2019 IEEE/ACM International Conference on Technical Debt (TechDebt)

Technical Debt Triage in Backlog Management

Terese Besker
Computer Science and Engineering,
Software Engineering
Chalmers University of Technology
Göteborg, Sweden
Besker@chalmers.se

Antonio Martini
University of Oslo
Programming and Software
Engineering
Oslo, Norway
antonima@ifi.uio.no

Jan Bosch
Computer Science and Engineering,
Software Engineering
Chalmers University of Technology
Göteborg, Sweden
Jan.Bosch@chalmers.se

***Abstract* — Remediation of technical debt through regular refactoring initiatives is considered vital for the software system's long and healthy life. However, since today's software companies face increasing pressure to deliver customer value continuously, the balance between spending developer time, effort, and resources on implementing new features or spending it on refactoring of technical debt becomes vital. The goal of this study is to explore how the prioritization of technical debt is carried out by practitioners within today's software industry. This study also investigates what factors influence the prioritization process and its related challenges. This paper reports the results of surveying 17 software practitioners, together with follow-up interviews with them. Our results show that there is no uniform way of prioritizing technical debt and that it is commonly done reactively without applying any explicit strategies. Often, technical debt issues are managed and prioritized in a shadow backlog, separate from the official sprint backlog. This study was also able to identify several different challenges related to prioritizing technical debt, such as the lack of quantitative information about the technical debt items and that the refactoring of technical debt issues competes with the implementation of customer requirements.**

Keywords—Technical Debt, Technical Debt Prioritization, Backlog Management

I. INTRODUCTION

Technical debt (TD) is a metaphor introduced by Ward Cunningham [11], where TD is described 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 even impossible [3].

Commonly, software companies need to consider the tradeoffs between the overall quality of the software and the costs of the software development process in terms of required time and resources.

Examples of this tradeoff can be seen in scenarios in which companies deliberately or indeliberately implement sub-optimal solutions to shorten the time-to-market or when resources are limited in practice. However, if TD is left unchecked in the software, TD can lead to large cost overruns, causing high maintenance costs due to internal software quality issues [5] and the inability to add new features [27]; it may even lead to a crisis point where a huge, costly refactoring or a replacement of the whole software needs to be undertaken [19]. This stresses the importance of regularly refactoring initiatives in the software. However, even if the best intention is to refactor TD as soon as possible after it has been identified or implemented, there is a tendency toward postponement of these refactoring tasks since, commonly, there are other important deadlines in the near future, where these refactoring tasks are often down-prioritized in favor of implementing new features [19].

The decision-making about *if* and *when* a TD item should be refactored is commonly performed as a part of the backlog management (BM) process, where each TD item is assessed and prioritized based on several different criteria. Thus, the BM process is important to facilitate the decision-making process to keep the level of TD at bay; it is also important to understand which factors influence these decisions and what challenges are faced during the prioritization of which TD items to refactor [17].

In general, TD has been extensively studied, and there are studies addressing and giving a theoretical recommendation about the prioritization process of TD. However, less attention has been paid to how the prioritization process of TD is carried out in practice in today's software industry. So far, only very few studies [9], [17] have investigated how the prioritization of TD is practically carried out in today's software industry.

To the best of our knowledge, this is the first empirical paper focusing on how the prioritization of TD is practically carried out in today's software industry together with an assessment of which issues are affecting the process in today's software industry.

This study aims to understand how software companies prioritize TD within their product BM process. In particular, we are interested in how the prioritization process of TD is carried out in practice and understanding which related factors influence the process together with potential challenges.

We, therefore, aim at answering the following research questions:

RQ1: How is the prioritization of technical debt carried out, and what factors influence the process?

This research question will explore how the prioritization process is carried out in practice and to what extent gut feelings influenced the decisions and whether they are managed proactively or reactively.

RQ2: What are the challenges of prioritization of technical debt?

978-1-7281-3371-3/19/$31.00 ©2019 IEEE
DOI 10.1109/TechDebt.2019.00010

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

This research question aims at identifying challenges related to the prioritization process of TD.

The remainder of this paper is structured in seven sections, as follows. Section 2 introduces the background related work. Section 3 describes the research methodology. Section 4 presents the research results. Sections 5 and 6 discuss the findings and threats to the validity of the study, respectively. Finally, Section 7 concludes this study.

II. Research Model and Background

Initially, we performed a literature review by searching for research publications addressing the prioritization process of TD. Based on the outcome of this review, we have a formed a conceptual model as presented in Fig.1. In total, five different dimensions were identified which are particularly interesting when portraying how the TD prioritization process is carried by practitioners and what related factors influence the process.

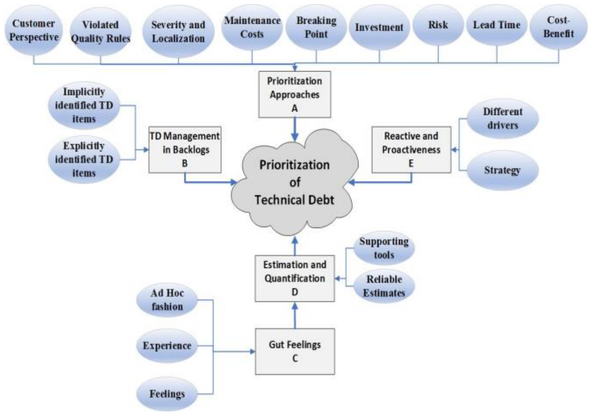
Customer Perspective
Violated Quality Rules
Severity and Localization
Maintenance Costs
Breaking Point
Investment
Risk
Lead Time
Cost-Benefit
Prioritization Approaches A
Implicitly identified TD items
Explicitly identified TD items
TD Management in Backlogs B
Prioritization of Technical Debt
Reactive and Proactiveness E
Different drivers
Strategy
Supporting tools
Reliable Estimates
Estimation and Quantification D
Ad Hoc fashion
Experience
Feelings
Gut Feelings C

Fig. 1. Conceptual Model — Prioritization of Technical Debt

A. *Different approaches to prioritizing TD*

From a theoretical perspective, the prioritization process of TD is relatively well represented by several academic studies. However, there is a current paucity of empirical research focusing on how the prioritization of TD is carried out in practice.

