Computer Science and Information Systems 22(1):369–399 https://doi.org/10.2298/CSIS240411012A

Unraveling the Organisational Debt Phenomenon in Software Companies*

Muhammad Ovais Ahmad[1], Osama Al-Baik[2], Abdelrahman Hussein[3], and Mwaffaq Abu-Alhaija[4]

[1] Department of Computer Science, Karlstad University,
Karlstad, Sweden
{ovais.ahmad}@kau.se

[2] Department of Software Engineering, Princess Sumaya University for Technology
Amman, Jordan
o.albaik@psut.edu.jo

[3] Department of Software Engineering, Al-Ahliyya Amman University
Amman, Jordan
a.husein@ammanu.edu.jo

[4] Department of Computer Science, Applied Science Private University
Amman, Jordan
m_abualhaija@asu.edu.jo

Abstract. Organizational debt (OD) is a major challenge to software organizations that seek to maintain agility, adaptability and sustainable competitiveness in the dynamic business environment. OD can be refers to suboptimal decisions, outdated procedures, misaligned structures and cultural barriers that limit an organization's ability to adapt and innovate quickly. This multifaceted process includes several factors, including ineffective workflows, knowledge silos, cultural issues, and inadequate resource utilization. When organizations are focused on short-term gains at the expense of long-term organizational health, the symptoms of organizational dysfunction may manifest themselves as reduced output, reduced quality and customer satisfaction. This study aims at assessing the extent of knowledge, factors, and consequences of organizational maladjustment in software organizations. A survey performed in three organizations identified several highly visible issues such as complex code, inconsistent UI, unclear requirements, and outdated processes. These themes often emerge due to exponential growth, prioritizing speed over quality, lack of cooperation and coordination, and outdated processes. The adverse effects of OD are comparable to technical debt, affecting the maintainability, user experience, and project management. This study also offers strategies for identifying, assessing, and mitigating OD through a combination of quantitative metrics, user feedback, and interdepartmental collaboration. By fostering a culture of continuous improvement, open communication, and cross-functional alignment, organizations can proactively address OD and create an environment conducive to innovation, quality, and customer-centricity.

Keywords: Organisational debt, software development, technical debt, process debt, nontechnical debt, agile, organisational agility.

* This is an extended version of our conference paper [6].

1. Introduction

The software development landscape is in a constant state of flux, demanding that organizations remain agile, adaptable, and competitive. Tight deadlines often lead to prioritizing features over code quality, and inadequate documentation creates a technical debt burden [2] [33], [34]. As market needs evolve and technological advancements accelerate, companies must navigate a complex interplay of technical and non-technical factors to sustain innovation and deliver value to customers [2]. Maintaining software in a suboptimal state, both technically and non-technically, can compromise its reliability and efficiency [33] [50] [2].

Short-term gains and a series of less-than-ideal decisions (e.g., prioritizing speed over proper documentation) can lead to the accumulation of "organizational debt" (OD) [2]. While technical debt (TD) has been extensively explored in academic literature [2][49][16], non-technical debt (NTD) remains less understood [33] [2][4][9]. NTD encompasses process debt, social debt, and people debt [13]. OD extends beyond TD and encompasses NTD, including process, social, and people debts. Liu et al. define "organizational debt" more narrowly, encompassing only social and process debt [36]. While TD focuses on codebase and architecture, OD covers broader organizational inefficiencies such as outdated processes and misaligned structures. NTD represents the unintended consequences of prioritizing short-term expediency over long-term sustainability [37]. This results in a gradual accumulation of outdated processes, misaligned structures, and cultural barriers within an organization. OD refers to the difference between a company's strategic plans and its actual ability to implement them in view of the ever-changing market needs [6] focusing on both technical and non-technical barriers to agility and innovation. This multifaceted concept extends beyond technical debt, encompassing a broad range of factors that can hinder an organization's ability to adapt, innovate, and maintain a competitive edge [36] [33][34][6]. As software companies grapple with the challenges posed by OD, a comprehensive understanding of its causes, consequences, and potential mitigation strategies becomes crucial. Recognizing the various manifestations of OD, such as inefficient workflows, knowledge silos, cultural misalignments, and inadequate resource allocation, allows organizations to take proactive steps. These steps can foster an environment conducive to continuous improvement, open communication, and cross-functional collaboration.

This study expands on the findings of our previous multivocal literature review (MLR) presented at the 39th ACM/SIGAPP Symposium on Applied Computing, Track on Lean and Agile Software Development [6]. This study seeks to provide insight into the impact of OD on software companies by investigating IT professionals' views on this issue. To this end, we will use data collected from a survey conducted in three different organizations in different fields. Moreover, this research aims to explore the ways of detecting, evaluating, and managing OD by using a set of quantitative measures, feedback from the users, and collaboration between departments. When an organization encourages learning and development, communication, and collaboration, they can prevent OD and establish a culture that supports change, quality, and customer focus. Finally, this approach can help achieving sustainable business performance and market sustainability in the context of the software development industry.

This study is organized into seven sections. Section 2 provides background information relevant to the research goal. Section 3 details the research methodology employed, including research design, data collection methods, and data analysis procedures. Section 4 presents result of our study. Section 5 addresses potential limitations of the study that could affect the validity of findings. Section 6 provides a discussion of results, interpreting them in the context of existing knowledge and highlighting their significance. Finally, Section 7 offers concluding remarks, summarizing the key findings and outlining potential future research directions.

2. Background

Software development is a complex process. Its complexity arises from the relationships and interactions among a large number of stakeholders, which have unpredictable behaviour. Additionally, each project is unique, with specific challenges stemming from the diverse backgrounds of individuals involved and the problem domain. Since technology and organisational processes are constantly evolving, these factors interact in unpredictable ways. Agile and Lean processes prioritize frequent delivery of value to customers by eliminating waste that hinders productivity and quality [10] [7] [11]. Treating team members as interchangeable 'resources' and constantly swapping them in and out of projects is ineffective [85]. In addition, they often blame the team for not following the process when problems occur. The "Just get it done" approach is the primary driver of debt in software development.

In 1992, Cunningham [21] introduced the concept of technical debt (TD) and described how "Shipping first-time code is like going into debt. A small debt accelerates development as long as it is promptly paid back with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on a not-quite-right code counts as interest on the debt. Entire engineering organizations can be brought to a standstill under the debt load of an unconsolidated implementation." Avgeriou et al. [15] elaborated on this definition as TD is a collection of design or implementation constructs that are expedient in the short term but set up a technical context that can make future changes more costly or impossible. TD represents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability. Other types of TD include architectural debt, code debt, and test debt [33][34].

Extensive research has been carried out on TD from a wide range of perspectives, including efforts, tools, types, management strategies, architectural aspects, agile development and prioritisation [2][49][40]. Rios et al. [50] reported 78 causes and 66 effects of TD. The main reported causes were deadlines, obsolete or incomplete documentation, lack of technical knowledge, inappropriate planning, lack of knowledge of technology, a focus on higher production at the expense of quality, and lack of a well-defined process. Avgeriou et al. [15][16] conducted comparisons of tools used for measuring TD, assessing their features, popularity, and empirical validity, and found that TD results in reduced maintainability and quality, amplified project costs, financial losses, bad code refactoring, team overload, difficulties in implementing the system and stakeholder dissatisfaction. Matkovic et al. [42] highlighted challenges contributing to technical debt, such as inadequate identification, monitoring, and refactoring of non-functional requirements, emphasizing the importance of business analysis and architectural decisions. Marjanović et al. [39] emphasized the need for updating organizational processes, developer training, and tools to improve application security. Poth et al. [47] discussed the issue of organizational silos in large enterprises, proposing self-service kits as a potential solution for scaling knowledge to autonomous teams.

Khomyakov et al. [31] specifically explored TD measurement and analysis tools, focusing on quantitative methods that could be automated. Besker et al. [18] address architectural TD and combined research efforts to produce new insights with a specific interest in the topic. The findings indicate a lack of efficient management principles for architectural debt. In agile software development, Behutiye et al. [17] stress the significance of timely delivery and design matters, which may precipitate TD pitfalls. The adverse outcomes of TD in agile software development comprise reduced output, system quality, and escalated maintenance costs. Lenarduzzi et al. [34] found that code debt and architectural debt are the most common types of TD studied. Nevertheless, research into other types of TD, including test debt and requirement debt, was limited.

The Danish government faced challenges as a third of its critical IT systems were inadequate, lacking proper documentation, security, and requiring software and hardware upgrades [26]. Sweden's government agencies also struggled with outdated IT systems, affecting 70 percent of their setups [14]. According to Klinger et al. [33] "the decision to acquire TD is not made by technical architects, but rather by non-technical stakeholders who cause the project to acquire new technical debt or discover existing TD that wasn't previously visible" (p. 35). Klinger's highlight an important aspect of OD, which is that non-technical stakeholders play a crucial role in the accumulation of debt within organizations. Software project success or failure depends on both technical and non-technical factors. TD has received more attention in research and practice compared to non-technical debt (NTD) in software development [2].

