

A Multivocal Literature Review on Non-Technical Debt in Software Development: An Insight into Process, Social, People, Organizational, and Culture Debt

Hina Saeeda*, Muhammad Ovais Ahmad*, Tomas Gustavsson**

**Department of Computer Science, Karlstad University, Sweden*

***Business School, Karlstad University, Sweden*

Hina.saeeda@kau.se, Ovais.Ahmad@kau.se, Tomas.gustavsson@kau.se

Abstract

Software development encompasses various factors beyond technical considerations. Neglecting non-technical elements like individuals, processes, culture, and social and organizational aspects can lead to debt-like characteristics that demand attention. Therefore, we introduce the non-technical debt (NTD) concept to encompass and explore these aspects. This indicates the applicability of the debt analogy to non-technical facets of software development. Technical debt (TD) and NTD share similarities and often arise from risky decision-making processes, impacting both software development professionals and software quality. Overlooking either type of debt can lead to significant implications for software development success. The current study conducts a comprehensive multivocal literature review (MLR) to explore the most recent research on NTD, its causes, and potential mitigation strategies. For analysis, we carefully selected 40 primary studies among 110 records published until October 1, 2022. The study investigates the factors contributing to the accumulation of NTD in software development and proposes strategies to alleviate the adverse effects associated with it. This MLR offers a contemporary overview and identifies prospects for further investigation, making a valuable contribution to the field. The findings of this research highlight that NTD's impacts extend beyond monetary aspects, setting it apart from TD. Furthermore, the findings reveal that rectifying NTD is more challenging than addressing TD, and its consequences contribute to the accumulation of TD. To avert software project failures, a comprehensive approach that addresses NTD and TD concurrently is crucial. Effective communication and coordination play a vital role in mitigating NTD, and the study proposes utilizing the 3C model as a recommended framework to tackle NTD concerns.

Keywords: systematic reviews and mapping studies, software quality

1. Introduction

Software development is inherently a sociotechnical process, where the successful completion of software projects relies on the symbiotic relationship between technical capabilities and non-technical aspects of software development [1, 2]. This includes considering social aspects, as defects in software often arise from cognitive errors and miscommunication within and outside of organizations [3]. Such defective software leads to the accumulation

of technical debt and additional maintenance costs [4]. In software engineering, the term “technical debt” metaphorically describes the consequences of rushing software project development, resulting in defects and costly maintenance [5].

Over the past decade, both academia and industry have shown great interest in technical debt (TD) [3, 6], exploring various dimensions such as TD effort [7], TD tools [8], TD management strategies [5, 9], managing architectural TD [10], TD in Agile development [11], TD management elements [12], and TD prioritization [13]. Surprisingly, non-technical aspects also contribute to TD, as non-technical stakeholders play a role in driving projects to acquire TD [14]. Therefore in SD, the debt metaphor is used to describe issues prevalently – technical debt (i.e., code debt and code smells) and other debts (i.e., process debt, social debt, people debt, organisational, and cultural debt) [7, 10, 11, 13]. Software projects’ success or failure combines technical and non-technical elements [11, 12]. Despite the significant attention given to TD in software [5–7, 10, 12, 14, 15], there remains a substantial research gap concerning the study of non-technical debt (NTD) [3, 7, 11, 13, 16–18]. NTD, such as process debt [7], people debt [16, 17], social debt [3, 11, 18–20], organizational debt [21], and cultural debt [13, 22] have not received sufficient consideration within the technical debt domain. A 2022 systematic mapping review identified a scarcity of scientific studies on NTD [1]. The review reported only 17 scientific studies on NTD and requested further empirical investigation as well as other forms of literature reviews to cover NTD in SD.

Therefore, this study aims to investigate NTD by conducting a multi-vocal literature review. According to Garousi et al. [23], when there is an absence of scientific evidence on any topic, it is recommended to conduct a multivocal literature review, as grey literature can provide valuable insights, perspectives, and empirical evidence that may not be available through traditional peer-reviewed sources. Including grey literature can lead to a more comprehensive understanding of the research topic. This multi-vocal literature review (MLR) provides a state-of-the-art of various NTD types, their causes, and mitigation strategies. The review extends and cross-validates the findings of a previously conducted systematic mapping review [1], enhancing the strength of the research outcomes by investigating similar research questions from different perspectives. By including scientific and grey literature, this review offers additional insights by capturing diverse perspectives and theoretical and practical insights. To achieve our study goals, we are investigating the research questions designed and reported in [1] that serve as the guided foundation for this MLR.

- RQ1 – What is the current state-of-the-art research on the different NTD types in software engineering?
- RQ2 – What are the reported causes of NTD in software engineering?
- RQ3 – What are the reported NTD mitigating strategies?
- RQ4 – What are the possible future directions for NTD in software development?

The present study is built upon previous research (doi.org/10.5220/0011772300003464) on the topic of NTD in SD. In this current version of our work, we aim to expand the scope of the prior investigation by offering a more comprehensive analysis. Specifically, we provide a detailed thematic division of the identified instances of NTD, including their underlying causes and potential solutions. Moreover, we conduct a thorough comparative analysis with a study referenced as [1]. The purpose of this comparison is to demonstrate how our research replicates, extends, and validates the existing body of work in this domain.

The rest of the study is structured as follows: The design and methodology of our investigation used in the research are explained in Section 2. Section 3 discusses the findings, and Section 4 is based on a discussion and conclusion. Section 5 examines threats to validity and how they were resolved. Finally, Section 6 represents future work.

2. Research method

This 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. [23]. The complete systematic MLR process is illustrated in Figure 1, which consists of three phases: planning, conducting, and reporting. Each phase of the MLR study is discussed in detail in the rest of the section. Our MLR search was conducted on October 10, 2022, and analysis and reporting were completed by December 2022.

2.1. Planning the MLR

The primary purpose of conducting an MLR study is to focus on the “classification and thematic analysis of both scientific and grey literature on a software engineering topic” [23]. In this context, Garousi et al. [23] compared a systematic and a multivocal literature review study. A typical systematic literature review is motivated by a specific research topic that may be answered empirically. On the other hand, multivocal literature research examines a larger spectrum of software engineering challenges using grey and scientific literature. The following two processes (i.e., Motivation and Objectives and Research Questions) comprise the MLR planning phase, as depicted in Figure 1.

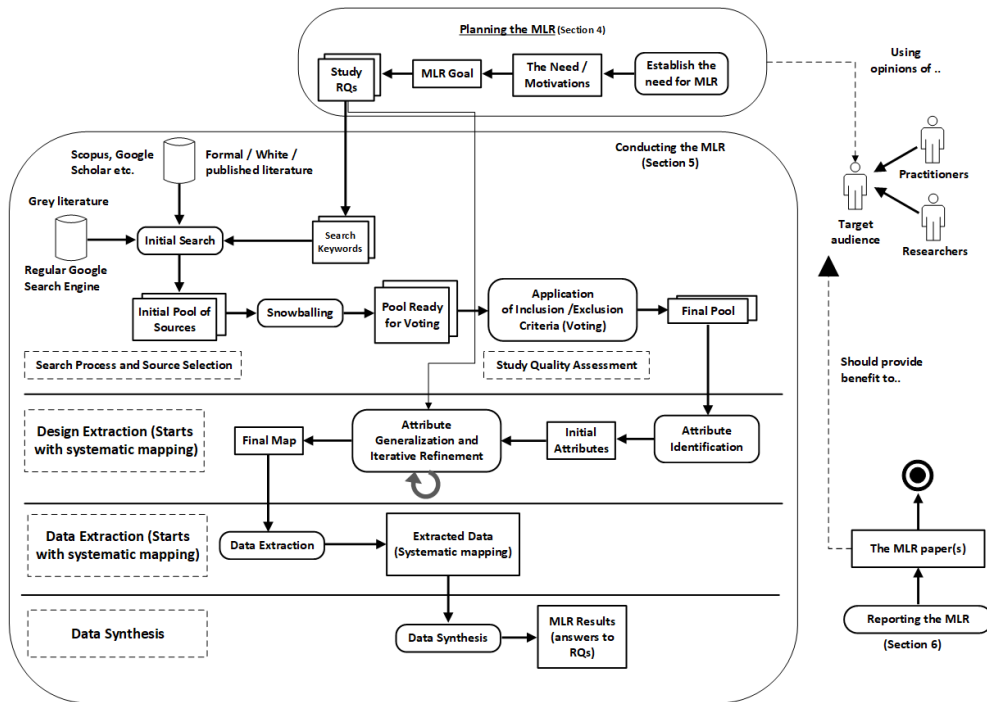


Figure 1. Complete MLR process

Motivation

MLR is useful in finding what is happening in an under-discovered phenomenon. MLR incorporates all accessible literature (including, but not limited to: blogs, white papers, articles, and academic literature) [23]. Therefore, MLR is vital for expanding research by

including non-scientific (grey) literature normally excluded in scientific studies. Academics and industry have been interested in TD for the last twenty years. Whereas the NTD is still in its infancy, it is necessary to investigate this niche and important research area further. In this case, an MLR study is motivated to learn about different types of NTD in software engineering and why they happen. What types of prevention or mitigation strategies may be utilized to prevent them?

The main objectives of this study are to understand: 1. What NTD is and how it can impact software engineering projects. 2. Identify the main types of NTD in software engineering projects. 3. Identify the main causes of NTD in software engineering projects. 4. Determine how to prevent or mitigate NTD in software engineering projects. 5. Determine the possibilities for future NTD research. These questions serve as the foundation for this study.

2.2. Conducting the MLR

At first, we conducted a pilot search on debt in software engineering using Google Scholar. The aim was to determine the existence of any secondary studies on the given topic. We conducted this search using the following string: (“process debt” OR “social debt” OR “people debt” OR “organizational debt” OR “culture debt”) AND (“Software”). The search was conducted in October 2022, and Google Scholar yielded 3080 results. It was evident from the search that the current focus of research is predominantly on TD, whereas not a single review was found on NTD.

2.2.1. Search strategies and data sources

The designed search string defined the scope of our study. The designed string includes the search terms “population” and “intervention” based on (PICO) criteria suggested by Kitchenham et al. [24], 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 designed search string was (“process debt” OR “social debt” OR “people debt” OR “organizational debt” OR “culture debt”) AND (“software”).

