Journal of Software Engineering Research and Development, 2026, 14:2, doi: 10.5753/jserd.2026.5462

Management of Technical Debt in Startups: A Systematic Mapping

Déverson Rogério Rando [**Universidade Estadual de Maringá** | ***deverson.rando@ifpr.edu.br***]
Gislaine Camila Lapasini Leal [**Universidade Estadual de Maringá** | ***gclleal@uem.br***]
Guilherme Corredato Guerino [**Universidade Estadual de Maringá** | ***guilherme.guerino@ies.unespar.edu.br***]
Renato Balancieri [**Universidade Estadual de Maringá** | ***rbalancieri@uem.br***]

Abstract

This study explores the state of the art of Technical Debt (TD) in software startups, seeking to understand the approaches, methods, and techniques used to manage this issue. TD is a concept that refers to suboptimal technical decisions made to accelerate development, such as implementing code or design that later requires revision to avoid future problems. In startups, where speed and innovation are essential for growth, the pressure to quickly release products often leads to the accumulation of TD. Although this offers short-term benefits, such as faster feature releases, in the long term, it can compromise product quality, increase maintenance costs, and reduce the ability to scale the business. We conducted the study through a systematic literature mapping and analyzed articles from the Scopus, IEEE Xplore, and ACM Digital Library databases. The authors selected fourteen studies published between 2017 and 2024. The analysis revealed that practitioners and researchers still lack standardized practices and tools for efficiently managing TD. Furthermore, the research highlights the need for more empirical studies that consider the specific context of startups, where limited resources and the need for accelerated innovation create unique challenges for TD management. The mapping carried out highlights the fragmentation of current approaches and the lack of a unified framework for managing TD in startups. The authors therefore conclude that there is a need to develop and empirically validate strategic models that integrate technical, human, and business aspects to prove more effective solutions for this specific context.

Keywords: *Technical Debt, Software Startups, Technical Debt Management, Software Quality*

1 Introduction

Software startups, companies focused on innovation and the development of high-tech products, face significant challenges due to the pressure to innovate at a rapid pace. According to Klot et al. (2019), these companies often struggle with backlog management while balancing technical debt. In the Brazilian context, Guerino et al. (2024, 2023) highlight that professionals recognize both the usefulness and the challenges of UX work in startups. Furthermore, Paternoster et al. (2014) emphasize that startups adopt rapid development cycles with continuous testing and iterations to achieve product-market fit. This pressure to launch products quickly frequently leads to technical shortcuts and, consequently, the accumulation of technical debt (TD), which can compromise product quality, increase maintenance costs, and hinder scalability as the company grows.

These technical compromises are defined by the concept of Technical Debt, introduced by Cunningham (1992), which draws an analogy with financial debt: just as borrowed capital must be repaid with interest, the effort "borrowed" when delivering suboptimal solutions must be paid back through increased future maintenance effort. TD arises when teams prioritize speed over robustness to meet deadlines, resulting in long-term burdens that impact maintainability and quality.

This scenario aligns with the findings of Berg et al. (2018), who conducted a systematic mapping study on software startups and identified low maturity in engineering practices, particularly those affecting long-term technical sustainability. Their results underscore the importance of investigating how TD is managed in practice and the implications of such decisions in the startup context.

In such dynamic environments, TD becomes commonplace. It is not unusual for startups to knowingly deliver software with unresolved issues, expecting to address them later. The need to balance fast delivery with acceptable quality forces recurring decisions to postpone architectural or technical improvements, as observed by Klotins et al. [M5].

Market competition further intensifies this tension. To meet tight deadlines, startups often cut corners, resulting in debt that, while offering short-term gains, introduces future costs: more failures, complexity, and resistance to change. Gomes (2022) highlight that, as the company evolves, TD compounds, raising bug incidence, slowing adaptation, and increasing maintenance overhead. Effective management practices, such as continuous refactoring and disciplined development, are crucial to sustaining innovation in the long term. Therefore, TD management is critical to software project success, as it directly affects productivity, cost, and quality, and must be proactively addressed to strike a balance between agility and long-term stability, as pointed out by Aldaeej and Seaman [M2] and Besker et al. [M3].

Complementing this understanding, studies on TD management in Small and Medium Enterprises (SMEs) offer valuable insights for startups, given the similarities in agility and resource constraints. For instance, Hamed and et al. (2023) propose a structured approach with five phases: prevention, identification, monitoring, prioritization, and repay-

ment. This framework suggests that even in fast-paced environments, it is possible to systematically monitor and mitigate TD.

Other systematic studies have explored TD from multiple perspectives. Li et al. (2015) investigated its implications for software management. Ribeiro et al. (2016) focused on decision-making for maintainability. Murillo et al. (2023) updated strategies for identification and resolution. Albuquerque et al. (2022) applied intelligent techniques to manage related challenges. Further contributions address TD in agile estimates, as shown by Klimczyk and Madeyski (2020), links to code smells and architecture degradation, discussed by Das et al. (2021), and tool features for TD management, explored by Saraiva et al. (2021). However, few studies have explicitly focused on the constraints that shape TD in startups, such as resource scarcity and accelerated delivery cycles.

A broader review conducted by Murillo et al. (2023) analyzed publications from 1992 to 2022 and observed growing use of automated tools for identifying and quantifying TD. Yet, it emphasizes that effective management must also integrate human, cultural, and organizational dimensions, and that more empirical validation is needed.