NTD refer to a range of debt types that can arise in software companies, including process, social, people, cultural [2] [8] [6]. NTD stems from suboptimal decisions that prioritise short-term gains over long-term sustainability, which can impede software development activities. In recent systematic review [2] to explore NTD in software engineering, focusing on social, process, and people debt. Their study analyzed 42 primary studies and identified 29 causes and 31 effects of NTD. It is highlighted that NTD is often interconnected and can significantly influence software development activities. The study revealed that NTD can lead to reduced productivity, decreased software quality, and increased development costs. It is important address NTD alongside TD for more effective software development and propose strategies for managing NTD, including improved communication, knowledge sharing, and process optimization. Our literature review provided detailed background to various NTD types such as social, process, and people debts in software engineering [2] [8] [8] [6]. Appendix B provided glossary of various debt types, below we only highlight definitions and key leading causes, we reported details in [2].

Process debt is a "suboptimal action or event with short-term advantages but long-term detrimental effects" [17]. Teams incur process debt when conducting stand-up meetings solely for status updates to leaders, limiting the meetings' full potential. The leading causes of process debt are lack of process competencies, process divergence, and external dependencies (i.e. technology, tools and external trends) [2].

People debt refers to "people issues that, if present in the software organization, can delay or hinder some development activities", for example, expertise focussed on a few personnel, as an effect of delayed training and hiring [2]. People's debt causes examples are lack of knowledge, experience, commitment and lack of psychological safety, inadequate management decisions and low developer morale [2].

"Cultural debt is a technical decision that borrows against the organization's culture. Such decisions can introduce team divisions, deteriorate communication or even weaken leadership effectiveness". The leading causes of cultural debt include hiring the wrong people, dismissing complaints, charging discrepancies, and giving unequal rewards.

Social debt refers to "the presence of sub-optimality in the development community, which causes a negative effect on software development" [52]. An example of social debt is lone wolf, and radio silence/bottleneck [2] [5]. Social debt causes are communication, collaboration and coordination challenges, and community smells [2] [9].

Introducing new policies for hybrid work can expose managers to the potential of incurring OD [37]. These policies have the power to shape the norms that develop among employees. If managers fail to address unfavourable norms promptly, they will be faced with the consequences of this OD in future. Liu et al [37] research identified 23 mechanisms, grouped into eight general coordination categories, and assessed their impact on aspects such as shared mental models, team coordination, team cohesion, and team learning.

3. Research Method

In this section, we present the methodology used for our research. Our initial step involved conducting a multivocal literature review (MLR) focused on OD within software companies, as we previously reported in our work [6].

MLR typically encompass a wide array of easily accessible material. Given that this approach incorporates diverse perspectives, including non-academic sources, it is crucial to establish a clear objective before proceeding. In our case, the primary aim of the MLR was to gain insight into how software industry professionals conceptualize and describe the phenomenon of "organizational debt".

We adhered to the MLR guidelines proposed by Garousi and Mantyla [6][7] [12] [53]. Considering the novelty of "organizational debt" as a topic and the scarcity of academic literature, we opted to use Google search engine rather than traditional academic databases (such as SpringerLink, ACM, or IEEE eXplore). Google search capabilities proved effective in locating relevant grey literature. In June 2023, we conducted our search using the query ("organizational debt" OR "organisational debt" AND "software"). This search string was based on our study scope, incorporating both "population" and "intervention" terms. We included "software" to ensure coverage of studies discussing software, its development, engineering aspects, or software-intensive products, services, and systems. The term ("organizational debt" OR "organisational debt") was employed to capture all OD-related papers.

From the search results, we examined the first 120 sources. This approach aligns with Haddaway et al. [27] observation that grey literature researchers often focus on the top 50 results. We scrutinized each link, recording relevant findings in a Microsoft Excel spreadsheet. We excluded sources that were non-English, videos, advertisements, catalogs, duplicates, research profiles, or outside the software engineering domain. Our final selection comprised 22 blog posts out of the initial 120 sources, which are documented in [6]. Notably, our search strategy did not yield any relevant scientific articles.

For data analysis, we employed thematic analysis techniques, focusing on four key themes: OD definitions, causes, consequences, and mitigation strategies. Within each theme, we conducted an open analysis of the data. It is worth noting that we didn't use pre-defined themes or codes for causes, effects, and mitigation strategies; instead, these emerged organically from the data. The initial MLR findings were published in [6]. Building upon this foundation, we subsequently expanded our research through an exploratory survey.

The survey instrument itself was developed based on our OD literature review in software engineering, as reported [6]. The development process adhered to best practices for exploratory survey design, as suggested by Kitchenham and Pfleeger [32]. To ensure methodological rigor, the survey followed a structured series of steps, including expert reviews and pilot testing. These steps were crucial in refining the survey's content and ensuring its alignment with the study's objectives. However, it is important to clarify that the pilot testing was distinct from the main survey and was conducted solely to test the clarity and reliability of the instrument, not to collect final survey data.

Expert reviews were conducted by professionals with over ten years of experience in software engineering and project management, ensuring that the instrument was reviewed by domain experts. Their feedback resulted in adjustments to question phrasing, the inclusion of additional response categories, and refinement of the open-ended questions. The profile of these experts included senior software engineers and project managers. Experts recommended rephrasing some of the closed-ended questions to avoid ambiguity, improving the clarity of operational definitions related to OD, and expanding the scope of questions to cover a broader range of organizational levels (e.g., operational, managerial, executive). Additionally, they suggested restructuring open-ended questions to encourage more detailed and insightful responses from participants.

After incorporating the experts' feedback, the survey was subjected to pilot testing with a small sample of participants to assess the overall clarity and usability of the instrument. The pilot test focused on understanding whether participants could easily interpret the questions and provide responses that align with the survey objectives. Importantly, the data collected from the pilot test were not included in the final survey results, as the purpose of this phase was to refine the instrument rather than to gather actionable data. Adjustments based on pilot test feedback included simplifying technical jargon and restructuring several questions to improve response accuracy.

The respondent pool included project managers, senior managers, and other key stakeholders in software organizations, ensuring a broad representation of perspectives within the software engineering industry. The survey consisted of 17 questions divided into six sections:

1. Participants background
2. Organizational Information,
3. Awareness and Identification of Organizational Debt
4. Causes and Consequences of Organizational Debt
5. Mitigation Strategies and Practices
6. Additional Comments and Feedback.

The survey included a mix of closed-ended questions and open-ended questions to gather both quantitative and qualitative data (See Appendix A, for survey questions).

The survey was distributed electronically to nine different software organizations across various regions. The organizations were selected based on their size (medium-sized) and willingness to participate in the study. Responses were received from five of the nine organizations contacted. Two organizations were excluded from the analysis due to an insufficient number of responses (three or fewer responses received). The remaining three organizations were located in Jordan (JoOrg), Canada (CaOrg), and the United States (UsOrg). These organizations were categorized based on international definitions for company size, where medium-sized companies are defined as having between 100 and 500 employees, as outlined in Table 3. In addition to the steps outlined in the research

Table 1. Participating Organizations

ID	Size	Responses	Included responses	Domain	Location
JoOrg	200-250	16	13	Education Technology	Jordan
CaOrg	300-350	21	16	E-Commerce	Canada
UsOrg	200-250	18	13	Healthcare	United States

method, the study also addressed validity threats by implementing strategies such as peer debriefing and member checking. Peer debriefing involved discussions with colleagues in software engineering to validate the coding framework, while member checking ensured that the participants' responses were accurately interpreted by returning initial results to a subset of respondents for verification. Furthermore, data triangulation was employed, comparing qualitative insights with MLR data to improve the robustness of the analysis.

3.1. Data Coding and Analysis

The closed-ended responses were coded numerically for quantitative analysis. In accordance with best practices for survey analysis in software engineering research [32], descriptive statistics such as frequencies and percentages were employed to provide an initial quantitative overview of the data. For open-ended questions, a thematic analysis approach was employed to identify common themes and patterns in the responses. We applied a deductive thematic analysis, as described by Kitchenham and Pfleeger [32] and Braun and Clarke [20], which allows researchers to analyze data using predefined theoretical frameworks to guide the process. This approach is particularly useful when examining data through the lens of established theories. In contrast to inductive methods, where themes emerge directly from the data, deductive analysis facilitates a more targeted examination of key research questions and theoretical constructs [3][20].

We compared these themes with the original ones that were reported by Ahmad and Al-Baik [6] focusing on their relevance to OD. Specifically, the open-ended responses were first read thoroughly to gain familiarity with the data. In line with qualitative research guidelines, the initial coding process involved identifying recurring ideas, concepts, or patterns in participants' responses, which were then organized into codes. These codes were subsequently refined and consolidated into broader themes that aligned with the study's research objectives. The coding process was iterative, with codes and themes being revised as new insights emerged from the data. In order to ensure the robustness of our findings, methodological triangulation was applied, as recommended by Kitchenham and Charters [32]. This involved comparing and integrating data from MLR and open-ended survey responses to generate a holistic understanding of OD in software organizations.

To enhance the credibility and trustworthiness of the qualitative research, peer debriefing and member-checking procedures were rigorously applied. Peer debriefing involved discussions with an independent coder knowledgeable in OD, allowing for the verification of the identified themes and codes, thereby improving the credibility and confirmability of the analysis. Additionally, a purposive sample of participants was asked to review the interpretations of their responses and provide feedback, a process known as member checking. This step further strengthened the validity of the qualitative data by ensuring that the interpretations accurately reflected participants' perspectives. Moreover, efforts were made to improve the reliability of the analysis by adhering to a systematic process of comparison between qualitative and quantitative data. This methodological approach, recommended by Kitchenham and Pfleeger [32], allows for a more nuanced understanding of the phenomenon under investigation and ensures a reliable synthesis of both qualitative and quantitative data. By using both data types, the study was able to address the research questions comprehensively, reducing bias and enhancing the validity of the conclusions.

