Multivocal Literature Review on Non-Technical Debt in Software Development: An Exploratory Study

Hina Saeeda[1][a], Muhammad Ovais Ahmad[1][b] and Tomas Gustavsson[2][c]

[1]*Department of Computer Science, Karlstad University, Sweden*

[2]*Karlstad Business School, Karlstad University, Sweden*

Keywords: Technical Debt, Non-Technical Debt, Process Debt, Social Debt, People Debt, Organizational Debt, Software Development, Multivocal Literature Review, Systematic Review.

Abstract: Earlier research has focused on technical debt (TD). While numerous issues connected to non-technical aspects of software development (SD) that are equally worthy of ”debt” status are neglected. Simultaneously, these types of debts regularly develop significant challenges to be addressed, demonstrating that the debt metaphor may be used to reason about elements other than technical ones. It motivates us to create the new umbrella term ”Non-Technical Debt” (NTD) to investigate people, processes, culture, social, and organizational concerns under its cover. All types of debt are similar in some ways, and they are often caused by making risky decisions. Therefore, ignoring any one dimension of debt can have severe consequences on the successful completion of SD projects. This study investigates recent literature on the current state of knowledge about NTD, its causes, and mitigation strategies. By using a thematic analysis approach, we found five NTD types (i.e., people, process, culture, social, and organizational). We further identified their accumulation causes and discussed remedies for mitigation.

1 INTRODUCTION

Software engineering is a “sociotechnical” phenomenon involving social and technical aspects (Winter et al., 2014; Storey et al., 2020). For a project to be successful, technical skills and abilities must work effectively with non-technical (social) aspects (Tonin, 2018). Problems associated with poor code quality, management, organization, and resources compromise the quality of a project. Poor code quality is linked to TD, but problems with management, organization, and management are non technical. Both technical and non-technical issues are caused by cognitive errors and miscommunication within and across teams (Tamburri et al., 2015). These issues further lead to the accumulation of debt in SD projects. In software engineering, debt is metaphorically inherited to represent the extra cost or effort needed to fix quality issues. The debt metaphor in software development is used to describe socio-technical challenges: - technical debt (code debt and code smell) and other non-technical debts (social, cultural, and organizational) (Kruchten et al., 2012; Martini and Bosch, 2015). Therefore, technical debt is only one side of the coin. Debt can be either technical (code and design aspects), or non-technical (human and social aspects) (Chen, 2022; Ozkaya, 2016; Rios et al., 2018). TD is “the debt acquired by rushing software project development, resulting in defects and expensive maintenance costs ” (Kazman, 2019). Thus, debt extension to the software domain occurs due to software development decisions prioritizing speed or delivery over well-designed code (Martini and Bosch, 2017).

Academics and industries have been interested in TD for 20 years (Kruchten et al., 2012; Martini et al., 2019) and investigated it from various dimensions, for example, TD effort (Martini and Bosch, 2015), TD tools (Ozkaya, 2016; Hilty and Aebischer, 2015) , TD management strategies (Palomba et al., 2018), managing architectural TD (Rios et al., 2018), TD in Agile development (Holvitie et al., 2018), TD management elements (Melo et al., 2021), TD prioritization (Kruchten et al., 2012) and so on.

While studies on NTD exploration are still in their infancy. Research shows that TD is directly linked to the NTD as according,(Klinger et al., 2011, p.35)“the choice to gain technical debt is made by non-technical stakeholders, who drive the project to obtain new

[a] https://orcid.org/0000-0002-7562-338X
[b] https://orcid.org/0000-0002-7885-0369
[c] https://orcid.org/0000-0002-1512-6592

technical debt or uncover existing technical debt that was previously invisible”. Further, timeline pressure, negligence, an inadequate education, bad practices, non-systematic quality verification, or fundamental ineptitude are cited as the factors responsible of technical debt(Kruchten et al., 2012). It shows that non-technical (social) factors have a direct effect on technical debt in software development projects. This shows that software projects’ success is likewise dependent on technical and non-technical aspects of the SD, and the effects of TD and NTD are equally damaging, i.e., the extra cost generated by NTD is conceptually alike to TD (Vinsennau, 2016; Tonin, 2018). Therefore, NTD need adjustment and empirical research to investigate how they are accumulated and managed and how they are linked to the generation of TD. This study aims to explore prominent NTD types, causes, and mitigation strategies in SD through an exploratory multivocal literature review. This is the first multi-vocal literature review (Garousi et al., 2019) conducted on the topic. The rest of the study is structured as follows: section 2. Background, section 3. Design and methodology, Section 4. Result, Section 5. Discussion and Conclusion and Section 6. Threats to validity and limitations.

2 BACKGROUND

TD arises when development teams take shortcuts in various activities to accelerate the delivery of functionality that will require reworking later. TD metaphor mainly encompasses concerns related to code, testing, architecture, and so on (Yli-Huumo et al., 2017). Aside from TD, there are other types of debt that are not well studied and are produced by socio-technical factors (e.g., a lack of supporting practices, incomplete tasks in the development process, low external quality, and incomplete features or functionalities)(Li et al., 2015; Dargó et al., 2019). Socio-technical congruence may be the first rudimentary sign of social debt in specific development communities, and impact product quality (Cataldo et al., 2009). Researchers (De Souza and Redmiles, 2011) investigate awareness maintenance approaches as the socio-technical stream advances. These techniques are fundamentally comparable to the concept of social debt, as they aim to manage and maintain project knowledge to avoid delays and related ”debt”. Further (Bird et al., 2009) employ social-network analysis to examine coordination between groups of developers with socio-technical interdependence. Social debt refers to the cumulative, rising costs of the current state of affairs, coupled with unanticipated and negative consequences inside a developing community (Tamburri, 2019). According to (Tamburri, 2019), the causes of sub-optimal development communities might range from global distance to organizational constraints to incorrect or ignorant socio-technical choices (i.e., choices that affect both social and technical facets of software development). One or more bad sociotechnical choices can initiate the phenomenon (Tamburri et al., 2015). The poorer decisions that software development teams make, the more adverse effects are observed. As long as these poor choices remain unchanged, their repercussions will endure and intensify. So, development communities are made up of people, processes, and cultures, and all these factors are connected. For more informed decision-making, measuring the social debt (if any) associated with such decisions is essential. While (Cusick and Prasad, 2006), (Jaktman, 1998)(Andreou, 2003)investigate the impact of organizational decisions on collaboration and product quality. (Nagappan et al., 2008) demonstrate the effects of organizational structure and other “human” factors on software quality in practice. These studies signal more research in product quality and evaluation because they connect with social debt and organizational structures’ roles in productivity. For instance, (Cusick and Prasad, 2006) investigates the process of determining if the current organizational structure of a company is effective (or in accordance) with specific actions of quality production . Further, process debt is explored and investigated as its issues can be attributable to inefficient processes that require adjustment (Martini et al., 2020) and require further empirical investigation. In addition to examining process, social, and organizational debt empirically , which were previously described as beyond the purview of technical debt, it is necessary to investigate people and cultural debt.