In this regard, Gandomani et al. (2024) highlight that adopting agile methods in startups requires specific adaptations due to uncertainty and limited resources. They propose tailoring practices according to team size, maturity, and product stage, emphasizing that TD management in startups requires not only technical but also contextually relevant methodological choices. This perspective underscores the importance of examining how startups can effectively manage TD in their unique contexts.

Therefore, this study investigates a still-underexplored problem: how startups can effectively manage TD while maintaining their pace of innovation and growth. To this end, it systematically maps the current state of knowledge on TD in software startups, focusing on its identification, management, and mitigation through both technical and organizational lenses. The study follows the protocol of Kitchenham (2007), covering literature from 2017 to 2024.

Our findings indicate that despite increased interest in automating TD identification and prioritization, there is a significant lack of standardization in practices and tools. The literature remains fragmented, with little empirical validation and insufficient attention to the particular challenges faced by startups. For researchers, this highlights the need for integrated, context-sensitive models. For practitioners, it signals the urgency of flexible strategies that preserve agility without neglecting sustainability.

To achieve this, the following section presents the methodology adopted for conducting the mapping study, followed by the results and a discussion of the approaches and challenges found in the literature regarding TD management in startups. The final section outlines the main implications and proposes directions for future research.

2 Methodology

We followed the guidelines proposed by Kitchenham (2007). The goal of this mapping is to investigate the state of the art on technical debt in startups by examining the primary approaches, methods, and techniques adopted in this context. The Kitchenham protocol provides rigorous guidance to ensure the quality and comprehensiveness of reviews, supporting all stages of the process, from formulating research questions to study selection, data extraction, and result synthesis.

2.1 Research Questions

This section presents the rationale behind the research questions that guided the mapping process.

Q1. What are the impacts of TD in the context of software startups?

RQ1 investigates the impacts of TD in the context of software startups, including definitions, metrics, and consequences for software development processes. It also addresses how TD influences productivity, quality, and project cost, along with the organizational and cultural challenges of its management. Identifying effective mitigation strategies and research gaps is essential to understanding and addressing this complex phenomenon in software engineering.

Q2. What approaches, methods, and techniques have been used to manage TD in startups?

RQ2 explores the main approaches, methods, and techniques used to manage TD in startups, including practices such as continuous refactoring, test automation, code review, static analysis, continuous integration, and continuous delivery. It also examines the use of management frameworks and decision-making models for prioritizing and reducing TD. Understanding the effectiveness, limitations, and implementation challenges of these approaches is crucial for advancing TD management.

Q3. What are the challenges and opportunities associated with managing TD in startups?

RQ3 focuses on the practical challenges teams face when managing TD, such as prioritization difficulties, lack of leadership support, and resistance to change. It also addresses the negative impacts of TD on software quality, team productivity, and maintenance costs. Additionally, it explores improvement opportunities through agile practices, process automation, team education, and the development of more effective tools and metrics.

2.2 Inclusion and Exclusion Criteria

To ensure the scientific rigor and contextual relevance of the selected studies, a set of clear inclusion and exclusion criteria was defined and consistently applied throughout the selection process. Table 1 summarizes these criteria.

We rigorously applied these criteria throughout all selection phases to ensure that only high-quality and contextually aligned studies were included in the final mapping.

Table 1. Inclusion and Exclusion Criteria for Study Selection

ID	Criterion Type	Description
C1	Inclusion	Studies that explicitly address technical debt (TD) in the context of software startups and present approaches, methods, or techniques for managing, measuring, analyzing, or reducing TD.
C2	Exclusion	Studies that mention TD only peripherally or outside the context of software startups.
C3	Exclusion	Studies that do not report empirical research, such as case studies, surveys, or controlled experiments.
C4	Exclusion	Studies not published between 2017 and 2024.
C5	Exclusion	Non–peer-reviewed publications (e.g., editorials, opinion pieces, blog posts).
C6	Exclusion	Publications not written in English.
C7	Exclusion	Studies lacking methodological transparency or a clear empirical basis.
C8	Exclusion	Duplicate records or publications with inaccessible full text.

2.3 Search Strategy

The authors conducted the search in three major electronic databases: Scopus, IEEE Xplore, and ACM Digital Library. These databases were selected due to their comprehensive coverage of software engineering literature and their relevance for studies involving technical and architectural aspects of software development, particularly in startup contexts.

The search targeted articles published in English between 2017 and 2024. The authors retrieved studies using combinations of keywords ”technical debt,” ”software debt,” ”architecture debt,” ”code debt,” ”technical debt management,” ”technical debt reduction,” ”technical debt measurement,” ”technical debt analysis,” ”startup,” and ”start-up.”

The primary keywords were derived directly from the core concepts of the research: ”technical debt” and ”startups.” Synonyms and related terms, such as ”software debt,” ”architecture debt,” and ”code debt,” were included to capture terminological variations. To target studies that go beyond merely mentioning technical debt, terms like ”management,” ”reduction,” ”measurement,” and ”analysis” were added.

The search string was constructed iteratively. An initial version was developed based on the research questions and refined through pilot searches to evaluate its coverage and precision. This refinement process included validating whether the string retrieved known relevant studies from prior literature. The final version of the search string was tested for compatibility with each database to ensure reliable results.