Numerous papers propose different prioritization approaches to assist the decision-making process when prioritizing TD [33], [10], [24], [27], [14], [1], [15].

Several of these publications investigate different aspects of the TD that need to be considered during the process. Frequently, cost-benefit analysis is described related to the prioritization process [27], but also other perspectives are mentioned. For instance, Falessi and Voegele [13] focus on the prioritization of violated quality rules. Other criteria that are proposed when prioritizing TD are, for example, the payment of debt items based on the impact the TD has on the software from a customer perspective and the nature of the debt in terms of its severity and localization [24].

In another study, Chatzigeorgiou et al. [8] adopt a prioritization strategy focusing on the timing of repaying the debt in terms of identifying a breaking point where the interest of the TD items reaches the level of the corresponding principal. Further on, Snipes et al. [28] highlight the importance of developing a cost-benefit analysis that also incorporates the financial aspects of the decision aspects during the prioritization of TD.

Guo and Seaman [2] adopt a different decision-making strategy determining the optimal collection of TD items that should be incurred or held, from the perspective of incurring TD as an investment. Also, the need for taking the business perspective into account when prioritizing TD is discussed in a study conducted by Reboucas et al. [23]. According to another study carried out by Martini and Bosch [19] addressing different types of information needed by project owners and architects during the prioritization activity of architectural TD (ATD) shows that vital aspects, such as lead time, maintenance costs, and risk are important factors to consider during the prioritization process.

The major difference between the above-listed studies and ours is that we focus our study on how the prioritization process of TD is carried out in software companies in practice.

B. *The presence of TD items in backlogs*

Schmid [26] recommends that TD items should be identified in a TD backlog, and a study by Ernst al. [12] surveying 1831 software engineers and architects found that 31% of the respondents in the survey stated that TD was an implicit part of their backlog and 25% stated that TD was an explicit part of their backlog. Unfortunately, it is not clear in that study how the explicitly and the implicitly identified TD items differ. In comparison to that study, our study aims to understand what different types of backlogs the TD items are identified in and how they differ in management and during the prioritizing process. In our recent study [17], we concluded that TD is either documented in a dedicated backlog for TD issues or in a feature backlog where TD items are mixed with features.

C. *The influence of gut feeling*

Both Van [30] and Guo, Spínola, and Seaman [15] state that decisions related to TD are largely based on a manager's gut feeling, rather than hard data gathered through appropriate measurement. This notion is echoed by Carriere, Kazman, and Ozkaya [7] who state that decisions related to cost and scheduling of architectural TD are often done in an ad hoc fashion, based largely on the experience and gut feelings of the architects.

However, to the best of our knowledge, no study addresses the influence of gut feeling on the prioritization process of TD for refactoring.

D. *Estimation of the value of refactoring TD*

Cost and value estimations of refactoring initiatives are essential for managing TD since these estimations assist in planning the work processes and also prevents potential cost and schedule overruns. However, making cost and benefit decisions regarding TD is challenging [7], and several authors highlight the difficulties of obtaining such reliable estimates [7], [18], [21], and there are only very few supporting software tools available to assist such activities.

E. *Reactive- and proactiveness of TD management*

A reactive approach refers to what currently is happening to the present project, whereas a proactive approach focuses mainly on how to improve future projects [30]. When

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

making decisions related to prioritizations of TD refactoring initiatives, these decisions can take a reactive and/or proactive approach.

In general, TD management is considered to be largely reactive since it often must cause significant pain on multiple fronts before it is addressed [12]. Martini, Bosch, and Chaudron [20] state that refactorings are often overlooked in prioritization and they are often triggered by development crises in a reactive fashion.

However, none of these studies examines whether the prioritization of TD is driven by a strategy that is reactive or proactive.

III. Research Methodology

The goal of this study is to understand how software companies prioritize TD within their product BM process. To achieve that, we chose a mixed research methodology by acquiring both quantitative and qualitative data by performing a multiple case-study on four software-developing companies.

The study includes focus interviews with 17 practitioners specializing in different application domains. Additionally, we collected data using a survey (n=17).

A. *Case Selection*

Following the classification provided by Yin [32], this study is an embedded case study in which multiple units of analysis are studied in one case [31]. The sample population was selected using a non-probability sampling technique [31], where the selection of participant companies was obtained using convenience sampling. The selection of participants was conducted by first reaching out to contact persons to whom we had direct access within our network. After describing the study, this person then suggested colleagues with knowledge about the companies' prioritization process who were invited to participate in the study. A brief description of the participating companies and respondents is presented below and summarized in Table I. For confidentiality reasons, the companies have been anonymized in this study.

TABLE I. Companies and Respondents

Company Description			
ID	***Application Domain***	***Company Size***	***Roles***
A	Telecommunications	Large	• Manager Architecture Unit • Program Manager • Product Guardian • Software Architect
B	Automotive	Large	• Solution Lead • Delivery Leader • Solution Architect • IT Architect • Business Analysis
C	Mobility Solutions	Large	• Software Project Manager • Technical Team Leader • Software Architect
D	Packaging solutions and food processing	Large	• Developer • Line Manager • Software Engineer • Software Engineer • Developer

Company A is located in Sweden, and they are developing software using the Scaled Agile Framework (SAFe). Using this approach, their BM approach includes the use of a hierarchical backlog structure in terms of Epics.

The software architects have dedicated board meetings on a weekly basis where (among other things) they prioritize technical debt. This company has worked actively with remediation initiatives of technical debt for several years and has, in general, a high awareness of the negative consequences of technical debt.

The interviewees work both with legacy software with a maintenance focus and with newer software having a strong customer-requirement focus.

Company B is located in Sweden. Their BM process is synced among different development teams within the same department in the same country.

The company adopted an agile software development approach where a BM process is used for planning and prioritizing the development activities within each iteration. This company has an overall backlog which is broken down into several minor team backlogs, which are managed by each team.

The interviewees from this company represent both managers who work with the overall backlog (team A) and developers working with the team backlogs (team B). The items in the backlogs have a direct connection to external customer requirements.