The rationale for using the term “software” is that this study will cover studies that discuss software, software development, and systems. So, the search will include all documents with the word “software” in the title, abstract, and keyword. At the same time, the terms process debt, people debt, social debt, and organizational debt were used to include all NTD-associated sources. While we selected NTD (process, people, and social debts) from the existing systematic mapping review [1] on the topic and extended the scope of the study by adding two NTDs (cultural and organizational debts) as well. The overall motivation for selecting these five NTD types is based on the proposed “Hexagonal socio-technical systems framework” (adapted from Davis et al. 2014) [25] where they highlighted people, process, culture, and organization (technology and infrastructure) elements as the most critical sociotechnical aspects. As software development is an extensive interactive activity, we added the social debt [3] to cover the communication, collaboration, and cooperation challenges among people (directly or indirectly linked with the software development process). To ensure a broad overview of the topic, the selection criteria for including papers

in our study focus more on relevance. According to Lenarduzzi et al. [26], there are other types of debts, including service debt, suitability debt, and environmental sustainability debt, that can be studied under the umbrella of the NTD. However, in the current study, we are restricting our scope to only include the strongly linked social-technical elements of the software development process. Restricting the scope of our study has positive effects by enabling focused exploration, increased rigor, and efficient resource allocation. It helped us to concentrate on important research elements, leading to comprehensive findings. However, there are drawbacks to consider, including limited generalizability and potential oversight of relevant aspects.

The search string was designed to retrieve results from the Google search engine. We preferred the Google search engine as it is faster and a good source for collecting grey literature. We aimed to keep the search string simple to be as inclusive as possible with a new topic like non-technical debt in software engineering. Therefore, we did not restrict the search to particular years. We found 110 results with 11 Google pages, each with 11 links to further resources. Figure 2 shows the complete research conduction phases.

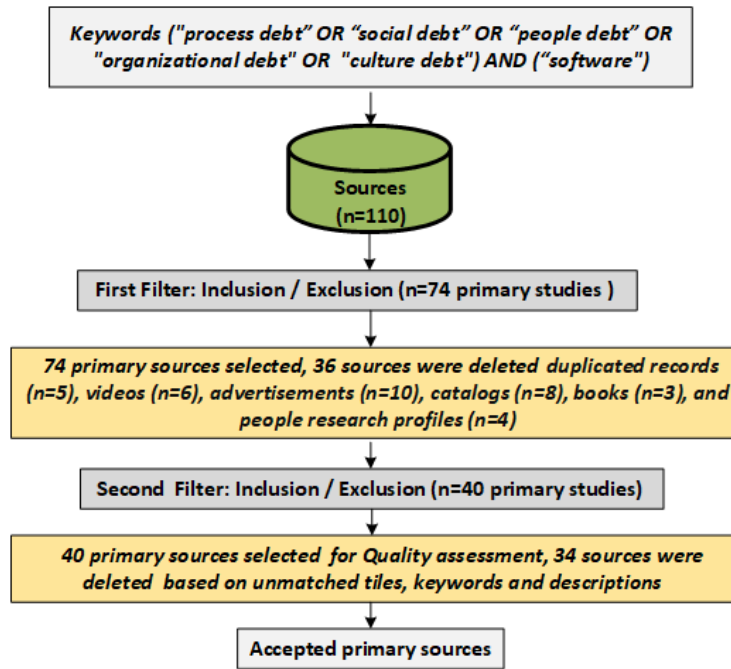


Figure 2. Research conduction process

2.2.2. Primary studies selection procedure and application of inclusion/exclusion criteria

In the first round, we excluded 36 records out of 110 based on the exclusion criteria given in Table 1. The detailed breakdown of excluded records is duplicated records ($n = 6$), videos ($n = 7$), advertisements ($n = 11$), catalogs ($n = 8$), and people research profiles ($n = 4$). After the first round, we obtained 74 records out of 110 and included them in subsequent steps.

Table 1. Inclusion and exclusion criteria

Inclusion	Exclusion
Sources contributing to the understanding of non-technical debt in software development, no matter to what extent the topic is discussed	Sources not written in English
Empirical and non-empirical sources, either qualitative or quantitative, analyzing any NTD in SD	Videos, advertisements, catalogs or keynotes, newspapers, duplicate sources
Blogs, white papers, and experiences report “people, process, social, cultural and organizational debt	Position and research profiles, non-software engineering domain sources

2.2.3. Quality assessment

To apply quality assessment criteria, the 40 primary studies [S1–S40] were divided into two categories: grey literature (GL) and scientific literature (SL). We adopted the 11-factor quality assessment criteria (Table 2) proposed by Dybå et al. [27] for scientific literature. At the same time, we adopted the quality assessment checklist of grey literature (Table 3) from Garousi et al. [23]. Each criterion was graded on a binary (“1” or “0”) grade, in which “1” indicates “yes” to the question, while “0” means “no.” Both checklist criteria measured the extent to which the quality of the 12 SL and 28 GL sources could be appropriately assessed. Two research separately consider the 40 primary sources. This technique helps to limit the degree of subjectivity and report the results more objectively. Researchers combined their results and solved a few conflicts in the discussion session.

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?

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 [7] 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 studies. All primary publications adequately detailed “data collecting” and “data analysis”, except [3]. While [7] lacks design and sampling phases. While “research findings” and “research value” criteria were applicable and fulfilled by all papers. Three papers [7, 16, 17] failed to discuss the researcher-participant connection explicitly. None of the papers received a complete score on the quality evaluation, but few publications received two or three negative responses. We divided grey literature into three tiers. We have 9 primary sources in 1st tier GL, which cover high outlet control/high credibility, including thesis, reports, and white papers. Two primary sources come under the umbrella of 2nd tier GL, which covers moderate outlet control/moderate credibility,

Table 3. Quality assessment check list for GL

1.	Is an individual author associated with a reputable organization?
2.	Has the author published other work in the field?
3.	Does the author have expertise in the area? (e.g., job title principal software engineer)
4.	Does the source have a clearly stated aim?
5.	Does the source have a stated methodology?
6.	Do authoritative, documented references support the source?
7.	Does the work cover a specific question?
8.	Does the work refer to a particular population?
9.	Is the work balanced in a presentation?
10.	Is the statement in the sources as objective as possible?
11.	Do the data support the conclusions?
12.	Does the item have a clearly stated date?
13.	Does it enrich or add something unique to the research?
14.	Does it strengthen or refute a current position
15.	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

including Q/A sites, Wiki articles, and workshops. There is 17 3rd-tier GL that cover low outlet control/low credibility. None 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 lack a methodology section and documented references, whereas thesis and seminar reports have written methodology sections. All sources provide information on specific NTD issues, cover the software development population, and are balanced in the overall presentation.

2.2.4. Data extraction and analysis

After completion of the quality analysis, data extraction was performed in MS Excel sheets based on the primary sources types, year of publication, NTD types found, NTD causes, and mitigation strategies. After the data extraction, data analysis was done using thematic analysis techniques [28]. The thematic analysis yielded primarily five themes, each highlighting NTD types. Further codes (discussed in the result section) were created against each NTD type, cause, and mitigation strategy.

3. Results

Our MLR study presents the results from analyzing the 40 primary sources [S1–S40] (see Appendix A). The presented results begin with demographic information, i.e., (i) type of literature, (ii) type of sources, and (iii) publication by year, and then proceed to a detailed assessment based on the thematic analysis.

3.1. Demographics

Figure 3 shows the two broader categories of our primary sources, GL ($n = 28$) and SL ($n = 12$). Figure 4 further shows the detailed breakdown of these two categories. The highest number of resources are cited from web blogs ($n = 17$), and the second highest number of resources are found in journals ($n = 6$), conferences ($n = 6$), and theses ($n = 6$).

Also, seminar reports ($n = 3$) and two workshops are reported on the topic. This clearly shows high practitioners' interest in the topic under study. Figure 5 shows that before 2018 fewer relevant studies were captured, whereas active research efforts have been evident since 2019.

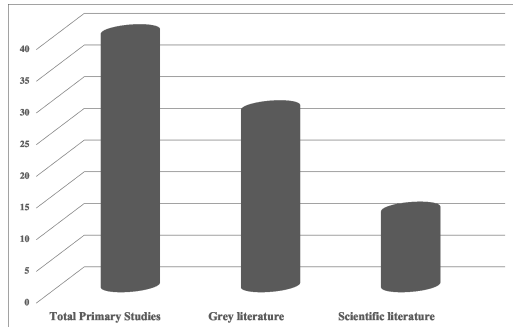


Figure 3. Type of literature

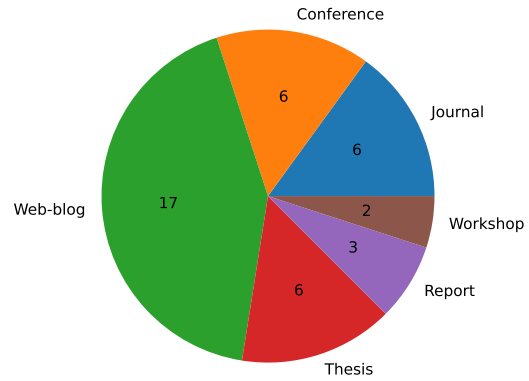


Figure 4. Types of primary sources

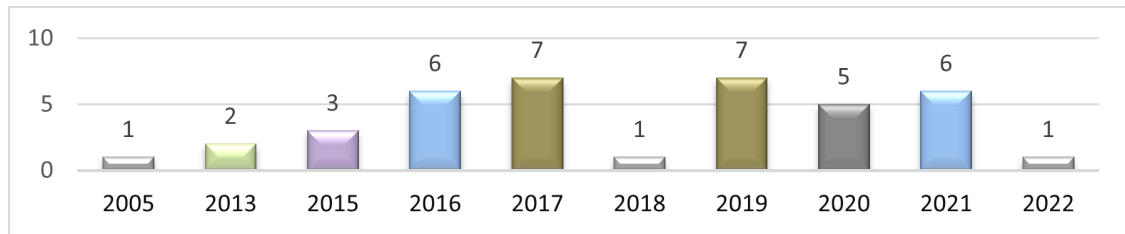


Figure 5. Publication's trend

3.2. NTD state of the art

In this section, we discuss five different types of NTD (i.e., people, process, social, cultural, and organization) with the help of relevant examples. This section answers RQ1 – What is the current state-of-the-art research on the different NTD types in software engineering?

3.2.1. Process debt

“Process debt refers to issues that, if present in software development, might delay or impede specific development operations” [29]. A software process is represented by a series of work phases applied to the design and development of a software product [30]. A process that operated effectively and efficiently a year ago may not be as productive as before, as changes in expectations, people, resources, and tools necessitate the modification of processes to achieve optimal performance. During these changes, the number of process bottlenecks, wasteful stages, and superfluous procedures add to process debt [31]. It occurs when a process is poorly understood and controlled [2]. Likewise, process debts are related to the creation of technical debt [7, 32]. Martini et al. [33] researched process debt in fine detail and revealed a number of its causes and mitigation strategies. An example of process debt is when teams hold stand-up meetings to report status so leaders know what is happening [16]. Team members can quickly show their leader the project's status. The

debate focuses on recording rather than distributing information and addressing team interdependencies, causing process debt [16].