Search String:

```
(TITLE-ABS-KEY ( "technical debt" ) OR
TITLE-ABS-KEY ( "software debt" ) OR
TITLE-ABS-KEY ( "architecture debt" )
OR TITLE-ABS-KEY ( "code debt" ) AND
TITLE-ABS-KEY ( "technical debt management"
) OR TITLE-ABS-KEY ( "technical debt
reduction" ) OR TITLE-ABS-KEY ( "technical
debt measurement" ) OR TITLE-ABS-KEY (
"technical debt analysis" ) OR TITLE-ABS-KEY
( "startup" ) OR TITLE-ABS-KEY ( "start-up"
)) AND PUBYEAR > 2009 AND PUBYEAR <= 2024 AND
(LIMIT-TO ( SUBJAREA , "COMP" ) OR LIMIT-TO (
SUBJAREA , "DECI" ) OR LIMIT-TO ( SUBJAREA ,
"ENGI" )) AND (LIMIT-TO ( DOCTYPE , "ar" ))
```

2.4 Study Selection

The study selection process followed a structured and rigorous approach to ensure the inclusion of only relevant and high-quality studies. This process was divided into four main phases:

Phase 1 – Initial Screening: Titles, abstracts, and keywords of articles retrieved from electronic databases were screened. The inclusion and exclusion criteria were applied to retain studies that explicitly addressed TD in software systems, particularly those focused on approaches, methods, and techniques for managing, measuring, analyzing, or reducing TD. Studies with only peripheral mentions of TD or lacking a clear focus on startups were excluded.

Phase 2 – Introductory Review: The introduction and conclusion sections of the remaining studies were reviewed to gain a better understanding of their context and main contributions. This step allowed a more accurate application of the inclusion and exclusion criteria.

Phase 3 – Full-Text Review: The full texts of shortlisted studies were analyzed, and the inclusion/exclusion criteria were reapplied. At this stage, detailed data were extracted, including study objectives, methods, results, and conclusions.

Phase 4 – Snowballing: The authors employed a complementary snowballing strategy to expand the literature coverage. Both backward and forward snowballing techniques were used:

- Backward Snowballing: References of the included studies were examined to identify additional relevant works.
- Forward Snowballing: Citation indexes were used to locate more recent articles citing the selected studies.

All articles identified through snowballing were subjected to the same inclusion and exclusion criteria. This multi-phase process aimed to maximize literature coverage and ensure robustness of the final set of selected studies.

To ensure alignment with the startup context, studies addressing broader domains were only included if they provided distinct findings, empirical data, or discussions explicitly related to software startups. In such cases, only the startup-relevant content was extracted and annotated during data collection.

The authors mapped each selected study to one or more research questions, guiding both data extraction and synthesis.

Figure 1 illustrates the selection process, showing how the initial pool of studies was progressively filtered based on the inclusion and exclusion criteria, ultimately resulting in the final set of articles considered relevant for mapping TD management in software startups.

2.5 Data Extraction

Data extraction involved the systematic collection of relevant information from each study included in the review. The authors extracted data on the article title, authors, year of publication, study objective, methods used, main results, conclusions, and the approaches, techniques, challenges, and opportunities related to TD management. This information was essential for synthesizing the available evidence and answering the research questions defined in the mapping protocol.

To organize and facilitate this process, the StArt tool (State of the Art through Systematic Reviews) Hernandes et al. (2012) was employed. A customized data extraction form was generated using Microsoft Excel, featuring detailed fields aligned with each research question, including study metadata, methodological aspects, and TD-related findings.

The first author performed the initial data extraction. To ensure consistency and minimize bias, three other researchers independently reviewed the extracted data involved in the study. Any discrepancies were discussed collectively until consensus was reached, thus ensuring the integrity of the data.

In cases where studies addressed both startups and broader contexts, only the findings explicitly related to software startups were extracted and annotated. Each data item was also mapped to one or more research questions to guide the synthesis phase and maintain analytical coherence.

This structured and collaborative extraction process facilitated a comprehensive and reliable synthesis of the evidence, thereby enhancing the robustness of this mapping study.

2.6 Quality Assessment

In order to ensure the reliability and transparency of this systematic mapping, a quality assessment of the selected studies was conducted. The objective of this evaluation was to identify the extent to which the studies provide methodological rigor, clarity, and practical relevance to the research questions.

We assessed study quality using the short checklist recommended by Kitchenham (2007) SLR guidelines, which adopt Yes/Partly/No answers for quality questions. In addition, each criterion was aligned with Wohlin et al. (2000) validity dimensions to ensure coverage of construct, internal, external and conclusion validity. We scored Yes=1, Partially=0.5, No=0; total scores < 2 denote low quality, 2–3.5 medium, and > 3.5 high quality. The five criteria were defined as follows: (QA1) clarity of objectives (construct validity); (QA2) context description (external validity); (QA3) methodological rigor (internal validity); (QA4) support of results by data/analysis (conclusion validity); and (QA5) practical relevance for startups (transferability).

Each criterion was scored using a three-level scale: *Yes (1 point)*, *Partially (0.5 points)*, or *No (0 points)*. The maximum possible score per study was 5 points. Studies scoring below 2 points were considered low quality, between 2 and 3.5 medium quality, and above 3.5 high quality.

The quality assessment revealed that **78.6% of the selected studies achieved high quality** and **21.4% medium quality**, with no studies classified as low quality. This predominance of high-quality studies reinforces the robustness of the findings presented in the following sections. These results support the reliability of the answers to the research questions while also highlighting opportunities for improving methodological transparency and practical relevance in future work

To support transparency and reproducibility, the detailed scores attributed to each individual study are provided in Appendix A.