4. Results

In this section, we first review the trends of OD awareness and try to refine the definition of OD that was established in our MLR [6]. Secondly, we discuss OD causes and effects as reported by the survey participants and examine the alignment to the ones that were previously reported in Ahmad and Al-Baik [6]. Thirdly, we discuss OD identification, assessment metrics, and mitigation strategies as reported by the research participants. Finally, we provided future directions and potential OD research agenda.

4.1. Results from Literature Review

OD Concept and Definition In our conference papers, we systematically assembled twenty-two blog postings centred on OD, authored by software engineering professionals. The concept of OD has gained increasing attention in recent years, as evidenced by the growing number of publications on the topic. A clear pattern emerges as six out of 13 articles were written in the past couple of years 2020-2023, highlighting the increasing discussion surrounding the OD phenomenon.

The conceptual roots of OD trace back to 2015 when Steve Blank extended the metaphor of TD and characterized OD as "worse" [19]. Notably, this idea finds antecedents in Ben Horowitz's conceptualization of "management debt" dating back to 2012 [6]. Subsequent contributors like Dignan broadened the scope, asserting that OD is not confined to startups but holds significance on a broader organizational scale. The MLR results [6] offer a plethora of OD definitions, reflecting the nuanced perspectives of software industry professionals; Table 1 below shows a summary of these definitions.

We synthesis these definitions and proposed: "OD refers to the difference between a company's strategic plans and its actual ability to implement them in view of the ever-changing market needs" [6]. OD, however, is a multifaceted phenomenon that encompasses a wide range of technical and non-technical aspects within an organization. While

TD primarily refers to the accumulation of shortcuts and compromises made in the codebase, OD encompasses a broader scope that includes inefficient processes, outdated policies, cultural misalignments, and suboptimal organizational structures. This reveals a nuanced understanding of OD that goes beyond the traditional concept of TD.

OD Causes and Consequences: As reported by Ahmad and Al-Baik [6], Table 2, summarises the OD causes, consequences and mitigation strategies. It was also reported that OD is not frequently measured in software organizations, nor valued because it's extremely expensive [46]. Organisations that are not responsive to change accumulate OD: decreased agility, reduced competitiveness, negative effect on employee morale and increased resistance to change, and eventually lead to inefficiency and bureaucracy [6]. At a higher level, OD manifests itself in two ways:

1. Obsolescence occurs when structures and policies become unfit for purpose. While rules and structures may have initially served the organisation well, they can become entrenched and inflexible, impeding adaptability and innovation.

2. Accumulation occurs when policies and procedures are constantly added but never removed. When employees are unsure of their responsibilities and accountabilities, it leads to confusion and inefficiency. Overcompensating some employees while neglecting others can create perceptions of unfairness, leading to demotivated teams and reduced overall productivity.

OD Mitigation Strategies: Identifying and mitigating organisational debt requires a systematic approach given its multifaceted nature. Organisational debt symptoms can be spotted through regular performance monitoring, employee surveys, and audits of processes [6]. A comprehensive evaluation of various organisational components is vital for deeper insights.

Quantitative performance metrics offer warning signs such as prolonged declines in productivity, increasing software defects, lags in new feature releases, product quality issues, and rising customer complaints. Comparing metrics over time and against competitors highlights underperformance. Periodic audits help assess process efficiency, redundancy, and alignment with objectives. Surveys and interviews to gather employee perspectives on pain points complement the top-down analysis. The utilization of both quantitative and qualitative data allows for the cross-validation of findings regarding the state of organizational components.

4.2. Practitioners Survey Results

Demographics: A total of 54 responses were collected from Jordan (JoOrg), Canada (CaOrg), and the United States (UsOrg). Majority of the responses were from 21 responses from CaOrg, followed by 18 responses from UsOrg and 16 responses from JoOrg. However, 3 responses from JoOrg, 5 responses from CaOrg, and 5 responses from UsOrg were excluded due to incomplete responses, resulting in a final sample of 42 included responses. The study included a diverse group of participants from various roles (See Table 4).

Table 2. OD Definitions, adopted from our MLR [6]

Proposed Definition in Grey Literature	Year
“OD is all the people/culture compromises made to ‘just get it done’ in the early stages of a start-up”	2015
“Organizational debt is any structure or policy that no longer serves an organization”	2020
“Organizational debt is the accumulation of changes and decisions leaders should have made but did not”	2016
“The interest companies pay when their structure & policies stay fixed and/or accumulate as the world changes”	2016
“Management Debt is incurred when you make an expedient, short-term management decision with an expensive, long-term consequence”	2022
“Organizations may intentionally or unintentionally incur organizational debt through management actions, governance process changes, internal process changes, or large-scale organizational changes when short-term advantages are sought at the expense of ’doing things right”	2017
“Organizational debt is the baggage that prevents people from delivering astonishing results”	2015
“Organizational debt - our organizations are also a good excuse to avoid changes, as we often look for someone who is going to help us, but we do not really want to give him or her the power to implement the changes”	2019
“Organizational debt is sibling of technical debt, for example a toxic culture, struggling leader etc.”	2020
“Organizational debt: things that should’ve been done to ensure health & efficiency, but weren’t”	2021
“Organizational debt, an analogy! During the execution of organizational changes (transformations, reorganizations, changes in ways of working etc.) shortcuts are taken that lead to frustration, more time and money etc. It’s the same thing as technical debt”	2021
“Organizational Debt is the interest companies pay when their structure and policies 1) stay fixed and/or 2) accumulate as the world changes”	2016
“Organizational debt is a holistic concept, and it is more than technical debt and also different from bureaucracy. Organizational debt is a networked concept that fosters the blame-free identification of cross-functional and cross-department weak points”	2023

Table 3. OD Causes, Consequences, and Mitigation Strategies from MLR [6]

Theme	Cause	Consequences	TD Analogy	Mitigation Strategies
Pressure to "Just Get It Done"	Root cause is urgency to complete tasks	Reduced Organizational Effectiveness: Decreased speed, capacity, engagement, flexibility, and innovation	Rushed to suboptimal solutions	Avoid rigid organizational charts, set work-in-progress limits, and use Kanban.
Organizational Growth Challenges	Compromises in personnel, culture, and leadership practices	Decline in Key Performance Metrics: Decreased agility, slow delivery, reduced competitiveness	Rapid growth without proper scalability measures	Encourage individual workers to tailor their roles for flexibility and adaptation
Cost and Measurement Issues	OD not frequently measured or valued due to perceived high cost	Diminished Agility, Slow Adaptation, and Reduced Competitiveness: Hinders responsiveness and efficiency	Avoiding investments in tools and processes.	Establish a feedback culture, nurturing internal trust and encouraging open feedback
Poorly Managed Change	Inadequate adaptation or ineffective change management initiatives	Poorly managed changes lead to resistance and ineffectiveness	Quick, temporary fixes instead of comprehensive changes	Practice continuous participatory governance: Involve people in co-designing roles, structures, and policies
Lack of Collaboration Culture	Failure to seek input from stakeholders during change initiatives	Structures and policies become unfit, hindering adaptability	Lack of collaboration, outdated systems and processes	Regularly monitor performance, behavioral, and innovation metrics
Siloed Change Efforts	Independent change efforts without coordination	Fragmentation, duplicated efforts, and lack of synergy due to independent change efforts	Fragmented solutions that don't integrate well	Implement programs like "process bounty" to encourage to highlight hindrances, fostering a culture of identifying and rectifying OD sources.
Leadership Decisions Avoiding Disruption	Hesitation to address underperforming employees and avoiding necessary changes	Mediocrity and resistance become accepted norms, hindering necessary changes	Leadership avoids necessary improvements	Offer leadership training, change management, etc.

The largest groups were Project Managers and Product Owners, each comprising 14.29% of the sample (6 participants each), and CEOs (4.76%). This focus on managerial-level participants ensures understanding of OD from a strategic perspective. Other roles included Software Developers (11.90%) and Quality Engineers (9.52%). This distribution highlights the variety of perspectives included in the study, ensuring a well-rounded view of challenges faced by different levels within software organizations.

Table 4. Participant Titles and Demographics

Title	**N**	**Percentage**
Quality Engineer	4	9.52%
Graphic Designer	2	4.76%
Network Engineer	3	7.14%
Project Manager	6	14.29%
UX Researcher	2	4.76%
Software Developer	5	11.90%
Business Analyst	4	9.52%
Security Officer	3	7.14%
Product Owner	6	14.29%
Customer Support Team Lead	2	4.76%
CEO	2	4.76%
Marketing Professional	3	7.14%
Total	42	100%

Table 5 provide an over of demographic characteristics of our survey participants. The majority of participants were male (66.67%), with females making up 33.33% of the sample. This distribution reflects the current gender disparity often seen in the software engineering industry. A significant proportion of participants held Graduate degree

Table 5. Demographic profile of respondents (N=42)

	Characteristic	**Frequency**	**Percentage**
Gender	Male	28	66.67%
	Female	14	33.33%
Age (years)	22-25	10	23.81%
	26-29	12	28.57%
	30-33	8	19.05%
	34 and above	12	28.57%
Education Level	Graduation (Bachelor)	22	52.38%
	Post-Graduation(Masters/MPhil)	14	33.33%
	Doctorate	6	14.29%
Software Development (Years)	Less than 1	4	9.52%
	1-2	8	19.05%
	3-5	12	28.57%
	6-7	10	23.81%
	7+	8	19.05%

(52.38%). Those with Post-Graduation degrees (Masters/MPhil) made up 33.33% of the sample, and 14.29% held Doctorate degrees. This variety in educational background underscores the breadth of expertise represented in the study. Participants' experience in software development varied widely.

