university of groningen

faculty of science and engineering

Identifying requirements for technical debt tools

Fadziso Manwa

university of groningen

faculty of science and engineering

University of Groningen

Identifying requirements for technical debt tools

Bachelor's Thesis

To fulfill the requirements for the degree of
Bachelor of Science in Computing Science
at the University of Groningen under the supervision of
dr. Daniel Feitosa (Computing Science, University of Groningen)
and
João Paulo Biazotto (Computing Science, University of Groningen)

Fadziso Manwa (S4213742)

July 24, 2024

Abstract

Technical debt (TD) describes the extra effort associated with software maintenance due to early project shortcuts. TD can be managed with the use of TD management tools. Despite this, TD management tools lack the features and functionalities that practitioners need. An evaluation of software engineering studies has revealed that TD and TD management are addressed on Stack Exchange question-and-answer (Q&A) websites. Therefore, this study aims to determine what practitioners have discussed regarding TD management on the Stack Exchange and if these conversations reveal software requirements that could aid in the advancement of TD management tools. 543 TD-related questions were retrieved and analysed for this study using a dataset that was obtained from three Stack Exchange websites: Stack Overflow, Software Engineering, and Project Management. 67 software requirements, three themes for classifying the requirements, and five approaches to responding to questions containing requirements were found as a result of the question analysis. The results of this study demonstrate that Stack Exchange websites can provide requirements for TD tools, which could facilitate their development.

Contents

List of Figures

List of Tables

1 | Introduction

Ward Cunningham coined the term **technical debt** (TD) as a metaphor to describe the additional work or expense associated with software systems that result from early software project shortcuts [8, 16]. In some cases, allowing technical debt to accumulate can result in short-term benefits, such as reducing the risk of delivering a project outside the approved time-frames and increasing productivity [28]. However, TD that has not been addressed in the long term will have negative results and may become apparent in the form of quality degradation, deceleration of software processes, reduced functionality, and software bugs [10].

Although TD is inevitable, it can be managed to reduce the amount of problems that may arise if it were left to accumulate. TD management techniques, which entail increasing TD awareness, detection & repayment of TD, and the prevention of accumulation of TD, have been developed to combat the long-term effects of TD [27]. During software development, performing these techniques can become tedious and laborious; therefore software have been developed to aid developers in managing TD [27]. These include tools that provide an overview of the internal structure of the code, tools that evaluate software and identify problems by pointing out rule violations, code clone detectors, metric tools, quantification and visualisation tools, and refactoring tools [27].

Even though these tools benefit and support TD management, more automation is needed regarding TD management practices. An example of a practice with little support is the prevention of TD which entails the actions taken to avoid undesired TD [6]. This calls for the extension of existing tools or the development of new tools designed to assist in managing TD [6]. To offer better solutions for automating TD management practices, it is critical to comprehend the requirements that practitioners have. These requirements can be found by combing through discussions practitioners have since these could provide more insight into software engineering topics including TD and TD management.

A source of such discussions is the Stack Exchange network[1]. The software engineering literature has revealed that the Stack Exchange question-and-answer websites can offer practical, insightful points of view on a variety

[1] https://stackexchange.com/

of software engineering topics [2, 4, 24]. Analysing these websites provides insightful information about the issues users encounter and the unresolved concerns surrounding a certain topic [2, 4, 24].

Although there has been research on TD discussions on the Stack Exchange [2, 23, 15, 9], very few explicitly focus on TD tools. Therefore, this study aims to determine what practitioners have discussed regarding TD management and TD tools on the Stack Exchange network. Specifically, we seek to establish whether these conversations reveal software requirements that could aid in the advancement of TD tools by categorising TD-related questions and examining the categories related to TD management for software requirements. The identification of these requirements would provide a foundation for the development of TD tools specialized to automate technical debt management methods.

We will examine the relevant related literature in Chapter 2. Chapter 3 details the research settings and methodological approach. Chapter 4 reveals the results, and Chapter 5 discusses the results, limitations, and recommendations. To summarize, Chapter 6 presents the conclusion, and Chapter 7 presents the acknowledgements.

2 | Related Work

This chapter provides a brief overview of studies that have examined questions from Stack Exchange Q&A websites to gain an understanding of various topics in software engineering (Section 2.1) and TD concepts and management (Section 2.2

2.1 Stack Exchange

In an empirical study, Wan et al. [29] proposed to capture and compare the popularity and impact of discussion topics in Stack Exchange communities. Wang et al. [30] study the factors for fast answers on Stack Exchange Q&A websites. Martins et al. [18] present an empirical study aimed at understanding how test smells and test refactorings are discussed on the Stack Exchange network. Bhatia et al. [5] present a study on the management of bugs on Stack Exchange. Ahmadi et al. [1] targeted Stack Exchange to identify the primary needs of software project managers. Sulír & Regeci [26] analyze the questions and answers on Software Engineering for topics and quantities of the questions, historical trends, and authors' sentiments. Posnett et al. [22] conducted an empirical study on the tenure of posters and quality of answers on Stack Exchange.

2.2 TD discussions in Stack Exchange

There has been research done specifically on discussions about TD that take place on the Stack Exchange network. The research dives into the essence and the possible contributions the discussions may provide for future insights regarding TD phenomenon [15].

In an observational study on the usage of the term "technical debt" on the Stack Exchange Network, Alfayez et al. [2] found 578 TD-related questions, the questions could be categorised into 14 different categories such as TD tools, TD repayment, TD representation, and TD definitions. The study identified 636 unique tags for TD-related questions, with "*SonarQube*" being the most used tag, followed by "*technical debt*'', "*java*", "*agile*", and "*scrum*" among the top-10 most used tags. Furthermore, the findings showed that the most challenging questions for users were those classified under the categories of

TD consequences, TD incurring, and TD tools, and that most of the questions found to be related to TD had to do with TD tools.

A study by Kozandis et al. [15] was conducted to understand how users requested support with respect to the found TD. After reviewing and analysing 415 questions from the Stack Exchange websites, the authors found that architecture, code, and design TD are the most referenced TD types on Stack Overflow. Additionally, they found that all TD types reflect the common length of Stack Overflow questions and that predictive models can accurately detect and classify TD questions and their binary urgency, but not TD types. Moreover, they found that most of the TD questions displayed some degree of urgency and that 29 themes surfaced from the questions with TD repayment and TD management being the most recurrent.

Other research studies delve into more specific sector-oriented questions on TD. One such study by Santos et al. [23] compiled and analysed 79 TD discussions on Agile software development (ASD) from Stack Exchange Project Management. They pointed out 8 types of TD in the context of ASD and identified 51 indicators of ASD-related TD, for instance, poorly written code, design problems, and bug occurrence. They found that the most commonly discussed types of TD are process and people debt and that Product Owner and Development Team are the most important roles concerning ASD-TD. In another article, which researched the scope of TD in security questions found on Stack Overflow, it was found that 38% of the 117 233 questions they had extracted were security-related TD questions [9].

Studies more focused on TD management topics found in the Stack Exchange have also been conducted to bridge the gap between refactoring as a research topic and how it is adapted in practice. An empirical study by Peruma et al. [20] found thatCode Optimization, Architecture and Design Patterns, Unit Testing, Tools and IDEs, and Database are the top-five topics most associated with discussions about refactoring on Stack Overflow. From such topics, Tools and IDEs were the most popular; however, this topic was also realised as the most difficult with the highest number of questions without an accepted answer or any answer at all.