3 RESEARCH METHOD

The following section outlines the *Multivocal Literature Review* (MLR) technique adopted in this study. We follow the established MLR guidelines and procedures proposed by (Garousi et al., 2019). The complete systematic MLR process consists of three phases: planning, conducting, and reporting. The remainder of the section goes into great detail on each phase of the MLR study. We performed our MLR search, on April 10, 2022, and finished the analysis and reporting until the end of October, 2022.

3.1 Planning the MLR

Using grey and scientific literature, multivocal literature research examines a larger spectrum of software engineering challenges. The following two processes (i.e., motivation, objectives, and research questions) make up the MLR planning phase.

Motivation: (Garousi et al., 2019) suggest including grey literature in reviews when relevant knowledge is not reported adequately in scientific literature. MLR is useful in finding what is happening in an under-discovered phenomenon. MLR incorporate all types of accessible literature includes, but is not limited to: (blogs, white papers, articles, and academic literature) (Garousi et al., 2019). Therefore, our MLR is vital for expanding research by including non-scientific literature to know more about NTD in SD.

Objectives and Research Questions:
The main objectives of this study are to understand: 1. Identify the main types of NTD in software development projects. 2. Identify the leading causes of NTD in software development projects. 3. Determine the ways to prevent or mitigate NTD in software development projects. 4. Determine the possibilities for future NTD research. The following research questions serve as the foundation for this study.

RQ1- What are the different prominent NTD types in software development projects?

RQ2- What are the causes of NTD accumulation in software development projects?

RQ3- What are the mitigation strategies for NTD handling in software development projects?

3.2 Conducting the MLR

The MLR conduction process is based on several phases, as mentioned in Figure 1: Research Conduction Process.

Search Strategies and Data Sources: The designed search string includes the search terms 'population', and 'intervention' based on (PICO) criteria suggested by (Kitchenham et al., 2009). Where population refers to the application area, "software," and intervention represents NTD types. Based on intervention, we selected five key terms for finalizing the search string (i.e., process debt, social debt, people debt, organizational debt, and cultural debt). Finally, the term software ensures we do not include research from other domains, like social sciences or economics. The finalized search string was ("process debt" OR "social debt" OR "people debt" OR "organizational debt" OR "culture debt") AND ("Software"). The term "software" is used since this research will involve studies that address software, software development, and software systems. As a result, all publications containing "software" in the title, abstract, or keyword will be included in the search. At the same time, the terms process debt, people debt, social debt, and organizational debt were used to include all NTD-associated sources. We finalized these specific keywords based on our pilot search results.

The search string was designed to retrieve results from Google, which was preferred due to its speed and relevance and its ability to collect grey literature. To be inclusive, the search string was kept simple and not restricted to specific years, resulting in 110 results across 11 pages.

Inclusion /Exclusion Criteria: Sources relevant to understanding NTD in SD and analyzing various forms of NTD (people, process, social, cultural, organizational) are included in the research. Sources not in English, videos, ads, catalogs, duplicates, research profiles, and outside of the software engineering domain are excluded. 36 records were excluded in the first round, and 74 remained for analysis. 40 were deemed relevant in the second round, and the rest were excluded, with a double check by two authors.

Quality Assessment: To apply quality assessment criteria, the 40 primary studies (shown in Appendix A: Primary List of Studies) were divided into two categories: grey literature (GL) and scientific literature (SL). We used the 11-factor quality rating criteria established by (Dybå and Dingsøyr, 2008) for scientific literature shown in Table2. Whereas we adopted the quality assessment checklist of grey literature from (Garousi et al., 2019) as shown in Table1. Each criterion was given a binary (1 or 0) grade, where 1 implies the answer is yes and 0 means the answer is no. Both checklist requirements gathered data on how well it was possible to evaluate the 12 SL and 28 GL sources' quality.

For the scientific studies, based on the screening criterion, each of the 12 studies received a score of 1; each study offered a clear research objective and background for the investigation. However, one paper (Martini and Bosch, 2017) lacked an adequate discussion of its research methodology and did not employ proper sampling. No relevant control group was found for comparing treatments in the primary papers. All primary publications adequately detailed data collecting and data analysis, except (Tamburri et al., 2013). While (Martini and Bosch, 2017) is lacking design and sampling phases. while research findings and research value criteria are applied to and fulfilled by all papers. Three papers (Martini and Bosch, 2017; De Souza and Redmiles, 2011; Tamburri et al., 2013) failed to discuss the researcher-participant connection explicitly. None of the papers received a complete score on the quality evaluation, but a few publications received two or three negative responses.

Table 1: Quality Assessment Check List for GL.

1. Is the publishing organization reputable?
2. Is an individual author associated with a reputable organization?
3. Has the author published other work in the field?
4. Does the author have expertise in the area? (e.g., job title principal software engineer)
5. Does the source have a clearly stated aim?
6. Does the source have a stated methodology?
7. Is the source supported by authoritative, documented references?
8. Does the work cover a specific question?
9. Does the work refer to a particular population?
10. Does the work seem to be balanced in presentation?
11. Is the statement in the sources as objective as possible?
12. Are the conclusions supported by the data?
13. Does the item have a clearly stated date?
14. Does it enrich or add something unique to the research?
15. Does it strengthen or refute a current position
16. 1st tier GL: High outlet control/ High credibility: thesis, reports, white papers
17. 2nd tier GL : Moderate outlet control/ Moderate credibility: Q/A sites, Wiki articles, workshop
18. 3rd tier GL : Low outlet control/ Low credibility: Blog posts

For GL quality assessment, we divided grey literature according to (Garousi et al., 2019) in three tiers. We have 9 number of primary sources in 1st tier GL, which cover high outlet control/ high credibility, including thesis, reports, and white papers. Two primary sources come under 2nd tier GL, covering moderate outlet control/credibility, including Q/A sites, wiki articles, and workshops. There are 17, 3rd tier GL that cover low outlet control/ Low credibility. Not a single of the GL sources received a complete score on the quality evaluation but reached the minimum threshold, which indicates credible sources as a whole. All the grey literature sources clearly stated their goal. Web blogs also obviously lack a methodology section and documented references, whereas thesis and seminar reports have clearly written methodology sections. All of the sources provide information on specific NTD issues, cover the software development population, and are balanced in the overall presentation. Most web blog conclusions lack data support and do not include cited links. But all of the sources present new information about NTD.

Table 2: Quality Assessment Check List for SL.

1. Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?
2. Is there a clear statement of the aims of the research?
3. Is there an adequate description of the context in which the research was carried out?
4. Was the research design appropriate to address the aims of the research?
5. Was the recruitment strategy appropriate to the aims of the research?
6. Was there a control group with which to compare treatments?
7. Was the data collected in a way that addressed the research issue?
8. Was the data analysis sufficiently rigorous?
9. Has the relationship between the researcher and participants been adequately considered?
10. Is there a clear statement of findings?
11. Is the study of value for research or practice?

Data Extraction and Analysis: After completing the quality analysis, detailed data analysis was conducted based on thematic analysis techniques (Braun and Clarke, 2006). The thematic analysis yielded primarily five themes, each highlighting NTD types. Further codes were created against each NTD type in the form of its cause and mitigation strategies. The raw coding for causes and mitigation strategies helps us inductively consolidate it into the five main NTD themes. This helps us group various causes and solutions to provide recommendations.

4 RESULTS

Based on the aforementioned research questions (section 3.1), the result of our MLR study is presented in this section.

The analysis begins with demographic information, i.e., (i) type of literature, (ii) type of sources, and (iii) publication by year, and then proceeds to a detailed assessment based on thematic analysis.

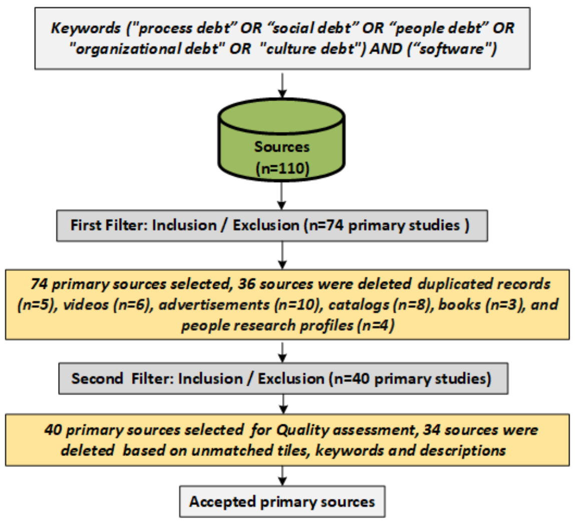
Keywords ("process debt" OR "social debt" OR "people debt" OR "organizational debt" OR "culture debt") AND ("software")
Sources (n=110)
First Filter: Inclusion / Exclusion (n=74 primary studies)
74 primary sources selected, 36 sources were deleted duplicated records (n=5), videos (n=6), advertisements (n=10), catalogs (n=8), books (n=3), and people research profiles (n=4)
Second Filter: Inclusion / Exclusion (n=40 primary studies)
40 primary sources selected for Quality assessment, 34 sources were deleted based on unmatched tiles, keywords and descriptions
Accepted primary sources

Figure 1: Research conduction process.

4.1 Demographic

The publication trend shows fewer relevant studies before 2015, and to our surprise, in 2018, we captured only one relevant record. On the other hand, active research efforts have been evident since 2019, as shown in Figure 2. This highlights the developing interest of practitioners in the NTD.

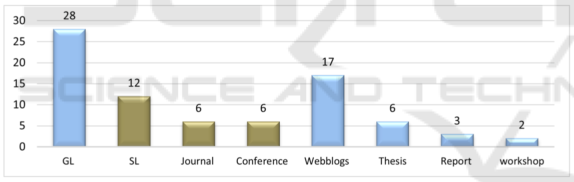

Figure 2: Type of literature and sources.

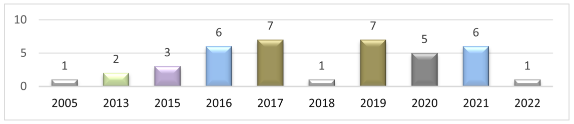

Figure 3: Publication's trend.

Figure 3, shows that the highest number of resources are weblogs (n = 17). Other sources include journals (n = 6), conferences (n = 6), thesis (n = 6), experience reports (n = 3), and workshop papers (n = 2).

4.2 NTD State of the Art

This section collectively encompasses insight into RQ1 (Types of the NTD) and RQ2(Causes of NTD accumulation).
Process Debt: Process debt is "a suboptimal action or event with short-term advantages but long-term detrimental effects"(Martini et al., 2020). One illustration of Process debt is when teams conduct stand-up meetings to present status so that bosses are aware of what is happening(De Souza and Redmiles, 2011). Team members might demonstrate to their leader that they are aware of project developments as a short-term benefit. The long-term detrimental effect is that conversation focuses more on documentation than information exchange and addressing pertinent task interdependence (De Souza and Redmiles, 2011). According to (Martini et al., 2020), "activity-specific debt", "role debt", "documentation debt", "unsuitable processes," "synchronization debt" and "infrastructure debt" are types of process debt. The basic causes of process debt are lack of competencies to design, manage and execute processes as well as lack of follow-up assessment (Bird et al., 2009). Processes are pending and not executed correctly (Melo et al., 2021; Yli-Huumo et al., 2017) inappropriate processes, and outdated processes are other leading causes of process debt (Wenger E, 2002). Irregular modifications in the development process usually lead to knowledge depletion, and process debt (Irwin and Adrian, 2011; De Souza and Redmiles, 2011). Another important aspect is understanding and communicating a specific process value in an organization (Bird et al., 2009). If not communicated properly, it can lead to process debt. Further, the border between software and hardware development is getting closer daily, requiring careful process design. Understanding contexts and situations in process design is important, for example, when processes are designed only for the software development team and ignore the collaborating hardware teams, it is a remedy for process debt. The main reason is that individuals in each of these teams have different experiences, maturity, motivation, and leadership styles (they might like or not be guided by a process) (Bird et al., 2009). The human factor in the process plays a critical role in the accumulation or avoidance of process debt. In reference to software development, it is the apparent distance between team members with less power and those with more power, like those with more experience or who make decisions to design and implement the specif processes. In the end, it messes up the software process and hurts the money of the organization. There are priggish members or nit-picker colleagues who require pointlessly precise conformance or excessive propriety from others, which is unpleasant. The attitude irritates co-workers and hinders the software development process in return (Espinosa, 2021). Organizations and companies can also contribute to process debt, i.e., when a process is not carried out by the individuals who are supposed to and when roles

created for a process do not have clear matches in the organization. It leads to the process debt(Espinosa, 2021).

People Debt: "People debt refers to people's problems in a software company that can slow or impede developmental operations"(Melo et al., 2021). It is characterized by shortcuts and settling for sub-optimal strategies and poor communication and coordination" One example of people debt is expertise concentrated in too few people due to delayed training and/or hiring (Yli-Huumo et al., 2016). Poor communication and coordination within and outside a software development team lead to people's debt. Lack of discussions between business people, architects, and development teams results in people's debt. For example, business people may feel that the development team focuses solely on delivering new features and ignores refactoring old solutions, or they may not be interested in discussing priorities. (Yli-Huumo et al., 2016).