The largest group had 3-5 years of experience (28.57%), followed by those with 6-7 years (23.81%). This mix of experience levels ensures that the study encompasses both relatively new and experienced professionals, providing a comprehensive view of how OD affects in their work.

The survey results also provide a view of OD awareness across various organizational levels. A significant proportion of respondents (60%) indicated familiarity with the concept of OD, highlighting a growing recognition of the term within the software industry. However, 25% of participants reported only partial familiarity, and 15% were entirely unfamiliar with OD. This suggest that while awareness is increasing, there remain knowledge gaps that could hinder effective OD identification and mitigation.

Moreover, when asked to describe their organization's current level of OD, 50% of respondents identified OD at the operational level, indicating that day-to-day processes, workflows, and procedures are where inefficiencies are most prevalent. 35% reported OD at the managerial level, suggesting that mid-level management also grapples with decision-making inefficiencies and misaligned structures.

Only 15% recognized OD at the executive level, reflecting that strategic misalignment exists, it is less prevalent compared to operational and managerial layers. These findings underscore the need for targeted OD mitigation strategies that focus on operational and managerial inefficiencies, where the debt is most concentrated, while ensuring alignment at the executive level to drive long-term organizational adaptability and competitiveness.

Table 6 presents a comprehensive overview of OD based on practitioners' survey. It outlines five key themes related to OD: complex codebase, inconsistent UI, unclear requirements, outdated content, and inability to adapt to change. For each theme, the table details the causes, consequences, TD analogies, and mitigation strategies. In the following sections, we will discuss each of these themes in detail.

We will present the specific causes that lead to OD, the consequences, and strategies that practitioners recommend for mitigating these challenges. This analysis will provide valuable insights into the nature of OD and offer practical approaches for addressing it in software development organizations.

OD Concept from Practitioners Survey: From a developers' perspective, OD can manifest as complex and poorly documented codebases, siloed knowledge, and a lack of clear code ownership, hindering maintainability and the integration of new features. As developers mentioned, "*Difficulty integrating new features due to complex code structure" (Jo8), "Difficulty meeting sprint goals due to last minute feature additions or changes*" (Us12), and "*Long code review times due to a large codebase*" are indicators of OD (Ca15).

For designers, OD may present itself as inconsistencies in the user interface, outdated design guidelines, and a lack of user research, leading to design debt and usability issues. A UX designer noted, "*Inconsistency in user interface (UI) elements across different parts of the application can indicate design debt.*" (Ca8) and "*Inconsistent brand application*

Table 6. OD Causes, Consequences, and Mitigation Strategies Based on Practitioners Survey

Themes	**Causes**	**Consequences**	**TD Analogy**	**Mitigation Strategies**
Complex codebases, Siloed knowledge, unclear code ownership	Rapid uncontrolled growth, prioritization of speed over quality, lack of clear communication & collaboration, outdated processes and structures	Reduced maintainability, difficulty integrating new features, longer development cycles	Impacting code quality & maintainability	Dedicating time for code refactoring, implementing code reviews, investing in test automation
Inconsistent UI, outdated design guidelines, lack of user research	Rapid growth, prioritization of speed over quality, lack of communication between design & development teams	Design debt leading to usability issues, user frustration	Impacting the user experience	Fostering collaboration between design and development teams, integrating user research findings into development process
Unclear requirements, inefficient communication	Lack of clear collaboration and cooperation, outdated processes and structures	Missed deadlines, frustrated teams, difficulty adapting to changes	Process debt impacting project management & communication	Streamlining workflows, establishing knowledge management practices, collaborating with product teams
Outdated content, misaligned messaging	Prioritization of speed over quality, limited resources	Ineffective communication, difficulty aligning content with product features	Process debt impacting content creation & management	Investing in content creation tools, establishing clear content creation workflows
Inability to adapt & change, hindered innovation	Lack of agility and understanding agile concepts, limited resources and competing priorities	Reduced competitiveness, difficulty responding to market shifts	Accumulated debt across various aspects hindering overall agility	Fostering a culture of continuous improvement, invest in staff training, promoting user-centric mindset

across different platforms, potentially indicating outdated design assets or style guides." (Jo4).

Project managers often grapple with OD in the form of unclear requirements, scope creep, and inefficient communication between stakeholders. In response to what symptoms of OD exist in the participating organizations, a project manager from CaOrg stated, "*Frequent scope creep and missed deadlines due to unclear project requirements*" (Ca5), while a project manager from JoOrg mentioned "*Frustration among team members due to inefficient workflows or unclear priorities.*" (Jo5), and from the UsOrg, a project manager stated "*Project teams struggling to adapt to changes due to inflexible project management methodologies.*" (Us5).

As for marketing and content writer specialists working in the software domain, OD can manifest as outdated content, misalignment between messaging and product features, and inefficient content creation workflows. A marketing professional from CaOrg mentioned, "*Difficulty aligning marketing campaigns with new product releases due to slow content creation workflows*" as an indicator of OD (Ca6), while another marketing professional from JoOrg mentioned "*Inconsistent branding across different platforms due to outdated style guides or lack of clear communication.*" as a symptom of OD (Jo7).

While TD primarily impacts the maintainability, evolvability, and quality of the codebase, OD has far-reaching consequences that can affect an organization's ability to adapt to change, innovate, and remain competitive in the market. OD encompasses a broader range of factors, including suboptimal decisions, outdated processes, misaligned structures, and cultural barriers, which collectively hinder an organization's agility and overall performance. Each of these terms within the OD definition can be understood as follows: Suboptimal decisions refer to short-term choices that prioritize immediate gains at the cost of long-term sustainability. Outdated processes reflect workflows and procedures that no longer align with current organizational goals or market demands. Misaligned structures refer to organizational hierarchies or teams that are no longer optimal for innovation and flexibility.

Finally, cultural barriers involve resistance to change or a lack of openness to new ideas, which stifles innovation. Understanding how each element of OD (suboptimal decisions, outdated processes, etc.) influences organizational behavior is crucial for developing holistic approaches that address both technical and non-technical challenges. By refining management practices to account for OD, organizations can incorporate strategies that specifically target these inefficiencies.

Considering the above discussion and definitions, OD is define as the accumulation of suboptimal decisions, outdated processes, misaligned structures, and cultural barriers that impede an organization's ability to adapt and innovate effectively. The inclusion of 'and' in the definition does not imply that all components must coexist for OD to be present. This definition integrates the technical and non-technical aspects of debt, making it applicable in practice. Rather, each component represents a potential source of OD, which can be assessed individually. For instance, an organization may assess its OD by evaluating its decision-making processes, structural efficiency, and adaptability to market changes. The organization's ability to adapt effectively can be defined as its capacity to restructure processes, workflows, and team dynamics in response to internal and external changes. Innovation, on the other hand, can be defined as the organization's capacity to introduce new products, services, or processes that deliver value in line with evolving market demands.

The gradual build-up of unresolved decisions and unimplemented actions that should have been undertaken by leaders. This accumulation results in structures, policies, and processes that no longer align with the organisation's objectives, ultimately impeding its progress and adaptability in the face of changing circumstances. This accumulation hinders an organization's ability to maintain optimal performance, agility, and responsiveness. As a result, OD exacerbates the gap between its intended strategic plans and the practical capacity to meet evolving market demands. The means that OD encompasses a broader range of organizational factors compared to TD and NTD. While TD focuses on code quality and maintainability, OD extends beyond technical aspects to include organizational inefficiencies that limit agility and competitiveness. This expanded view helps identify and mitigate debt beyond technical aspects, integrating social, process, and structural considerations.

OD Causes and Consequences: The survey results highlight that OD significantly affects an organization's agility, adaptability, and competitiveness. Agility refers to the ability to respond quickly to market changes, while adaptability denotes the capacity to adjust processes and structures in response to internal and external pressures. Competitiveness is defined as the organization's ability to maintain or improve its position relative to competitors. Rapid growth and the prioritization of speed over quality emerged as significant contributors to the accumulation of OD across multiple departments.

The survey found that organizations burdened by OD were less agile and adaptable, often facing challenges in maintaining competitiveness due to outdated processes and misaligned structures. Participants noted that OD led to slow decision-making, reduced innovation, and lower responsiveness to customer needs, further highlighting its detrimental impact on organizational performance.

In the development realm, respondents reported a primary cause of technical debt as Ca15 cited "*Rapid growth leading to pressure to ship features quickly, sometimes at the expense of code quality*" and Us1 testified "*In my experience, rapid feature releases with tight deadlines can lead to technical debt as corners are sometimes cut*" (Us1). Similarly, Ca7, a product owner, mentioned that "*Rapid feature releases with tight deadlines can lead to technical debt due to shortcuts taken.*" Even in the marketing department, Ca6 noted that "*Limited resources might lead to slow content creation and outdated materials*," indicating that the pressure of rapid growth can contribute to process debt.