A study by Gama et al. [12] investigated how developers commonly identify TD items in their projects. They found that SO professionals commonly discuss the identification of TD, revealing 29 different low-level indicators to recognise TD items in code, infrastructure, architecture, and tests. They grouped low-level indicators based on their themes, producing a set of 13 high-level indicators such as the presence of bad coding and the lack of good design practices. In addition, they classified all low- and high-level indicators into three different categories (Development Issues, Infrastructure, and Methodology) according to the type of debt each of them was intended to identify.

These research findings present an overarching theme of the various TD issues that practitioners face and how they discuss with each other when look-

ing for assistance. We note that although various studies have looked into TD in Stack Exchange discussions, few explicitly focus on TD tools even though some studies we highlighted reveal that discussions about TD tools are common on the Stack Exchange network. Thus, this study presents a step forward in addressing this shortcoming and advances the state of the art by revealing the practitioner-specified requirements for TD tools.

3 | Methodology

This chapter provides insight into the framework of the study by divulging the selected research questions, data sources, data collection procedures and data analysis approach.

3.1 Research Questions

As stated previously, TD management practices can become laborious, therefore, TD tools have been developed to aid developers in managing TD [27]. Although these tools support TD management and have been shown to bring improvements in TD management when used in training [7], more automation is needed in order to satisfy practitioners needs. The review of the literature reveals that there is little focus placed specifically on TD tool discussions on the Stack Exchange. This study thus aims to find out what has been discussed by practitioners on the Stack Exchange network about TD and if these discussions reveal software requirements that could promote the development of TD tools. Based on our aim, we have established the following two research questions:

- RQ1 - What is discussed about technical debt management in the Stack Exchange network?
- RQ2 - What requirements for TD tools are reported in the Stack Exchange network?

3.2 Research Data

This section of the study focuses on research data, mainly revealing data sources, data collection, and data analysis procedures.

3.2.1 Data Sources

This study's dataset was obtained from three Stack Exchange Q&A websites: Project Management[1] (PM), Software Engineering[2] (SE), and Stack Overflow[3]

[1] https://pm.stackexchange.com/
[2] https://softwareengineering.stackexchange.com/
[3] https://stackoverflow.com/

(SO). These websites were chosen because they contain questions on specific programming problems, software tools commonly used by programmers and software development methods and practices. Questions regarding TD are most likely to be found in these sections of software engineering. Furthermore, previous research on TD discussions in the Stack Exchange, by Alfayez et al. [2] and Kozandis et al. [15] , utilised these three Q&A websites as data sources.

In this section, a brief background is given on the Stack Exchange network and the three Q&A websites: PM, SE & SO. This information was extracted from the meta description of each website.

Stack Exchange

The Stack Exchange network currently consists of 183 question-and-answer websites in various fields, each site focusing on a specific topic [4]. The websites are categorized according to the following classifications: Technology, Culture & recreation, Life & arts, Science, Professional and Business. On each website, a user can ask a question and the question will be answered by other users. Users, questions, and answers are also evaluated through a reputation system[5]. Good answers are voted up and rise to the top. This results in the best answers appearing first so that they are always easy to find. The individual who asked can mark one answer as "accepted". This does not specifically indicate that it is the best answer; it may just indicate that the solution worked for the individual who asked. The Stack Exchange provides a specific question format to avoid questions that are primarily opinion-based or that are likely to generate discussion rather than answers. Questions that do not adhere to the format or need improvement may be closed until they are fixed. All questions are tagged with their subject areas with each having a limit of up to 5 tags since a question might be related to several subjects and comments are also utilized to ask for more information or clarify a question or answer.

Project Management

PM is a Q&A website for project managers and academics. It was established in 2011 and the website covers several topics, including project management frameworks, how to use tools to address problems linked to project management and the profession or practice of project management [2].

Software Engineering

SE is a Q&A website for professionals, academics, and students working within the systems development life cycle. It was established in 2010 and the website is focused on topics relating to software development methods

[4] https://stackexchange.com/about

[5] Programs that let members of online communities grade one another to establish a reputation for trust.

& practices, requirements, architecture & design, quality assurance & testing and configuration management, build, release & deployment.

Stack Overflow

SO is a Q&A website for professional and enthusiast programmers, academics, and students. It was established in 2008 and the website is mainly based on questions that cover specific programming problems, software algorithms and software tools commonly used by programmers. It is the biggest, most well-known and most established Stack Exchange website [2].

3.2.2 Data Collection & Analysis

An illustration of the data collection and analysis stages is depicted in Figure 3.1 and explained in this section.

The data collection and analysis process is made up of five stages. In the first stage, data is downloaded from SO, SE and PM and a set of TD questions is extracted using an automated search process. In the second stage, the TD-related question set is refined through a manual verification process. During the third stage, the set of TD-related questions is manually coded according to a set of established TD question categories. The fourth stage is a manual search for the TD tool requirements on questions from a subset of the TD question categories. In the fifth and final stage an analysis of the requirements is carried out. An in-depth description of the stages is provided below.

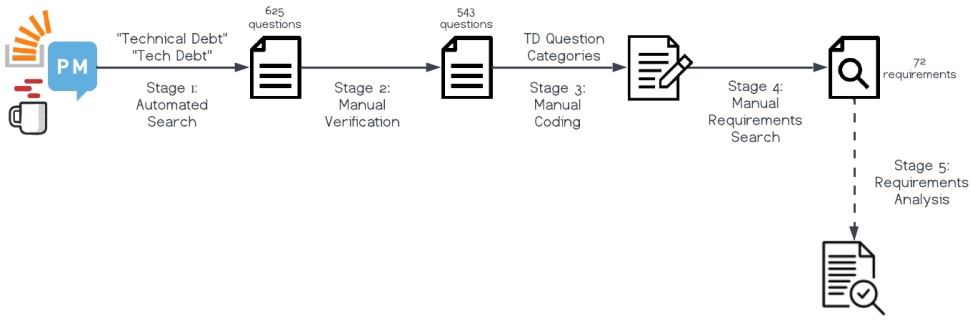

Figure 3.1: Data Collection & Analysis Overview

Stage 1: Automated Search

SO, SE and PM 7z files were downloaded from the Stack Exchange Data Dump[6]. The dump that was used contains data up until 2024-04-02. An existing set of Python scripts provided by Feitosa et al [11] was used for the automated data extraction. Their version of the scripts are supplied with

[6] https://archive.org/details/stackexchange

SO files and search for a given keyword in the tags of each SO question. Following the extraction of the questions, the scripts tie each question to its corresponding set of answers, comments, and post history.

Before the automated search was performed, the scripts were extended to search for a given keyword through the tags, title, and body of a question. The scripts were also extended to look through not only SO but SE and PM as well. The keywords used were "technical debt" and "tech debt". These keywords were chosen because they were highlighted by Alfayez et al. [2] as popular keywords that referenced technical debt in SO.

Stage 2: Manual Verification

The initial set of TD-related questions was manually examined to ensure the relevancy of TD-related questions that are automatically identified. Specifically, the content of each question was manually inspected and evaluated by looking at how the term TD was utilized in the question. If a question was centred around TD or if TD was an important factor in the question, then it was included in our final TD-related question dataset, otherwise, it was excluded.

For instance, consider the following question: *"In real world, business initiatives always take higher priority as there are associated ROIs and deliver something tangible to the users. But there are technical initiatives and projects that need to be done to keep up with the different versions of software, upgrading to a newer platforms, architecture re-factoring etc. How can we plan, prioritize and manage such competing initiatives? Is there a model to quantify technical debt and its impact to the business?"* In this case, the person is concerned about how to quantify TD and its impact on the business, then this questions is relevant for our study.