2.7 Data Synthesis and Analysis

The data synthesis and analysis were conducted using both qualitative and quantitative methods. The qualitative synthesis focused on answering the research questions by categorizing the main findings of each study and organizing them into emerging themes and patterns identified in the literature. This process aimed to capture the current state of knowledge regarding TD, as well as the approaches, methods, and techniques used to manage it, and the challenges and opportunities reported in the context of software startups.

In parallel, a quantitative synthesis was performed to summarize numerical evidence reported in the primary studies. This included frequency counts of approaches and techniques, the distribution of study types, and the prevalence of specific challenges and opportunities related to TD management.

To assess the methodological rigor of the included studies, a quality appraisal was conducted based on three main criteria: (i) clarity in the presentation of objectives, methods, and results; (ii) relevance to the theme of technical debt in startups; and (iii) internal consistency of findings. Each study was classified into one of three categories—high, medium, or low quality—to support a more reliable synthesis and interpretation of the evidence.

The results were synthesized descriptively, highlighting the diversity of approaches and tools used to manage TD, the challenges encountered by startups, and the opportunities for improvement. The authors provide a comprehensive and structured overview with this synthesis of the current state of research, offering valuable insights for both academics and practitioners concerned with sustainable software development in startup environments.

Table 2 compiles the selected studies, including metadata such as authorship, publication year, venue, and focus area related to TD in startups.

3 Results

A total of 427 papers were initially retrieved: 197 from Scopus, 197 from the ACM Digital Library, and 33 from IEEE Xplore.

Table 2. Studies on TD in Startups.

ID Paper	Title	Authors	Year	Venue
[M1]	Towards Effective Technical Debt Decision Making in Software Startups	Aldaeej, A.	2020	40th International Conference on Software Engineering
[M2]	The Negative Implications of Technical Debt on Software Startups: What They Are and When They Surface	Aldaeej, A.; Seaman, C.	2023	Proceedings of the 5th International Workshop on Software-Intensive Business: Towards Sustainable Software Business
[M3]	Embracing Technical Debt, from a Startup Company Perspective	Besker, T.; Martinia, A.; Lokuge, R. et al.	2018	Proceedings 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME): 415-425
[M4]	Trade-Offs in Managing Risk and Technical Debt in Industrial Research Labs: An Experience Report	Gauthier, F.; Jordan, A.; Krishnan, P.; Hassanshahi, B.; Süß, J.G.; Bae, S.; Lee, H.	2020	Proceedings of the 3rd International Conference on Technical Debt
[M5]	Exploration of Technical Debt in Start-Ups	Klotins, E.; Unterkalmsteiner, M.; Chatzipetrou, P.; Gorschek, T.; Prikladnicki, R.; Tripathi, N.; Pompermaier, L.B.	2018	Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice
[M6]	Value-based technical debt management: An exploratory case study in start-ups and scale-ups	Njima, M.; Demeyer, S.; Smolander, K.; Grunbacher, P.; Hyrynsalmi, S.; Jansen, S.	2019	2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms and Ecosystems (IWSiB '19)
[M7]	Integrating Technical Debt Management and Software Quality Management Processes: A Framework and Field Tests	Ramasubbu, N.; Kemerer, C.	2018	IEEE Transactions on Software Engineering (TSE)
[M8]	Technical Debt in Brazilian Software Startups: Perceptions of Professionals in Paraná	Santos, Adauto; Kuspil, Jonathan; Rando, Deverson; Santos, Dionnes; Leal, Gislaine Camila; Balancieri, Renato; Oliveirajr, Edson	2023	Proceedings of the XXII Brazilian Symposium on Software Quality
[M9]	A rule-based decision model to support technical debt decisions: A multiple case study of web and mobile app startups	Aldaeej, Abdullah; Seaman, Carolyn	2024	Information and Software Technology
[M10]	Startups Transitioning from Early to Growth Phase: A Pilot Study of Technical Debt Perception	Bajohr, Stefan; Spohrer, Kai; Mendez, Daniel; Penzenstadler, Birgit; Zimmermann, Thomas	2020	ICSOB (Lecture Notes in Business Information Processing)
[M11]	Startups and Technical Debt: Managing Technical Debt with Visual Thinking	Marcos Chicote	2017	SoftStart (IEEE/ACM Workshop)
[M12]	Technical Debt Trade-Off: Experiences from Software Startups Becoming Grownups	Cico, Orges	2019	ICSOB (Lecture Notes in Business Information Processing)
[M13]	The Perception and Management of Technical Debt in Software Startups	Apa, Cecilia; Jeronimo Junior, Helvio; Nascimento, Luciana M.; Vallespir, Diego; Travassos, Guilherme Horta	2020	Software Startups (LNBIP)
[M14]	Toward a Technical Debt Relationship with the Pivoting of Growth Phase Startups	Cico, Orges; Besker, Terese; Martini, Antonio; Nguyen Duc, Anh; Souza, Renata; Bosch, Jan	2021	PROFES (LNCS)

Note: Systematic Mapping of studies related to Technical Debt in the context of Software Startups.