Lack of clear communication and collaboration was another recurring theme highlighted by respondents from various roles. Ca5, a project manager, pointed out that ". *Us6, a UX researcher, stated that "Outdated user personas that don't reflect the current user base could cause the product to miss user needs*," suggesting a lack of communication between user research and development teams. Ca14, a customer support team lead, mentioned that "*Lack of clear communication between development and customer support teams could be hindering the creation of user-friendly self-service options*," leading to process debt in customer support. Us9, a network engineer stated that "*Limited communication between network operations and development teams can lead to mismatched network requirements*," resulting in debt that impacts performance and scalability.

Outdated processes and structures were also identified as significant contributors to organizational debt. Ca15, a software developer, noted that "*Lack of clear code ownership makes refactoring and maintaining code quality challenging*," indicating outdated processes for code management. Ca11, a business analyst, mentioned "*A lack of standardized processes across departments, potentially resulting in redundancies and inefficiencies*," as a potential source of process debt. Ca10, a security officer, cited "*Difficulty keeping up with the evolving threat landscape due to outdated security tools*," as a contributor to security debt.

Limited resources and competing priorities were frequently cited as factors hindering efforts to address organizational debt. Developers like Ca15 mentioned "*Limited resources and time for code refactoring and technical debt reduction*" as a factor contributing to technical debt accumulation. Ca16, a quality engineer, stated that "*Limited resources and time for QA to develop and maintain automated tests*" led to testing debt. Ca4, a graphic designer, noted that "*Limited time and resources for the design team to undertake large-scale organization initiatives*" hindered efforts to mitigate design debt. The limited resources were also reported on both, human and non-human resources, Us9 a network engineer from UsOrg, stated "*Frequent network slowdowns, bottlenecks during peak usage periods, and difficulty integrating new applications due to limited network capacity.*" and "L*imited staff resources might make network modernization a complex and time consuming process*."

Lack of agility and inability to adapt to market changes were also reported to introduce insufficient flexibility and incapacity to adjust to shifts in the market. The organization's outdated legacy systems and bloated processes hinder its ability to stay up with more agile competitors, making it slow and inflexible. The CEO of JoOrg attested that "Reduced agility and adaptability to changing requirements due to technical limitations" is a major OD impact on the organization (Jo2). The CEO of CaOrg acknowledged that "*I believe a strong focus on employee development and training on new technologies will help us stay agile and adapt to changing market demands*" (Us2).

These observations from various departments and roles illustrate the multifaceted nature of OD and highlight how factors such as rapid growth, communication barriers, outdated processes, and resource constraints can contribute to the accumulation of different types of OD within software organizations.

OD Identification Identifying the presence of OD within an organization often involves recognizing various indicators and symptoms across different domains. One key area is inefficient processes and outdated policies, where teams might encounter "*Difficulty finding clear and up-to-date documentation for internal systems" or "Repetitive tasks that could potentially be automated"* (Ca3), or "*Outdated design tools or asset libraries requiring workarounds and slowing down workflows*" (Jo4). Project managers may also observe "*Frequent scope creep and missed deadlines due to unclear project requirements*" (Ca5) or "*...team members struggle to adapt to changes due to inflexible project management methodologies*" (US5).

Technical debt and code quality issues can also serve as indicators of OD. Developers might face "*Difficulty integrating new features due to complex code structure*" (Ca7) or experience "*Long code review times due to a large codebase*" (Ca1). Additionally, teams may encounter "*difficulty troubleshooting issues caused by legacy code or features not documented clearly*" (Us11), further highlighting the presence of technical debt.

User experience and customer satisfaction issues can be telling signs of OD as well. Design teams might receive "*User complaints about unintuitive features or cluttered interface*" (Ca8) or "*Difficulty getting timely feedback on design concepts, leading to revisions and delays*" (Jo4), or "*Inconsistent brand application across different platforms due to lack of design guidelines*" (Us4).

QA teams could struggle with "*Difficulty reproducing reported bugs due to insufficient test data or unclear defect documentation*" (Ca16). Customer support personnel might encounter "*Frequent reports of slow loading times or bugs in the software*" (Ca13) experience "*Long resolution times for complex customer inquiries due to limited knowledge base information*" (Ca14), or experience "*Frequent escalations to developers due to lack of readily available solutions in the internal knowledge base*" (Us11).

Moreover, employee frustration and morale concerns can be indicative of OD within an organization. This might manifest as "*Employee surveys indicate frustration with slow internal tools and processes*" (Ca2), or *"Frustration among developers due to inconsistent coding standards and legacy code*" (Ca5). Support teams could also face *"Difficulty accessing clear and up-to-date product information"* (Ca13), while development teams may show signs of *"decreased developer morale and productivity due to rework caused by last-minute changes"*(Us12), contributing to overall employee dissatisfaction.

OD Assessment: To effectively assess and monitor OD levels, organizations can implement various metrics and processes. Project data analysis can involve *"Track[ing] project metrics like cycle time, defect escape rates, and time spent on bug fixes versus new feature development"* (Jo1, Jo5, Jo9, Ca5, Ca12, Us5, and Us6) or *"Analyz[ing] project data to identify potential bottlenecks, delays, and inefficiencies"* (Ca12) or *"... track[ing] project metrics like schedule variances, defect rates, and team member utilization... "* (Jo5).

User feedback and experience metrics can be gathered through "*regular user surveys and interviews*" (Ca9), tracking "*user task completion rates, error messages, and usability issues*" (Ca9), or monitoring "*website traffic, engagement metrics, and competitor marketing strategies*" (Ca6).

Customer satisfaction and support metrics can provide valuable insights, such as tracking "*customer satisfaction surveys and identify[ing] recurring pain points"* (Ca14), monitoring *"customer service ticket volume regarding repetitive issues"* (Ca2), or analyzing or *"website traffic and user engagement with different content"* (Us7).

Interdepartmental collaboration and knowledge sharing can be facilitated by *"establish[ing] clear communication channels and protocols for collaboration between departments*" (Ca11), conducting *"workshops and retrospectives to identify areas for process improvement"* (Ca12), and encouraging *"open feedback loops and knowledge sharing across teams and departments"* (Us1).

Additionally, organizations should focus on security and compliance monitoring by *"track[ing] the number of unpatched vulnerabilities and time taken to address them"* (Ca10), *"conduct[ing] regular penetration testing to identify potential security weaknesses"* (Ca10), and monitoring *"compliance with security regulations and industry best practices"* (Ca10).

By leveraging a combination of these metrics and processes, organizations can gain valuable insights into potential sources of organizational debt, identify areas that require improvement, and monitor the effectiveness of their OD mitigation strategies over time, enabling them to make informed decisions and implement targeted interventions to address organizational debt proactively.

OD Mitigation Strategies: Identifying and mitigating organisational debt requires a systematic approach given its multifaceted nature. Organizations can dedicate specific time periods for *"addressing technical debt and code refactoring"* (Ca15) to incrementally improve their codebase. Implementing rigorous "*code reviews with a focus on code quality, maintainability, and identifying potential issues early on*" (Ca1) can prevent the accumulation of future technical debt. Furthermore, "*investing in test automation tools and frameworks*" (Ca16) can reduce manual testing workloads and improve overall test coverage, ensuring better quality assurance.

Fostering practices that promote knowledge-sharing and collaboration among developers is also crucial. Respondents highlighted the benefits of " *focus on collaboration across teams is essential to prevent knowledge silos and maintain a healthy codebase*" (Us1), and the CEO of JoOrg stated "*Clearer communication and collaboration across teams could be beneficial*" for facilitating knowledge transfer and enabling better collaboration on codebase maintenance.

Addressing process debt often involves optimizing processes and improving knowledge management practices. "*Conducting process optimization workshops*" (Ca14) can help organizations streamline workflows, eliminate redundancies, and improve overall efficiency. Establishing "*robust knowledge management practices, such as creating comprehensive documentation, centralized knowledge bases, and facilitating knowledge sharing across teams*" (Us3) can mitigate process debt stemming from knowledge gaps and inefficient information flow.

Moreover, "*collaborating with product teams to identify opportunities for improving self-service resources based on customer feedback*" (Ca14) can reduce the burden on support teams and enhance the overall customer experience. "*Integrating user research findings into the product development process and establishing user research libraries*" (Ca9) can ensure that user needs are adequately addressed, preventing the accumulation of user experience-related debt.

Mitigating cultural debt requires fostering an environment that promotes continuous improvement, open communication, and cross-functional collaboration. "*Encouraging open communication, feedback loops, and cross-functional collaboration across teams and departments*" (Ca1) can break down silos, promote transparency, and leverage collective expertise to address organizational challenges holistically.

Additionally, "f*ostering a culture of continuous improvement through practices like retrospectives, workshops, and encouraging participatory decision-making*" (Ca12) and "*Conducting regular reviews to identify areas for improvement*" (Jo5), enables organizations to proactively identify areas for improvement and adapt to changing needs and requirements.