Company C is located in Sweden, but their head office is located in another European country. The overall BM process is synchronized between the different countries. The company uses an agile software development approach where a BM process is used when prioritizing the development tasks.

The interviewees from this company represent two different development teams from two different departments. The first team A develops software which is mainly a part of an internally used platform, and thought does not have an external (outside the company) customer. This teamwork mainly on legacy software, where maintenance tasks are most common.

The second team B works directly towards an external customer, where the user stories are based on the customer requirements. This team implements continuously new features.

Company D is located in Sweden, but the company has sites in several parts of Europe. The company supplies complete systems for automation services with extensive in-house software development of embedded, real-time software systems. The overall BM process is synchronized between different development teams, located in different countries, and sharing the same sprint backlog. The software development is today strongly driven by requirements from external customers.

B. *Research Design*

As illustrated in Fig.2, this study's research design was divided into five phases. The following sections describe these phases and the related research methods used in each stage.

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

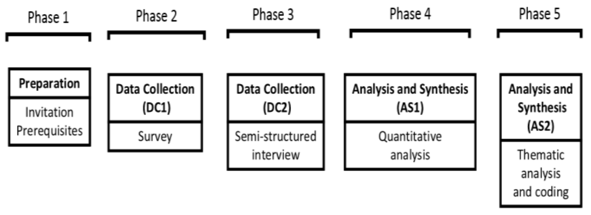

Fig. 2. Visualization of the research model and method used in each phase.

1) Preparation

First, the study was presented and discussed during a workshop with software practitioners from software companies within our network, all having an extensive range of software development. Secondly, an invitation to participate in the study was emailed to the participants in the workshop.

2) Data Collection (DC1 and DC2). The data collection in this study was conducted in two phases.

a) Survey (DC1)

The first data collection phase (DC1) collected data using a survey questionnaire from 17 software practitioners working at four software developing companies. All the practitioners were directly involved in the process of prioritizing TD in their companies' backlog. The survey proceeding was semi-structured, where one of the researchers was available and could clarify and explain the objectives of the survey [22].

As illustrated in Table II, the survey included seven statements assessing the prioritization of TD with respect to the earlier stated research questions. The respondents were asked to indicate their agreement for each statement in relation to their current working situation, using a 6-point Likert scale (not at all, to a small extent, to some extent, to a moderate extent, to a large extent, and to a very large extent).

TABLE II. SURVEYED STATEMENTS

ID	Statement	RQ
ST1	To what extent do you consider your current BM process also includes the prioritization of technical debt?	1
ST2	To what extent do you consider the "gut feeling" having an influence when making prioritization decisions about technical debt?	1
ST3	To what extent do you think that the technical debt's negative effects could be reduced if you did the prioritization of your backlog differently (we are not saying how, just wondering if you think it would be possible or not)	2
ST4	How difficult do you think it is to do the estimation of the value of a refactoring of a technical debt issue?	2
ST5	To what extent do you consider the way you work today with the TD management to be a reactive approach?	1
ST6	To what extent do you consider the way you work today with the TD management to be a proactive approach?	1
ST7	To what extent do you feel that you can influence the decision process of prioritizing the technical debt?	1 and 2

b) Interviews (DC2)

In the second phase of the data collection (DC2), the survey results were complemented with semi-structured interviews to enrich the collected information and gain a deeper understanding of how TD is prioritized within the BM process.

In total, we conducted focus interviews (as suggested by [25]) with the same 17 practitioners who had earlier participated in the previous data collection phase (DC1). The motivation for selecting focus interviews was based on this methods suitability to uncover factors that influence opinions, behavior, or motivation [16]. The interview questions were developed to cover the same taxonomies as the survey and the identified aspect of our conceptual model as presented in Fig. 1. The interview guide was defined before and used in all interviews. Examples of the interview questions are presented in Table III.

The interviews were conducted between November 2107 and January 2018. The interviews were held either in Swedish or English, and each interview lasted between 90 and 120 minutes. To obtain a more accurate rendition of the interviews, all interviews were digitally recorded and transcribed verbatim. All interviewees were asked for recording permission before starting, and they all agreed to be recorded and to be anonymously quoted for this paper. The interviewer also informed the interviewees that participation was voluntary and that the interviewee could cancel the interview at any time. Before starting the interviews, the interviewer presented the objective of the study and provided related background information about the research area.

TABLE III. INTERVIEW QUESTIONS

Prioritization Aspect	***Examples of Interview Questions***
TD Management in Backlogs	• *Briefly describe how your BM works today. What kind of issues are included? One or more backlogs? Who is doing what and when?* • *The strategy for prioritization which you use today, what is the background of it? Used for a long time? Created by whom? Changed recently (or ever)?* • *Do you have a fixed amount of time in each sprint for refactoring of TD today? Pros/Cons with this? How do you know that is enough time, do you do any follow-up on the spent hours?*
Gut Feelings	• *What extent of influence has the "gut feeling" when making prioritization decisions?*
Estimation and Quantification	• *How do you estimate the value from removing TD? What does this gut feeling consist of? Any examples?*
Reactive and Proactiveness	• *Do you consider the way you work today with the TD management to be a reactive or proactive approach?* • *When you do the prioritization of TD today, how long time in advance do you consider (meaning looking at forthcoming features)?* • *Are your roadmap of future features being affected by present TD?*
Prioritization Approaches	• *When prioritizing TD for refactoring, which different factors do you take into consideration and which ones are the most important?*

3) Analysis and Synthesis (AS1 and AS2). The analysis in this study was conducted in two different phases.

a) Survey (AS1)

The data collected in the surveys during phase 2 (DC1) were analyzed and synthesized quantitatively, i.e., by interpreting the numbers collected from the survey answers. The techniques for analyzing and summarizing the quantitative data include different methods, such as

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

determining measures of central tendency (e.g., median and mode) and measures of variability (e.g., frequencies) [29].

a) Interviews (AS2)

The analysis and the synthesis of the interview data collected in phase 3 (DC2) were analyzed using thematic analysis [6] to identify, analyze, and report patterns and themes within the data.