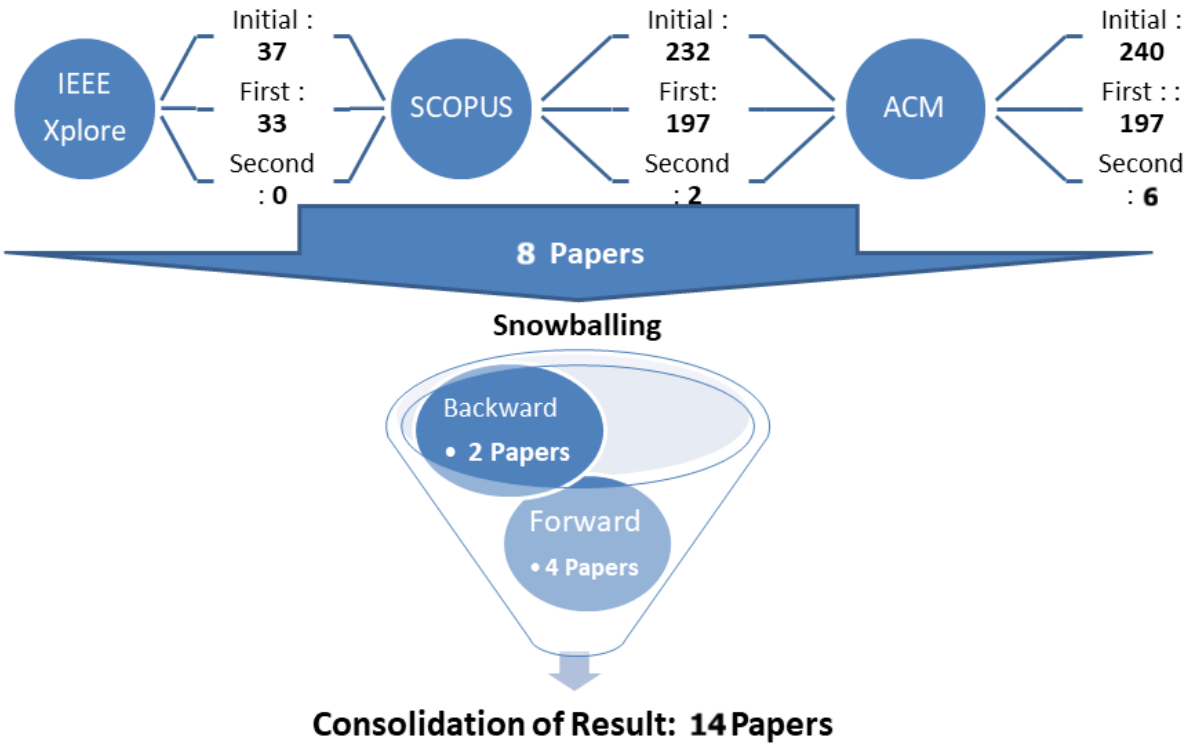

Figure 1. Study identification and selection process. Source: Author (2025).

Following the initial screening, The authors included nine studies that met all the criteria in the review. To broaden the scope and reduce the risk of missing relevant works, a snowballing strategy was applied in both directions. Backward snowballing, based on the reference lists of the included articles, identified two additional studies. Forward snowballing, which analyzed papers that cited the selected studies, resulted in the inclusion of 4 more articles. All additional studies were evaluated using the same inclusion and exclusion criteria. As a result, the final set comprised 14 primary studies. This expanded dataset significantly strengthened the review conducted by the authors, providing a more comprehensive understanding of TD management practices and challenges in the startup context.

The selected studies present a wide range of approaches and contexts related to TD, reflecting the diversity of methodologies and techniques adopted to address this issue. These studies, published between 2017 and 2024, span multiple venues relevant to software engineering and startup ecosystems.

Most studies employed empirical methods, including case studies, qualitative and quantitative research, and controlled experiments. The participants involved ranged from software developers to project managers, offering multiple perspectives on how TD is perceived and managed across organizational roles and maturity levels.

This synthesis of the main characteristics of the selected studies provides a solid foundation for future investigations. It contributes to improving TD management practices, especially in the fast-paced and resource-constrained environment typical of software startups.

3.1 Q1. What are the impacts of TD in the context of software startups?

TD is a recurring phenomenon in software startups, where pressure for rapid releases and limited resources often lead to suboptimal technical decisions. While these trade-offs may be strategic in the early stages, enabling faster delivery and validation, they tend to generate severe consequences as the product and organization evolve.

The reviewed studies highlight a wide range of negative impacts of TD in startups, affecting technical, organizational, and strategic dimensions. These impacts can be categorized into three main groups: **operational**, **business**, and **strategic**.

Table 3 summarizes the primary adverse outcomes reported in the literature, organized into operational, business, and strategic categories.

Operational impacts The analysis of the selected studies revealed that TD generates a variety of negative conse-

Table 3. Summary of Reported Negative Impacts of Technical Debt in Startups.