The inability to create a team-friendly environment frequently results in an isolated and broken workflow, which rarely leads to team efficiency or production (Melo et al., 2021). Lack of communication may affect teams' performance (Melo et al., 2021; Tamburri and Di Nitto, 2015; Martini et al., 2019). In remote work without face-to-face connection, individuals miss a sense of shared purpose and are more indifferent to their employers (Ladewig, 2019). Loneliness is one of the primary obstacles that distant workers may need to overcome. This is especially true if they are not accustomed to working alone throughout the day. Egalitarian organizations evaluate ideas based on merit and prioritize open communication that ends up with cliques, internal politics, and a breeding ground for big personalities (Bellotti, 2021). When the communication pathways grow around personal relationships rather than chains of command, individuals get frustrated (Yli-Huumo et al., 2016). People leave because they burn out over time, especially software engineers, and the debt in relationships piles up (Bellotti, 2021). Leaving people can have a detrimental effect on team performance because, in flat organizations, no one understands to whom tasks need to be transferred or who should fill the vacated position (Bellotti, 2021). Further, people with false-positive results ignore the warning sign and prioritize tasks that help them get ahead at the bottom of the backlog, and it is the ignored activities where the debt is created (Bellotti, 2021).

Culture Debt: "Cultural debt is a technical decision that borrows against the organization's culture". Such decisions can introduce team divisions, deteriorate communication, or even weaken leadership effectiveness" (Chen, 2022; Sutton, 2019). An example of cultural debt is an inconsiderate culture of hiring the wrong people (Chen, 2022), dismissing complaints, charging discrepancies, and giving unequal rewards(Jaktman, 1998; Hosking, 2017; Hosking, 2021). Cultural debt is very damaging when managers do not understand what type of organizational culture they are trying to build (Chen, 2022). A lack of cultural understanding in managers makes a path for a weak culture that has a multiplier effect because managers are recruiting, managing, and leading people in their organizations. Less management investment is a major reason for weak cultures and cultural debts in the software industry.

Cultural debt is also caused once you hire the wrong people; it is very hard to "fix" (Chen, 2022). You can revert code, but you cannot revert culture. For example, you cannot just reverse a lack of diversity by hiring more people from underrepresented groups if 95 percent of your organization is already made up of a specific population (any specific ethnic group). One wrong choice can send a signal of ill feelings through the staff as a whole. Cultural debt has very high interest rates. Sometimes, even if a culture misfit is fired or leaves, you will never be able to pay off the debt you have racked up. Optimism, a strong work ethic, and interpersonal skills can be more important than any other skills a candidate may have to indulge in the working culture. The new hires who failed in the first 18 months did not do so because they did not have the right technical skills. Instead, they failed because they did not want to learn or because they lacked the right temperament or emotional intelligence.

Social Debt: It referred to "the presence of sub-optimality in the development community, which causes a negative effect on SD" (Tamburri et al., 2013). Social debt is analogous to technical debt in many ways. "Social debt refers to unplanned project expenditures associated with a non-optimal developer community, or weak team interactions" (Yli-Huumo et al., 2016; Tamburri et al., 2013). An example of social debt is the lack of adequate communication between different parts of the organization (e.g., between the development and operations teams) (Tamburri et al., 2013).

The causes of social debt are directly related to how people act in software development teams. For example, tense social interactions, bad behavior, and bad products are all signs of social debt (Espinosa, 2021). Such deficiencies in social interactions, including a lack of communication, coordination, and cooperation, lead to compromised leadership decisions resulting in unfavourable long-term outcomes

(Dreesen et al., 2021). The four commonly discussed social debt causes are (Tamburri et al., 2013; Espinosa, 2021): i. Black Clouds: Organizations don't make it easy for people to talk to each other, and effective communication between teammates doesn't lead to the sharing of knowledge. ii. Organizational silos are associated with a lack of task coordination and interconnection with each other and with non-conducive conditions for effective communication among teammates. Lone Wolf: It occurs when defiant teammates perform their duties regardless of their teammates. For example, unsanctioned architecture decisions across the development process contribute to project delays. Radio silence or bottleneck : It refers to interactions between leaders and teammates that are formally structured. For example, if one person acts as a go-between for several teams, there will be too much communication and many delays.

Organizational Debt: It referred to "the accumulation of changes that leaders should have made but didn't" (Bellotti, 2021; Sutton, 2019). Organizational debt occurs when there is a pressure of just let it get done"(Sutton, 2019; Dignan, 2017; Bellotti, 2021). Some of the major leading causes of organizational debt are condescending behavior, disgruntlement, or rage quitting (Kazman, 2019) of team members. Slow or rigid organizations build up debt because they rely on their old systems, have persistent skill gaps, lose leaders often, and don't like taking risks (Dargó et al., 2019; Sutton, 2019; Vinsennau, 2016).

Organizational debt leads to technical debt because of deferred investments due to budget uncertainty, poor governance and architecture, and organizational restructuring (Vinsennau, 2016). Organizations accumulate debt when a new idea, process, or way of working is introduced enthusiastically but, during execution, management ceases it. This leads to a less frequent feedback loop, halfway through the implementation of a new way of working, and budget cuts result in old behaviors creeping back in with frustration and "change fatigue" (Vinsennau, 2016). Typically, command-and-control organizations are rife with technical debt too. Flat organizations prioritize open communication and promote all ideas' evaluation based on merit without middle managers to gunk things up, which results in cliques and internal politics (Bellotti, 2021). Uber's organizational debt is shown by the fact that they kept hiring people who didn't have enough experience to lead teams and think about the company as a whole(Nagappan et al., 2008).

4.3 Mitigation strategies for NTD

This section encompasses insight into the RQ3 (Mitigation strategies for identified NTD types).

Process Debt Mitigation: For process debt management, the three main steps require careful execution. First, it is important to understand the process debt conceptually. It is essential to have a person in charge of managing processes who has an in-depth understanding of and knows the importance of those processes (Martini et al., 2020). Second, the process must be designed with purpose and value. The process that was made just for management's sake will fail in the end. Third, it's important to measure the process, which includes the development team, customers, and management-related processes. Sprint burn-down charts and defect trends can be used to figure out how to get things done on time and how good they are. If not the whole process, some of its parts might need to be automated. For example, the preparation, compilation, and processing of multiple Excel sheets are sometimes reported as labor-intensive. With the help of one AI-based system, reporting, decision-making, and productivity will go up, reducing process debt and enabling functional augmentation and virtualization (Kavas, 2021). Also, communicate the value of the new process to the teams and allocate sufficient time to use it in their context (Eaden., 2017). Any reform that is rushed through can only result in backlash and misery. Getting everyone to understand the benefit of a new process and adopt it can take a long time, but it will be worth in the end to have a better process that will help organizations and teams achieve the goal (Eaden., 2017).

