is more expensive in the beginning but it becomes more lightweight when the process is repeated. In [26], the TD management process of several companies is analyzed with reported results similar to our cases, for example, the limited use of measurements and lack of a systematic process. However, in contrast with our work, the study does not focus on TD tracking; it reports a broad snapshot of current practices and does not take change management perspective into account. For example, we report information such as the quantified cost of managing TD, the reasons why organizations start tracking TD, and the preparation activities and costs necessary to track TD. We present a maturity model, SAMTTD, that, taking change management aspects into account, allows for the transfer of knowledge to practice. This is visible in the additional four levels added in our model. We can consider the fourth step in our model as an especially important addition to our work because we found evidence of a systematic process using TD-specific documentation not reported in [26]. Also, none of the cited studies reports quantitative answers from as many as 226 practitioners, which also show trends and statistical results reported here.

There are a few studies regarding Technical Debt tracking and tools in the literature. As for tools, most of the recent findings report tools created by researchers (e.g. [27–29]). The experience reports are usually related to the evaluation of the tool in a specific context and, therefore, cannot be considered as state-of-practice (at least, not yet). This is understandable as new tools are being developed while this manuscript is being written, and the attention to TD by software organizations is quite recent. As for tracking, three initiatives have been reported in the literature [28,30,31]. The first one, [28], presents a tool called DebtFlag, which allows tracking TD and its propagation. However, the evaluation of such a tool in practice has yet to be reported. The second one, [30], reports the evaluation of a tool (AnaConDebt) to assess and track TD. A first study has been done in an industrial environment, but more studies are needed to understand whether the tool is usable in practice. Finally, the last paper, [31], reports a new method to analyze the TD reported in code comments. Although some of the features of the semi-automatic approach seem interesting, it is not clear how many TD items are covered by comments and whether this approach can be used in practice (the paper does not report a practical use of the method with an evaluation from the practitioners). For example, if we look at the survey conducted in this paper, currently only around 1% of the participants (three) state that they track TD using comments.

4.3. Limitations and threats to validity

Here we report the main threats to validity regarding this study, according to [11]: *construct validity, internal validity, external validity*, and *reliability*.

Construct validity is concerned with the investigation device and the validity of the data with respect to the RQs that are investigated. In a survey, this is usually one of the main threats to the validity of the results, as participants might interpret definitions and other terms differently from each other. Although this phenomenon is unavoidable, we took a few approaches to mitigate the consequences. As for the misunderstandings related to the interpretation of what TD is, we have reported, before the questions, short definitions of the issues and management activities that are associated with TD according to the most up-to-date literature. In other words, we did not ask questions on “Technical Debt” directly but, instead, on more concrete issues that are associated with it. In our experience, this should have reduced the possibility that the respondents would consider TD as something else, for example, bugs or missing features (something that might happen in practice, according to our experience). We also provided, in the last part of the survey, a definition operationalized from the various existing formal definitions. We asked a question about whether the practitioners were familiar with TD according to the definition, and they mostly agreed. Although this does not ensure that the practitioners had answered with full knowledge of what Technical Debt is, we believe that the two mitigation strategies together contributed to reducing the threats to construct validity.

There is a threat of construct validity also when mapping the respondents to the levels in the SAMTTD, as we did not ask this question directly to the participants. To mitigate this threat, we used multiple evidences from various quantitative and qualitative answers, and we can reliably say that no company is using integrated measurements of TD, which place the respondents necessarily from level 1 to 4. We have thoroughly assessed the number of respondents for level 3 regarding the usage of a tracking tool. By definition, the respondents in level 1 do not know what TD is, and this datum comes directly from the answers related to their familiarity with TD. Level 4 includes the few practitioners who have confidently reported how they track TD. These practitioners have also been interviewed, which yielded a description of what systematic process they used. Consequently, level 2 contains the remainder of the respondents not included in levels 1, 3, and 4.

As for the results concerning testing hypotheses statistically, it is important to notice that, in most cases, we could not reject the null hypotheses that the results would depend on the background of the respondents (roles, company, etc.). This means that we could not find enough evidence in this dataset to support the rejection of the null hypotheses, but the reader should be warned that we also did not prove the opposite hypotheses. In summary, we cannot claim that the background played a role in the results.

Finally, it is important to report the threats to external validity. We investigated mostly large companies involved in the development of embedded systems and from the Scandinavian area. This entails three possible threats.

- It is possible that, in other domains (e.g., web development), the percent of the companies in the maturity steps would differ. To mitigate this threat, we have included a company developing "pure" optimization software. In this case, we did not find a statistical difference with respect to the other companies. However, more research is needed to understand if there is a difference.
- Companies in other countries, with different contexts and cultural backgrounds, might answer the survey differently or have different ways of managing Technical Debt. However, all the companies investigated in this study employ developers from all over the world and have distributed development. It is therefore likely that the background of the participants in the survey would actually be more heterogeneous than the organizations themselves, who are only Scandinavian.
- Small companies might behave very differently with respect to Technical Debt management.