Category	Outcome	ID Paper
Operational	Feature delivery delays	[M14], [M2], [M3],[M5], [M8], [M9], [M10], [M12], [M13]
Operational	Rising maintenance costs	[M14], [M2], [M3],[M5], [M6], [M8], [M9], [M10], [M12], [M13]
Operational	Reduced product quality	[M14], [M2], [M3],[M5], [M6], [M8], [M9], [M10], [M12], [M13]
Business	Reduction in team productivity	[M14], [M2], [M5], [M6], [M9], [M12]
Business	Increased business risk	[M14], [M2], [M3],[M5], [M6], [M8], [M9], [M12]
Business	Competitiveness loss due to TD accumulation	[M9], [M2], [M3], [M5], [M6]
Strategic	Inability to perform technological pivot	[M14]
Strategic	Limitations on sustainable growth	[M14], [M3], [M5] , [M6], [M9], [M10]
Strategic	Potential for unwanted or forced pivots	[M14]

Note: Categorized impacts observed across selected literature studies.

quences in software startups. In this section, we first present the operational impacts, which encompass issues directly affecting day-to-day development activities, such as delivery speed, maintenance costs, and product quality. The following items synthesize the most frequently reported operational issues identified across the reviewed studies.

- **Feature delivery delays:** Accumulated TD increases code complexity, making it harder to implement new functionalities, thereby compromising *time-to-market*, which is crucial in competitive environments. Evidence from the corpus indicates that unresolved TD slows down feature delivery and limits responsiveness to market demands [M14], [M2], [M3],[M5], [M8], [M9], [M10], [M12], [M13].
- **Rising maintenance costs:** The longer TD remains unresolved, the higher the cost of future bug fixes, testing, and integration, which is particularly problematic in resource-constrained startup environments. Multiple studies report that accumulated TD leads to increased maintenance effort and operational overhead [M14], [M2], [M3],[M5], [M6], [M8], [M9], [M10], [M12], [M13].
- **Reduced product quality:** TD related to architecture, testing, and documentation negatively affects maintainability, scalability, and system robustness. The corpus consistently reports quality degradation as a direct consequence of TD accumulation in startups [M14], [M2], [M3], [M5], [M6], [M8], [M9], [M10], [M12], [M13].

Business impacts Beyond operational challenges, TD also produces a wide range of business impacts that directly influence the sustainability and growth of startups. These effects extend beyond code-level issues, affecting team efficiency, reliability, and overall competitiveness. In this section, we outline the main business-related consequences reported in the analyzed studies.

- **Reduced team productivity:** Accumulated TD forces development teams to spend increasing amounts of time dealing with rework, legacy issues, and technical workarounds, reducing their ability to deliver new value efficiently. Evidence from the corpus indicates that unresolved TD significantly decreases productivity in startup teams [M14], [M2], [M5], [M6], [M9],[M12].
- **Increased business risk:** TD contributes to system instability, higher failure rates, and increased rework, which can negatively affect customer satisfaction, operational reliability, and decision-making during critical growth phases. Several studies report that unmanaged TD elevates both technical and business risks in startups [M14], [M2], [M3], [M5], [M6], [M8], [M9],[M12].
- **Competitiveness loss due to TD accumulation:** High levels of accumulated TD reduce startups' agility and limit their ability to respond quickly to market changes, introduce innovations, or scale their products. The analyzed studies consistently associate excessive TD with reduced competitiveness and constrained innovation capacity [M9], [M2], [M3], [M5], [M6].

Strategic impacts Finally, beyond operational and business concerns, TD can also have far-reaching strategic impacts that shape the long-term direction and sustainability of startups. These effects influence critical decisions related to technology evolution, scalability, and strategic positioning. In this section, we present the main strategic consequences reported in the analyzed studies.

- **Inability to perform technological pivot:** Structural TD can constrain architectural flexibility and limit the adoption of new technologies, making it difficult for startups to perform strategic technological pivots when required. Evidence from the corpus indicates that accumulated TD may lock startups into suboptimal technical paths, hindering necessary transformations [M14].
- **Limitations on sustainable growth:** High levels of TD can create technical bottlenecks that restrict scalability, slow down system evolution, and limit long-term growth potential. Several studies associate unresolved TD with constrained sustainable growth in startups, particularly during transitions between growth stages [M14], [M3],[M5], [M6], [M9], [M10].
- **Potential for unwanted or forced pivots:** When TD

reaches critical levels, startups may be compelled to undertake disruptive and unplanned strategic changes, such as major rewrites or abrupt shifts in their technical roadmap. The analyzed studies report that excessive TD can trigger forced pivots with significant cost and risk implications [M14].

In addition to these impacts, several studies noted that TD can harm team morale and increase developer turnover, especially in startups with immature processes. Tolerance for TD also varies across lifecycle stages: in early phases, some debt may be acceptable if accompanied by control mechanisms and plans for resolution. However, if neglected, TD can significantly hinder a startup's trajectory.

The challenges of TD in startups are amplified by their inherent characteristics: scarce resources, need for speed, high turnover, and rapid evolution. Ineffective TD management can compromise both software quality and overall business viability. Therefore, monitoring and mitigating technical debt is a strategic imperative for sustaining growth in software startups.

Given these findings, it is crucial to advance research on best practices, tools, and decision-making strategies tailored to different stages of startup evolution. TD poses a substantial challenge for startups, and failure to manage it effectively can compromise not only product quality but also long-term business sustainability. The reviewed studies emphasize the importance of treating TD as a strategic concern and call for further investigations into solutions that can help mitigate its risks across the startup lifecycle.

3.2 Q2. What approaches, methods, and techniques have been used to deal with TD?

The analysis of the selected studies reveals a wide range of adaptive approaches, structured methods, and specific techniques adopted to manage TD in startups. This diversity reflects both the multifaceted nature of TD and the specific challenges faced by these organizations, such as high uncertainty, time-to-market pressure, and limited resources.