People Debt Mitigation: People's debt management is challenging as it is more connected with human behaviour, psychology, and well-being. Instead of waiting and seeing policy, the people's debt requires immediate action; otherwise, its adverse effects would spread to individuals in the organization swiftly (Hilty and Aebischer, 2015; De Souza and Redmiles, 2011). A channel must be established for continuous monitoring and communication because it speeds up progress reporting and facilitates faster decision-making. Also, frequent and effective communication is important for building teams, getting people involved, solving problems, and solving conflicts. Communication needs to be clear and open with stakeholders for better decision-making. For instance, understanding the people's debt and frequent communication between businesspersons and engineering teams about their decisions lessen the business pressure on engineers (Yli-Huumo et al., 2017). People's well-being is connected to their social ties and support; long periods of isolation at work are linked to stress, depression, and low morale. To minimize people's debt, joint work groups are recommended for daily discussion, and social well-being (Ladewig, 2019).

Culture Debt Mitigation: You can revert code, but you cannot revert culture (Chen, 2022). To minimize cultural debt, a collaborative and cooperative environment is required. Creating natural workplaces through clear communication and an organized workplace aids in the regulation of cultural debt (Coleman, 2019). Cultural standards and values must be considered in communication and information-sharing materials in multicultural teams (O'Keeffe, 2017). Invest in managers who develop the culture and creative mindset in the company. A correct and collaborative mindset helps to minimize cultural debt(Chen, 2022; Wenger E, 2002). Further, establishing a culture of continuous testing(Nagappan et al., 2008) and enabling knowledge exchange with multicultural audiences(O'Keeffe, 2017) ease the effects of cultural debt.

Social Debt Mitigation: There are several organizational strategies, frameworks, models, tools, and guidelines to help monitor and mitigate social debt. Social debt mitigation strategies are linked to collaborative work environments (Ladewig, 2019; Ozkaya, 2016), social network analysis (Tamburri et al., 2015), honest and open communication in teams, and intense collaboration (Dreesen et al., 2021). Practitioners must be equipped with the tools necessary to diagnose and manage social debt in their development communities(Tamburri et al., 2013). Some of the tools reported to detect and manage social debt are: GEEZMO alarms managers and supervisors about circumstances affecting teammates' mood; CodeFace4Smell detects organizational silos, black cloud, Lone wolf, and Radio silence social debt causes; DAHLIA, with key aspects, includes decision popularity, decision awareness to investigate some of the reasons of social debt (Espinosa, 2021).

Organizational Debt Mitigation: The organizations need to keep a watchful eye on their decisions and changes that need to be identified, prioritized, measured, and monitored. Monitoring facilitates faster decision-making and business improvement by speeding up reporting. Real-time exception detection allows for real-time responses. While communication is important for trust, team building, improved relationships, problem-solving, and conflict resolution. Robust document management guarantees that everyone in a company, regardless of department or team, understands its storage, review process, and up-to-date status, and that, if any discourse actions are required, (Yli-Huumo et al., 2017; Dreesen et al., 2021). Adhering to an organizational structure that is based on validated frameworks(Martini and Bosch, 2017) and amending organizational charts can help streamline organisational processes, improve decision-making, manage multiple locations, promote employee performance, and focus on customer service and satisfaction (Martini and Bosch, 2017). Social network analysis tools (Tamburri et al., 2013) can effectively forecast, manage, and handle debt in organizations. (Tamburri et al., 2015). Automating debt identification tools and testing tools can be used to monitor the debt sources continuously and find opportunities to overcome them (Tamburri, 2019). Organizations need adaptability and a high degree of flexibility to survive in the software business. Cloud-based services ensure the continuity of organizational processes, reduce costs, and foster increased collaboration. Cloud-based services are scalable, and provide automatic software updates. It is not only efficient but also beneficial to the environment, and it provides automatic software integration (Kerv, 2022). The organizational culture can be improved by creating and communicating meaningful values to employees, conducting proper selection procedures, improving orientation and on-boarding for teams, enabling and empowering employees in skills and decisions, engaging employees in training, coaching according to their needs and domains, and communicating effectively and efficiently within teams (Casey, 2020).

5 DISCUSION AND CONCLUSION

Recently, social debt has been investigated by (Tamburri et al., 2015) and process debt by (Martini et al., 2020). Both provided similarities to the TD analogy, but for cultural, people, and organizational debt, we need more studies to find their similarities with TD. While technical debt resides within the system codebase, NTD seems more pervasive and intertwined with people, organizations, and their working processes and cultural issues. For example, a socio-technical decision generates both technical and social debt. As a result, in addition to generating technical debt, each socio-technical decision can result in any or all of the NTD types (i.e., people, process, culture, social, and organizational).

TD outcomes are measured financially while NTD outcomes are measured by their consequences. Further research is needed to quantify NTD in software projects. NTD is more difficult to fix than TD, contributes to TD accumulation, and has both short and long-term effects. NTD and TD are closely linked to human factors such as software architecture and their impact on business and culture(Tamburri et al., 2015; Martini et al., 2020; Espinosa, 2021).

NTD reflects and weighs heavily on the human and social aspects since it is caused by factors such as cognitive distance (lack of or excessive communication), mismatched architecture, and cultural and organizational systems (Espinosa, 2021). NTD must be addressed together with TD to avoid major consequences in software project failure. The development communities often interchange cultural and organizational terms, despite their different meanings and contexts. Cultural debt in software development can result from carelessness in adopting policies and practices, while organizational debt can stem from sluggish or inflexible systems, ageing or legacy systems, and poor architectural choices. Insufficient skills to upgrade and modernize systems and infrastructure is also linked to this debt. In return, it leads to an inability to integrate systems and upgrade given organizational setups. (Vinsennau, 2016).

In software development communities, there are two cultural perspectives: work culture and employees' cultural background. The combination of these factors can lead to the selection of sub-optimal processes and result in process debt (Martini et al., 2020). Therefore, poor cultural and organizational choices are directly proportional to the selection of inefficient software development processes. For the people debt, the leading causes found were directly associated with inefficient collaboration (Ladewig, 2019), insufficient communication (Melo et al., 2021), shortcuts in communication (Yli-Huumo et al., 2017), and lack of inter-team coordination (Martini et al., 2019). Therefore, it is clear that many issues linked to the causes of process debt are based on a lack of communication and coordination. The same pattern is reported in triggering social debt, i.e., lack of suitable communication among important sides of the organization (Martini et al., 2019) and missing social connections or reduction of communication (Dreesen et al., 2021). The same applies to organizational debt, as the main reason for organizational debt causes are associated with uneven information sharing among teams (Yli-Huumo et al., 2017)and a lack of healthy communication (Eaden., 2017). The second most important pattern we found is the relation of organizational debt with culture and software process, as poor organizational cultures are linked with hindering software development progress (Eaden., 2017; Hosking, 2017). Intuitively, smells that exist in community members' interactions hinder communication.