3.2.2. People debt

“People debt refers to people’s problems in a software company that can slow or impede developmental operations” [5]. Human (people) factors in SD can be investigated psychologically, cognitively, managerially, and technically. These factors have organizational, interpersonal, and individual impacts. When building systems, enterprises accumulate people’s debt by ignoring their people’s needs. Unresolved people’s issues [34], such as two people who had to be put on different teams because they could not work with each other, the person promoted whose head’s only just been above water for the last six months is the typical causes of people debt [35–37]. It refers to all the ideas, goals, and objectives a corporation had for employees but abandoned. It also refers to missed opportunities to improve employee lives and jobs. One common example of people debt is expertise concentrated in too few people due to delayed training and/or hiring [2, 12, 16, 20]. Nokia’s sharp fall, unveiled by an INSEAD study [38], resulted from a toxic culture of silence that the employees were experiencing that caused them to be in denial about the progress of their competition. Boeing’s catastrophic failure resulted from an environment of fear, where engineers were unwilling to discuss problems and failures [34]. People’s debt is also directly linked to the creation of TD [39].

3.2.3. Culture debt

“Cultural debt is making a technical decision that borrows against the organization’s culture. Such decisions can introduce team divisions, deteriorate communication or even weaken leadership effectiveness” [13, 22]. Corroding culture affects morale and alienates partners, customers, and employees. The biggest threat to a company’s ability to seize an opportunity isn’t the wrong people, product, pricing, competition, or market forces; it’s Culture Debt [35, 40]. Leaders often say, “Their employees are their greatest asset.” If the culture is correct, if it is shattered and you hire out of step with it, everything will fail. These “biggest assets” will make unwise decisions and fail. The right people in the right atmosphere will always win [21, 40]. Uber had cultural debt as before 2017, the company’s services, such as ordering a car, tracking a vehicle, and driver software, were packaged into one software. When part of the software crashed, the whole system went down [41]. The company switched software to recover swiftly and stabilize the business. The corporation bought 30 000 decentralized programs, which caused the technical debt. To cover the technical debt, they employed inexperienced workers without much training. The company’s cultural debt produced decentralized communication, inadequate leadership, and disorganization.

3.2.4. Social debt

“Social debt is a cumulative and increasing cost in the current state of things, connected to invisible and negative effects within a development community” [16]. Tamburri et al. [3, 11, 18, 20] investigate social aspects of debt under the term “social debt.” It is associated with unforeseen project costs, similar to technical debt, as it depicts the cost accumulation of software projects due to community causes, i.e., suboptimal working environments

and conditions [42, 43]. Community causes contribute to the accumulation of social debt, which impacts the people who work on software development tasks and the quality of the software they produce [3, 27]. This chain of events can potentially jeopardize the business continuity of software development organizations. An example of social debt is a lack of effective communication between different parts of the organization (for example, between development and operations teams). Another example has an architecture team that is disconnected from the SD team and, therefore, might suggest architectural solutions that are not realistic, as they do not take into consideration the details and the requirements elicited during the implementation phase [3, 36].

3.2.5. Organisational debt

Organizational debt is “the accumulation of changes that leaders should have made but didn’t” [36]. It turns out that organizational debt can kill the company faster than other debts [42]. Organizational debt is all the people/culture compromises made to “just get it done” in the early stages of a startup. Growing companies must understand how to recognize and “refactor” organizational debt [27, 43, 44]. Organizational change requires individuals to change, whether introducing a new tool, restructuring business processes, or even larger transformation. These changes add to organizational debt as well [44]. The common organizational debt example is doing what’s quick and convenient. It is understood that it does not scale and is not the ultimate solution, but it works now [5]. Further organizational debt example is obsolete processes. The company implemented a solution that worked at the time but now has better, more efficient options, but systems and processes rely on the old ways. Organizational debt is also directly linked to the generation of TD as well [34, 38, 45].

3.3. NTD accumulation causes

This section presents different identified common causes for accumulating the NTD. This section answers RQ2 – What are the reported causes of NTD in software engineering?

3.3.1. Process debt causes

Our study identified 14 process debt causes that are divided into three themes named process divergence, organizational and external dependencies (see Table 4). Among the identified causes, six were found to be discussed in both GL and SL, which include inefficient processes, outdated processes, sub-optimal processes, power distance, shortcuts, and quick fix norms prevalent in software companies. On the other hand, the GL introduced three new causes: lack of software culture, external trends, and technology and tools.

Process divergence. According to Martini et al. [7, 33], process designers develop defective processes, process executors deviate from well-designed processes, and infrastructure flaws can cause problems in process implementations. Process divergence is hard and can lead to inadequate planning, prioritization, and incompetency. Further causes of process debt are obsolete and suboptimal processes and lack of follow-up assessment, design, management, and execution [6, 7]. When process experts optimize processes, they add to process debt by missing some important steps. Unaccustomed staff causes confusion and process debt, too [7]. Inefficient processes include wasted time, customer delays, approval waits, batching delays, redundant steps, effort duplication, errors, and rework [15, 46, 48]. Obsolete or

Table 4. Process debt causes

Process debt themes	Causes	GL ref.	SL ref.
Process divergence	Process incompetence		[33]
	Inefficient processes	[46]	[15]
	Outdate process	[29]	[47]
	Changing process	[48]	[20]
	Sub-optimal processes	[6]	[7]
	Lack of follow-up assessment		
	Lack of prioritization		
	Cost of process changes		
Organizational challenges	Value neglecting		
	Lack of software culture	[12]	
	Power distance	[49]	[50]
	Shortcuts and quick fix norms	[45]	[48]
External dependencies	Technology and tools		[7]
	External trends		

old processes include using outdated or time-consuming manual processes when a simple technological replacement could save time and costs [46, 48]. Inadequate defect analysis, documentation, or test case management causes process debt. No one uses more efficient methods when shortcuts and quick solutions are the norms. This diminishes productivity when employees repeat the same “shortcut” when a permanent, more efficient approach might have been established from the start. Changing a software process changes activity, artifact, and role. One element’s change may affect others due to interdependencies. Changing an activity (e.g., adopting agile) can influence final production. Unanalyzed changes in the processes can impair development [7, 47]. The value a process brings to stakeholders and the organization is sometimes unclear. Stakeholders disregard the process when such messages aren’t properly communicated, causing process debt [6, 48]. Existing processes are hampered by different enterprises, units, teams, domains, and events. If these conditions and circumstances aren’t considered, process debt and costly effects arise [7, 47]. Understanding contexts and scenarios in process design are vital; for example, when processes are built just for the software development team and neglect partnering hardware teams, it causes misunderstandings and generates process debt.

Organisational challenges. Power distance, lacking software culture, and neglecting values add to the accumulation of process debt. Power distance refers to the degree to which lower-ranking members of an organization “accept and expect the unequal distribution of authority”. In the context of software development is the perceived distance between less powerful teammates and power-holder teammates, such as experienced teammates or decision-makers [12, 50]. The absence of a software culture in many organizations is a significant problem. This indicates that processes carefully crafted for use in other contexts may not be the most effective in all software development environments [7, 29]. Another issue is interacting with organizations with different cultures, interests, and power. It can lead to the negligence of important values; examples are processes not followed by open-source organizations developing a software system component used by the development team (e.g., the lack of a correct library versioning) or other stakeholders that may be interested in receiving data to compute analytics without knowing about the burden for the developers [49].

External dependencies. The influence of external trends, technology, and tools directly affects debt accumulation. How an organization adopts new processes is influenced by

external trends such as greater global competition, changing demographics, changing customer concerns, and volatile stock markets. Process debt can occur when an organization adopts processes that are not fit for the organization based on these external trends. Having the technology to support the processes, automate them, and make their steps easier is also essential. Additional process debt can be caused by a lack of technology and tools, such as when an outdated configuration management solution does not deliver quickly [7].

3.3.2. People debt causes

Our MLR identified 10 people's debt causes that are divided into three themes named behavioral, work, and 3C challenges (see Table 5). It is evident that people's debt is heavily investigated in the SL compared to GL, where seven cases are reported. On the other hand, GL offers two new causes named people's poor behavior and remote work. Only two of the people's debts are identified in both SL and GL named race to ship as many features and poor inter-team coordination.

Table 5. People debt causes

People debt themes	Causes	GL ref.	SL ref.
Behavioural challenges	Frustrated and poor-performing teams		[12]
	Poor customer responsiveness		[12]
	People's poor behaviour	[36]	
Work challenges	Remote Work	[51]	
	Race to ship as many features	[51]	[16]
	Leaving people	[36, 51]	
3C challenges	Insufficient collaboration		[5]
	Insufficient communication		[5, 16]
	"Shortcuts" in communication		[5]
	Poor inter-team coordination	[38]	[5, 16]

Behavioural challenges. A team member's poor conduct negatively impacts the performance of other team members. Inversely, poor performance can also contribute to behavioral issues, particularly when team members become dissatisfied and angry about their poor performance or believe that an unfair standard has been established, which leads to poor productivity [12]. Low productivity has several detrimental repercussions on the workplace, including the financial impact on profitability and structural consequences on employee morale [12, 36]. Further poor behaviors of the people, such as excessive self-indulgence, a lack of self-control, exploiting others, and low motivation and effort, can be correlated with various antisocial, immoral, and imprudent behaviors that impede the software development process [36]. While poor customer responsiveness is based on the service provider's inability to provide in-time service, this is based on both the speed of interaction and the speed of the service fulfillment [12, 36].

Work challenges. In remote work without face-to-face connection, individuals miss a sense of shared purpose and are more indifferent to their employers. People's poor behaviors and attitudes toward work, not maintaining a positive attitude, shortcuts in communication, and remote working are also important reasons for people's debt [51]. Numerous disadvantages are linked with remote employment, as it directly impacts individuals' health. Loneliness is one of the primary obstacles that distant workers may need to overcome. When employees are not accustomed to working alone throughout the day, people could ask a coworker a brief

inquiry or run into someone in the hallway to discuss casually while working in the office. Many employees miss a sense of shared purpose without face-to-face connection and are more indifferent to their employers. Another working challenge is the ambitious managers who want to ship as many features as possible, which makes the development of the feature crawl over time. Also, when people leave, no one knows to whom the work should be transitioned. The combination of strict deadlines and excessive workloads causes software professionals to burn out and quit their careers prematurely [16]. Software engineers quit their jobs due to inadequate compensation in terms of money, technical difficulties, and growth opportunities. This has a detrimental effect on team performance; for example, the departure of team members is when no one understands who the task should be transferred to and who should fill the vacated position [36].

Communication, collaboration, and coordination (3C) challenges. During software development, “shortcuts in communication with people” refers to the incapacity of team members to interact effectively and appropriately [38]. This can occur when team members bypass proper communication channels, engage in extremely brief communication intervals, or avoid communication. Shortcuts in communication are taken to save time. Nevertheless, adopting communication shortcuts can impact multiple phases of the software development life cycle. For instance, omitting steps when speaking with the client to collect software requirements could result in a missed or insufficient requirements analysis. Failure to foster a collaborative environment at work results not only in a loss of benefits but also in a slew of disadvantages. The inability to create a team-friendly environment frequently results in an isolated and broken workflow, rarely leading to team efficiency or production [5]. When team members lack coordination, production can suffer, processes become more difficult, and work finishing can take longer [16].