Table 4 provides an overview of the main approaches identified, describing their purpose, characteristics, and sources. These approaches are further detailed and discussed in the subsections that follow.

Structured and Systemic Approaches Structured and systemic approaches to TD management in startups emerge as alternatives to the informal practices typically observed in these environments, aiming to ensure greater predictability and control over decision-making. By integrating quality assurance, project management, and quantitative metrics, these approaches seek to align agility with sustainability, enabling companies to maintain delivery speed without compromising the technical evolution of their products. In this context, different strategies—such as formal processes, Lean methods, and the use of objective indicators—offer pathways for more consistent and evidence-based TD management.

Startups that pursue greater process formalization tend to adopt process-based approaches, which integrate TD management into existing quality and project management workflows. This integration enables enhanced predictability and control, as discussed by Aldaeej[M1].

The Lean approach, grounded in agile principles, emphasizes value delivery with minimal waste. It leverages metrics such as development speed and process variance to optimize workflows and mitigate unnecessary TD accumulation. Aldaeej[M1] proposes Lean-based strategies to support effective TD management in startups.

Quantitative strategies rely on objective indicators — including cycle time, delivery rate, and team productivity — to assess the operational impact of TD. These metrics inform data-driven prioritization of remediation actions, as highlighted by Aldaeej[M1].

Valuation-Oriented Strategies and Decisions Within the scope of TD management in startups, valuation-oriented strategies play a crucial role by framing decisions not only as technical but also as business choices. These approaches aim to balance the costs and benefits of incurring, maintaining, or repaying TD, offering structured guidance for allocating scarce resources in high-uncertainty environments. By incorporating economic and contextual factors into decision-making, they enable startups to treat TD strategically, aligning remediation efforts with organizational goals and growth stages.

Valuation-oriented approaches assess the costs and benefits of maintaining or repaying TD, often employing probabilistic models or cost-benefit analyses. Aldaeej[M1] emphasizes that such techniques support the strategic allocation of resources and informed decision-making about deferring or addressing debt.

Aldaeej and Seaman [M9] propose the use of decision tree models to formalize this process, incorporating factors such as organizational maturity, business impact, product criticality, and resource availability.

Contextual and circumstantial management strategies further highlight that TD decisions should be adapted to the startup's stage of growth, code stability, and market dynamics, as also reinforced by Aldaeej and Seaman [M9].

Visual and Collaborative Practices Visual and collaborative practices have emerged as effective complements to traditional TD management approaches in startups. By making abstract concepts visible and fostering collective awareness, these practices bridge the gap between technical and non-technical stakeholders, promote shared responsibility, and enhance transparency in decision-making. They create a participatory environment in which TD can be continuously monitored and addressed, aligning remediation efforts with team performance and organizational goals.

An innovative example is the Visual Thinking technique presented by Chicote [M11], which uses physical representations (e.g., adhesive tape) to make TD tangible to team members. This facilitates communication between technical and non-technical stakeholders, enhances engagement, and clarifies the impact of debt.

Other collaborative practices include the use of visual tracking tools, such as "tape boards" and Scrum velocity

Table 4. Approaches to Manage TD in Startups

Category	Approach	Description	ID Paper
Contextual	Context-Aware Management	Adapting TD decisions to the startup's stage, available resources, product stability, and time-to-market constraints to support more effective decision-making.	[M14], [M4], [M9], [M10]
General Practices	TD Management Activities	Core activities such as identification, measurement, prioritization, monitoring, and repayment of TD.	[M14], [M2], [M3], [M4],[M5] , [M6], [M7], [M8], [M9], [M11],[M12], [M13]
Operational-Formalization	Formalization of Practices	Establishing formal or semi-formal TD management practices to balance agility and control as startups mature.	[M13]
Operational-Monitoring	Continuous Monitoring and Proactive Decision-Making	Monitoring indicators such as defect rates, code growth, or delivery metrics to support decisions on incurring or repaying TD.	[M9]
Operational-Monitoring	TD Tracking	Applying explicit mechanisms to track TD items and their impact, supporting informed and proactive management.	[M14]
Operational-Tools	Tool-Supported Management	Using project and task management tools (e.g., Jira, Trello) to organize TD-related activities and maintain dedicated backlogs.	[M13],[M5]
Preventive	Choosing Appropriate Technologies	Selecting mature technologies with strong community support to reduce risks of obsolescence and structural TD.	[M10],[M5]
Preventive	Proactive Management During Rapid Development	Combining rapid prototyping with planned refactoring to enable fast delivery while controlling TD accumulation.	[M14], [M11]
Structured-Systemic	Lean Approach	Applying lean principles and flow-based metrics (e.g., velocity, cycle time) to support TD-related decision-making.	[M1]
Structured-Systemic	Process-Based Approach	Integrating TD management into existing software development and quality assurance processes.	[M1]
Structured-Systemic	Quantitative Approach	Using quantitative metrics to assess TD impact and prioritize remediation actions based on measurable outcomes.	[M1]
Valuation-Oriented	Decision Tree Model	Applying rule-based decision models to guide TD-related decisions according to maturity, criticality, and resources.	[M9]
Valuation-Oriented	Value-Oriented Approach	Evaluating TD through cost-benefit and value-based perspectives to support prioritization decisions.	[M6], [M1]
Visual-Collaborative	Visual Thinking Technique	Using physical and visual representations (e.g., adhesive tape) to make TD visible and foster shared awareness within teams.	[M11]

Note: Inventory of identified approaches and strategies for Technical Debt management in startups.

charts, which enable continuous monitoring of TD and its relationship to team performance.