The thematic analysis was conducted using a six-phase guide. As a first step, the audio-recorded qualitative data collected from interviews in DC2 were transcribed, and we familiarized ourselves with the data through careful reading of the transcripts. The second step involved the construction of initial codes from the data, where we organized the data into distinct groups. In this phase of the analysis, a qualitative data analysis (QDA) software package called Atlas.ti was used. The third phase focused on identifying themes by sorting the codes into potential themes and collating all the relevant coded data extracts within each theme. Each extract of data was assigned to at least one theme and, in many cases, to multiple themes. For example, the citation "*in that* [the backlog], *we also enter internal things like refactorings and other requirements that the customer does not have, but the product needs"* was coded as "*Recorded item in Backlog*" in the theme "Backlog Management." The fourth phase focused on the revised set of candidate themes and the fifth phase focused on identifying the essence of each theme and determining what aspect of the data is captured by each theme. This phase also stressed the importance of not just paraphrasing the content of the data extracts but also identifying what is interesting about them and why. The final phase of the thematic analysis took place when we had a set of fully developed themes; it involved the final analysis and write-up of the publication.

Fig. 3 shows a subset of the outcome of the analysis as they emerged out of the analysis process where the mapping between different code hierarchical levels and the taxonomy are graphically presented. The figure represented a curtailed and reduced part of the data collection model (not completely displayed due to space limitation).

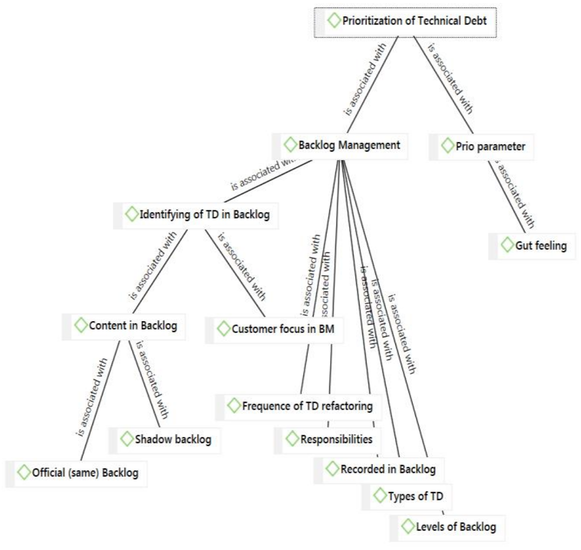

Fig. 3. Sub-set of Coding Scheme

IV. Results and Findings

This section first presents the results from the survey, and then the results from the interviews are presented and organized according to the previously defined research questions.

A. Survey Results

In the survey, the participants were asked to rate seven sets of 6-point Likert Scale statements (not at all, to a small extent, to some extent, to a moderate extent, to a large extent, and to a very large extent) to indicate how they consider their current prioritization of TD process is carried out. The ratings provided by the respondents for each of the survey statement (see Table II) are presented in Fig. 4.

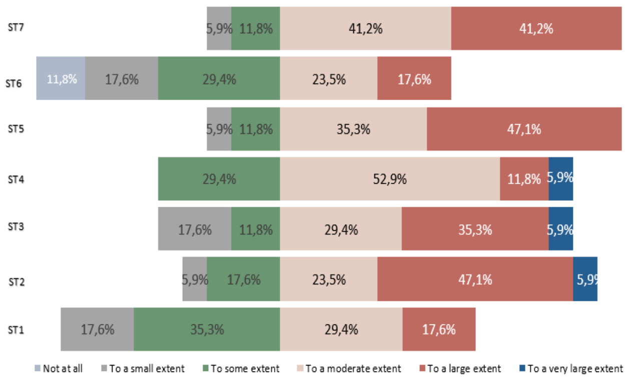

Fig. 4. Summary of the responses to the survey

The first survey result shows (ST1) that all the respondents' BM process included prioritization of TD to some extent, and that three (17.6%) respondents state that their BM process includes the prioritization of TD to a large extent.

According to the survey result (ST2), the gut feeling among the respondents has a profound impact when making decisions related to the prioritization of TD. Thirteen respondents (76.5%) state that gut feeling influences the decision-making of prioritizations related to TD to a large or very large extent.

A substantial part of the survey respondents considers that the amount of TD could be reduced if the prioritization process was carried out differently. According to the result in ST3, all twelve (70.6%) out of seventeen respondents stated that they believed that the negative effects due to TD could be reduced (to a moderate extent, to a large extent, or a very large extent) if they did the prioritization of TD items in their backlog differently.

Survey statement ST4 addresses the extent to which the respondents find the estimation of the prioritization to be difficult. The estimation of the value of a refactoring of a TD issue was considered to be quite difficult. The survey result shows that twelve (70.6%) respondents state that they find the estimation of the value of a refactoring of a TD issue difficult to a moderate, to a large, or a very large extent.

According to the survey result for ST5, the respondents' current TD management approach is mostly reactive, where fourteen (82%) respondents stated that they consider their prioritization process to be reactive to a moderate or a large extent.

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

On the corresponding statement ST6, assessing to what extent the current prioritization process is considered proactive, seven respondents (41.2%) state that their approach is proactive to a moderate or to a large extent.

Finally, the result of statement ST7 reveals that fourteen (82%) of the respondents feel they can influence the decision process of prioritizing TD to a moderate or a large extent.

B. *RQ1: How is the prioritization of TD carried out and what factors influence the process?*

Our analysis has identified many elements that describe how the prioritization process of TD is carried out in practice. We list these elements according to the research model presented in Fig.1.

1) *Backlog management and the representation of TD*

Deciding which activities should be prioritized and which can be postponed or ignored is one of the most frequently discussed questions during the BM process in software organizations.

All investigated companies used a BM approach when planning the software developing activities. Generally, two types of backlogs were used: the product backlog as an ordered list including everything that is known to be needed in the product and the sprint backlog as the set of backlog items selected to be implemented in the next coming sprint. Meanwhile, the content in the product backlog has a focus on product-level functionality described by user stories; the sprint backlog includes a collection of tasks associated with the user stories from the product backlog.