3.3.3. Culture debt causes

Our MLR identified 7 cultural debt causes that are divided into two themes named organizational culture and management challenges (see Table 6). It is evident that cultural debt is heavily investigated in the GL and reported 6 causes. At the same time, unique causes identified in SL are named un-participatory culture. Here it is also worth mentioning that our MLR extended to cover cultural debt, which was not included in a recent NTD review conducted by Ahmad and Gustavsson [1]. This expansion is motivated by recognizing that a positive and productive work environment is essential for maximizing the effectiveness of processes and people involved in SD.

Table 6. Cultural debt causes

Culture debt themes	Causes	GL ref.	SL ref.
Org-culture challenges	Un-mindfulness in adopting culture	[52]	
	Un-participatory culture		[47]
	Individualistic culture	[22, 53]	
	Weak organisational culture	[13, 21]	
Management challenges	Managers lacking understanding of culture	[13]	
	Underinvestment in core HR functions	[13]	
	Hiring wrong people	[13, 54]	

Organisational culture challenges. Culture debt results from improperly implementing policies and procedures. Deferred investments because of budget uncertainty, inadequate

governance, organizational restructuring, and the need to act rapidly to meet emergent threats are common cultural debt drivers in organizations [13, 53]. Unparticipatory culture oppressed active decision-making and goal-setting. Unparticipative environments make employees feel less ownership over their work. They're more likely to ignore a problem or opportunity than assume someone else's responsibility. Individualistic cultures stress the needs of the individual over the needs of the group. In this type of culture, people are seen as independent and autonomous [21, 22]. Social behavior tends to be dictated by the attitudes and preferences of individuals. However, people with strong individualistic values and beliefs within an individualistic culture would have smaller social support networks, lower emotional competence, lower intentions to seek help from various sources, and poorer mental health [13, 22]. Individualism has also drawbacks where employees become too self-reliant, and a lack of emphasis on cooperation and teamwork leads to inefficiency in production [22]. Another big challenge faced is weak culture. It refers to values and beliefs not strongly and widely shared within the organization [13]. This implies that individual members of the organization rely more on personal principles, norms, and values. Further, the poorly implemented solution causes organizational culture to poorly identify the actions required, schedule the actions to identify the resources required, put measures in place to counter adverse consequences, and review the plans, etc. [13, 53].

Management challenges. When management teams don't know what culture they are trying to build in their organizations, it leads to cultural debt. Decision-makers have a cumulative impact on organizational culture. Managers can not guide the employees on how the company functions and is seen as a whole when they lack a proper understanding of the culture [13, 53]. Having adequate investments for managerial tasks is very important. Investing in management is important for cultural strength. Fewer management investments are a major reason for cultural debts in the software industry [13, 21]. Lack of investment in management causes degradation in revenue, branding, and workplace environment [13, 22]. The wrong hiring made by managers can have a serious and long-term impact on an organization. If a wrong hire is made, it can cause disruption in the team, increase recruitment and training costs, and decrease morale and productivity. It can also lead to a lack of trust in the leadership and a decrease in the quality of work. Additionally, the wrong hire can result in bad decision-making and cause the organization to miss out on potential opportunities [13].

3.3.4. Social debt causes

In the social debt category, we identified 17 causes that are divided into three themes named social confines, community smells, and organizational challenges (see Table 7). SL literature reported the majority of these cases, whereas GL reported only two new causes of social debt, named social isolation and decision in-communicability.

Social confines. Too much time alone at work and fewer collaborative talks contribute to social isolation [42, 51]. This hampers collaboration, idea-sharing, and teamwork. Social pressure is when one person or group influences another; for example, argument, persuasion, conformity, and demands are some social pressure examples [42]. Two types of social pressures exist 1. Workplace peer pressure, such as comparisons and competition with peers and 2. Psychological pressure: pressure caused by own thinking, such as overthinking, etc. Social pressure encourages improved performance and perfection. Seeing others succeed motivates others to achieve well. But extra and consistent social pressure on teams can lead to low self-esteem, lack of confidence, confusion about one's place in a social group,

Table 7. Social debt causes

Social debt themes	Causes	GL ref.	SL ref.
Social confines	Social isolation	[52]	
	Social pressure on teams		[47]
	Poor social structures		[50]
Organisational challenges	Uncooperativeness of the development community		[20]
	Organizational barriers		
	Uninformed socio-technical decisions		
	Architectural changes		
	Decision incommunicability	[36]	
	Global distance		[5]
	Lack of proper communication in organizations		[16]
Community smells	Omissions in social interactions		[43]
	Power distance		[42]
	Priggish members		
	Prima donnas		
	Radio silence or bottleneck		
	Haring villainy		
	Solution defiance		

etc. A social structure is a network of (social) relationships, habits, and ways of thinking among people working toward a common goal. It helps people with the same organizational aim to communicate information. Communication deteriorates, work outputs are delayed, and profitability is damaged when a company's social structure fails. To run effectively, it is necessary to frequently examine the social and organizational structure to ensure it matches the business's needs [42, 55].

Organisational challenges. The development community's unwillingness to work with technical experts causes organizational problems. Software development professionals rarely get along with techies. Unaware developers can create a "us vs. them" situation [3]. They blame each other for problems [3]. Organizational barriers impede employee knowledge flow and can lead to commercial failure [16, 20]. Organizational barriers include rules, policies, hierarchical positions, facilities, and complex systems. Employees must send queries in the organization's preferred language, medium, and manner of communication. The policy describes how employees should behave and communicate to stay employed [16, 18]. Uninformed socio-technical choices include poorly communicated organizational decisions and misinterpreted team findings. These ill-informed assessments can affect an organization's social and technical interdependence and the development community. It might lead to a lack of shared attention on communication and collaboration in achieving technical performance and job quality [3, 42]. Poor organizational or sociocultural conditions prevent the development network from communicating directly with key stakeholders [43]. Incommunicability is linked to communication and affected by social and organizational issues (e.g., organizational filtering protocols or nondisclosure agreements). Incommunicability traits, including community and smells, that impose communication obstacles (e.g., corporate silos or limited software practitioner communication cause social debt [5]. Organizational architecture increases societal debt. Architecture decisions can be determined "by osmosis," using information from every communication link in the development and operations network [12]. Critical information loss is almost inevitable. In the ensuing communication chain, important information, logic, and needs can be lost [20]. Inadequate communication, omission of social connections, and other issues contribute to social debt [5, 16].

Community smells. Priggish members refer to pedant teammates demanding of others pointlessly precise conformity or exaggerated propriety, which frustrates teammates and affects the software development process [18, 42]. Prima donnas work in isolation and don't welcome changes or support from teammates. The outcome prevents the organization from innovative solutions or processes and effective communication and collaboration [3, 42]. Another form is a lone wolf, where individuals work regardless of their peers due to poor communication. The negative results of such work are unsanctioned architecture decisions across the development process, code errors, and project delays [20, 42]. Radio silence or bottleneck can occur when tasks and communications are formally performed in a complex organization [42]. For example, a team member working as a unique information intermediary for different teams leads to communication overload and massive delays [42]. Sharing villainy is an environment where the goal of sharing reliable knowledge or information is challenging. When organizations cannot offer a decent working environment and lack encouraging knowledge sharing, the result is that the team finds it difficult to complete project activities [42]. Solution defiance can be called team conflicts and a lack of respecting others' opinions regarding a potential solution. Each organization has teams that might be more diverse in various ways. Teams conflict in decision-making meetings when a team or individual is too rigid in their technical expertise, organizational cultural beliefs, values, and norms [55].

3.3.5. Organisational debt causes

Organizational debt leads to TD because of deferred investments due to budget uncertainty, poor governance and architecture, and organizational restructuring. On the other side, TD can also cause organizational debt. Non-technical debt refers to the broader organizational inefficiencies and operational shortcomings that hinder the organization's performance. Here's how system challenges can be related to non-technical debt in the organization. The inability to integrate systems, poor maintenance practices, the inherent complexity of systems, and complex, difficult-to-operate systems are considered causes of organizational debt. These factors can contribute to inefficiencies, increased costs, and hindered productivity within an organization. These system challenges can directly contribute to organizational debt. For a clear division of organizational debt themes, we have divided organizational challenges into two main types, i.e., organisational structure and system challenges [36]. Details of the organisational debt causes are provided in Table 8.

Organisational structure challenges. Small and inflexible organizations lack the knowledge and capacity to upgrade and modernize [52]. Poor maintenance, inadequate investment, system complexity, and changing mission requirements affect them [44]. Bad organizational and architectural choices lead to bad decisions, performance evaluations, and compensation structures [58]. A very large and flat organizational structure has obstacles like a lack of hierarchical or flat structures: motivational issues, blurred decision-making processes, a lack of knowledge of areas of responsibility, and inconsistent processes and procedures. Flat organizations eliminate expectations. Some workers depart because they can't advance [36]. Internal politics (office or workplace politics) is inevitable. It refers to persons competing for prestige or power in the workplace. Politics decreases individual and organizational output. Politics harms the workplace [2]. Team members use their available informational resources through information sharing. Information sharing promotes innovation, efficiency, and new ideas by reducing repetition. Everyone benefits when employees share their knowledge and generate searchable content. Uneven information exchange across teams can

Table 8. Organizational debt causes

Organizational debt themes	Causes	GL ref.	SL ref.
Organizational structure challenges	Sluggish or inflexible organizations	[52]	
	Bad organizational, architectural choices	[52]	
	Lack of skills to upgrade organizations	[44]	
	Extensive and flat organization	[36]	
	Internal politics	[36]	
	Uneven information sharing among teams	[2]	
	Lack of management commitment	[56]	
	Unclear changes	[57]	
	Un structured Information	[57]	
	Lack of healthy communication	[41]	
	Organizational culture hindering progress	[41]	
Organizational-systems challenges	Inability to integrate systems	[52]	
	Poor maintenance	[52]	
	The inherent complexity of the systems	[52]	
	Complex, difficult to operate systems		[2]

harm the organization by hiding information from management [56]. Lack of management commitment generates an inefficient organization that hinders information sharing and cooperation. Lack of software implementation and maintenance resources affects quality [57]. When people don't grasp why change is needed, anxiety, scepticism, and resistance rise, and most significant changes are justified by financial returns (e.g., acquisitions add revenues; cost reductions increase margins). However, the rationale for large-scale change must be clear and convincing for all important stakeholders [41]. Unstructured information is difficult for people and computers to interpret. Unstructured information often causes workarounds that modern businesses don't understand. A lack of communication can produce misunderstandings, missed opportunities, conflict, disinformation, and mistrust, making staff feel defeated. Poor organizational communication causes friction, frustration, and confusion, producing a stressful climate where individuals aren't driven to collaborate or be productive. Culture impacts people's performance. Organizational culture is seen as a technique to get things done or as typical organizational features that shape member behavior and improve (or hinder) strategic achievement and performance [52]. Ambiguity, poor communication, and inconsistency are common cultural challenges. These can create a hostile and unpleasant workplace, leading to harassment, bullying, and high turnover. Culture drives growth and performance. Unhealthy workplace culture impairs engagement, retention, and performance. It hinders business when procedures and processes are structured to fit a legacy technology's capabilities. If you lack current collaborative tools like video conferencing or group chat, you may choose a local team over one with the best skills.