Finally, cooperation is compromised by the smells existing in communities' structures. Businesses with inefficient processes and outdated software are also linked with organizational debt (Dignan, 2017). So the analysis shows that "pinpointing" and separating different types of debt, i.e., technical debt and non-technical debt in SD, is challenging as they are greatly linked. All NTD contributes to TD (Tamburri et al., 2015; Martini et al., 2020; Hilty and Aebischer, 2015; Palomba et al., 2018; Yli-Huumo et al., 2016) and both cause equal damage. It is also evident that one type of NTD causes another type of NTD, i.e. culture debt and organization debt can lead to process debt(Martini and Bosch, 2017; Martini et al., 2020), people debt can lead to organizational debt, and culture debt(Marlow, 2017; Falchuk, 2019), and social debt can lead to people debt (Chen, 2022).

We found a pattern in mitigation strategies for identified NTD types; for instance, many problems can be handled with communication, collaboration and coordination (i.e., the 3C model)(Fuks et al., 2008). However, Further investigation is needed for effective NTD handling strategies beyond communication and collaboration.

6 VALIDITY THREATS AND LIMITATIONS

We follow validity threat framework presented by (Dreesen et al., 2021). We are addressing threats to external validity, threats to construct validity, and threats to conclusion validity. Construct validity refers to obtaining the right measures for the studied concept (Dreesen et al., 2021). To minimize this threat, a data collection process was designed to support data recording. Two other researchers were also involved in the whole process, which helped to lessen this threat even more. External validity concerns the extent to which the study results are generalizable (Dreesen et al., 2021). Because our MLR primary studies were largely based on online sources (grey literature), their applicability to the broader area of practices and general disciplines of TD and NTD is limited. We tried to minimise external validity threat by following guidelines proposed by (Garousi et al., 2019). The conclusion validity of the research is a concern as it is related to researchers' bias or misinterpretation of data. To minimize this risk, the authors included two researchers in the analysis process and performed a full audit and trial of 40 sources. However, as it is an MLR, most of the studies are not subject to peer review, and results may be anecdotal and lack research rigor. Excluding all records not written in English may also limit the study as relevant literature in other languages may be missed.

7 FUTURE WORK

Future studies can expand on examining the relationship between social and people's debt with various theories, such as social capital, psychological safety, or control theory. Research could also focus on developing a classification system of NTDs, the consequences of NTDs on software development projects(Ahmad and Gustavsson, 2022), and exploring software ecological debt, known as "greenability". Future studies can also aim to promote human-friendly software development practices by considering software usability engineering techniques to study software usability debt(Saeeda et al., 2018). We also recommend expanding the investigation of NTD to software development challenges in large-scale agile projects(Saeeda et al., 2015).

ACKNOWLEDGMENTS

This work was supported by NODLA Project funded by the Knowledge Foundation Sweden.

REFERENCES

Ahmad, M. O. and Gustavsson, T. (2022). The pandora's box of social, process, and people debts in software engineering. *Journal of Software: Evolution and Process*, page e2516.

Andreou, A. S. (2003). Promoting software quality through a human, social and organisational requirements elicitation process. *Requirements Engineering*, 8(2):85–101.

Bellotti, M. (2021). Hunting tech debt via org charts.

Bird, C., Nagappan, N., Gall, H., Murphy, B., and Devanbu, P. (2009). Putting it all together: Using socio-technical networks to predict failures. In *2009 20th International Symposium on Software Reliability Engineering*, pages 109–119. IEEE.

Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

Casey, K. (2020). What causes technical debt – and how to minimize it.

Cataldo, M., Mockus, A., Roberts, J. A., and Herbsleb, J. D. (2009). Software dependencies, work dependencies, and their impact on failures. *IEEE Transactions on Software Engineering*, 35(6):864–878.

Chen, A. (2022). Cultural debt.

Coleman, B. (2019). Culture debt is one of the most toxic threats to business, and your startup is probably victim to it.

Cusick, J. and Prasad, A. (2006). A practical management and engineering approach to offshore collaboration. *IEEE software*, 23(5):20–29.

Dargó, Z. et al. (2019). Technical debt management: Definition of a technical debt reduction software engineering methodology for smes.

De Souza, C. R. and Redmiles, D. F. (2011). The awareness network, to whom should i display my actions? and, whose actions should i monitor? *IEEE Transactions on Software Engineering*, 37(3):325–340.

Dignan, A. (2017). How to eliminate organizational debt – building strong organizations.

Dreesen, T., Hennel, P., Rosenkranz, C., and Kude, T. (2021). "the second vice is lying, the first is running into debt." antecedents and mitigating practices of social debt: An exploratory study in distributed software development teams. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 6826.

Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9-10):833–859.

Eaden., M. (2017). When testers deal with process debt: Ideas to manage it and get.

Espinosa, E. A. C. (2021). *Understanding Social Debt in Software Engineering*. PhD thesis, The University of Alabama.

Falchuk, B. (2019). What's the greatest threat to your organization?" culture debt.

Fuks, H., Raposo, A., Gerosa, M. A., Pimentel, M., Filippo, D., and Lucena, C. (2008). Inter-and intra-relationships between communication coordination and cooperation in the scope of the 3c collaboration model. In *2008 12th International Conference on Computer Supported Cooperative Work in Design*, pages 148–153. IEEE.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106:101–121.

Hilty, L. M. and Aebischer, B. (2015). Ict for sustainability: An emerging research field. *ICT innovations for Sustainability*, pages 3–36.

Holvitie, J., Licorish, S. A., Spínola, R. O., Hyrynsalmi, S., MacDonell, S. G., Mendes, T. S., Buchan, J., and Leppänen, V. (2018). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, 96:141–160.

Hosking, M. (2017). Transformation troubles and non-technical debt.

Hosking, M. (2021). "cultural debt" hurting your organization's growth?

Irwin, K. and Adrian, S. (2011). Does socio-technical congruence have an effect on software build success? a study of coordination in a software project. *IEEE Transactions on Software Engineering*, 37(7):307–324.

Jaktman, C. B. (1998). The influence of organisational factors on the success and quality of a product-line architecture. In *Proceedings 1998 Australian Software Engineering Conference (Cat. No. 98EX233)*, pages 2–11. IEEE.

Kavas, I. (2021). Don't go back to the office without fixing your process debt.

Kazman, R. (2019). Managing social debt in large software projects. In *2019 IEEE/ACM 7th International Workshop on Software Engineering for Systems-of-Systems (SESoS) and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES)*, pages 1–1. IEEE.

Kerv (2022). Cloudthing — organisational debt & why it makes digital transformation hard-cloudthing.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15.

Klinger, T., Tarr, P., Wagstrom, P., and Williams, C. (2011). An enterprise perspective on technical debt. In *Proceedings of the 2nd Workshop on managing technical debt*, pages 35–38.