Typically, the sprint backlog includes a mix of different items (e.g., new features, bugs/defects, and improvements) and is labeled accordingly. Only team B at company B categorized their backlog items using the term technical debt. The rest of the companies categorized TDs by labeling them as "Improvements." Nevertheless, a significant amount of the content of these improvements was considered by the interviewees to be TD.

However, among the different companies and teams, the strategy of whether including TD items in the "official" main sprint backlog or putting them into an unofficial "shadow" backlog differed. Only company B (team A) and company C (team A) included improvement items in their ordinary sprint backlog to some extent. In these teams, only the TD items classified as critical and urgent improvements were added to the ordinary sprint backlog, and the rest of the TD items were managed in a shadow backlog. The usage of shadow backlogs was also used at companies A and D, where these shadow backlogs were commonly not managed together within the ordinary BM management processes. This shadow backlog was described as "*A shadow outside as one tries to catch up and drive aside*." This shadow backlog could be managed either on a team or team member individual level (or both).

Several of the interviewees described the reason for using shadow backlogs as an individual decision and a way of being able to handle TD even though the company (or the team) did not support an official representation of TD in the backlogs.

2) *The strategy behind the backlog management process*

The result indicates that the strategy of managing refactorings of TD in backlogs includes different aspects regarding both the way the TD items existed in the backlogs and how the time, effort, and resources for the refactoring of them were allocated.

Commonly the teams prioritized and selected improvement items from the shadow backlog and introduced them directly into the sprint backlog when they considered the items to be in need of implementation or refactoring, and the team estimated that they had available time for it. However, in some cases, the main sprint backlog was never populated with TD items since the shadow backlog was managed on the side, totally separated from the management of the main backlog. A participant from company B described how they transferred TD items from the shadow backlog into the ordinary backlog: "*We also try to add* [to the sprint backlog] *internal things like refactoring of technical debt and other requirements that the customer does not have, but as the product needs.*"

Interestingly, the strategy of using shadow backlogs was not described or communicated within any of the companies' ordinary BM process, and the background of using the shadow backlogs was not clear to the respondents. *"It is quite interesting* [why we have shadow backlogs] *because we have always talked about that everything should be included in the* [official] *backlog. It is very important. We have spread that message throughout the whole organization."*

Except for company D, the companies' BM strategy explicitly includes allocating time and resources for refactoring activities. However, this is done in different ways. For instance, at Companies B and C (team C2a), they have on several occasions set aside a full or partial sprint only focusing on improvements from the backlog. At company A, they have a BM strategy allocating 15% of the time in each sprint for improvement work where the developers are specifically encouraged to refactor items from their shadow backlogs. However, this strategy only allows the team to use this amount of time, meaning that there is no obligation to spend the time; furthermore, there is no follow-up on time spent. As one interviewee from company A stated, *"This is an old estimation, not sure where it comes from and whether it is enough time."*

Company D does not have an explicit strategy for addressing refactoring activities. Their strategy is based on a clustering approach in which the required refactoring activities related to or needed for a specific prioritized feature or bug are refactored.

3) *TD prioritization aspects*

The companies' shadow backlogs include several TD items which all require refactoring activities. However, these TD items differ in terms of several factors, and these factors need to be considered when prioritizing which item should be selected for refactoring during the next sprint iteration and which refactoring could be postponed.

Commonly, if the TD items were minor in terms of required effort, the team members could prioritize them directly within their teams. Meanwhile, if the TD items were considered to be of a more severe nature, or were estimated to require significant effort, these items were commonly discussed during specific backlog grooming meetings. During these dedicated backlog grooming meetings, none of the companies used a rational and defined prioritization approach to assist the decision-making when prioritizing TD.

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

The companies described that this step in their BM process was strongly influenced by "gut feelings" among the participants making the decisions. The prioritization process was described to be heavily influenced by the different participants' roles and their empowerment to make decisions during these meetings. For example, one interviewee from company A said, "*I think that if we would have the same meeting tomorrow, with the same people, we could end up prioritizing differently, the decision is highly dependent on the discussions that occur in the room, in that specific moment.*"

Another interviewee at company D described the empowerment of the decision-maker's experience. *"In my experience, it's usually the most experienced guy that has the biggest impact* [when prioritizing TD]. *We don't actually need a big consensus among the participants."*

However, even if the prioritization of the TD process was described to a large extent as being influence by meritocracy, several different aspects were mentioned as taken into account when making these decisions, such as the criticality, the urgency, of the TD items where the previous experience and knowledge of the professionals making these decisions had a huge impact on the outcome of the prioritization of TD items.

Our result shows that the decisions based on gut feeling are far from being taken on an ad hoc basis without any firm strategy. It is evident that the different companies focused on different prioritization aspects when doing the prioritization of their TD items. Table IV illustrates the different factors that were described as taken into consideration as a part of the practitioners' gut feelings when prioritizing TD.

TABLE IV. PRIORITIZATION FACTORS – AS PART OF GUT FEELINGS

Current Factors Influencing the TD Prioritization		
Company	***Prioritization Aspect***	***Examples of Argumentation***
A	Risk Assessment	*"We want to fix issues that potentially could make us reach a crisis point. It is also important to focus on the potential risk of actually doing the refactoring."*
B	Product or Business Needs	*"When doing our prioritization, we assess "what is the most important* [TD item to refactor], *either from our* [developmemnt team] *point of view or a business point of view."*
B	Resource Utilization	*"Things gets prioritized because of available resources. We do not want anyone to sit and wait only."*
C	Software Quality	*"I focus a lot on different compromised quality 'ilities,' such as maintainability and flexibility during the prioritization of technical debt."*
C	Financial	*"Management is always interested in ROI. Thus, if you invest money, you should be able to get it back relatively quickly."*
D	Product or Business Needs	*"We do consider future perspectives. About the urgency, the request coming from the surroundings. So, it's really based on experience, on the discussion, on personal feelings."*

Looking at the prioritization aspects in Table IV, it is apparent that there are several different prioritization aspects among the companies, even if these aspects were not explicitly communicated within any of the investigated companies.