Organisational system challenges. A lack of or poorly designed integration can cause duplicate data, sluggish order processing, fulfillment delays, disgruntled customers, and profit loss. A lack of system integration hinders system performance and treasury operations with human work [52]. System integration challenges are caused by insufficient expertise, required money or investments, resources, inadequate communication/planning, after-go-live maintenance, and sophisticated technical concerns. Lack of integration produces information silos that obscure company performance. Inefficiencies hinder decision-making and raise redundancies [2]. Poor maintenance means failing to keep organizations working. Routine maintenance is preventive, predictive, or scheduled [52]. Maintenance is key to quality assurance and a company's long-term profitability. Unmaintained resources can cause

instability and slow production. Malfunctioning machinery or breakdowns can be expensive. A software system's complexity isn't an accident. This intrinsic complexity originates from four elements: the complexity of the problem area, the difficulty of managing the development process, software flexibility, and discrete system behavior difficulties [52]. Software complexity makes development management more difficult. Complex systems have many interacting parts. Hence they lack predictable causation. These components might alter over time, generating unpredictability in relationships. This causes unforeseen difficulties, defects, security failures, or crashes that are hard to examine. Detailed system descriptions, risk evaluations, and demand specifications are often wrong for systems where unexpected events occur [2].

Hence, the inability to integrate systems, poor maintenance practices, the inherent complexity of systems, and complex, difficult-to-operate systems are recognized causes of organizational debt. These factors directly contribute to inefficiencies, increased costs, and hindered productivity within an organization, thus adding to the burden of organizational debt. The inability to integrate systems effectively results in data inconsistencies, manual workarounds, and limited information flow, leading to higher costs, reduced productivity, and an accumulation of organizational debt over time. Poor maintenance practices, such as neglecting software updates and security patches, lead to degraded system performance, increased downtime, and higher maintenance costs, all of which contribute to organizational debt. The inherent complexity of systems, particularly legacy systems, requires specialized knowledge, training, and support, adding overhead costs and creating difficulties in system configuration, customization, and troubleshooting. Similarly, complex and difficult-to-operate systems with poor user interfaces and convoluted workflows impede employees' ability to perform tasks efficiently, resulting in errors, reduced productivity, and frustration. The time spent navigating these complex systems and seeking workarounds adds to inefficiencies and organizational debt.

3.4. NTD mitigation strategies

In this section, we are examining mitigation approaches for NTDs from the current literature. We will discuss each type of NTD management approach separately. This section answers RQ3 – What are the reported NTD mitigating strategies?

3.4.1. Process debt mitigation

We identified 5 process debt mitigation strategies (see Table 9). Only one mitigation strategy was identified in GL named measurement of process, whereas the remaining 4 are from SL. It is important to note that SLR [1] aims to provide information on how to prevent process debt and address architecture issues, requirement mismatches, process divergence, and organizational challenges. The current MLR, on the other hand, provides more specific recommendations and considerations for mitigation strategies. It highlights the importance of process documentation, monitoring, automation, market adaptation, and process design in managing process debt. It emphasizes the need for organizational restructuring involving end users. The MLR also suggests following conceptual models and tracking process productivity. The MLR offers broader principles and concepts for effective process debt management. It highlights the importance of process productivity tracking and conceptual models, which are not explicitly mentioned in [1].

Effective process debt management strategies are strongly tied to improving software development processes, such as process documenting, process monitoring, regular auditing, early detection of risks, and measuring process appropriateness [5, 29]. Process automation can minimize process debt [33], and by embracing new and valuable technologies and tools, process automation can lead to functional augmentation and virtualization, making the process more understandable [33]. It is also a good idea to employ new processes to avoid process debt if market trends and needs change. Following the agile approach, for example, meeting early time to market, accepting dynamic requirements changes, engaging end users throughout the development process, and so on, all while revamping and improving overall organizational structures, aid in avoiding process debt. The changes to the development process are linked to the restructuring of the entire organization [2]. These redesigns are generally associated with organizational transformation, necessitating large-scale, broad-scale modifications.

Table 9. Process debt mitigation strategies

Mitigation Strategies	GL ref.	SL ref.
Measurement of process	[29]	[5]
Automation of process		[2]
Continuous process assessment		[33]
Conceptual model for understanding the process debt		
Adopting new process		

An example is an organization's transition from a traditional software process to an agile software development process [33]. It is critical to have someone in charge of managing processes and supervisors who appreciate the importance of processes. Its primary purpose should be a continual evaluation that aids in proper process monitoring. On the other hand, a process must be designed with a specific purpose and value rather than just as a mandatory management tool imposed by the business. A thorough study is essential before selecting a process [16]. Researchers recommend following conceptual models to avoid process divergence and maintain track of process productivity in software development projects to understand process debt better [33].

3.4.2. People debt mitigation strategies

We identified 5 people's debt mitigation strategies (see Table 10). In the GL, two new mitigation strategies were identified, named work clubs' ideas for people's psyche and well-being and adopting the tradition of handling people's debt. Both GL and SL are reporting on educating business people about engineering people's decisions. In contrast, the remaining two strategies named continued monitoring and communication and managing dependencies, are reported in SL sources. Here it is important to note that the current MLR, in comparison to [1], delves deeper into the interpersonal and social aspects of managing people's debt, highlighting the significance of collaboration, resource provision, appreciation, education, and social support in creating a positive and supportive work environment.

People's debt management is difficult and uncontrollable since it is linked to human behavior, psyche, and well-being. But the people's challenges can be handled by encouraging collaboration among team members [59]. This could include having regular meetings where each person can discuss their progress and areas of improvement. Encourage team members

to work together to find solutions to problems and foster open communication among team members [2, 60]. Clear communication between stakeholders and developers makes gathering adequate and clear requirements easy. Open communication also leads to better collaboration and monitoring of interdependencies between teams [2]. This could include having team members provide honest feedback to each other and discussing their thoughts on how to improve the development process. Ensure all team members are on the same page regarding the project's goals. Everyone should understand what needs to be done and when it needs to be done [60]. This can help prevent the team from getting bogged down in the details and conflicts. Ensure everyone has the necessary resources to do their tasks [34]. This could include ensuring everyone can access the right tools and technology and providing support and guidance when needed [2]. Appreciate each team member for their hard work and dedication. This could include recognizing individual contributions and celebrating team successes. This can help motivate people to continue to do their best [15, 42]. Educating people adequately is also compulsory to avoid people's debt, such as educating businessmen about engineering people's decisions and to lessen business people's pressure on engineers by educating and communicating to them the technical perspectives [2]. People's well-being is strongly connected to their social ties and support; long periods of isolation at work are linked to stress, depression, and low morale. To keep people out debt, joint work groups or venues are recommended for daily discussion and social well-being. It is also important to continuously identify, prioritize, understand, and handle people's debt in organizations. So, there is a need to adapt to the tradition of people's debt understanding and handling.

Table 10. People debt mitigation strategies

Mitigation strategies	GL ref.	SL ref.
Continued monitoring and communication		[59, 60]
Work clubs' idea for people psyche and wellbeing	[39]	
Adopting tradition of handling people debt	[34]	
Managing dependencies		[37]
Educating business people about engineering people decisions	[2]	[42]

3.4.3. Culture debt mitigation strategies

We identified 4 culture debt mitigation strategies (see Table 11). Both GL and SL are reporting on three common mitigation strategies: handling issues collaboratively, delivering accurate information for intended multicultural audiences, clear communication, and an orderly business environment. While GL identified an additional mitigation strategy, creating the right mindset in the company. All of these findings are new to the existing literature as cultural debt is out of scope in [1].

To effectively handle cultural debt, organizations should focus on creating the correct mindset among their members. This includes fostering a growth mindset that embraces challenges and establishes trust within the working relationships. It is important for team members to feel comfortable discussing ideas, disagreements, and solutions. Encouraging diversity of thought is crucial for promoting creative problem-solving and innovation within teams [12, 13]. Encouraging team members to share different perspectives and ideas to see a problem from all angles and establishing clear communication between team members is one of the most effective ways to address cultural differences [54]. Respecting

each other's cultural backgrounds and values may include being mindful of language and communication styles and understanding different approaches to problem-solving and decision-making, specifically by solving difficulties collaboratively across teams [22, 57] and enabling knowledge exchange with multicultural audiences [53, 56]. Creating more natural workplaces through clear communication and an organized workplace also aids in regulating cultural debt. Investing in management for a better working culture and emphasizing organizational transparency provides a good culture [31].

Table 11. Cultural debt mitigation strategies

Mitigation strategies	GL ref.	SL ref.
Creating right mind set in company	[12, 13]	
Handle issues collaboratively	[50]	[47]
Deliver accurate information for intended multicultural audience	[57]	[53, 56]
Clear communication, orderly business environment	[22]	[5]

3.4.4. Social debt mitigation strategies

We identified 8 social debt mitigation strategies (see Table 12). Both GL and SL are reporting on three common mitigation strategies: social network analysis for debt analysis, collaborative communication, and guidelines for managing team composition and improving the description of architectural decisions. SL separately reports metrics for software architecture communicability and frameworks to follow for social debt mitigation, including the CAFFEA framework, architectural tactics, DAHLIA, and socio-technical quality framework. GL additionally reports combined work environments (hybrid) for social debt mitigation.

The current MLR expands the scope on topics not covered in study [1]. It emphasizes the significance of organizational strategies, frameworks, models, and guidelines that are crucial for monitoring and mitigating social debt. It recognizes the value of collaborative work environments and social network analysis as integral components of these strategies. It highlights the importance of fostering open communication, promoting collaboration, respecting diverse opinions, and implementing effective conflict-resolution strategies as essential measures for managing social debt. Additionally, the MLR acknowledges the significance of employing specific tools to diagnose and manage social debt within development communities.

In line with these findings, several organizational strategies, frameworks, models, tools, and guidelines help monitor and mitigate social debt. Social debt mitigation strategies are linked to collaborative work environments [32, 51] and social network analysis [20, 50]. Honest and open team communication and intense collaboration [11, 43]. Software development teams can manage social problems by encouraging open communication and collaboration, respecting the opinions of others, and using effective conflict resolution strategies [18, 19].

Open communication allows teams to share ideas and identify potential problems, while collaboration encourages everyone to work together to find solutions. Respect for the opinions of others is essential for teams to progress without disagreements or misunderstandings [19, 43].