"*Investing in employee training and development programs*" (Ca2) can help upskill teams on new technologies, methodologies, and best practices, ensuring they remain agile and adaptable. Promoting a "*user-centric mindset by incorporating user research, design* *thinking workshops, and prioritizing user experience throughout the product development lifecycle*" (Ca8,) can prevent the accumulation of design debt and ensure a superior user experience.

To mitigate security debt, organizations should prioritize "*implementing security awareness training programs for all employees*" (Ca10) to promote best practices and reduce vulnerabilities. "*Piloting automated security patching processes*" (Ca10) can address vulnerabilities promptly and reduce the risk of security breaches. Furthermore, "*integrating security considerations earlier in the development lifecycle and investing in modern security tools and resources*" (Ca10) can enhance the overall security posture of the organization.

Across all these mitigation strategies, the importance of fostering a culture of continuous improvement, open communication, and cross-functional collaboration cannot be overstated. By embracing these principles, organizations can effectively combat organizational debt, improve operational efficiency, enhance product quality, and better adapt to changing market demands and customer needs.

4.3. Future Research Directions

While this study consolidated understanding of OD, several fruitful avenues exist for further investigation based on current knowledge gaps:

- Develop metrics to quantify OD, enabling rigorous tracking and benchmarking. Combine productivity data with indicators of culture, innovation, and TD. Establish validated scales to measure dimensions like employee engagement, psychological safety, organizational agility, and leadership effectiveness [24] [45] [38]. Statistical modelling can relate these metrics to OD.
- Conduct empirical studies on the impact of OD on workforce motivation, attrition, fatigue, and burnout [35] [44] [41]. Use questionnaires and ethnographic methods to gather insights. Relate debt to tangible individual performance metrics like productivity, absenteeism, and error rates.
- Investigate through case studies the relationship between OD and customer satisfaction [22] [13], especially in software-intensive service organizations. Survey data can correlate debt to metrics like call resolution times, complaint rates, churn, and net promoter scores.
- Examine through controlled experiments the role of OD in software project success/failure. Vary team structures and processes to reveal optimal configurations. Productivity, quality, cost, and schedule metrics assess performance.
- Estimate the economic costs of OD through case studies and cost modelling across software companies [30] [25]. Assess opportunity costs from delayed innovations. Relate to the total cost of ownership models.
- Explore whether TD quantification techniques [15][16][40] can be extended to provide estimates of OD. Comparative studies could be conducted to evaluate the precision of these techniques when applied to different types of debt within software organizations.
- Organizational forgetting [29],a concept largely overlooked in this research stream. Organizational forgetting refers to the processes through which an organization intentionally or unintentionally loses knowledge, practices, or routines that no longer

serve its strategic goals [29]. In the context of OD, fostering deliberate organizational forgetting can be instrumental in reducing debt by eliminating outdated processes, suboptimal decisions, and misaligned structures. Future work could explore how organizational forgetting can be systematically applied to debt reduction efforts, improving organizational adaptability and performance.
- Design field studies of interventions such as restructured teams, revised workflows, and new planning processes to validate OD mitigation techniques [43]. These studies should measure the before-and-after effects of such interventions to determine their efficacy in reducing OD. However, it is important to recognize that OD, especially when accumulated in the early phases of an organization's lifecycle, is often difficult to address without external engagement. Engaging external consultants [1], even in a limited capacity such as providing strategic nudges or asking critical questions, can play a pivotal role in mitigating debt that is deeply ingrained in organizational structures and processes. To minimize OD effectively, future research should explore the role of external consultants in the early stages of a company's lifecycle.

Further research to address these gaps will provide more rigorous, empirically grounded insights to guide debt management in practice. It represents an emerging interdisciplinary arena spanning management science, organizational behaviour, anthropology, and software engineering [54]. Collaboration between academics and industry practitioners is needed to develop context-specific strategies rooted in both theory and pragmatism [48]. There are rich possibilities for cross-pollination between disciplines to uncover novel solutions [23]. With organizational agility and adaptability growing more crucial in turbulent conditions [28], understanding how to minimize friction and debt represents the key to sustaining innovation and competitiveness [51].

5. Validity Threats and Limitations

The study presented in this manuscript has several potential validity threats and limitations that should be acknowledged. One significant threat is the external validity, which stems from the relatively small sample size, with only 42 responses obtained from three software organizations. Although these responses provided valuable insights into OD, a larger and more diverse sample, spanning different industries and geographical regions, would improve the generalizability of the findings.

The survey relied on self-reported data from participants, which may be subject to personal biases, perceptions, and interpretations. According to Kitchenham and Pfleeger [32], reliance on self-reported data introduces a potential threat to internal validity, as participants' responses may not always accurately reflect the true state of OD within their organizations. To mitigate construct validity threats, we employed several strategies, including peer debriefing, which was conducted with subject matter experts from diverse sectors of software engineering. These experts provided critical feedback on the coding and interpretation of the survey data, ensuring a robust analysis. Furthermore, member checking was performed by sharing the coded data with a subset of participants to validate the accuracy of the interpretation and ensure alignment with their original responses. This procedure enhanced the credibility of the qualitative findings. Additionally, data triangulation was applied by comparing the qualitative data from open-ended survey questions with the MLR results. Triangulating these two data sources helped corroborate the findings, although the inherent subjectivity in qualitative data analysis—where researchers' interpretation might introduce bias—cannot be entirely eliminated [32].

The study provided a cross-sectional snapshot of OD at a certain period of time. Kitchenham and Pfleeger [32] suggests, a cross-sectional design may limit the ability to observe long-term trends or changes in OD over time. A longitudinal study, where organizations are observed over an extended period, would provide a deeper understanding of how OD evolves and the long-term effectiveness of mitigation strategies. This is particularly relevant as OD is a dynamic construct that fluctuates with organizational and environmental changes. Therefore, the findings of this study are valid for the period during which the research was conducted, but future studies may need to revisit and update the results over time.

Despite these limitations, the study offers insights into the awareness, causes, consequences, and mitigation strategies for OD in software organizations. Future research should address these limitations by incorporating a larger, more diverse sample, utilizing multiple data collection sources, and adopting longitudinal research designs to better capture the evolving nature of OD and its effects on organizations.

6. Discussion

The notion of OD has attracted more attention by the software organizations as they face difficulties in sustaining the key values such as flexibility, adaptability, and competitiveness in the context of the changing environment. This paper set out to investigate the level of awareness, factors, impacts, and risk management measures of OD within software organizations so as to confirm the findings of Ahmad and Al-Baik [6].

The results of this study show that OD has far-reaching consequences beyond TD, particularly in its impact on organizational agility and competitiveness. Organizations that accumulate OD tend to have outdated structures and processes, which slow down their ability to respond to market changes and customer demands. Agility is reduced as organizations struggle to implement changes efficiently, while competitiveness suffers because of their inability to innovate and adapt quickly. Addressing OD through continuous improvement and cross-functional collaboration is crucial for maintaining organizational performance and market relevance. As for the issues, participants pointed out several signs and manifestations of OD in different areas such as ineffective workflows, outdated procedures, poor code quality, user interface and experience problems, and dissatisfied employees. This goes to show that OD affects an organization's performance and its capacity to create value for clients in a profound way. Table 7, offered omparison of literature review and survey results.

The factors that contributed to the development of OD were numerous, which included growth, poor communication and coordination, obsolete methods and structures, and resource constraints as some of the frequently mentioned reasons. The following factors lead to the creation of various types of debts like technical, process, and cultural debts that affect an organization's ability to be agile [2]. Interestingly, OD have negative impacts not only in technical domains but also in other domains like competitiveness, employee morale, resistance to change and inefficiency. This observation stresses the need for han-

Table 7. Comparison of literature review and survey results

Themes	Causes from review	Causes from survey	Similarities	Differences
Rapid Growth	Organizational growth challenges	Rapid uncontrolled growth	Both acknowledge rapid growth as a cause	Table 2 focuses on general organizational growth challenges, while Table 5 emphasizes uncontrolled growth
Poor Communication and Collaboration	Lack of collaboration culture, siloed change efforts	Lack of clear communication and collaboration	Both emphasize poor communication	Table 2 mentions siloed change efforts, while Table 5 highlights lack of communication between specific teams
Urgency to Complete Tasks	Pressure to "Just Get It Done"	Prioritization of speed over quality	Both recognize urgency as a cause	Table 2 refers to general urgency, while Table 5 focuses on speed over quality
Measurement and Cost Issues	OD not frequently measured or valued due to perceived high cost	Not found	Not found	Survey does not mention measurement and cost issues explicitly
Ineffective Change Management	Poorly managed change, leadership decisions avoiding disruption	Outdated processes and structures	Both address change management	Literature includes leadership decisions, while survey focuses on outdated processes
Technical and Design Debt	Not found	Complex codebases, inconsistent UI, outdated design guidelines	Both acknowledge technical aspects	Survey provides detailed technical causes not mentioned in the review

dling OD preventatively as it tends to spread throughout the organisation and threaten its sustainability.

This paper also sought to find out different approaches of recognizing, evaluating, and managing OD. People stressed the need to use number indicators, consumers' responses, and customer satisfaction rates to identify potential causes of OD, as well as, collaborate with different departments. These findings support previous studies that have been done on the best practices for the management of NTDs [2]. Additionally, fostering a culture of continuous improvement, open communication, and cross-functional collaboration emerged as crucial elements in combating OD effectively. Mitigating TD involved practices such as dedicated time for code refactoring, rigorous code reviews, and investment in test automation, consistent with established software engineering best practices [34]. Addressing debt required process optimization, robust knowledge management practices, and collaboration with product teams. Mitigating cultural debt necessitated promoting open communication, fostering a culture of continuous improvement, and prioritizing user experience throughout the product development lifecycle.