4) Pro- and reactiveness

In general, all companies strived toward making proactive decisions during the prioritization process, even if they indicated that, in practice, their decision commonly was taken mostly reactively. Only company B had an overall spoken strategy from upper management stating that all decisions related to the evolution of the software should have a specific proactive focus, where the goal of each decision aimed at surviving in the long term (between 10-20 years) perspective. At the rest of the companies, the prioritization process was described as quite reactive where the TD items were first prioritized when the consequence of them was obvious and caused a direct negative effect. For example, one interviewee shared this, "*Our prioritization decisions are not linked to any future solution at all; you just do what you see, reactively, what you see, here and now.*"

The lack of available information about future product features was commonly expressed as a reason for being more reactive but also other reasons were mentioned by some interviewees: "*If you're proactive, you're just referred to as a cost, but if you're reactive, you'll be seen as a guy that fixes things, so you only get the reward if you're reactive actually"*. Also, the number of stakeholders and users operating the software described having an influence on the reactiveness and the proactiveness: "*Some time ago when we didn't have a lot of users, then I think we were more proactive in what we did. And now when we have more users, we are more reactive in that sense."*

Although most of the interviewees described their prioritization process by not taking the evolution of the software into consideration, a proactive approach of the prioritization process of TD was described as a highly desired approach. One interviewee at company C said, *"You want to be proactive, but you do not often have all the data and information available* [about the future of the software*], so it's often popping up things, and first then you can act."*

C. RQ2: What Are the Challenges of Prioritization TD?

In this section, we explore how software companies' reason about the challenges associated with the prioritization process of TD. In general, the prioritization process of TD was rated high among the challenges related to the overall prioritization process. While analyzing the identified challenges, it became clear that several of them were related and, therefore, we have grouped them accordingly.

1) Predicting the future of the software

One general major concern for the interviewees was that they lacked information about how the software was going to evolve in the future, and thus they found it difficult to prioritize necessary and urgent TD items for refactoring. Yet another challenge related to this area refers to the challenge of motivating the prioritization of TD without a clear picture of the future of the software due to uncertainty. The following quote from company C describes this challenge: "*I'm always saying that, in this project, we may use parts of this software in other projects in the future as well. But then I get questions like "Yes, but it will never happen. Why do you say that? Why should we put this cost now for governance, it will never happen." No, but my gut feeling says we should do it. However, if we had figures saying that there is a 20% chance that we will reuse part of this software in the future, that would have helped us making those prioritization decisions more accurate."*

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

2) Available information about the TD items

The most critical limitation for the prioritization of the TD process was expressed in terms of the lack of available information about the TD items and its related negative consequences, both from a current perspective but also from a future perspective. One interviewee said, "*I would like to integrate both the costs of TD and probability* [of the occurrence of the TD interest payment] *and use more business decisions….Both costs from a current status perspective but also from a future cost perspective"*.

Several interviewees stated that the lack of information was the missing key to understanding the value of refactoring of TD and thereby being able to reduce the negative consequences of TD. Having access to reliable and updated estimates and/or quantifications of the negative effects of the present TD items was described as crucial for being able to improve the process of prioritizing TD. However, none of the companies had access to or were able to produce this kind of information for the identified TD items. Neither did any of the companies use any supporting software tools to assist in the decision-making process of prioritizing TD in their backlog.

3) TD refactoring competition with customer requirements

Our findings indicate that the pressure of delivering customer value and meeting delivery deadlines forces the software teams to down-prioritize TD refactorings continuously in favor of implementing new features rapidly.

Furthermore, our findings also indicate that this pressure depends largely on whether the user of the software is an internal or external customer. If the user is an internal customer, the pressure is much lower than if the user is an external customer. A lower pressure induces TD refactorings to receive more attention and thus be more easily prioritized. The teams at company C exemplify this situation, where the two teams have different types of customers. As one of the team members put it, "*It is the opposite situation for us in our team* [having an internal customer]. *They* [the other team, having an external customer] *work significantly more with a customer focus in their backlog. We have a little more free stuff, and therefore, we can prioritize specifically technical debt issues. We don't suffer from such time pressure, and we can also control our documentation better, and are more flexible, and we ourselves are also able* to influence to a larger extent the decisions that are made*"*.

V. Discussion

Through this study, we have identified several challenges software practitioners face when prioritizing TD within their BM process. We also examined what different BM strategies were used and what factors influence the decision-making process of prioritizing TD.

When TD is present in the software, the only significantly effective way of reducing it is to refactor it. However, to be refactored, the refactoring activities of the identified TD items needs to be prioritized in competition with, for example, implementation of new features. From a software practitioners' point of view, the process of prioritizing TD is surrounded by several ambiguities and difficulties.

Firstly, a starting point of our findings shows that TD items commonly are not present in the main sprint backlog; they are often documented in quite ad-hoc managed shadow backlogs. Even if the TD items sometimes are escalated by transition from the shadow backlog into the main backlog, this transition is not clear in terms of *when* and *if* the TD item should be transferred. This way of administrating TD items could potentially lead to the TD items becoming not fully visible and known during the prioritization process, and thereby, not given sufficient attention. There is also a risk that the professionals prioritizing the work for each sprint are not aware of the present TD items (since they are locally managed in the teams), and therefore, these items will never be considered during the prioritization process.

Secondly, the TD prioritization decision-making process has earlier been studied by other researchers, but until now, not from a concrete and practical perspective. Our findings show that when taking these prioritizations of TD decisions, the practitioners rarely base their decisions on pre-defined procedures or guidelines. Rarely do they conduct any cost-benefit analysis or calculate/estimate TD as an investment or calculate/estimate the current or future interest rate in terms of maintenance costs. Neither do they carry out any severity analysis. A possible explanation for these results may be the lack of adequate information about the TD items to assist such activities and a lack of delegated responsibility for providing such information coupled with upper management not focusing strategically and explicitly on remediation of TD in the software. Quite the contrary, our findings show that the decisions related to prioritization of TD are heavily influenced by gut feelings and on meritocracy where the experience of the decision-makers have a huge impact on the outcome.