Finally, effective conflict resolution strategies, such as brainstorming or seeking outside help, allow teams to work through disagreements and ensure everyone is on the same

Table 12. Social debt mitigation strategies

Mitigation strategies	GL ref.	SL ref.
Frameworks:		
1. CAFFEA framework		
2. Architectural tactics		[43, 50]
3. DAHLIA		
4. Socio-technical quality framework		
Combined work environments (hybrid)	[32, 51]	
Metric for software architecture communicability		[11]
Social network analysis for debt analysis	[18]	[20]
Collaborative communication	[19]	[43]
Guidelines: 1. Managing team composition 2. Improving the description of architectural decisions	[18]	[42, 50]
Tools:		
1. GEEZMO		
2. CodeFace4Smell		
3. YOSHI		
Models:		
1. Statistical		
2. Social networks		

page. Practitioners must have the tools to diagnose and manage social debt in their development communities [18]. Some of the tools reported to detect and manage social debt are GEEZMO which 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 [42].

3.4.5. Organisational debt mitigation strategies

We identified five organizational debt mitigation strategies (see Table 13). Among the strategies identified, only one was reported in both the SL and GL studies, namely monitoring, communication, and documentation. The remaining four strategies were exclusively documented in the GL, including adhering to the organizational model, maintaining and revising organizational charts and cloud architecture, and enhancing organizational culture. Notably, our study, the current MLR, focuses specifically on organizational debt mitigation strategies, which were not encompassed in a recent review on NTD [1].

Table 13. Organisational debt mitigation strategies

Mitigation strategies	GL ref.	SL ref.
Monitoring, communication and documentation	[45]	[43]
Following organizational model	[53, 57]	
Maintaining and revising organizational charts	[36, 44]	
Cloud architecture	[57]	
Improving organizational culture	[41, 49]	

The mitigation strategies emphasize on the importance of monitoring decisions and changes that need to be identified, prioritized, measured, and monitored. Such a monitoring

process facilitates faster decision-making and drives business improvement by expediting reporting. Real-time exception detection plays a crucial role in enabling organizations to respond promptly to emerging challenges and opportunities [57]. These findings highlight the significance of robust document management and adherence to an organizational structure based on validated frameworks.

Effective communication within an organization is essential for building trust, fostering teamwork, improving relationships, enhancing problem-solving abilities, and resolving conflicts [53, 57]. Robust document management practices ensure that all individuals within the company, regardless of their department or team, understand the storage, review process, and up-to-date status of documents. This ensures clarity and alignment and, if necessary, enables timely actions to be taken. Adhering to an organizational structure based on validated frameworks and updating organizational charts can streamline processes, enhance decision-making, manage multiple locations, drive employee performance, and prioritize customer service and satisfaction [43]. Social network analysis tools offer effective means to forecast, manage, and address debt within organizations [20]. Automating debt identification and testing tools enable continuous monitoring of debt sources and provide opportunities to proactively overcome them. In the dynamic software business, organizational adaptability and flexibility are crucial for survival and success [18, 36, 44].

Further 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 [57]. 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 [41, 55, 61].

4. Future work

Our MLR proposes several potential future directions that can further advance understanding of NTD and develop effective mitigation strategies. This section outlines key areas for future research in the field of NTD and answers RQ4. First, it would be interesting to apply social capital theory and control theory that can provide deeper insights into how social environments and relationships influence the accumulation of social and people debt. By leveraging these theories, researchers can explore the types of resources available through social networks and how they can be utilized to reduce debt. Second, develop a comprehensive taxonomy for categorizing and identifying distinct NTD types. This taxonomy can serve as a foundation for classifying NTD and enable the development of specialized approaches to tackle the unique challenges posed by each type. Stakeholders can benefit from this taxonomy by better understanding NTD types, their effects, and effective mitigation strategies. Third, investigating the effects of different NTD types on TD accumulation is crucial for comprehending how NTD leads to the creation of TD. Empirical studies are needed to explore the relationship between NTD and TD, considering factors such as poor planning, rushed coding, inadequate testing, and lack of refactoring. This research can shed light on how various issues contribute to the development of TD. Fourth, it's worth exploring other types of debts that can be studied under the umbrella of NTD, such

as service debt, sustainability debt, and environmental sustainability debt. Investigating service debt can provide insights into trade-offs related to service-oriented architectures, service-level agreements, and service dependencies. Understanding and managing service debt can contribute to the development of strategies and practices that ensure reliable and efficient service delivery. Exploring sustainability debt could focus on the long-term sustainability and maintainability of software systems. It aims to identify approaches that design and develop more sustainable software systems, both in terms of their technical aspects and their impact on the environment and society. Investigating environmental sustainability debt encompasses the ecological impact of software development activities, including energy consumption, resource utilization, and carbon emissions. Research in this area can contribute to the development of eco-friendly software engineering practices, reducing the environmental footprint of software systems. By pursuing these future research directions, researchers and practitioners can advance the understanding of NTD, develop effective mitigation strategies, and promote more sustainable and efficient software development practices.

5. Implications

Our MLR not only demonstrates the scarcity of research efforts on NTD, but it also demonstrates the direct relationship of NTD to TD in software development projects. It directs researchers to investigate further the relationship of NTD to TD in terms of causes, mitigation techniques, and consequences. This study elucidates the dangers of ignoring NTD while studying and developing TD. The highest number of grey literature studies shows the practitioners' increasing interest in the topic. However, it also forecasts that practitioners lack a holistic view of the different types of NTD and struggle to find a correlation between them. Most practitioners are aware of the TD and NTD relationship to it. Still, they do not understand how different NTD are connected. Therefore our study implicates the need for further investigation of TD to NTD and NTD to NTD relationships by conducting more industrial case studies. Our study also emphasizes practitioners' education and training about NTD and coping strategies for handling NTD in software development. Thus, we urge the companies to participate in research projects in the future to target research goals regarding the prevention and mitigation of NTD that are relevant to the software industry.

6. Threats to validity

We are addressing threats to external validity, threats to construct validity, and threats to conclusion validity. A data collection process was designed to support data recording to minimize the construct validity threat. Two researchers were involved in the whole process, which helped to lessen this threat even more. 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 minimize external validity threats by following the guidelines proposed by [23]. Conclusion validity is related to researchers' bias or misinterpretation of data. This is a major risk and cannot be eliminated. However, we took several steps to minimize this threat, such as having two researchers involved in the analysis, which helps limit subjective opinions. Further, a full audit and trial of 40 sources were maintained, and conclusions were drawn collaboratively.

7. Discussion and conclusion

The topic of debt is relatively new compared with other domains, such as software quality and testing. However, despite significant studies published on the TD concept, NTD has remained less explored. Among the different types of debt proposed in work by Lenarduzzi et al. [5], social debt has been investigated by [3] and process debt by [33]. While in a recent review conducted by Ahmad and Gustavsson [1], they extended the scope of previous studies by including an investigation of people's debt as well. However, they identified a scarcity of scientific studies on NTD [1]. They reported only 17 scientific studies on NTD and requested further empirical investigation and other forms of literature reviews on the topic. This MLR investigated NTD to extend the recently conducted systematic mapping review [1]. To achieve our study goals, we are investigating the research questions designed and reported in [1] that serve as the guided foundation for this MLR.

While TD resides within the system codebase [61], NTD seems more pervasive and intertwined with people, organizations, their working processes, and cultural issues. Software development is a socio-technical phenomenon based on socio-technical decisions. A socio-technical decision generates technical and any or all NTD types (i.e., people, process, culture, social, and organizational). TD outcomes are measured in monetary values, while consequences can measure NTD outcomes. We intentionally used the word “consequence” rather than “value” for NTD as its measurement is beyond the scope of monetary values. Therefore, more research is needed to understand how the effects of NTD can be measured or quantified in software projects. The results show that NTD seems harder to fix than TD. NTD contributes to TD accumulation, and its effects are both short- and long-term. NTD and TD are strongly intertwined with human dimensions – software architectures and their impact on businesses and cultures [3, 33, 42]. This highlights the complexity of managing debt in software development and underscores the need for further exploration and understanding of NTD to effectively mitigate its impact. 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 [42]. This signal that NTD must be dealt with alongside TD to avoid severe consequences of software project failure.

Non-technical debt (NTD) significantly influences the human and social aspects of software projects, stemming from factors such as cognitive distance, architectural discrepancies, and cultural and organizational systems [42]. This highlights the need to address NTD alongside technical debt (TD) to mitigate the potentially severe consequences of project failure. In comparison to the existing review [1], our study uncovered a range of significant causes for process debt. These causes encompass process divergence, inefficient and outdated processes, changing process requirements, sub-optimal practices, lack of follow-up assessment, prioritization issues, costs associated with process changes, neglect of value considerations, power distance, reliance on shortcuts and quick fixes, as well as the influence of technology, tools, and external trends. This expanded our understanding of process debt causes and extended the scope of knowledge in this domain beyond the technology-focused causes identified in the systematic mapping review [1].

Regarding process debt mitigation, the SLR aims to provide general insights into preventing process debt and addressing architecture-related issues, requirement mismatches, process divergence, and organizational challenges [1]. In contrast, our study offers more specific recommendations and considerations for effective mitigation strategies. It highlights the importance of process documentation, monitoring, automation, market adaptation, and

thoughtful process design in managing process debt. Additionally, our study emphasizes the need for organizational restructuring that involves end users and suggests following conceptual models and tracking process productivity. These broader principles and concepts for effective process debt management supplement the recommendations provided by the recent review [1], as they explicitly address aspects such as process productivity tracking and the use of conceptual models.

For the people debt, the leading causes found were directly associated with inefficient collaboration, insufficient communication [51], shortcuts in communication [5, 42], and lack of inter-team coordination [16]. Therefore, it is clear that many issues linked to the causes of people's debt are based on a lack of communication and coordination. Behavioral challenges associated with people's debt arise from poor conduct and performance, decreasing productivity and customer responsiveness. Work challenges emerge from remote work disadvantages, such as a lack of shared purpose, poor attitudes, and ambitious managers causing feature development delays. Communication, collaboration, and coordination challenges manifest as shortcuts, inadequate teamwork, and coordination issues, impeding workflow and prolonging work completion. These challenges collectively impact the software development process, team performance, and productivity, necessitating effective solutions and strategies to mitigate their negative effects. The current MLR extends the understanding of people's debt in software development by highlighting specific behavior, work, and communication challenges compared to the recent review [1], which focuses on identifying factors such as knowledge gaps, inadequate management, and morale issues contributing to people's debt. For the mitigation of people's debt challenges, the current MLR, in comparison to review [1], delves deeper into the interpersonal and social aspects of managing people's debt, highlighting the significance of collaboration, resource provision, appreciation, education, and social support in creating a positive and supportive work environment.

The MLR expands the scope on topics of social debt not covered in the recent review [1]. It emphasizes the significance of organizational strategies, frameworks, models, and guidelines that are crucial for monitoring and mitigating social debt. It recognizes the value of collaborative work environments and social network analysis as integral components of these strategies. It highlights the importance of fostering open communication, promoting collaboration, respecting diverse opinions, and implementing effective conflict-resolution strategies as essential measures for managing social debt. Additionally, the current study acknowledges the significance of employing specific tools to diagnose and manage social debt within development communities.