The findings of this study contribute to the growing body of knowledge on OD and provide valuable insights for software organizations seeking to enhance their long-term success. By recognizing the multifaceted nature of OD and implementing proactive mitigation strategies, organizations can minimize the accumulation of debt and foster an environment conducive to innovation, quality, and customer satisfaction. However, it is important to acknowledge the limitations of this study, such as the relatively small sample size, reliance on self-reported data, and the potential for response biases.

Future research should aim to address these limitations by employing larger and more diverse samples, triangulating data from multiple sources, and conducting longitudinal studies to further strengthen the understanding of OD and its implications. Additionally, the evolving nature of OD necessitates ongoing research to keep pace with the dynamic landscape of software development and organizational practices. Exploring the quantification of OD, estimating its economic costs, and examining the relationship between OD and factors such as workforce motivation, customer satisfaction, and project success could provide valuable insights for effective debt management. Collaboration between academia and industry practitioners is crucial in this endeavour, as it fosters the integration of theoretical frameworks and practical insights, ultimately leading to the development of context-specific strategies rooted in both rigour and pragmatism.

7. Conclusion

This study demonstrates that OD extends beyond technical debt, encompassing a broad range of organizational inefficiencies, including outdated processes, misaligned structures, and cultural barriers. OD integrates both technical and non-technical debt, influencing organization's ability to maintain agility, adaptability, and competitiveness. The accumulation of OD hinders decision-making, slows innovation, and weakens the organization's competitive edge. By recognizing and addressing OD through proactive measures such as interdepartmental collaboration and continuous improvement, organizations can reduce their debt and create an environment conducive to long-term success. This study has shed light on the multifaceted nature of OD, its causes, consequences, and potential mitigation strategies within software organizations.

The survey findings revealed a growing awareness of OD among software professionals, recognizing its impact on various aspects of an organization, including inefficient processes, code quality issues, user experience challenges, and employee frustration. The causes of OD were found to be diverse, with rapid growth, communication barriers, outdated processes, and resource constraints contributing to the accumulation of different types of debt, such as technical debt, process debt and cultural debt.

Notably, the consequences of OD extend beyond technical aspects, affecting an organization's competitiveness, employee morale, adaptability, and overall efficiency. This underscores the importance of proactively addressing OD to ensure long-term success and sustainability. The study explored various strategies for identifying, assessing, and mitigating OD, emphasizing the importance of leveraging quantitative metrics, user feedback, customer satisfaction data, and interdepartmental collaboration. Fostering a culture of continuous improvement, open communication, and cross-functional collaboration emerged as crucial elements in combating OD effectively.

Mitigating technical debt involved practices such as dedicated time for code refactoring, rigorous code reviews, and investment in test automation. Addressing process debt required process optimization, robust knowledge management practices, and collaboration with product teams. Mitigating cultural debt necessitated promoting open communication, fostering a culture of continuous improvement, and prioritizing user experience throughout the product development lifecycle. While this study contributes to the growing understanding of OD, it is essential to acknowledge its limitations and the need for further research. Future studies should aim to address these limitations by employing larger and more diverse samples, triangulating data from multiple sources, and conducting longitudinal studies to provide a more comprehensive and dynamic understanding of OD. Additionally, exploring the quantification of OD, estimating its economic costs, and examining its relationship with factors such as workforce motivation, customer satisfaction, and project success could yield valuable insights for effective debt management. Collaboration between academia and industry practitioners is crucial in this endeavour, fostering the integration of theoretical frameworks and practical insights to develop context-specific strategies rooted in both rigour and pragmatism.

In conclusion, the concept of organizational debt represents a critical challenge for software organizations striving for agility, adaptability, and long-term success. By recognizing the multifaceted nature of OD, implementing proactive mitigation strategies, and fostering a culture of continuous improvement and open communication, organizations can minimize the accumulation of debt and foster an environment conducive to innovation, quality, and customer satisfaction. Ongoing research and collaboration between academia and industry are essential to further advance the understanding and effective management of organizational debt in the software development domain.

Acknowledgments. This work was supported by the KK-stiftelsen (grant no. 20200253), Helge Ax:son Johnsons Stiftelse, and STINT - Swedish Foundation for International Cooperation in Research and Higher Education (IB2020-8720).

References

1. Adizes, I., Cudanov, M., Rodic, D.: Timing of proactive organizational consulting: difference between organizational perception and behaviour. Amfiteatru Economic 19(44), 232 (2017)

2. Ahmad, M.O., Gustavsson, T.: The pandora's box of social, process, and people debts in software engineering. Journal of Software: Evolution and Process p. e2516 (2022)
3. Ahmad, M.O.: A deep dive into self-regulated learning: Reflective diaries role and implementation strategies. Communications of the Association for Information Systems 54(1), 868–888 (2024)
4. Ahmad, M.O.: Psychological safety, leadership and non-technical debt in large-scale agile software development. In: 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS). pp. 327–334. IEEE (2023)
5. Ahmad, M.O.: 5g secure solution development and security master role. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering. pp. 629–633 (2024)
6. Ahmad, M.O., Al-Baik, O.: Beyond technical debt unravelling organisational debt concept. In: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing. pp. 802–809 (2024)
7. Ahmad, M.O., Dennehy, D., Conboy, K., Oivo, M.: Kanban in software engineering: A systematic mapping study. Journal of Systems and Software 137, 96–113 (2018)
8. Ahmad, M.O., Gustavsson, T.: Nexus between psychological safety and non-technical debt in large-scale agile enterprise resource planning systems development. In: Conference on Practical Aspects of and Solutions for Software Engineering. pp. 63–81. Springer (2023)
9. Ahmad, M.O., Gustavsson, T., Saeeda, H.: Customised roles in scrum teams for the development of secure solution. In: 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 362–369. IEEE (2023)
10. Al-Baik, O., Miller, J.: Kaizen cookbook: The success recipe for continuous learning and improvements. In: 2016 49th Hawaii International Conference on System Sciences (HICSS). pp. 5388–5397. IEEE (2016)
11. Al-Baik, O., Miller, J.: Integrative double kaizen loop (idkl): towards a culture of continuous learning and sustainable improvements for software organizations. IEEE transactions on Software Engineering 45(12), 1189–1210 (2018)
12. Al-Tarawneh, A.M.: An analysis of the positive and negative effects of cyberpsychotherapy: A systematic review. In: 2024 2nd International Conference on Cyber Resilience (ICCR). pp. 1–4. IEEE (2024)
13. Anderson, E.W., Sullivan, M.W.: The antecedents and consequences of customer satisfaction for firms. Marketing science 12(2), 125–143 (1993)
14. Audit, T.S.N.: Föråldrade it-system–hinder för en effektiv digitalisering (2019)
15. Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C.: Managing technical debt in software engineering (dagstuhl seminar 16162). Dagstuhl reports 6(4) (2016)
16. Avgeriou, P.C., Taibi, D., Ampatzoglou, A., Fontana, F.A., Besker, T., Chatzigeorgiou, A., Tsintzira, A.A.: An overview and comparison of technical debt measurement tools. IEEE software 38(3), 61–71 (2020)
17. Behutiye, W.N., Rodríguez, P., Oivo, M., Tosun, A.: Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Information and Software Technology 82, 139–158 (2017)
18. Besker, T., Martini, A., Bosch, J.: Managing architectural technical debt: A unified model and systematic literature review. Journal of Systems and Software 135, 1–16 (2018)
19. Blank, S.: Organizational debt is like technical debt–but worse (2015)
20. Braun, V., Clarke, V.: Thematic analysis. American Psychological Association (2012)
21. Cunningham, W.: The wycash portfolio management system. In: Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum). ACM, Vancouver, British Columbia, Canada (1992)
22. DeLone, W.H., McLean, E.R.: The delone and mclean model of information systems success: a ten-year update. Journal of management information systems 19(4), 9–30 (2003)