Further, our findings show that the prioritization process is also highly dependent on individual practitioners' power to justify and argue for a certain item in the backlog to be prioritized. This leads to polarization of the TD prioritization process because it is often perceived as selecting one person's opinion over another's. Further, our result indicates that decisions related to prioritization of TD often are not backed up with validation. This result may be explained by the fact that the companies lack sufficient guidelines and supporting frameworks guiding the prioritization process of TD and directing the discussion into specific areas of interest.

Thirdly, our results show that refactoring activities of TD get less attention if the software will serve an external customer, compared to an internal customer since, in these cases, the focus on delivering customer value is being prioritized in favor of refactoring activities of TD. This result may be explained by the fact that management is not fully aware of the magnitude of the consequences having TD in the software can cause, both in terms of, for example, compromised software quality, lowered developer productivity, increased maintenance cost, and project delays [4].

The upper section of the visualization in Fig. 5 shows contributing causes of why the prioritization of TD is down prioritized or neglected and the lower section presents the identified related challenges for the prioritization of TD in backlogs. This figure can assist in developing actions that sustain the enhancement of the TD prioritization process.

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

The figure illustrated that there are several identified causes that potentially have an impact on the TD prioritization process. In the future, we intend to continue with the validation and refinement of the causes by applying them to additional industrial cases and investigating the validity of the findings.

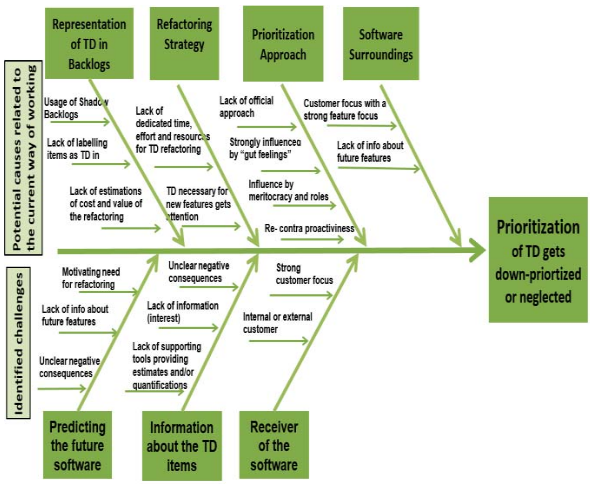
Potential causes related to the current way of working
Identified challenges
Representation of TD in Backlogs
Refactoring Strategy
Prioritization Approach
Software Surroundings
Usage of Shadow Backlogs
Lack of labelling items as TD in
Lack of estimations of cost and value of the refactoring
Lack of dedicated time, effort and resources for TD refactoring
TD necessary for new features gets attention
Lack of official approach
Strongly influenced by "gut feelings"
Influence by meritocracy and roles
Re- contra proactiviness
Customer focus with a strong feature focus
Lack of info about future features
Prioritization of TD gets down-priortized or neglected
Motivating need for refactoring
Lack of info about future features
Unclear negative consequences
Unclear negative consequences
Lack of information (interest)
Lack of supporting tools providing estimates and/or quantifications
Strong customer focus
Internal or external customer
Predicting the future software
Information about the TD items
Receiver of the software

Fig. 5. Identified Causes and Challenges Related the Prioritization Process of TD in Backlogs

A. Implications for practitioners and researchers

When looking at the process of prioritizing TD items during the BM process from a practitioner's perspective, we conclude with some important implications and recommendations for practitioners and researchers, particularly useful during prioritization of the TD in the BM process:

- To give the refactoring of TD activities adequate attention, it is crucial that the professionals prioritizing features and improvements (e.g., TD items) in the backlog are aware of the magnitude of the negative consequences TD brings to the software, both from a short-and a long-term perspective.
- The use of ad hoc managed shadow backlogs leads to the need to define *if* and *when* this TD item should be prioritized and whether shadow backlogs should be transferred to the official sprint backlog. Otherwise, there is a risk of missing or down-prioritizing these TD items.
- It is necessary to agree and communicate a shared vision and purpose of the product and its features to all practitioners who are involved in prioritizing TD (both using a shadow backlog and the official sprint backlog). With such proactive prioritization approach, aligning the product roadmap to specific prioritization tasks will make the decision-making more transparent and understandable.
- A guiding framework or approach in which different prioritization aspects are included can support the grooming process of identifying TD issues that are important to prioritize based on different goals and deliverables. The use of such a framework or approach would also contribute to traceability enhancement.

VI. Study Limitations

It is important to consider the validity of the results in a case study [25]. The findings in this study are subject to at least three different limitations.

First, the limited number of participating companies and the qualitative nature of the investigation means that our findings need to be interpreted cautiously. Although the findings cannot be generalizable to all software companies, they provide good evidence about the identified five dimensions on which the research questions were based, and the findings also point to areas of further research. Furthermore, we plan to expand our sample in the future to reach a higher degree of validation of our results.

Secondly and correspondingly, the goal of collecting quantitative data using a survey was not meant to give precise, measurable results but rather to indicate the way the practitioners perform their prioritization of TD today.

Thirdly, the selection of the companies was based on an available convenience sample among companies within our network that was available and had a specific interest in participating in this study. However, the participating companies represented all software companies from different business areas and with a software development process including regular BM prioritization activities.

Taken together, these are all limitations that can be considered acceptable in light of the exploratory purpose of this study. We preferred to gain a deep and rich understanding of the context of a few cases to build a holistic first theory rather than surveying the topic on a high level only.

VII. Conclusions

This study set out with the aim of assessing how the prioritization of TD is carried out in practice by practitioners in today's software industry. The results of this investigation show that the prioritization of TD in backlogs is commonly done in an unstructured way, where the identified TD items are often managed in "shadow backlogs," potentially resulting in the TD items being overlooked during the prioritization process.

Other findings show that the process regarding which TD item to prioritize is heavily influenced by gut feeling among the decision-makers, and whether the user of the software is an external or internal character also influences the prioritization strategy of TD. Several different challenges for prioritization of TD were revealed in this study, where one of the more significant challenges refers to the difficulty of estimating the value of each potential TD refactoring activity. Further on, even if the software practitioners have a vision of acting proactively when prioritizing TD while the circumstances and the organizational culture force them to act reactively upon the prioritization tasks. Overall, the study indicates that TD needs further attention in the overall BM process, and the findings can also assist in explaining why TD commonly is not sufficiently visible in the official backlogs and thereby not prioritized accordingly.