We noticed that cultural and organizational terms were interchangeably used in the development communities, even though these terms have different meanings and contexts. While they can also have the exact reasons and mitigation strategies in specific contexts, i.e., carelessness in adopting policies, practices, and culture can lead to cultural debt in software development communities. It is also linked to the different causes of organizational debt, i.e., sluggish or inflexible systems, aging or legacy systems, and bad architectural choices associated with carelessness in adopting culture. This also leads to insufficient skills to upgrade and modernize the systems and infrastructure. In return, it leads to an inability to integrate systems and upgrade given organizational setups [52]. While we also noticed that there are two kinds of cultural perspectives in software development communities. (I) Work culture and (II) The employee's cultural background.

There is also a relationship between cultural and organizational impacts on selecting sub-optimal processes, which leads to the process debt in the result [33]. Therefore, poor

cultural and organizational choices are directly proportional to the selection of inefficient software development processes. Here it's important to note that both organizational and cultural debt that our MLR includes in the scope for investigation is not included in a recent NTD review conducted by Ahmad and Gustavsson [1]. The same pattern is reported in triggering social debt, i.e., lack of suitable communication among important sides of the organization [16] and missing social connections or reduced communication. The same applies to organizational debt, as the main reason for organizational debt causes is associated with uneven information sharing among teams [2] and a lack of healthy communication [48]. 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 [41, 48]. 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 [41, 57]. 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 interlinked. NTD contributes to TD, and both cause equal damage. We further noticed that "All NTD contributes to TD," highlighting the interrelated nature of the technical and non-technical debt. It is also evident From the literature [3, 7, 11, 12, 16, 18, 20, 33, 55, 55, 62] that one type of NTD causes another type of NTD, i.e., culture debt, and organization debt can lead to process debt [7, 16, 33], people debt can lead to organizational debt, and culture debt [2, 53], and social debt can lead to people debt [13]. While they may be distinct types of debt, they are not mutually exclusive. In fact, non-technical debt can contribute to technical debt, as seen in the example above. Non-technical debt can create constraints that limit the ability to address technical debt or cause technical debt to be incurred in the first place. There is a clear connection between NTD and large-scale agile development. The challenges regarding testing strategies, specifically integration, regression, and user acceptance testing found in large-scale agile development, are reported as "test debt". At the same time, sprint-related challenges in agile projects are categorized as non-technical debt as well [1].

Acknowledgment

This research was performed within the Non-Technical Debt in Large-Scale Agile Software Development (NODLA) Project, funded by the Knowledge Foundation, Sweden.

References

- [1] M.O. Ahmad and T. Gustavsson, "The Pandora's Box of social, process, and people debts in software engineering," *Journal of Software: Evolution and Process*, 2022, p. e2516.
- [2] J. Yli-Huumo, *The role of technical debt in software development*, Ph. D. Thesis, Lappeenranta University of Technology, 2017.
- [3] D.A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "Social debt in software engineering: Insights from industry," *Journal of Internet Services and Applications*, Vol. 6, 2015, pp. 1–17.
- [4] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of Systems and Software*, Vol. 101, 2015, pp. 193–220.
- [5] A. Melo, R. Fagundes, V. Lenarduzzi, and W.B. Santos, "Identification and measurement of requirements technical debt in software development: A systematic literature review," *Journal of Systems and Software*, 2022, p. 111483.

- [6] P. Kruchten, R.L. Nord, and I. Ozkaya, “Technical debt: From metaphor to theory and practice,” *IEEE Software*, Vol. 29, No. 6, 2012, pp. 18–21.
- [7] A. Martini and J. Bosch, “The danger of architectural technical debt: Contagious debt and vicious circles,” in *12th Working IEEE/IFIP Conference on Software Architecture*. IEEE, 2015, pp. 1–10.
- [8] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing technical debt in software engineering,” in *Dagstuhl Reports*, Vol. 6, No. 4. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.
- [9] W. Cunningham, “The WyCash portfolio management system,” *ACM SIGPLAN OOPS Messenger*, Vol. 4, No. 2, 1992, pp. 29–30.
- [10] N. Rios, R.O. Spínola, M. Mendonça, and C. Seaman, “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*, 2018, pp. 1–10.
- [11] D.A. Tamburri, “Software architecture social debt: Managing the incommunicability factor,” *IEEE Transactions on Computational Social Systems*, Vol. 6, No. 1, 2019, pp. 20–37.
- [12] J. Yli-Huumo, A. Maglyas, and K. Smolander, “How do software development teams manage technical debt? – An empirical study,” *Journal of Systems and Software*, Vol. 120, 2016, pp. 195–218.
- [13] A. Chen, *Cultural Debt*, 2022. [Online]. <https://www.careerfair.io/reviews/cultural-debt>
- [14] T. Klinger, P. Tarr, P. Wagstrom, and C. Williams, “An enterprise perspective on technical debt,” in *Proceedings of the 2nd Workshop on Managing Technical Debt*, 2011, pp. 35–38.
- [15] J. Yli-Huumo, A. Maglyas, and K. Smolander, “The effects of software process evolution to technical debt – Perceptions from three large software projects,” *Managing Software Process Evolution: Traditional, Agile and Beyond – How to Handle Process Change*, 2016, pp. 305–327.
- [16] A. Martini, V. Stray, and N.B. Moe, “Technical, social and process debt in large-scale agile: An exploratory case-study,” in *Agile Processes in Software Engineering and Extreme Programming – Workshops: XP Workshops*. Montréal, QC, Canada: Springer, 2019, pp. 112–119.
- [17] Z. Li, P. Liang, and P. Avgeriou, “Architectural debt management in value-oriented architecting,” in *Economics-Driven Software Architecture*. Elsevier, 2014, pp. 183–204.
- [18] D. Tamburri, *From Technical to Social Debt: Analyzing Software Development Communities using social networks analysis*, 2015. [Online]. <https://www.slideshare.net/DamianTamburri/from-technical-to-social-debt-analyzing-software-development-communities-using-socialnetworks-analysis>
- [19] T. Mejía, *Social Debt: the difficult commitment*, 1998. [Online]. <https://www.socialwatch.org/book/export/html/10623>
- [20] D.A. Tamburri and E. Di Nitto, “When software architecture leads to social debt,” in *12th Working IEEE/IFIP Conference on Software Architecture*. IEEE, 2015, pp. 61–64.
- [21] C.B. Jaktman, “The influence of organizational factors on the success and quality of a product-line architecture,” in *Proceedings 1998 Australian Software Engineering Conference*. IEEE, 1998, pp. 2–11.
- [22] B. Sutton, *Overcoming Cultural and Technical Debt*, 2019. [Online]. <https://sloanreview.mit.edu/audio/overcoming-cultural-and-technical-debt/>
- [23] V. Garousi, M. Felderer, and M.V. Mäntylä, “Guidelines for including grey literature and conducting multivocal literature reviews in software engineering,” *Information and Software Technology*, Vol. 106, 2019, pp. 101–121.
- [24] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey et al., “Systematic literature reviews in software engineering – A systematic literature review,” *Information and Software Technology*, Vol. 51, No. 1, 2009, pp. 7–15.
- [25] M.C. Davis, R. Challenger, D.N. Jayewardene, and C.W. Clegg, “Advancing socio-technical systems thinking: A call for bravery,” *Applied Ergonomics*, Vol. 45, No. 2, 2014, pp. 171–180.

- [26] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, and F.A. Fontana, “A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools,” *Journal of Systems and Software*, Vol. 171, 2021, p. 110827.
- [27] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Information and Software Technology*, Vol. 50, No. 9–10, 2008, pp. 833–859.
- [28] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative Research in Psychology*, Vol. 3, No. 2, 2006, pp. 77–101.
- [29] L. McGuire, *What Is Process Debt, and Why Is It a Problem*, 2022. [Online]. <https://www.formstack.com/blog/process-debt>
- [30] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2011.
- [31] S.W. Wenger E, McDermott RA, *Cultivating Communities of Practice: a Guide to Managing Knowledge*. Harvard Business School Publishing, 2002. [Online]. <https://hbswk.hbs.edu/archive/cultivating-communities-of-practice-a-guide-to-managing-knowledge-seven-principles-for-cultivating-communities-of-practice>
- [32] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, “Putting it all together: Using socio-technical networks to predict failures,” in *2009 20th International Symposium on Software Reliability Engineering*. IEEE, 2009, pp. 109–119.
- [33] A. Martini, T. Besker, and J. Bosch, “Process debt: A first exploration,” in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2020, pp. 316–325.
- [34] D. Blomstrom, *How to Recognise and Reduce HumanDebt*, 2022. [Online]. <https://www.infoq.com/articles/human-debt>
- [35] P. Vinayak, *Everything you need to know about Cultural debt*, 2021. [Online]. <https://e2ehiring.com/blogs/everything-you-need-to-know-about-cultural-debt>
- [36] M. Bellotti, *Hunting Tech Debt via Org Charts. Knowing where to look for problems*, 2021. [Online]. <https://bellmar.medium.com/hunting-tech-debt-via-org-charts-92df0b253145>
- [37] L. Pirzadeh, “Human factors in software development: A systematic literature review,” Master’s thesis, 2010.
- [38] INSEAD, “Company strategic planning/Interviews/INSEAD,” 2015.
- [39] G. Marlow, *People debt is like technical debt – eqsystems.io*, 2017. [Online]. <https://eqsystems.io/2017/04/people-debt-like-technical-debt>
- [40] B. Coleman, *Culture Debt Is One of the Most Toxic Threats to Business, and Your Startup Is Probably Victim to It*, 2019. [Online]. <https://www.inc.com/bernard-coleman/culture-debt-is-one-of-most-toxic-threats-to-business-your-startup-is-probably-victim-to-it.html>
- [41] M. Hosking, *Transformation troubles and non-technical debt*, 2017. [Online]. <https://www.linkedin.com/pulse/transformation-troubles-non-technical-debt-matt-hosking/>
- [42] E.A.C. Espinosa, *Understanding Social Debt in Software Engineering*, Ph.D. dissertation, The University of Alabama, 2021.
- [43] T. Dreesen, P. Hennel, C. Rosenkranz, and T. Kude, ““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*, 2021, p. 6826.
- [44] J. Trouw, *Organisational debt an analogy*, 2021. [Online]. <https://www.linkedin.com/pulse/organisational-debt-analogy-jaap-trouw>
- [45] S. Blank, “Organizational debt is like technical debt – but worse,” 2015. [Online]. <https://www.forbes.com/sites/steveblank/2015/05/18/organizational-debt-is-like-technical-debt-but-worse-2/?sh=6ea3ce447b35>
- [46] S. Priestnall, *What is Process Debt?*, 2020. [Online]. <https://www.linkedin.com/pulse/what-process-debt-steve-priestnall>
- [47] J.A. Miko, *Collaboration strategies to reduce technical debt*, Ph.D. dissertation, Walden University, 2017.
- [48] M. Eaden, *When Testers Deal With Process Debt: Ideas to Manage It and Get Back to Testing Faster*, 2017. [Online]. https://www.ministryoftesting.com/articles/8d79968d?s_id=15650023