In contrast, consider the next question:*"I hear a lot of terms which aren't well known amongst programmers (or perhaps the ones I work with at work aren't very good apart from a few), such as "technical debt" (which I studied and even see first hand at work). What other obscure/not-well-known terms are there? This is especially useful to know as interviewers sometimes mention complex terms and if I don't know what they mean, it can screw up the interview as it is in progress."* In this question, practitioners' main goal is discover "*obscure terms*" used by the software engineering community, and TD is just mentioned as an example.

Stage 3: Manual Coding

The final TD-related question set was qualitatively analysed at this stage using open and axial coding. These coding methods are used to derive the TD question category of each question. With open coding, the questions were taken and broken up into labelled discrete parts according to the main TD issue cited in the question. To exhibit the process of open coding an example is given below.

"*I recently started at a new company, with a handful of programmers. Its a medium*

sized company, with around 70 employees, but IT only has 9-10, and there are 3 "programmers" beside myself. However, these guys have very limited experience and are doing a lot of stuff really terribly. For example, one of our projects is a PHP website. The majority of the code is stored in a 20,000 line PHP controller, with 6000 lines of JavaScript embedded in the PHP.

I keep making small suggestions here and there but nobody has been listening, everyone says they are too busy to implement my suggestions. The thing is, they shouldn't be that busy, and wouldn't be if things were done right. They spend most of their time fixing things that keep breaking. If each project was built correctly, I could do it all myself.

What approach should I take to convince these guys or the manager that things need to change, and that changing things will save a bunch of time? Should I skip trying to convince my coworkers and go straight to the manager, with a business-y proposal on how the company will save a bunch of money if they start doing things correct?"

An example of open coding is taking the phrase "*What approach should I take to convince these guys or the manager that things need to change, and that changing things will save a bunch of time?*" and giving it a code that describes the main matter of the question. In this case, the code would be *Convincing management*.

Axial coding [14] was used to map each TD-related question to a category based on one of the 13 TD question types presented by Alfayez et al. [2]. The chosen question types are as follows: TD communication, TD consequences, TD definitions, TD documentation, TD identification, TD incurring, TD management, TD monitoring, TD prevention, TD prioritisation, TD quantification, TD repayment and TD tools. The main method used to achieve the axial coding was to review each question and determine how well the code within it matched a description of a category. The question would be added to the appropriate category if the code corresponded to the description of that category.

Stage 4: Manual Requirements Search

Once the initial round of manual coding was completed, a search for TD tool software requirements was conducted. In this stage, the categories of TD questions that fell under the TD management practices presented by Li et al. [17] were identified. The TD management practices are identification, quantification, documentation, communication, prioritisation, repayment, monitoring, and prevention. The category TD Tools was included because our search focused on the requirements for TD tools, therefore, making it a key category. Once the categories were isolated, the search for potential requirements was conducted. This was done by applying codes on text that reveal what a TD tool could do, the service or services it offers, and the limitations imposed on its operation. It should be noted that some questions may not have explicitly mentioned a TD tool but rather that the activity men-

tioned could be added as a potential requirement for TD tools.

In order to formulate the requirements, a requirements template, specified by Pohl et al. [21], was used. An illustration of the template is shown in Figure 3.2.

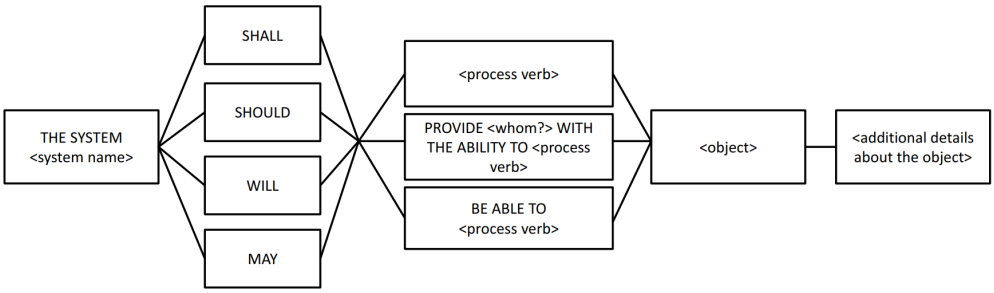

Figure 3.2: Requirements Template [21]

In the template, *the system* is a tool in which the requirement is applied. This is followed by the legal obligation represented by the modal verbs *shall, should, will* and *may*. These are used to make a distinction between requirements that are required by law, requirements that are highly recommended, requirements for the future, and requirements that are desirable, respectively. Next is the core of each requirement characterised by *process verb*. This is the functionality that the requirement specifies. The core is followed by the *object* which can be used to complete the core statement. For example, the information about what is being printed and where it is printed completes the process verb *print*. The *additional details about the object* refer to information about the object that enhances the requirement.

Stage 5: Requirements Analysis

Once formulated, the requirements were analysed for recurring trends that could provide information on how requirements could be classified. Each question with requirements was examined further by going through the answers and comments attached to it. The question was therefore coded according to whether one of the answers or comments provided a TD tool as a solution, a manual solution without any mention of TD tools, or no solution at all.

4 | Results

The results of this exploratory study are shown in this chapter. The set of TD-related questions that was obtained from three Stack Exchange Q&A websites served as the basis for the results. The study's research questions led to the division of the results into two sections, namely the TD management discussion results and the TD tool requirements results.

4.1 Results RQ1: TD Management discussions

The automated search stage resulted in the extraction of 625 TD-related questions and, at the end of the manual verification stage, 543 TD-related questions had been identified. The set of TD-related questions was classified into the 13 categories compiled by Alfayez et al.[2]. The 13 categories and the codes that highlight the key components of each category are described in detail below. An overview of the categories is provided in Table 4.1.

The first category is *TD communication*. Questions under this category are based on conveying information about TD and the long-term negative consequences it has for all parties involved. The codes used in this category are Convincing Management and Explaining TD. Convincing Management is related to persuading the management or a specific client to repay TD such as "*The questions then, is how to you as a developer sell to your manager that these areas need addressed and make a business case to get the time to address them now, rather than having to just incrementally improve here and there?*". Explaining TD refers to explaining the effects of TD to colleagues. An example of this is "*So how can I put this across to him in a way that will sink in how important this issue is?*". The TD communication category had a total of 37 questions. 28 coded with Convincing Management, and 9 with Explaining TD.

The *TD consequences* category highlight the variety of issues that can develop as a result of TD. The codes used for this category were Circumventing TD and Consequences. Circumventing TD highlighted questions intent on ignoring TD instead of addressing it such as "*there is a big technical debt card to look into better ways, but thats for another discussion*". Phrases describing the effects of TD and the increase in TD were coded with Consequences. An example of this is "*The result is that we deliver software that has bugs and the technical debt keeps growing. We use technology that was considered as a recent version*

Category	Codes	Amount per code	Total
Communication	Convincing Management	28	37
	Explaining TD	9	
Consequences	Circumventing TD	37	55
	Consequences	18	
Definitions	Understanding TD	21	21
Documentation	Documenting TD	25	30
	Recording TD	4	
	Representing TD	1	
Identification	Identifying TD	4	4
Incurring	Hacks	19	31
	Shortcuts	4	
	Adding TD	8	
Management	Managing TD	8	8
Monitoring	Tracking TD	5	6
	TD Organisation	1	
Prevention	Avoiding TD	23	28
	Steps To Prevent TD	5	
Prioritisation	Prioritising TD	2	2
Quantification	Measuring TD	2	13
	Quantifying TD	11	
Repayment	Paying Off TD	53	128
	Refactoring	24	
	Solution Enquiry	17	
	Steps To Solve Issues	31	
	Implementing Practices	3	
Tools	Fixing TD Tool Errors	105	186
	Functionalities In Tools	72	
	Tool Recommendations	9	

Table 4.1: TD question categories overview

in 2006 or so.". The TD consequences category had a total of 55 questions. 37 coded with Circumventing TD, and 18 with Consequences.