Therefore, the reader must be aware that there are some limitations to the extent to which we can generalize from these results.

There are also threats to the reliability of the results, or else, the results might be biased depending on an interpretation given by the authors, method, or source of evidence (e.g., if we asked only developers but not managers), as reported below.

- There is a threat in the quantities estimated by the respondents with respect to Q1. We do not know what the given estimations are based on since most of the participants do not explicitly track TD and their time spent on it. However, as the demographic data show, many participants can count several years (more than 10) of software development experience. Estimations are based on experience, and they are referenced to the practitioners' last projects, which limits a possible retrospective bias. Practitioners are used to estimating the amount of work that has been done or that is upcoming, which mitigates the threat that the estimated effort would be very distant from the real one.
- As for the authors' interpretation, we have made sure that, especially for the qualitative data analysis, we have applied observer triangulation: Two or more authors have analyzed the interviews and either separately coded the statements or checked the other authors' codes. Although this does not remove the threat completely, it is the main strategy used when qualitative data analysis is involved in the study.
- Relying only on quantitative data might miss important details that are necessary to understand the results or might show correlations that are not related to any real causality. For example, we could not find reasons from the quantitative background data that would explain the variance in the amount of time that the participants are employing to manage TD. However, we could combine the quantitative results to qualitative answers coming from some of the organizations participating in the survey, which helped explain the factors related to their maturity by analyzing the interviews.
- Finally, there is a threat of reliability of the results, as the percentage of developers participating in the survey was larger than other roles. This means that the results might be skewed by the developers' biases. However, to mitigate this threat, we performed a chi-square test to understand if the distribution of the answers would depend on the roles of the respondents. The test did not support such a hypothesis, meaning that there was not a statistically significant difference between different responding roles (different roles gave similar answers). By having such roles participating in the survey, we could apply a mitigation strategy denoted as source triangulation.

5. Conclusion

According to 226 respondents in 15 software organizations, practitioners estimate spending, on average, a substantial amount of time trying to manage TD (25%), although such an amount is affected by some variance. Software companies in Scandinavia are more familiar with the TD metaphor with respect to previous studies, and they track TD more. The awareness of TD in the system seems to be somewhat known by the developers, independent of which approach is used. Tools such as backlogs (the most popular approach) and static analyzers help reduce the management overhead of approximately

7%. However, only 26% of the respondents use tools to track TD, and only 7.2% of them created a systematic process for doing so. These low numbers are due to a lack of knowledge of what must be implemented, in terms of tools and processes, to introduce a TD tracking approach in the organization, as well as a lack of awareness of what the negative effects of TD are before they occur. Moreover, we studied some approaches and found that an initial investment in *preparing* for the introduction of TD is necessary, which makes starting TD tracking less appealing for managers who need to fund the activities. However, although there are some obstacles to overcome, some of the companies are proactively and strategically implementing a solution to make TD visible, which shows that it is practical to introduce such approaches. To help this process for other practitioners, we propose a Strategic Adoption Model (SAMTTD) based both on the evidence collected across this study in combination with current literature. The model can be used by practitioners to assess their Technical Debt tracking process and to plan the next steps to improve their organization.

Acknowledgements

We thank the Software Center companies and Matthias Tichy for his valuable insights. The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 712949 (TECNIOspring PLUS) and from the Agency for Business Competitiveness of the Government of Catalonia.