Kruchten, P., Nord, R. L., and Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *Ieee software*, 29(6):18–21.

Ladewig, S. (2019). The dark side of working from home — the startup.

Li, Z., Avgeriou, P., and Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220.

Marlow, G. (2017). People debt is like technical debt.

Martini, A., Besker, T., and Bosch, J. (2020). Process debt: a first exploration. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 316–325. IEEE.

Martini, A. and Bosch, J. (2015). The danger of architectural technical debt: Contagious debt and vicious circles. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*, pages 1–10. IEEE.

Martini, A. and Bosch, J. (2017). Revealing social debt with the caffea framework: An antidote to architectural debt. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 179–181. IEEE.

Martini, A., Stray, V., and Moe, N. B. (2019). Technical-, social-and process debt in large-scale agile: an exploratory case-study. In *International Conference on Agile Software Development*, pages 112–119. Springer.

Melo, A., Fagundes, R., Lenarduzzi, V., and Santos, W. (2021). Identification and measurement of technical debt requirements in software development: a systematic literature review. *arXiv preprint arXiv:2105.14232*.

Nagappan, N., Murphy, B., and Basili, V. (2008). The influence of organizational structure on software quality. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 521–530. IEEE.

Ozkaya, A. P. K. P. (2016). I seaman c. *Managing technical debt in software engineering Dagstuhl Rep*, 6(4):110.

O'Keeffe, D. (2017). An empirical case study of technical debt management: A software services provider perspective.

Palomba, F., Tamburri, D. A., Fontana, F. A., Oliveto, R., Zaidman, A., and Serebrenik, A. (2018). Beyond technical aspects: How do community smells influence the intensity of code smells? *IEEE transactions on software engineering*, 47(1):108–129.

Rios, N., Spínola, R. O., Mendonça, M., and Seaman, C. (2018). The most common causes and effects of technical debt: first results from a global family of industrial surveys. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10.

Saeeda, H., Arif, F., and Minhas, N. M. (2018). Usability software engineering testing experimentation for android-based web applications: usability engineering testing for online learning management system. In *Application Development and Design: Concepts, Methodologies, Tools, and Applications*, pages 397–415. IGI Global.

Saeeda, H., Arif, F., Minhas, N. M., and Humayun, M. (2015). Agile scalability for large scale projects: Lessons learned. *J. Softw.*, 10(7):893–903.

Storey, M.-A., Ernst, N. A., Williams, C., and Kalliamvakou, E. (2020). The who, what, how of software engineering research: a socio-technical framework. *Empirical Software Engineering*, 25(5):4097–4129.

Sutton, B. (2019). Overcoming cultural and technical debt.

Tamburri, D. A. (2019). Software architecture social debt: Managing the incommunicability factor. *IEEE Transactions on Computational Social Systems*, 6(1):20–37.

Tamburri, D. A. and Di Nitto, E. (2015). When software architecture leads to social debt. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*, pages 61–64. IEEE.

Tamburri, D. A., Kruchten, P., Lago, P., and van Vliet, H. (2013). What is social debt in software engineering? In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 93–96. IEEE.

Tamburri, D. A., Kruchten, P., Lago, P., and Vliet, H. v. (2015). Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*, 6(1):1–17.

Tonin, G. S. (2018). *Technical debt management in the context of agile methods in software development*. PhD thesis, PhD thesis, University of Sao Paulo.

Vinsennau, S. (2016). Decouple to innovate how federal agencies can unlock it value & agility by remediating technical debt.

Wenger E, McDermott RA, S. W. (2002). Cultivating communities of practice: a guide to managing knowledge. harvard business school publishing.

Winter, S., Berente, N., Howison, J., and Butler, B. (2014). Beyond the organizational 'container': Conceptualizing 21st century sociotechnical work. *Information and Organization*, 24(4):250–269.

Yli-Huumo, J. et al. (2017). The role of technical debt in software development.

Yli-Huumo, J., Maglyas, A., and Smolander, K. (2016). The effects of software process evolution to technical debt—perceptions from three large software projects. In *Managing Software Process Evolution*, pages 305–327. Springer.

Appendix A: Primary Studies (PS) List

[PS1]	Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016). How do software development teams manage technical debt? – An empirical study. Journal of Systems and Software, 120, 195–218. https://doi.org/10.1016/J.JSS.2016.05.018
[PS2]	Martini, A., & Bosch, J. (2017). Revealing social debt with the CAFFEA framework: An antidote to architectural debt. Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings, 179–181. https://doi.org/10.1109/ICSAW.2017.42
[PS3]	Kyle Ladewig. (2019). Social Debt. The dark side of working from home — The Startup — Medium. https://medium.com/swlh/social-debt-17bf03a269a
[PS4]	Cultural Debt. (n.d.). Retrieved May 1, 2022, from https://www.careerfair.io/reviews/cultural-debt
[PS5]	Avgeriou, P., Kruchten, P., Ozkaya, I., & Seaman, C. (2016). Managing Technical Debt in Software Engineering 16162. https://doi.org/10.4230/DagRep.6.4.110
[PS6]	Melo, A., Fagundes, R., Lenarduzzi, V., & Santos, W. (2021). Identification and Measurement of Technical Debt Requirements in Software Development: a Systematic Literature Review. https://doi.org/https://doi.org/10.48550/arXiv.2105.14232
[PS7]	Tamburri, D. A. (2019). Software Architecture Social Debt: Managing the Incommunicability Factor. IEEE Transactions on Computational Social Systems, 6(1), 20–37. https://doi.org/10.1109/TCSS.2018.2886433
[PS8]	Thelma Mejía. (2016). Social Debt: the difficult commitment. https://www.socialwatch.org/book/export/html/10623
[PS9]	Accenture. (2016). DECOUPLE TO INNOVATE How federal agencies can unlock IT value & agility by remediating technical debt. https://www.accenture.com/_acnmedia/PDF-85/Accenture-Decoupling-to-Innovate.pdf
[PS10]	Tonin, G. S. (2018). Technical debt management in the context of agile methods in software development. PhD Thesis. Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brazil. DOI: https://doi.org/10.11606/T.45.2018.TDE-30072018-142720
[PS11]	Besker, T., Ghanbari, H., Martini, A., & Bosch, J. (2020). The influence of Technical Debt on software developer morale. Journal of Systems and Software, 167, 110586. https://doi.org/10.1016/J.JSS.2020.110586
[PS12]	Marianne Bellotti. (2021, December 20). Hunting Tech Debt via Org Charts. Knowing where to look for problems. https://bellmar.medium.com/hunting-tech-debt-via-org-charts-92df0b253145
[PS13]	Yli-Huumo. (2017). THE ROLE OF TECHNICAL DEBT IN SOFTWARE DEVELOPMENT. Ph.D. Thesis. Lappeenranta University of Technology, Lappeenranta, Finland. https://lutpub.lut.fi/bitstream/handle/10024/136260/Jesse%20Yli-Huumo%20A4.pdf?sequence=2
[PS14]	Steve Priestnall. (2020, November 19). What is Process Debt? — LinkedIn. https://www.linkedin.com/pulse/what-process-debt-steve-priestnall/
[PS15]	Dargó, Z. (2019). Technical debt management: Definition of a technical debt reduction software engineering methodology for SMEs. School of Science, Aalto University, Espoo, Finland
[PS16]	McKeeby. (2017). Collaboration Strategies to Reduce Technical Debt. School of sciences, Walden University, USA.
[PS17]	MITSloan. (2019). Overcoming cultural and technical debt. Retrieved April 21, 2022, from https://sloanreview.mit.edu/audio/overcoming-cultural-and- technical-debt/.
[PS18]	Tamburri, D. A., Kruchten, P., Lago, P., & van Vliet, H. (2013). What is social debt in software engineering? 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013 - Proceedings, 93–96. https://doi.org/10.1109/CHASE.2013.6614739
[PS19]	Tamburri, D. A., Kruchten, P., Lago, P., & Vliet, H. van. (2015). Social debt in software engineering: insights from industry. Journal of Internet Services and Applications, 6(1). https://doi.org/10.1026/S13174-015-0024-6
[PS20]	O'keeffe , (2017). An Empirical Case Study of Technical Debt Management: A Software Services Provider Perspective. Masters Thesis. Dublin University, https://www.scss.tcd.ie/publications/theses/diss/2017/TCD-SCSS-DISSERTATION-2017-054.pdf.
[PS21]	Bernard Coleman. (2019). Culture Debt Is One of the Most Toxic Threats to Business, and Your Startup Is Probably Victim to It — Inc.com. https://www.inc.com/bernard-coleman/culture-debt-is-one-of-most-toxic-threats-to-business-your-startup-is-probably-victim-to-it.html