Questions in the *TD definitions* category were based on understanding the TD topics. The code used for this category was Understanding TD. An example of a question in this category is "*What is the definition of technical debt?*". The TD definitions category had a total of 21 questions.

The questions in the *TD documentation* category focus primarily on how to effectively portray and document TD. The codes used for this category were Documenting TD, Recording TD, and Representing TD. Documenting TD and Recording TD highlighted phrases based on keeping a log of TD such as "*Also how do you write a user story for refactoring code?*". Representing TD was based on ways to show TD. An example of this is "*How to represent technical debt in agile development using azure devops?*". The TD documentation category had a total of 30 questions. 25 coded with Documenting TD, 4 with Recording TD, and 1 with Representing TD.

TD identification includes questions that are concerned about methods that help in revealing TD. The code used for this category was Identifying TD. An example of a question in this category is "*To catch technical debt, I would like the compiler to throw an error if one of the include paths does not exist. Is this possible with either GCC or MSVC?*". The TD identification category had a total of 4 questions.

The questions in the category of *TD incurring* are focused on finding solutions that could lead to deliberate TD. The codes used for this category were Hacks, Shortcuts, and Adding TD. Hacks and Shortcuts highlighted questions with phrases asking for solutions that would solve an issue partially instead of fully addressing it at the expense of gaining TD such as "*My question is: what quick and dirty hacks exist to get this working properly?*". Adding TD was used on questions that described the act of actively adding TD to a project such as "*I am aware using annotation will lead to technical debt for the long run. However, I can't not justify the Java verbosity (although I love it at times when debugging a bug), this code is easier to read.*". The TD incurring category had a total of 31 questions. 4 coded with Adding TD, 19 with Hacks, and 8 with Shortcuts.

Questions in the *TD management* category deal with the actions done in TD management without concentrating on a particular management practice. The code used for this category was Managing TD. An example of a question in this category is "*how do you keep the quality of your product? What practices do you use?*". The TD management category had a total of 8 questions.

The questions in the *TD monitoring* category focus on the best ways to track and maintain TD. The codes used for this category were Tracking TD and TD Organisation. Tracking TD highlighted questions with the intent to observe TD such as "*In your practice, how do you effectively track and manage technical debt?*". TD Organisation was based on how to arrange TD. An example of this is "*What is the recommended practice for adding and organizing technical debt*

for a project?". The TD communication category had a total of 6 questions. 5 coded with Tracking TD, and 1 with TD Organisation.

Questions pertaining to *TD prevention* focus on tactics, or approaches that contribute to the avoidance of accumulation of TD. The codes used for this category were Avoiding TD and Steps To Prevent TD. Avoiding TD highlighted phrases that expressed the desire to avoid TD such as "*However I really don't like using static and it feels poorly implemented and a source of technical debt. What is a more elegant way to do this?"*. Steps To Prevent TD was used on questions asking for specific methods to avoid TD such as "*Which approach above incurs less technical debt in the long term?"*. The TD prevention category had a total of 28 questions. 23 coded with Avoiding TD, and 5 with Steps To Prevent TD.

The questions in the *TD prioritisation* category are about figuring out when a certain TD item reaches its durability limit. The code used for this category was Prioritising TD. An example of a question in this category is "*How to prioritize maintenance work and tech debt with something like User Pain?"*. The TD prioritisation category had a total of 2 questions.

The questions in the *TD quantification* category centre on how to measure TD. The codes used for this category were Measuring TD and Quantifying TD. An example of a question in this category is "*How can I quantify the amount of technical debt that exists in a project?"*. The TD quantification category had a total of 13 questions.

The questions in the *TD repayment* category focus on possible strategies to repay TD. The codes used for this category were Paying Off TD, Refactoring, Solution Enquiry, Steps To Solve Issues and Implementing Practices. Paying Off TD and Refactoring highlighted questions in which the questioner was in the process of paying off TD such as "*But it is really a nightmare to maintain right now and I would like to create a parallel project where I could start a fresh project, create a better structure and from there start migrating the old code to this new environment and refactor it as I go.*". Solution Enquiry and Steps To Solve Issues were based on questions asking for specific solutions to their problems, such as "*I need a solution or any idea for this problem and it can be implemented in the code or in the real live to identify the humans.*". Implementing Practices was based on asking for the best practices to follow in order to pay off TD. An example of this is "*Are there any recommended best practices?"*. The TD repayment category had a total of 128 questions. 53 coded with Paying Off TD, 24 with Refactoring, 17 with Solution Enquiry, 31 with Steps To Solve Issues, and 3 with Implementing Practices.

Questions under the category *TD tools* focus on tools that aid in managing TD. The codes used for this category were Fixing TD Tool Errors, Functionalities In Tools, and Tool Recommendations. Fixing TD Errors highlighted questions that focused on resolving technical errors that occurred during tool usage, such as "*Why I get this warning? is there something misconfigured?"*. Functionalities In Tools was used on questions that were focused on navigat-

ing around features in operational tools. An example of such a question is "*Is there a way of cleaning the Sonar DB for all history?*". Tool Recommendations was used on questions which asked about tools that could provide certain functionalities such as "*I want to determine the readability of the code that was written by an author. Thus, I am looking for a tool compatible with Python that will provide me with such functionality.*". The TD tools category had a total of 186 questions. 105 coded with Fixing TD Tool Errors, 72 with Functionalities In Tools, and 9 with Tool Recommendations.

Analysing the data shows that TD tools were the topic of discussion for the majority of the questions. The TD tools category contained 186 questions. This accounts for 34.3% of the TD-related question set. The second most discussed question category was about paying off of TD. There were 128 questions in the TD repayment category and they account for 23.6% of the question set. The category with the least number of questions was the category TD prioritisation with 2 questions, which represents 0.4% of the set of questions.

The code that was used the most was Functionalities In Tools. This code was used to label 72 questions. The second most used code was Paying Off TD which was used in 53 questions. The codes with the least mentions were Representing TD and TD Organisation. Both of these codes were used in 1 question, respectively.

It was also observed that a single question might cover certain information that could be classified as belonging to more than one question type. As a result, the total from the overview does not equate to the number of questions in the question set. An example of such a question is:

"*Does anyone know if there is some kind of tool to put a number on technical debt of a code base, as a kind of code metric? If not, is anyone aware of an algorithm or set of heuristics for it? If neither of those things exists so far, I'd be interested in ideas for how to get started with such a thing. That is, how can I quantify the technical debt incurred by a method, a class, a namespace, an assembly, etc.*"

In the example, the first half of the question relates to looking for a TD tool, whilst the second relates to quantifying TD.

4.2 Results RQ2: Requirements for TD tools

During the manual requirements search, 67 requirements-specific questions were identified. With these 67 questions, 67 requirements were identified.

Three requirement themes were identified throughout the requirements analysis stage: **Implied Tool Requirements, Requested Tool Requirements**, and **Existing Tool Requirements**. Also, further analysis was conducted to reveal the TD management categories in each theme.