References

[1] W. Cunningham, The WyCash portfolio management system, ACM SIGPLAN OOPS Messen. 4 (1992) 29–30.
[2] E. Tom, A. Aurum, R. Vidgen, An exploration of technical debt, J. Syst. Softw. 86 (6) (Jun. 2013) 1498–1516.
[3] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, J. Syst. Softw. 101 (Mar. 2015) 193–220.
[4] N. Brown, et al., Managing technical debt in software-reliant systems, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, 2010, pp. 47–52.
[5] T. Besker, A. Martini, J. Bosch, A systematic literature review and a unified model of Architectural Technical Debt, in: Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA, 2016.
[6] A. Martini, J. Bosch, M. Chaudron, Investigating architectural Technical Debt accumulation and refactoring over time: a multiple-case study, Inf. Softw. Technol. 67 (2005) 237–253.
[7] P. Kruchten, R.L. Nord, I. Ozkaya, Technical Debt: from metaphor to theory and practice, IEEE Softw. 29 (6) (2012) 18–21.
[8] Y. Guo, et al., Tracking technical debt—an exploratory case study, in: 2011 27th IEEE International Conference on Software Maintenance, ICSM, 2011, pp. 528–531.
[9] C. Seaman, et al., Using technical debt data in decision making: potential decision approaches, in: 2012 Third International Workshop on Managing Technical Debt, MTD, 2012, pp. 45–48.
[10] A. Martini, T. Besker, J. Bosch, The introduction of Technical Debt tracking in large companies, in: 2016 23rd Asia–Pacific Software Engineering Conference, APSEC, 2016, pp. 161–168.
[11] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (Dec. 2008) 131–164.
[12] A. Martini, J. Bosch, The danger of architectural Technical Debt: contagious debt and vicious circles, in: 2015 12th Working IEEE/IFIP Conference on Software Architecture, WICSA, 2015, pp. 1–10.
[13] D.A. Tamburri, P. Kruchten, P. Lago, H. van Vliet, What is social debt in software engineering?, in: 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE, 2013, pp. 93–96.
[14] T. Besker, A. Martini, J. Bosch, The pricey bill of Technical Debt—when and by whom will it be paid?, in: Proceeding of ICSME 2017, Shanghai, China, 2017.
[15] S. McConnell, Managing Technical Debt presentation at ICSE 2013. Online available: http://2013.icse-conferences.org/documents/publicity/MTD-WS-McConnell-slides.pdf (Accessed 12 April 2017).
[16] P. Avgeriou, P. Kruchten, I. Ozkaya, C. Seaman, Managing Technical Debt in software engineering (Dagstuhl Seminar 16162), Dagstuhl Rep. 6 (4) (2016) 110–138. Available online: http://drops.dagstuhl.de/opus/volltexte/2016/6693.
[17] U. Flick, An Introduction to Qualitative Research, SAGE, 2009.
[18] D. Moitra, Managing change for software process improvement initiatives: a practical experience-based approach, Softw. Process Improv. Pract. 4 (4) (Dec. 1998) 199–207.
[19] A. Martini, J. Bosch, A Multiple Case Study of Continuous Architecting in Large Agile Companies: Current Gaps and the CAFFEA Framework, 2016, pp. 1–10.
[20] A. Martini, J. Bosch, An empirically developed method to aid decisions on architectural Technical Debt refactoring: AnaConDebt, in: Proceedings of the 38th International Conference on Software Engineering Companion, New York, NY, USA, 2016, pp. 31–40.
[21] N.A. Ernst, S. Bellomo, I. Ozkaya, R.L. Nord, I. Gorton, Measure it? Manage it? Ignore it? Software practitioners and Technical Debt, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, New York, NY, USA, 2015, pp. 50–60.
[22] J. Holvitie, V. Leppänen, S. Hyrynsalmi, Technical Debt and the effect of agile software development practices on it—an industry practitioner survey, in: 2014 Sixth International Workshop on Managing Technical Debt (MTD), 2014, pp. 35–42.
[23] Z. Codabux, B. Williams, Managing Technical Debt: an industrial case study, in: 2013 4th International Workshop on Managing Technical Debt, MTD, 2013, pp. 8–15.
[24] N. Zazworka, R.O. Spínola, A. Vetro', F. Shull, C. Seaman, A case study on effectively identifying Technical Debt, in: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, New York, NY, USA, 2013, pp. 42–47.
[25] Y. Guo, R.O. Spínola, C. Seaman, Exploring the costs of technical debt management—a case study, Empir. Softw. Eng. 21 (1) (Nov. 2014) 159–182.
[26] J. Yli-Huumo, A. Maglyas, K. Smolander, How do software development teams manage technical debt? An empirical study, J. Syst. Softw. 120 (Oct. 2016) 195–218.
[27] R. Kazman, et al., A case study in locating the architectural roots of Technical Debt, in: Proceedings of the 37th International Conference on Software Engineering, vol. 2, Piscataway, NJ, USA, 2015, pp. 179–188.
[28] J. Holvitie, V. Leppänen, DebtFlag: Technical Debt management with a development environment integrated tool, in: Proceedings of the 4th International Workshop on Managing Technical Debt, Piscataway, NJ, USA, 2013, pp. 20–27.

[29] F.A. Fontana, R. Roveda, M. Zanoni, Tool support for evaluating architectural debt of an existing system: an experience report, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, New York, NY, USA, 2016, pp. 1347–1349.
[30] A. Martini, J. Bosch, The magnificent seven: towards a systematic estimation of Technical Debt interest, in: Proceedings of the XP2017 Scientific Workshops, New York, NY, USA, 2017, pp. 1–5.
[31] Q. Huang, E. Shihab, X. Xia, D. Lo, S. Li, Identifying self-admitted technical debt in open source projects using text mining, Empir. Softw. Eng. (May 2017) 1–34.