- [49] L.P. Gates, *Are We Creating Organizational Debt*, 2017. [Online]. <https://insights.sei.cmu.edu/blog/are-we-creating-organizational-debt/>
- [50] R. Kazman, “Managing social debt in large software projects,” in *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)*. IEEE, 2019, p. 1.
- [51] K. Ladewig, *The dark side of working from home*, 2019. [Online]. <https://medium.com/swlh/social-debt-17bf03a269a>
- [52] S. Vinsennau, *Decouple to innovate how federal agencies can unlock IT value and agility by remediating technical debt*, 2016. [Online]. https://www.accenture.com/_acnmedia/PDF-85/Accenture-Decoupling-to-Innovate.pdf
- [53] B. Falchuk, *What’s the Greatest Threat to Your Organization? Culture Debt*, 2019. [Online]. <https://bryanfalchuk.com/blog/culture-debt>
- [54] B. Coleman, *Culture Debt Is One of the Most Toxic Threats to Business, and Your Startup Is Probably Victim to It*, 2019. [Online]. <https://www.inc.com/bernard-coleman/culture-debt-is-one-of-most-toxic-threats-to-business-your-startup-is-probably-victim-to-it.html>
- [55] F. Palomba, D.A. Tamburri, F.A. Fontana, R. Oliveto, A. Zaidman et al., “Beyond technical aspects: How do community smells influence the intensity of code smells?” *IEEE Transactions on Software Engineering*, Vol. 47, No. 1, 2018, pp. 108–129.
- [56] D. O’Keeffe, “An empirical case study of technical debt management: A software services provider perspective,” M. Sc. thesis, University of Dublin, 2017.
- [57] A. Dignan, *How to Eliminate Organizational Debt – Building Strong Organizations*, 2017. [Online]. <https://culturestars.com/how-to-eliminate-organizational-debt>
- [58] N. Nagappan, B. Murphy, and V. Basili, “The influence of organizational structure on software quality: an empirical case study,” in *Proceedings of the 30th International Conference on Software Engineering*, 2008, pp. 521–530.
- [59] J. Cusick and A. Prasad, “A practical management and engineering approach to offshore collaboration,” *IEEE Software*, Vol. 23, No. 5, 2006, pp. 20–29.
- [60] C.R. De Souza and D.F. Redmiles, “The awareness network, to whom should i display my actions? and, whose actions should i monitor?” *IEEE Transactions on Software Engineering*, Vol. 37, No. 3, 2011, pp. 325–340.
- [61] K. Casey, *What causes technical debt – and how to minimize it*, 2020. [Online]. <https://enterprisersproject.com/article/2020/6/technical-debt-what-causes>
- [62] L.M. Hilty and B. Aebischer, “ICT for sustainability: An emerging research field,” *ICT Innovations for Sustainability*, 2015, pp. 3–36.

Appendix A. Primary studies

List of primary studies

- [S1] J. Yli-Huumo, A. Maglyas, and K. Smolander, “How do software development teams manage technical debt?—an empirical study,” *Journal of Systems and Software*, Vol. 120, 2016, pp. 195–218.
- [S2] A. Martini and J. Bosch, “Revealing social debt with the caffee framework: An antidote to architectural debt,” in *IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, 2017, pp. 179–181.
- [S3] K. Ladewig, *The dark side of working from home | The Startup*, 2019. [Online]. <https://medium.com/swlh/social-debt-17bf03a269a>
- [S4] S. Sachdev, *Cultural Debt*. [Online]. <https://www.careerfair.io/reviews/cultural-debt>
- [S5] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing technical debt in software engineering,” in *Dagstuhl Reports*, Vol. 6, No. 4. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.

- [S6] A. Melo, R. Fagundes, V. Lenarduzzi, and W.B. Santos, “Identification and measurement of requirements technical debt in software development: A systematic literature review,” *Journal of Systems and Software*, 2022, p. 111483.
- [S7] D.A. Tamburri, “Software architecture social debt: Managing the incommunicability factor,” *IEEE Transactions on Computational Social Systems*, Vol. 6, No. 1, 2019, pp. 20–37.
- [S8] T. Mejía, *Social Debt the difficult commitment=*, 1998. [Online]. <https://www.socialwatch.org/book/export/html/10623>
- [S9] S. Vinsennau, *Decouple To Innovate How Federal agencies can unlock IT value and agility by remediating technical debt*, 2016. [Online]. https://www.accenture.com/_acnmedia/PDF-85/Accenture-Decoupling-to-Innovate.pdf
- [S10] G.S. Tonin, *Technical debt management in the context of agile methods in software development*, Ph.D. dissertation, University of Sao Paulo, 2018.
- [S11] T. Besker, H. Ghanbari, A. Martini, and J. Bosch, “The influence of technical debt on software developer morale,” *Journal of Systems and Software*, Vol. 167, 2020, p. 110586. [Online]. <https://www.sciencedirect.com/science/article/pii/S0164121220300674>
- [S12] M. Bellotti, *Hunting Tech Debt via Org Charts. Knowing where to look for problems*, 2021. [Online]. <https://bellmar.medium.com/hunting-tech-debt-via-org-charts-92df0b253145>
- [S13] J. Yli-Huomo, *The role of technical debt in software development*, Ph.D. dissertation, Lappeenranta University of Technology, 2017.
- [S14] S. Priestnall, *What is Process Debt?*, 2020. [Online]. <https://www.linkedin.com/pulse/what-process-debt-stevepriestnall>
- [S15] Z. Dargó, “Technical debt management: Definition of a technical debt reduction software engineering methodology for smes,” Master’s thesis, Aalto University, School of Science, 2019.
- [S16] J.A. Miko, *Collaboration Strategies to Reduce Technical Debt*, Ph.D. dissertation, Walden University, College of Management and Technology, 2017.
- [S17] B. Sutton and P. Michelman, *Overcoming Cultural and Technical Debt*, MITSloan Management Review, 2019. [Online]. <https://sloanreview.mit.edu/audio/overcoming-cultural-and-technical-debt/>
- [S18] D.A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, “What is social debt in software engineering?” in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 93–96.
- [S19] D.A. Tamburri, P. Kruchten, P. Lago, and H.v. Vliet, “Social debt in software engineering: insights from industry,” *Journal of Internet Services and Applications*, Vol. 6, 2015, pp. 1–17.
- [S20] D. O’Keeffe, “An empirical case study of technical debt management: A software services provider perspective,” Master’s thesis, University of Dublin, 2017.
- [S21] B. Coleman, *Culture Debt Is One of the Most Toxic Threats to Business, and Your Startup Is Probably Victim to It*, 2019. [Online]. <https://www.inc.com/bernard-coleman/culture-debt-is-one-of-most-toxic-threats-to-business-your-startup-is-probably-victim-to-it.html>
- [S22] A. Dignan, *How to Eliminate Organizational Debt – Building Strong Organizations*, 2017. [Online]. <https://culturestars.com/how-to-eliminate-organizational-debt>
- [S23] D.A. Tamburri and E. Di Nitto, “When software architecture leads to social debt,” in *2015 12th Working IEEE/IFIP Conference on Software Architecture*. IEEE, 2015, pp. 61–64.
- [S24] A. Martini, V. Stray, and N.B. Moe, “Technical-, social-and process debt in large-scale agile: an exploratory case-study,” in *Agile Processes in Software Engineering and Extreme Programming–Workshops: XP 2019 Workshops, Montréal, QC, Canada, May 21–25, 2019, Proceedings 20*. Springer, 2019, pp. 112–119.
- [S25] A. Martini, T. Besker, and J. Bosch, “Process debt: A first exploration,” in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2020, pp. 316–325.
- [S26] S. Zimmeck, *Social Debt: Why Software Developers Should Think Beyond Tech*, 2019. [Online]. <https://sebastianzimmeck.medium.com/social-debt-why-software-developers-should-think-beyond-tech-df665d8401a5>

- [S27] P. Phelan, *Is “Cultural Debt” hurting your organization’s growth? (Part 1)*, 8W8 Global Business Builders. [Online]. <https://www.8w8.com/is-cultural-debt-hurting-your-organizations-growth-part-1/>
- [S28] J. Trouw, *Organisational debt an analogy*= <https://www.linkedin.com/pulse/organisational-debt-analogy>, 2021. [Online]. <https://www.linkedin.com/pulse/organisational-debt-analogy-jaap-trouw>
- [S29] D. Tamburri, *From Technical to Social Debt: Analyzing Software Development Communities using social networks analysis*, 2015. [Online]. <https://www.slideshare.net/DamianTamburri/from-technical-to-social-debt-analyzing-software-development-communities-using-socialnetworks-analysis>
- [S30] J. Holvitie, D. Tamburri, A. Goldman, S. Fraser, W. Snipes et al., “Social debt in software engineering: Towards a crisper definition,” Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Seminar 16162, 2016. [Online]. <https://www.dagstuhl.de/16162>
- [S31] R. Kazman, “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)*. IEEE, 2019, pp. 1–1.
- [S32] T. Dreesen, P. Hennel, C. Rosenkranz, and T. Kude, ““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*, 2021, p. 6826.
- [S33] F. Palomba, A. Serebrenik, and A. Zaidman, “Social debt analytics for improving the management of software evolution tasks,” in *16th Edition of the BELgian-NEtherlands Software EVOLution Symposium (BENEVOL 2017)*. CEUR-WS.org, 2017, pp. 18–21.
- [S34] M. Eaden, *When Testers Deal With Process Debt: Ideas to Manage It And Get Back To Testing Faster*, 2017. [Online]. https://www.ministryoftesting.com/articles/8d79968d?s_id=15650023
- [S35] E.A. Caballero Espinosa, *Understanding Social Debt in Software Engineering*, Ph.D. dissertation, The University of Alabama, 2021.
- [S36] W. Cunningham, “The wycash portfolio management system,” *ACM SIGPLAN OOPS Messenger*, Vol. 4, No. 2, 1992, pp. 29–30.
- [S37] I. Kavas, “Don’t go back to the office without fixing your process debt,” *Forbes*, 2021. [Online]. <https://www.forbes.com/sites/forbestechcouncil/2021/01/04/dont-go-back-to-the-office-without-fixing-your-process-debt/?sh=1afbfe9b74a4>
- [S38] *Organisational Debt and Why It Makes Digital Transformation Hard*, CloudThing, 2022. [Online]. <https://cloudthing.com/organisational-debt/>
- [S39] N. Almarimi, A. Ouni, and M.W. Mkaouer, “Learning to detect community smells in open source software projects,” *Knowledge-Based Systems*, Vol. 204, 2020, p. 106201. [Online]. <https://www.sciencedirect.com/science/article/pii/S0950705120304226>
- [S40] M. Hosking, *Transformation troubles and non-technical debt*, 2017. [Online]. <https://www.linkedin.com/pulse/transformation-troubles-non-technical-debt-matt-hosking/>