4.2.1 Implied Tool Requirements

This category details the requirements that fall under the classification in which a TD tool is not mentioned, but the requirement is rather inferred. 20 of the requirements are classified within this theme. A question that exhibits an implied tool requirement is "*I'm looking for a way to quantify where my team should spend it's time addressing technical debt in our codebase. One idea for this is to measure file churn (edits over time)*" In this example, the questioner does not mention explicitly that a tool is required, nonetheless, it is fair to claim that a tool could measure the amount of edits over a period of time. The resulting requirement would be *The tool may measure the number of times a file is edited*.

In the analysis of the Implied Tool Requirements, 6 categories of TD management were found: quantification, monitoring, documentation, repayment, management, and communication. An overview of the number of requirements in each category is shown in Table 4.2

Communication	Documentation	Management
1	6	1
Monitoring	**Repayment**	**Quantification**
3	5	8

Table 4.2: Management categories in Implied Tool Requirements

4.2.2 Requested Tool Requirements

This category includes requirements associated with questions for tools that could potentially offer a particular feature or functionality. 9 of the requirements are classified in this theme. A question that exhibits a requested tool requirement is "*Does anyone know if there is some kind of tool to put a number on technical debt of a code base, as a kind of code metric?*". The resulting requirement would be *The tool may quantify the TD of a codebase*.

In the analysis of the Requested Tool Requirements, 5 TD management categories were found: quantification, communication, management, repayment and monitoring. An overview of the number of requirements in each category is shown in Table 4.3

Communication	Management	Monitoring
1	1	2
Repayment	**Quantification**	
3	3	

Table 4.3: Management categories in Requested Tool Requirements

4.2.3 Existing Tool Requirements

This category details the requirements related to functionalities in tools that are operational in the software engineering field. The remaining 38 requirements are classified within this theme. A question that exhibits an existing tool requirement is "*Is it possible to get the technical debt per file in Sonar and preferably export it so it is possible to put in a chart?*" In this example, the questioner is looking to be able to export the TD metrics of every file scanned on an operational TD tool called SonarQube. The resulting requirement would therefore be *SonarQube may provide the TD per file*.

In the analysis of the Existing Tool Requirements, 4 TD management categories were found: monitoring, documentation, repayment, and quantification. The total amount of requirements for all categories is less than the number of requirements in the theme. The rest of the requirements pertain to the features of the tool, such as widgets and dashboards, which do not fall under a specific management category. An overview of the number of requirements in each category is shown in Table 4.4

Documentation	Monitoring
2	1
Repayment	Quantification
1	6

Table 4.4: Management categories in Existing Tool Requirements

An examination of the three requirement themes reveals that most requirements were associated with existing tools. There were 38 requirements that were placed within the Existing Tool Requirements theme which accounts for 56.7% of the identified requirements. The theme with the least requirements was Requested Tool Requirements with 9 requirements, which represents 13.4% of the requirements set.

An inspection of the management categories found within the themes reveals that there were three categories found in all three themes. These were quantification, repayment, and monitoring. In all three themes, the most popular category was quantification. In total, 17 requirements were associated with quantification, 6 with monitoring, 9 with repayment, 8 with documentation, 2 with communication and 2 with management. The categories communication and management had the least requirements.

4.2.4 Answers for questions with requirements

For the second part of the requirement analysis, the answers and comments to the requirement questions were analysed. It was noted that there were five ways that a question could be answered. The ways were assigned the codes: Answered - tool, Answered - no tool, Answered - alternative tool, Answered within tool, and Unanswered. Examining the responses to the requirement

questions revealed that 50 were answered and 17 had no solution. The codes of the types of responses are explained in more detail below.

The code **Answered - tool** describes a question in which one of the answers provides a TD tool as a solution. The questions that were eligible for the Answered - tool code were those with requirements under the themes Implied Tool Requirements and Requested Tool Requirements. An example of an answer within this category is "*Regarding profiling tools, if you decide to go that way you may take a look at xhprof. It has smaller size of the output files and web interface that you could embed into your app for continuous tracking*" . 14 requirement questions were answered with a TD tool suggestion.

The code **Answered - no tool** describes a question in which the answers provide a manual solution. The questions that were eligible for the Answered - no tool code were those with requirements under the themes Implied Tool Requirements and Requested Tool Requirements. An example of an answer coded with Answered - no tool is "*Firstly, you need to estimate the dev cost for the re-factoring as you would the sales driven feature request. This may well be tricky to get accurate if it's a large job but, assuming you have sufficiently experienced people in the 2 technologies, it should be doable. Secondly, you need an estimate of the cost of not re-factoring. If you were doing the estimates for me, I'd expect some level of metrics. For example, the difference between the average dev cost for VB code and for C# code over a quarter. Or some such. Will obviously depend on how accurately you track this. With these 2 numbers, you can estimate the pay back period that the re-factoring will give you i.e. at what point will the re-factoring turn from a net cost to a net gain*". 13 requirement questions were answered without tool suggestions.

The code **Answered - alternative tool** was used to label requirement questions that were based in tools and were answered by giving an alternative tool as a solution. The questions that were eligible for the Answered - alternative tool code were those with requirements under the theme Existing Tool Requirements, since these were based on TD tools. An example of an answer coded with Answered - alternative tool is "*The Eclipse Metrics plugin may get you close to what you're looking for. It'll give you a health check on your projects by reporting on different types of complexity (coupling, cyclomatic complexity, cohesion, length of methods and so on).*" 1 requirement question was answered with an alternative tool as a solution.

The code **Answered within tool** was used to label a requirement question that was based on tools and was answered by giving a solution within the confines of the tool mentioned in the question. The questions that were eligible for the Answered within tool code were those with requirements under the theme Existing Tool Requirements since these were based in TD tools. An example of a response with this code is "*If these are open source libraries that you don't want analyzed at all, you can exclude them from analysis altogether using sonar.exclusions. Else, you can add an exclusion pattern to avoid the creation of issues on those files, so that their technical debt will effec-*

tively be 0, while other metrics will be computed (lines of code, duplications etc.) - see sonar.issue.ignore.multicriteria". 20 requirement questions were answered with methods related to the tools mentioned in the questions.

The **Unanswered** code describes questions with no answer at all or questions with answers or comments that highlighted that the question had no solution. Questions with requirements in all three themes were eligible to be labelled by the code Unanswered. An example of a response coded with Unanswered is "*As of SonarQube 4.1 (January 2014), this is not possible yet.*". 17 requirement questions did not have a solution.

4.2.5 Proposed solutions for requirements

This section provides an overview of the requirements found in each theme in the form of tables. The requirements are shown in the first column, and the answers to the requirement questions are shown in the second. The requirements in Existing Tool Requirements start with the name of the tool mentioned in the question due to the requirement being specific to that tool. The other requirement themes are related to unspecified tools so they start with "The tool may...". Implied Tool Requirements are found in Table 4.6, Requested Tool Requirements in Table 4.5, and the Existing Tool Requirements in Table 4.7.

Requested Tool Requirements	
Requirement: **The tool may**	Question response code
quantify TD of a code base	Answered - tool
provide a metric that shows the degree of TD in an application	Answered - tool
manage TD	Answered - tool
remove unused files	Answered - tool
manage a Business Rules repository	Answered - no tool
offer static code analysis as a hosted service compatible with BitBucket and Mercurial	Unanswered
provide a view of added or removed TD	Answered - tool
measure code readability of a file	Answered - no tool
automate the migration of a codebase	Answered - tool

Table 4.5: Table of Requested Tool Requirements

An inspection of the requirements in each category revealed that the majority of requirements in the theme Existing Tool Requirements are associated with SonarQube. 29 requirements were related to SonarQube. This accounts for 76.3% of the requirements in the Existing Tool Requirements theme and 43.2% of the overall sum of requirements. Moreover, 15 of the requirement questions were coded with Unanswered, 1 with Answered - alternative tool, and 20 with Answered within tool.