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.

References

[1] N. S. R. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, “Identification and Management of Technical Debt: A Systematic Mapping Study,” Information and Software Technology, 2015.

[2] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, “The financial aspect of managing technical debt: A systematic literature review,” Information and Software Technology, vol. 64, 2015, pp. 52.

[3] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162),” Dagstuhl Reports, vol. 6, no. 4, 2016, pp. 110-138.

[4] T. Besker, A. Martini, and J. Bosch, "The pricey Bill of Technical Debt - When and by whom will it be paid? ," in IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 2017, pp. 13-23.

[5] T. Besker, A. Martini, and J. Bosch, "Time to Pay Up - Technical Debt from a Software Quality Perspective," in proceedings of the 20th Ibero American Conference on Software Engineering (CibSE) @ ICSE17, Buenos Aires, Argentina, 2017, p. pp. in print. .

[6] V. Braun and V. Clarke, "Using thematic analysis in psychology, Qualitative research in psychology, 3(2)," 2006, pp. 77-101.

[7] J. Carriere, R. Kazman, and I. Ozkaya, "A cost-benefit framework for making architectural decisions in a business context," in 2010 ACM/IEEE 32nd International Conference on Software Engineering, 2010, pp. 149-157.

[8] A. Chatzigeorgiou, A. Ampatzoglou, A. Ampatzoglou, and T. Amanatidis, "Estimating the breaking point for technical debt," in 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), 2015, pp. 53-56.

[9] Z. Codabux and B. Williams, "Managing technical debt: an industrial case study," presented at the Proceedings of the 4th International Workshop on Managing Technical Debt, San Francisco, California, 2013.

[10] Z. Codabux and B. J. Williams, "Technical Debt Prioritization Using Predictive Analytics," in 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 704-706.

[11] W. Cunningham, "The WyCash portfolio management system, in: 7th International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA ’92)," 1992, pp. 29–30.

[12] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? Manage it? Ignore it? software practitioners and technical debt," presented at the Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo, Italy, 2015.

[13] D. Falessi and A. Voegele, "Validating and prioritizing quality rules for managing technical debt: An industrial case study," in Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on, 2015, pp. 41-48.

[14] F. A. Fontana, V. Ferme, M. Zanoni, and R. Roveda, "Towards a prioritization of code debt: A code smell Intensity Index," in 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), 2015, pp. 16-24.

[15] Y. Guo, R. Spínola, and C. Seaman, “Exploring the costs of technical debt management – a case study,” Empirical Software Engineering, 2014/11/30, 2014, pp. 1-24.

[16] R. A. Krueger and M. A. Casey, Focus groups: a practical guide for applied research vol. 4. [updat]. Thousand Oaks, Calif: Sage Publications, 2009.

[17] A. Martini, T. Besker, and J. Bosch, “Technical debt tracking: Current state of practice a survey and multiple case study in 15 large organizations,” Science of Computer Programming, 2018.

[18] A. Martini and J. Bosch, "An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt," presented at the Proceedings of the 38th International Conference on Software Engineering Companion, Austin, Texas, 2016.

[19] A. Martini and J. Bosch, "Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners," in Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on, 2015, pp. 422-429.

[20] A. Martini, J. Bosch, and M. Chaudron, “Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study,” Information and Software Technology, vol. 67, 2015, pp. 237-253.

[21] A. Martini, E. Sikander, and N. Medlani, "Estimating and Quantifying the Benefits of Refactoring to Improve a Component Modularity: A Case Study," in 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2016, pp. 92-99.

[22] S. L. Pfleeger and B. A. Kitchenham, “Principles of survey research: part 1: turning lemons into lemonade,” SIGSOFT Softw. Eng. Notes, vol. 26, no. 6, 2001, pp. 16-18.

[23] R. Reboucas De Almeida, U. Kulesza, C. Treude, D. Cavalcanti Feitosa, and A. H. G. Lima, "Aligning technical debt prioritization with business objectives: A multiple-case study," in Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018, 2018, pp. 655-664.

[24] L. F. Ribeiro, N. S. R. Alves, M. G. d. M. Neto, and R. O. Spínola, "A Strategy Based on Multiple Decision Criteria to Support Technical Debt Management," in 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2017, pp. 334-341.

[25] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empirical Software Engineering, vol. 14, no. 2, 2009, pp. 131-164.

[26] K. Schmid, "A formal approach to technical debt decision making," in Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures, ed. Vancouver, British Columbia, Canada: ACM, 2013, pp. 153-162.

[27] C. Seaman, G. Yuepu, N. Zazworka, F. Shull, C. Izurieta, C. Yuanfang, et al., "Using technical debt data in decision making: Potential decision approaches," in Managing Technical Debt (MTD), 2012 Third International Workshop on, 2012, pp. 45-48.

[28] W. Snipes, B. Robinson, G. Yuepu, and C. Seaman, "Defining the decision factors for managing defects: A technical debt perspective," in Managing Technical Debt (MTD), 2012 Third International Workshop on, 2012, pp. 54-60.

[29] G. M. Sullivan and A. R. Artino, Jr., “Analyzing and interpreting data from likert-type scales,” Journal Of Graduate Medical Education, vol. 5, no. 4, 2013, pp. 541-542.

[30] V. Van, Software Engineering: Principles and Practice: John Wiley & Sons, 2008.

[31] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, Experimentation in software engineering: an introduction: Kluwer Academic Publishers, 2000.

[32] R. K. Yin, Case study research: design and methods vol. 5. London: SAGE, 2014.

[33] N. Zazworka, C. Seaman, and F. Shull, "Prioritizing design debt investment opportunities," presented at the Proceedings of the 2nd Workshop on Managing Technical Debt, Waikiki, Honolulu, HI, USA, 2011.

Authorized licensed use limited to: PUC MG. Downloaded on February 21,2022 at 18:39:23 UTC from IEEE Xplore. Restrictions apply.