[PS22] Aaron Dignan. (2017, March 15). How to Eliminate Organizational Debt – BUILDING STRONG ORGANIZATIONS. https://culturestars.com/how-to-eliminate-organizational-debt/

[PS23] Tamburri, D. A., & Nitto, E. di. (2015). When Software Architecture Leads to Social Debt. Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015, 61–64. https://doi.org/10.1109/WICSA.2015.16

[PS24] Martini, A., Stray, V., Moe, N.B. (2019). Technical-, Social- and Process Debt in Large-Scale Agile: An Exploratory Case-Study. In: Hoda, R. (eds.) Agile Processes in Software Engineering and Extreme Programming - Workshops. XP 2019 (pp. 112-119). Lecture Notes in Business Information Processing, vol 364. Springer, Cham. https://doi.org/10.1007/978-3-030-30126-2_14

[PS25] Martini, A., Besker, T., & Bosch, J. (2020). Process Debt: a First Exploration. Process Debt: A First Exploration, 2020-December, 316–325. https://doi.org/10.1109/APSEC51365.2020.00040

[PS26] Sebastian, Z. (2019). Social Debt: Why Software Developers Should Think Beyond Tech. https://sebastianzimmeck.medium.com/social-debt-why-software- developers-should-think-beyond-tech-df665d8401a5

[PS27] Peter Phelan. (2021). Is “Cultural Debt” hurting your organization’s growth? (Part 1) — 8W8 - Global Business Builders. https://www.8w8.com/is-cultural-debt-hurting-your-organizations-growth-part-1/

[PS28] Jaap Trouw. (2021, September 23). Organisational debt, an analogy. https://www.linkedin.com/pulse/organisational-debt-analogy-jaap-trouw/

[PS29] Damian Tamburri. (2015, January 15). From Technical to Social Debt: Analyzing Software Development Community https://www.slideshare.net/DamianTamburri/from-technical-to-social-debt-analyzing-software-development-communities-using-socialnetworks-analysis

[PS30] Damian Tamburri, B. C. ,Steven F. (2016). Social Debt in Software Engineering: Towards a Crisper Definition - UTU Research Portal - UTU Research Portal (Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Ed.). https://research.utu.fi/converis/portal/detail/Publication/18409819;jsessionid=8cdfd1a45b330f5 8497363848861

[PS31] Kazman, R. (2019). Managing Social Debt in Large Software Projects. 1–1. https://doi.org/10.1109/SESOS/WDES.2019.00008

[PS32] Dreesen, T., Hennel, P., Rosenkranz, C., & Kude, T. (2021). ”The second vice is lying; the first is running into debt.” Antecedents and mitigating practices of social debt: An exploratory study in distributed software development teams. Proceedings of the Annual Hawaii International Conference on System Sciences, 2020-January, 6826–6835. https://doi.org/10.24251/HICSS.2021.818

[PS33] Palomba, F., Serebrenik, A., & Zaidman, A. (2017). Social debt analytics for improving the management of software evolution tasks. In S. Demeyer, A. Parsai, G. Laghari, & B. van Bladel (Eds.), BENEVOL 2017 : BElgian-NEtherlands Software eVOLution Symposium, 4-5 December 2017, Antwerp, Belgium (pp. 18-21). (CEUR Workshop Proceedings; Vol. 2047). CEUR-WS.org. http://ceur-ws.org/Vol2047/BENEVOL_2017_paper_5.pdf

[PS34] Melissa Eaden. (2017). When Testers Deal With Process Debt: Ideas to Manage It And Get — MoT. https://www.ministryoftesting.com/dojo/lessons/when-testers-deal-with-process-debt-ideas-to-manage-it-and-get-back-to-testing-faster

[PS35] Eduardo Anel Caballero Espinosa. (2021). Understanding social debt in software engineering. Ph.D thesis. department of computer science in the graduate school of the University of Alabama http://ir.ua.edu/handle/123456789/8278

[PS36] Eduardo Anel Caballero Espinosa. (2021). Understanding social debt in software engineering. Ph.D thesis. department of computer science in the graduate school of the University of Alabama http://ir.ua.edu/handle/123456789/8278

[PS37] Ike Kavas. (2021, January 4). Don’t Go Back To The Office Without Fixing Your Process Debt. https://www.forbes.com/sites/forbestechcouncil/2021/01/04/dont-go-back-to-the-office-without-fixing-your-process-debt/?sh=792a028f74a4

[PS38] Anonymous. (2022). CloudThing — Organisational Debt & Why It Makes Digital Transformation Hard - CloudThing. https://cloudthing.com/organisational-debt/

[PS39] Almarimi, N., Ouni, A., & Mkaouer, M. W. (2020). Learning to detect community smells in open-source software projects. Knowledge-Based Systems, 204, 106201. https://doi.org/10.1016/J.KNOSYS.2020.106201

[PS40] Matt Hosking. (2017, November 15). Transformation troubles and non-technical debt. https://www.linkedin.com/pulse/transformation-troubles-non-technical-debt-matt-hosking/