The requirements in Implied Tool Requirements were divided relatively evenly between those defined with specific, distinct functionalities and those defined in a general manner. An example of a specific requirement is *The tool may filter log output to show unexpected exceptions in JUnit* whereas an exam-

Implied Tool Requirements	
Requirement : **The tool may**	Question response code
measure the amount of existing TD in a project	Answered - no tool
track completed tasks	Answered - no tool
track "who works on what"	Answered - no tool
quantify the impact of TD	Answered - no tool
quantify the value of refactoring	Answered - no tool
measure the number of times a file is edited	Answered - no tool
show the lines changed for an edited file	Answered - no tool
document TD	Answered - tool
provide visualizations of technical investment	Answered - no tool
detect dead code	Answered - tool
provide an estimate for ROI for TD repayment	Answered - tool
record TD in TFS	Answered - no tool
track TD	Answered - tool
measure the number of classes without Javadocs	Answered - tool
automatically integrate namespaces	Answered - tool
measure the amount of informative comments in a file	Answered - no tool
automatically generate simple input\output acceptance unit tests	Answered - tool
centralize common imports in React components	Answered - no tool
filter log outputs to show unexpected exceptions in JUnit	Unanswered
calculate the percentage of a codebase with comments	Answered - tool

Table 4.6: Table of Implied Tool Requirements

ple of a more generalised requirement is *The tool may document TD*. 11 of the 20 requirements in Implied Tool Requirements are related to general functionalities. Furthermore, 11 of the requirement questions were coded with Answered - no tool, 1 with Unanswered, and 8 with Answered - tool.

The requirements in the theme Requested Tool Requirements were also divided relatively evenly between specific and general definitions, with 5 out of a total of 9 requirements being defined with general functionalities. Additionally, 6 of the requirement questions were coded with Answered tool, 1 with Unanswered, and 2 with Answered - no tool.

Existing Tool Requirements	
Requirement	Question response code
Jira may keep track of a technical roadmap	Answered within tool
SonarQube may provide the TD per file	Answered within tool
SonarQube may provide a way to define TD	Answered within tool
SonarQube may automatically and dynamically calculate TD for individual programs	Answered - alternative tool
SonarQube may retrieve rules by category	Unanswered
SonarQube may calculate the TD of an architectural violation	Answered within tool
SonarQube may display TD in months and years	Unanswered
SonarQube may allow for the exclusion of files in the TD analysis	Answered within tool
SonarQube may retrieve 'Issues' by 'Characteristic'	Answered within tool
SonarQube may calculate the modularity metric from pre-existing metrics	Answered within tool
SonarQube may store project measures for each file in a database	Answered within tool
SonarQube may allow setting of TD for rules in which SQALE remediation function is linear	Unanswered
SonarQube may provide calculations for custome web rules	Answered within tool
SonarQube may facilitate for the addition of TD information for each rule	Answered within tool
SonarQube may provide a setting to change the number of decimal points in TD ratio	Unanswered
SonarQube may facilitate for the specification of TD for a custom FxCop rule	Unanswered
SonarQube may facilitate for the customization of the project level view	Unanswered
SonarQube may allow for the exportation of TD metrics for all projects	Answered within tool
SonarQube may facilitate for the removal of metrics from the dashboard	Unanswered
SonarQube may specify the last succesful analysisas Leak Period	Answered within tool
SonarQube may allow for the exclusion of files in coverage calculation	Unanswered
SonarQube may allow for configuration of time period in history diagram	Unanswered
SonarQube may provide a Technical Debt Pyramid Widget	Unanswered
SonarQube may automatically set issues as 'Resolve as Won't Fix'	Unanswered
TeamCity may provide a way to fail a build if Quality Gate fails	Answered within tool
Elixir Ecto may alias a column name	Answered within tool
SonarQube may provide the issue tag 'Will Fix Later'	Answered within tool
SonarLint may list the issues of changed files	Answered within tool
Jenkins may provide the user for each commit	Unanswered
SonarQube may report changes in coverage files	Unanswered
SonarQube may show code smell scores on the dashboard	Answered within tool
OpenCover may convert reports into NDepend compatible formats	Answered within tool
SonarApi may provide the estimated fix time for a project	Answered within tool
SonarQube may allow for the backdating of issues	Unanswered
SonarQube may track when a code smell was first identified	Unanswered
SonarQube may allow for the exclusion of files from the issues list	Answered within tool

Table 4.7: Table of Existing Tool Requirements

5 | Discussion

This chapter provides an interpretation of the results obtained, threats to the validity of the study are discussed, and the chapter ends with recommendations for future research.

5.1 Interpretation of Results

The TD management discussions results suggest that TD tools are the most discussed topic on the Stack Exchange websites we used. This is due to the fact that the TD tools category had 186 questions, which is 34.3% of the TD-related question set. The category with the second most questions was TD repayment with 128 questions, which is 23.6% of the TD-related question set. This suggests that the activity of paying off TD is a popular topic of discussion as well.

The code that was used the most was Functionalities In Tools. This shows that the most popular, indivisible, topic of discussion was centred around the features and capabilities of existing, operational, TD tools. The codes that were used least were TD Organisation and Representing TD, with each being used once. Questions coded with TD Organisation were put into the category TD monitoring and questions coded with Representing TD in the category TD Documentation. Even though the categories they belong to did not have the fewest questions, the separate activities of arranging and displaying TD were the least discussed singular issues by the users. This is because a code was utilised to highlight the primary TD issue or management activity in a question and one category could have many codes. It thus becomes possible to have a code with the least usage belonging to a category without the least questions.

In the requirements for TD tools results, 3 themes were discovered: Implied Tool Requirements, Requested Tool Requirements, and Existing Tool Requirements. From the themes discovered, the theme Existing Tool Requirements had the most requirements and the theme Requested Tool Requirements had the least requirements. It may show that most users already know about the current selection of TD tools and may be more inclined or prefer to ask for help with issues and improvements on the tools they would be using than to ask for alternative tools.

Inspecting the requirements in Implied Tool Requirements revealed that approximately half were defined in a general manner. This may be due to the nature of the theme. The requirements in this theme are inferred, therefore any activity that could be a functionality is recorded. This results in the identification of general functionalities such as "documenting TD". Approximately half of the requirements in Requested Tool Requirements were defined with general requirements as well. This suggests that nearly half of the users who requested tools did not have specific functionalities in mind but were rather looking for tools that performed general activities.

The examination of the requirements in Existing Tool Requirements revealed that the majority were related to the TD tool SonarQube. While this suggests that SonarQube is a popular TD tool amongst practitioners on the Stack Exchange, the tool also demands several improvements to properly support TDM.

From the eight management categories used to search for TD tool requirements, six contained requirements, mainly communication, documentation, management, monitoring, repayment and quantification. Of the six, three were found in all three requirement themes. These were quantification, repayment, and monitoring. This may suggest that the most common management activities practitioners look for in TD tools have to do with quantifying, paying off and monitoring TD. Quantification was the management category with the most requirements, indicating that practitioners are still most troubled by the TD management approach of quantifying TD in TD tools. Communication and management had the least requirements, suggesting that few practitioners are currently looking for TD tool features related to the conveying and general management of TD.

After reviewing the answers and comments on the requirements questions, 51 of the requirements questions were answered and 17 were unanswered. From the 51 answered questions, 37 were answered with manual solutions which shows that a majority of users answering TD-related requirements resort to manual solutions. This might suggest that there are practices and solutions that still need to be automated. The 17 unanswered requirements questions may also show that there are requirements that may be challenging to fulfil or may not have solutions.