23. Dodgson, M., Gann, D.M., Phillips, N.: Organizational learning and the technology of foolishness: The case of virtual worlds at ibm. Organization science 24(5), 1358–1376 (2013)
24. Edmondson, A.: Psychological safety and learning behavior in work teams. Administrative science quarterly 44(2), 350–383 (1999)
25. Ernst, N.A., Bellomo, S., Ozkaya, I., Nord, R.L., Gorton, I.: Measure it? manage it? ignore it? software practitioners and technical debt. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. pp. 50–60 (2015)
26. of Finance, D.M.: Regeringens kasseeftersyn på it-området (2017)
27. Haddaway, N.R., Collins, A.M., Coughlin, D., Kirk, S.: The role of google scholar in evidence reviews and its applicability to grey literature searching. PloS one 10(9), e0138237 (2015)
28. Helfat, C.E., Finkelstein, S., Mitchell, W., Peteraf, M., Singh, H., Teece, D., Winter, S.G.: Dynamic capabilities: Understanding strategic change in organizations. John Wiley & Sons (2009)
29. Holan, P.M.D., Phillips, N.: Organizational forgetting. Handbook of organizational learning and knowledge management pp. 433–451 (2012)
30. Kemerer, C.F., Slaughter, S.: An empirical approach to studying software evolution. IEEE transactions on software engineering 25(4), 493–509 (1999)
31. Khomyakov, I., Makhmutov, Z., Mirgalimova, R., Sillitti, A.: Automated measurement of technical debt: A systematic literature review. In: International Conference on Enterprise Information Systems. pp. 95–106 (2019)
32. Kitchenham, B.A., Pfleeger, S.L.: Personal opinion surveys. In: Guide to advanced empirical software engineering, pp. 63–92. Springer (2008)
33. Klinger, T., Tarr, P., Wagstrom, P., Williams, C.: An enterprise perspective on technical debt. In: Proceedings of the 2nd Workshop on managing technical debt. pp. 35–38 (2011)
34. Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., Fontana, F.A.: A systematic literature review on technical debt prioritization: strategies, processes, factors, and tools. Journal of Systems and Software 171, 110827 (2021)
35. LePine, J.A., Podsakoff, N.P., LePine, M.A.: A meta-analytic test of the challenge stressor–hindrance stressor framework: An explanation for inconsistent relationships among stressors and performance. Academy of management journal 48(5), 764–775 (2005)
36. Liechti, O., Pasquier, J., Reis, R.: Supporting agile teams with a test analytics platform: a case study. In: 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST). pp. 9–15. IEEE (2017)
37. Liu, Z., Stray, V., Sporsem, T.T.: Organizational debt in large-scale hybrid agile software development: A case study on coordination mechanisms. In: Agile Processes in Software Engineering and Extreme Programming–Workshops: XP 2022 Workshops, Copenhagen, Denmark, June 13–17, 2022, and XP 2023 Workshops, Amsterdam, The Netherlands, June 13–16, 2023, Revised Selected Papers. Springer (2023)
38. Luthans, F., Youssef-Morgan, C.M.: Psychological capital: An evidence-based positive approach. Annual review of organizational psychology and organizational behavior 4(1), 339–366 (2017)
39. Marjanović, J., Dalčeković, N., Sladić, G.: Blockchain-based model for tracking compliance with security requirements. Computer Science and Information Systems 20(1), 359–380 (2023)
40. Martini, A., Besker, T., Bosch, J.: Process debt: a first exploration. In: 27th Asia-Pacific Software Engineering Conference. pp. 316–325. IEEE (2020)
41. Maslach, C., Schaufeli, W.B., Leiter, M.P.: Job burnout. Annual review of psychology 52(1), 397–422 (2001)
42. Matkovic, P., Maric, M., Tumbas, P., Sakal, M.: Traditionalisation of agile processes: Architectural aspects. Computer Science and Information Systems 15(1), 79–109 (2018)
43. Morgeson, F.P., Hofmann, D.A.: The structure and function of collective constructs: Implications for multilevel research and theory development. Academy of management review 24(2), 249–265 (1999)

44. Mowday, R.T., Porter, L.W., Steers, R.M.: Employee—organization linkages: The psychology of commitment, absenteeism, and turnover. Academic press (2013)
45. Nadler, D.A., Tushman, M.L.: A model for diagnosing organizational behavior. Organizational Dynamics 9(2), 35–51 (1980)
46. Piqueres, C.: Managing debt: Organizational debt (2021), https://carlos-piqueres.medium.com/managing-debt-organizational-debt-a7a1578235f3
47. Poth, A., Kottke, M., Riel, A.: Self-service kits to scale knowledge to autonomous teams-concept, application and limitations. Computer Science and Information Systems 20(1), 229–249 (2023)
48. Rainer, A., Hall, T.: Key success factors for implementing software process improvement: a maturity-based analysis. Journal of Systems and Software 62(2), 71–84 (2002)
49. Rios, N., de Mendonca Neto, M.G., Spinola, R.O.: A tertiary study on technical debt: types, management strategies, research trends, and base information for practitioners. Information Software Technology 102, 117–145 (2018)
50. Rios, N., Spinola, R.O., Mendonça, M., Seaman, C.: The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from brazil. Empirical Software Engineering 25, 3216–3287 (2020)
51. Senge, P.M.: The fifth discipline: The art and practice of the learning organization. Broadway Business (2006)
52. Tamburri, D.A., Kruchten, P., Lago, P., Vliet, H.V.: Social debt in software engineering: insights from industry. Journal of Internet Services and Applications 6(1), 1–17 (2015)
53. Turab, N., Abu Owida, H., Al-Nabulsi, J.I.: Harnessing the power of blockchain to strengthen cybersecurity measures: a review. Indonesian Journal of Electrical Engineering and Computer Science 3(1), 593–600 (July 2024)
54. Wasserman, S., Faust, K.: Social network analysis: Methods and applications (1994)

APPENDIX A: Survey questions

Category	**Questions**
Participants background	What is your sex?
	What is your title?
	What is your age?
	How many years of experience do you have?
Organizational Information	What is your organization size?
	What domain is the organization operating in?
Awareness and Identification of Organizational Debt	Are you familiar with the concept of organizational debt (OD)?
	How would you describe your organization's current level of OD? (Operational, Managerial, Executive)
	What indicators or symptoms do you use to identify the presence of OD in your organization? (e.g., inefficient processes, outdated policies, employee dissatisfaction)
	Do you have any specific processes or methods in place to assess and monitor OD levels?
Causes and Consequences of Organizational Debt	In your opinion, what are the primary causes or factors contributing to the accumulation of OD in your organization? (e.g., rapid growth, lack of communication, outdated structures)
	What impact has OD had on your organization? (e.g., decreased productivity, employee turnover, difficulty adapting to change)
Mitigation Strategies and Practices	Does your organization have any strategies or practices in place to mitigate or reduce OD?
	If yes, please describe the strategies or practices you use to mitigate OD.
	How effective have these strategies or practices been in mitigating OD? (Very effective, Somewhat effective, Not effective)
	What challenges or obstacles have you faced in implementing strategies to mitigate OD?
Additional Comments and Feedback	Do you have any additional comments, suggestions, or insights regarding the identification, impact, or mitigation of organizational debt in software organizations?

APPENDIX B: Definition of various types of debt [2] [6]

Debt type	Definition
Technical debt	The debt incurred through the speeding up of software project development which results in a number of deficiencies ending up in high maintenance overheads
Non-technical debt	It is an umbrella term to cover a combination of social and technical aspects, such as process, social, and people debt.
Process debt	Refers to inefficient processes, for example, what the process was designed to handle may be no longer appropriate
Social debt	Social debt is analogous to technical debt in many ways: It represents the state of software development organizations as the result of "accumulated" decisions. In the case of social debt, decisions are about people and their interactions.
People debt	People debt Refers to people issues that, if present in the software organization, can delay or hinder some development activities. An example of this kind of debt is expertise concentrated in too few people, as an effect of delayed training and/or hiring
Organisational debt	OD refers to the difference between a company's strategic plans and its actual ability to implement them in view of the ever-changing market needs

Muhammad Ovais Ahmad is an associate professor at Karlstad University Sweden and adj. professor at the University of Oulu, Finland. He received his PhD and MSc degree in Information Processing Science from University of Oulu, Finland. Previously, he worked as a Professor of software engineering at the Gdansk University of Technology, Poland; research fellow and programme coordinator of the European Masters Programme in Software Engineering at the University of Oulu, Finland. Dr. Ahmad's research interests are focused on software development methodologies, evidence based software engineering, process assessment and improvement, software engineering curriculum and pedagogy, technology and digital services adoption and diffusion. He has published in journals such as the Journal of Systems and Software, Software Quality Journal, Information & Software Technology, Information Systems Frontiers, Information Technology and People, IET Software, Journal of Software: Evolution and Process, Journal of Cleaner Production, Transforming Government: People, Process and Policy, e-Informatica Software Engineering Journal.

Osama Al-Baik is a quality-oriented professional with over 20 years of experience and technical expertise in diverse range of technologies within multiple industry settings. He has held senior positions in international and domestic organizations. Dr. Osama received his Doctorate degree in Software Engineering and Intelligent Systems from University of Alberta, Canada; Master Degree in Software Engineering and Project Management from DePaul University, USA. Dr. Osama has also a PGP in AI and Machine Learning from University of Texas at Austin, USA. Dr. Osama's research interests have been in Software Engineering, Health Information Systems, Lean Operations, Process improvements and Optimization, Lean Product Development, Lean Service Excellence, and the Sociocultural aspects of Software Engineering.

AbdelRahman Hussein is an Associate Professor at the Department of Computer Science at Al-Ahliyya Amman University, Jordan. He received his first degree in Computer Science from Jordan University of Science and Technology, Jordan, in July 2000, master degree in Computer Science from Jordan University, Jordan in July 2003, and Ph.D. from the Anglia Ruskin University ARU), UK in 2010. His main research interests lie in the areas of VoIP, mobile Ad-Hoc networking, and E-learning.

Mwaffaq Abu Alhija is an Associate Professor at the Department of Computer Science, Applied Science Private University, Amman, Jordan. His main research interests lie in the areas of operating system design, distributed computing systems, multimedia communication and networking, mobile and wireless networks, data and network security, wireless sensor networks, Cybersecurity, and parallel computing.

Received: April 11, 2024; Accepted: December 20, 2024.