5.2 Implications

Examining the discussions surrounding TD and TD tools on Q&A websites offers three primary benefits relevant to this exploratory study.

First, researchers and developers can use the results as a guide on areas to focus on when researching and developing TD tools. For example, the existing tool requirements theme had the majority of requirements. This can direct researchers to focus on existing tools and the issues and discussions surrounding them. Developers can also focus on the extension of existing

tools in order to meet the needs of practitioners who use these tools.

Second, researchers and developers are alerted to tool requirements that still need to be addressed. For example, researchers could study unanswered questions and provide solutions to such questions. Also, developers could look into the unanswered questions to address TD tool requirements that have not been fulfilled yet either manually or by tools. Finally, tool vendors gain an understanding of what practitioners are requesting and focus on developing such functionalities in existing or new tools.

5.3 Threats to Validity

Construct Validity

Construct validity refers to how well a set of indicators represents an idea that cannot be measured [25]. The manual coding stage's classification of the TD-related questions into question types poses a potential risk to the construct validity of this study. This is because this stage may result in the chosen codes and code categories being inadequate representatives of the TD topics discussed in the questions. The TD-related question types presented by Alfayez et al. [2] were used for manual deductive coding to minimise any potential risks.

The automated search procedure that was used to find questions related to TD poses a risk to the study's construct validity due to the uncertainty of all questions retrieved being a true representation of TD. To attenuate potential threats related to the automated question search, not only the was the body of the questions considered but also their titles and tags. Manual verification of automated search results was performed to ensure that the data set was representative of TD.

External Validity

The ability of conclusions of a study to be used in general and real world situations is examined by its external validity [3]. The degree of identification of questions related to TD poses a possible risk to the external validity of the study. Our automated search yielded question results based on keyword matches discovered within a question. But there are numerous terminologies that may be used to refer to TD and the keywords we picked were only a small portion of the potential keywords. It's also feasible that TD could be described without relying on a specific list of terms. As a result, we acknowledge that not all questions related to TD have been found.

Reliability

The degree to which a research approach yields consistent and steady outcomes is known as reliability [19]. In terms of our study, this is related to the degree to which other researchers can reproduce our findings. This study

may contain bias because human judgement was used to determine a question's relevance to TD and to categorise it within TD-related question categories. This can compromise the reliability of the study because the reasons to categorise a question may differ from person to person, which could lead to different results. To ensure the replicability of our findings, we have provided all data, decisions and methods used in this study in the replication package [1].

Data collection and analysis procedures were performed by one person. This is a threat to the reliability of the study because the individual's judgement is not verified throughout the whole process, and their existing views are reinforced and alternative ideas are not considered. Ultimately, this may result in errors, such as questions being put into incorrect categories or valid questions being marked as irrelevant. To mitigate this threat, a supervisor evaluated the results of each stage, provided feedback, and gave confirmation to continue with the remaining stages.

5.4 Recommendations

Based on the approach that has been carried out in this study, further improvements can be made. To increase the scope of the retrieved TD questions, there are three suggestions. Firstly, research on the full extent of alternative TD terms could be implemented to widen the subset of keywords that can be used in the automated search stage. This could be done by reviewing more studies conducted on TD on the Stack Exchange and divulging the keywords and phrases they encountered. Secondly, machine learning could be adopted to address the cases in which TD may be described without any keywords being mentioned in the title, body or tag of a question. Thirdly, the data sources for the TD-related questions could be expanded past the Stack Exchange into more Q&A networks such as Codidact[2].

To address the possible bias that might occur from one person performing the manual verification, coding, and requirements search, two participants in the study can be employed to execute the same procedures, compare their results, and discuss any differences that might occur. In addition to this suggestion, Cohen's Kappa coefficient, a statistical technique that is frequently used to gauge how well two reviewers agree [13], can be used.

Another step that could have been taken in the study is the comparison of the resulting requirements with the activities and functionalities of operational TD tools such as SonarQube. This could be done in future research to obtain a more specific subset of requirements that are yet to be addressed.

[1] https://github.com/Fxdzie/StackExchange_Extraction
[2] https://codidact.com/

6 | Conclusions

The current research aimed to find out what practitioners on the Stack Exchange network have discussed about TD and whether these discussions reveal software requirements that could promote the development of TD tools.

The research questions were as follows:

RQ1: *"What is discussed about technical debt and technical debt management tools on the Stack Exchange network?"*

RQ2: *"What requirements for TD tools are reported in the Stack Exchange network?"*

543 questions TD-related questions were identified and their content was analysed. Manual coding was performed on the set of TD-related questions, and the set of questions was then divided into 13 categories of TD questions. This was followed by a manual requirements search. The results revealed that TD tools were the most discussed topic on SO,SE and PM. In addition, the data analysis resulted in the identification of 65 questions with a total of 67 requirements. Further analysis of the requirements questions revealed three themes in which the requirements could be categorised: Implied Tool Requirements, Requested Tool Requirements, and Existing Tool Requirements.

Furthermore, it was determined that most of the identified requirements were associated with operational TD tools, with SonarQube being the most popular. Lastly, five ways in which the requirements questions could be answered were detected, mainly: answered with a tool suggestion, answered with a manual solution, answered with an alternative tool, answered within a tool and unanswered.

This preliminary study provides proof that the requirements for the TD tools can be found on Stack Exchange Q&A websites. Furthermore, the study presents two main contributions. Firstly, this study provides a baseline for the extension or development of TD tools as it outlines specifications and functionalities, put forward by practitioners. Lastly, the study presents a step forward in addressing the literature gap regarding discussions about TD tools in the Stack Exchange network.

7 | Acknowledgements

I appreciate my supervisors, Daniel Feitosa and João Paulo Biazotto for sharing the scripts I used to carry out the automated search. Additionally, I am grateful to them for providing me with advice, criticism, and answers to my inquiries during the course of the study.

Bibliography

[1] Alireza Ahmadi et al. “Learning Software Project Management From Analyzing Q&A’s in the Stack Exchange”. In: *IEEE Access* 11 (2023), pp. 5429–5441. DOI: 10.1109/ACCESS.2023.3235953.

[2] Reem Alfayez et al. “What is asked about technical debt (TD) on Stack Exchange question-and-answer (Q&A) websites? An observational study”. In: *Empirical Software Engineering* 28.2 (2023), p. 35. ISSN: 1573-7616. DOI: 10.1007/s10664-022-10269-5. URL: https://doi.org/10.1007/s10664-022-10269-5.

[3] Chittaranjan Andrade. “Internal, External, and Ecological Validity in Research Design, Conduct, and Evaluation”. In: *Indian Journal of Psychological Medicine* 40.5 (2018). PMID: 30275631, pp. 498–499. DOI: 10.4103/IJPSYM.IJPSYM_334_18. eprint: https://doi.org/10.4103/IJPSYM.IJPSYM_334_18. URL: https://doi.org/10.4103/IJPSYM.IJPSYM_334_18.

[4] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. “What are developers talking about? An analysis of topics and trends in Stack Overflow”. In: *Empirical Software Engineering* 19.3 (2014), pp. 619–654. ISSN: 1573-7616. DOI: 10.1007/s10664-012-9231-y. URL: https://doi.org/10.1007/s10664-012-9231-y.

[5] Aaditya Bhatia et al. “A Study of Bug Management Using the Stack Exchange Question and Answering Platform”. In: *IEEE Transactions on Software Engineering* 48.2 (2022), pp. 502–518. DOI: 10.1109/TSE.2020.2994006.

[6] João Paulo Biazotto et al. “Technical debt management automation: State of the art and future perspectives”. In: *Information and Software Technology* 167 (2024), p. 107375. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2023.107375. URL: https://www.sciencedirect.com/science/article/pii/S0950584923002306.

[7] Yania Crespo et al. “The role of awareness and gamification on technical debt management”. In: *Information and Software Technology* 150 (2022), p. 106946. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2022.106946. URL: https://www.sciencedirect.com/science/article/pii/S0950584922000921.

[8] Ward Cunningham. “The WyCash portfolio management system”. In: *SIGPLAN OOPS Mess.* 4.2 (1992), 29–30. ISSN: 1055-6400. DOI: 10.1145/

157710.157715. URL: https://doi.org/10.1145/157710.157715.

[9] Joshua Aldrich Edbert et al. *Exploring Technical Debt in Security Questions on Stack Overflow*. 2023. arXiv: 2307.11387 [cs.SE].

[10] Neil Ernst, Rick Kazman, and Julien Delange. *Technical Debt in Practice: How to Find It and Fix It*. The MIT Press, Aug. 2021. ISBN: 9780262366304. DOI: 10.7551/mitpress/12440.001.0001. URL: https://doi.org/10.7551/mitpress/12440.001.0001.

[11] Daniel Feitosa et al. "Mining for cost awareness in the infrastructure as code artifacts of cloud-based applications: An exploratory study". In: *Journal of Systems and Software* 215 (Sept. 2024), p. 112112. ISSN: 0164-1212. DOI: 10.1016/j.jss.2024.112112. URL: http://dx.doi.org/10.1016/j.jss.2024.112112.

[12] Eliakim Gama et al. "Using Stack Overflow to Assess Technical Debt Identification on Software Projects". In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. SBES '20. Natal, Brazil: Association for Computing Machinery, 2020, 730–739. ISBN: 9781450387538. DOI: 10.1145/3422392.3422429. URL: https://doi.org/10.1145/3422392.3422429.

[13] Natasa Gisev, J. Simon Bell, and Timothy F. Chen. "Interrater agreement and interrater reliability: Key concepts, approaches, and applications". In: *Research in Social and Administrative Pharmacy* 9.3 (2013), pp. 330–338. ISSN: 1551-7411. DOI: https://doi.org/10.1016/j.sapharm.2012.04.004. URL: https://www.sciencedirect.com/science/article/pii/S1551741112000642.

[14] LaiYee H. *How To Do Open, Axial, & Selective Coding in Grounded Theory*. https://delvetool.com/blog/openaxialselective. [Accessed 30-03-2024].

[15] Nicholas Kozanidis, Roberto Verdecchia, and Emitza Guzman. "Asking about Technical Debt: Characteristics and Automatic Identification of Technical Debt Questions on Stack Overflow". In: *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '22. Helsinki, Finland: Association for Computing Machinery, 2022, 45–56. ISBN: 9781450394277. DOI: 10.1145/3544902.3546245. URL: https://doi.org/10.1145/3544902.3546245.

[16] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. "Technical Debt: From Metaphor to Theory and Practice". In: *IEEE Software* 29.6 (2012), pp. 18–21. DOI: 10.1109/MS.2012.167.

[17] Zengyang Li, Paris Avgeriou, and Peng Liang. "A systematic mapping study on technical debt and its management". In: *Journal of Systems and Software* 101 (2015), pp. 193–220. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2014.12.027. URL: https://www.sciencedirect.com/science/article/pii/S0164121214002854.

[18] Luana Martins et al. "Hearing the voice of experts: Unveiling Stack Exchange communities' knowledge of test smells". In: *2023 IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Soft-*

ware Engineering (CHASE). 2023, pp. 80–91. DOI: 10.1109/CHASE58964.2023.00017.

[19] Fiona Middleton. *Reliability vs. Validity in Research | Difference, Types and Examples — scribbr.com*. https://www.scribbr.com/methodology/reliability-vs-validity/. [Accessed 04-07-2024].

[20] Anthony Peruma et al. "How do i refactor this? An empirical study on refactoring trends and topics in Stack Overflow". In: *Empirical Software Engineering* 27.1 (2021), p. 11. ISSN: 1573-7616. DOI: 10.1007/s10664-021-10045-x. URL: https://doi.org/10.1007/s10664-021-10045-x.

[21] Klaus Pohl et al. *Requirements Engineering Fundamentals: A Study Guide for the certified professional for requirements engineering exam: Foundation level - IREB compliant*. Rocky Nook, 2016.

[22] Daryl Posnett et al. "Mining Stack Exchange: Expertise Is Evident from Initial Contributions". In: *2012 International Conference on Social Informatics*. 2012, pp. 199–204. DOI: 10.1109/SocialInformatics.2012.67.

[23] Eder Pereira Santos et al. "Technical Debt on Agile Projects: Managers' point of view at Stack Exchange". In: *Proceedings of the XXI Brazilian Symposium on Software Quality*. SBQS '22. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450399999. DOI: 10.1145/3571473.3571500. URL: https://doi.org/10.1145/3571473.3571500.

[24] Camila Costa Silva, Matthias Galster, and Fabian Gilson. "Topic modeling in software engineering research". In: *Empirical Software Engineering* 26.6 (2021), p. 120. ISSN: 1573-7616. DOI: 10.1007/s10664-021-10026-0. URL: https://doi.org/10.1007/s10664-021-10026-0.

[25] Dag I. K. Sjøberg and Gunnar Rye Bergersen. "Construct Validity in Software Engineering". In: *IEEE Transactions on Software Engineering* 49.3 (2023), pp. 1374–1396. DOI: 10.1109/TSE.2022.3176725.

[26] Matúš Sulír and Marcel Regeci. "Software Engineers' Questions and Answers on Stack Exchange". In: *2022 IEEE 16th International Scientific Conference on Informatics (Informatics)*. 2022, pp. 304–310. DOI: 10.1109/Informatics57926.2022.10083403.

[27] Girish Suryanarayana, Ganesh Samarthyam, and Tushar Sharma. "Chapter 8 - Repaying Technical Debt in Practice". In: *Refactoring for Software Design Smells*. Ed. by Girish Suryanarayana, Ganesh Samarthyam, and Tushar Sharma. Boston: Morgan Kaufmann, 2015, pp. 203–212. ISBN: 978-0-12-801397-7. DOI: https://doi.org/10.1016/B978-0-12-801397-7.00008-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128013977000084.

[28] Edith Tom, Aybüke Aurum, and Richard Vidgen. "An exploration of technical debt". In: *Journal of Systems and Software* 86.6 (2013), pp. 1498–1516. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2012.12.052. URL: https://www.sciencedirect.com/science/article/pii/S0164121213000022.

[29] Zhiyuan Wan, Xin Xia, and Ahmed E. Hassan. "What Do Programmers Discuss About Blockchain? A Case Study on the Use of Balanced LDA and the Reference Architecture of a Domain to Capture Online Discussions About Blockchain Platforms Across Stack Exchange Communities". In: *IEEE Transactions on Software Engineering* 47.7 (2021), pp. 1331–1349. DOI: 10.1109/TSE.2019.2921343.

[30] Shaowei Wang, Tse-Hsun Chen, and Ahmed E. Hassan. "Understanding the Factors for Fast Answers in Technical QA Websites: An Empirical Study of Four Stack Exchange Websites". In: *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 2018, pp. 884–884. DOI: 10.1145/3180155.3182521.