Prevention and Management During Rapid Development Prevention and management practices during rapid development are essential for startups that must balance speed and sustainability. These approaches focus on anticipating and mitigating TD risks before they accumulate, ensuring that the pace of delivery does not compromise long-term product quality. By adopting proactive measures and carefully selecting technologies, startups can sustain innovation while maintaining the stability and maintainability of their software systems.

Studies such as those by Cico et al. [M14] emphasize the importance of proactive TD management during fast-paced development cycles. These studies advocate for rapid prototyping followed by planned refactoring to validate business hypotheses without compromising long-term code quality, as also noted by Chicote [M11].

Another preventive strategy involves the deliberate selection of technologies with strong documentation, active community support, and proven stability. Such choices help mitigate the risks of obsolescence and the accumulation of structural TD, as highlighted by Besker et al. [M3], Klotins et al. [M5], and Bajohr et al. [M10].

Formalization and Operational Practices Formalization and operational practices represent a critical dimension of TD management in startups, as they seek to create structure without undermining agility. These approaches establish repeatable routines, tools, and controls that support transparency, coordination, and consistency in handling TD across different growth stages. By defining processes and leveraging management tools, startups can better align day-to-day operations with long-term quality and sustainability goals, while still adapting practices to their unique contexts.

The partial or complete formalization of management practices, as reported by Apa et al. [M13], reflects a deliberate effort to balance flexibility and control, particularly in startups transitioning from early-stage to growth phases.

Other operational practices include the adoption of visual management tools, such as Trello and Jira, which support the tracking of technical tasks related to TD and the organization of dedicated backlogs to address it. Klotins et al. [M5] and Apa et al. [M13] emphasize the importance of these tools for managing technical activities.

The findings indicate that TD management in startups is not homogeneous. It varies according to factors such as maturity stage, organizational culture, team technical profile, and business model. While automated practices such as refactoring, static code analysis, and quality control are commonly reported, the differentiating factor lies in each startup's capacity to tailor these practices to its specific context. This diversity is reinforced in the literature by Aldaeej and Seaman [M2], Besker et al. [M3], Gauthier et al. [M4], Klotins et al. [M5], Njima et al. [M6], Chicote [M11], Cico [M12], Apa et al. [M13], and Cico et al. [M14], as well as by empirical insights from Santos et al. [M8] and the decision models of Aldaeej and Seaman [M9].

Table 5 summarizes the methods and techniques used to identify, assess, and mitigate technical debt (TD) in software projects, particularly within the context of startups. TD refers to design or implementation decisions that accelerate development in the short term but require future rework to maintain software quality and maintainability.

Analysis-based Approaches Analysis-based approaches form the foundation of TD management in startups by providing systematic methods to identify, evaluate, and categorize potential problem areas before they escalate. Through different forms of analysis, these practices enhance visibility into the codebase, architecture, documentation, and testing processes, enabling teams to make informed decisions about prioritization and remediation. By applying these techniques, startups can monitor the health of their software systems and proactively mitigate risks associated with rapid development cycles.

- **Code analysis** relies on static and dynamic tools to detect issues in the code, such as duplication, excessive complexity, and poor style. Aldaeej[M1], Aldaeej and Seaman [M9], [M2], Besker et al. [M3], Bajohr et al. [M10], Cico [M12], Apa et al. [M13], and Cico et al. [M14] emphasize its importance for promptly identifying problematic areas.
- **Architecture analysis**, as discussed by Aldaeej[M1], Besker et al. [M3], Aldaeej and Seaman [M9], Bajohr et al. [M10], and Cico [M12], complements this by identifying deficiencies such as high coupling and low cohesion.
- **Documentation analysis**, highlighted by Aldaeej[M1] and Besker et al. [M3], evaluates the completeness and clarity of project documentation.
- **Test analysis** assesses test coverage and effectiveness, as noted by Aldaeej[M1] and Besker et al. [M3].
- **Defect classification**, also reported by Aldaeej[M1] and Besker et al. [M3], organizes issues into categories (e.g., code, design, documentation) to support prioritization.

Risk-based Approaches Risk-based approaches provide startups with a structured way to evaluate the potential impact of TD on both technical and business outcomes. By systematically identifying and prioritizing risks, these practices help teams focus their limited resources on the most critical debt items, balancing short-term delivery needs with long-term sustainability. Incorporating analyses such as risk assessment, business risk identification, and cost-benefit evaluation enables more informed and strategic decision-making in fast-paced development environments.

- **Risk analysis**, proposed by Aldaeej and Seaman [M9], supports the identification of potential TD-related risks and their business impact.
- **Business risk identification**, emphasized by Aldaeej[M1] and Besker et al. [M3], highlights the importance of prioritizing TD items that pose significant threats to business continuity.

- **Cost-benefit analysis**, described by Aldaeej[M1], Aldaeej and Seaman [M2], and Besker et al. [M3], evaluates trade-offs between repaying or tolerating TD.

Metrics-based Approaches Metrics-based approaches offer startups an evidence-driven way to understand and manage technical debt (TD) by quantifying its effects on software processes and outcomes. By tracking indicators related to productivity and quality, these practices enable teams to monitor efficiency, detect early signs of degradation, and prioritize remediation efforts based on measurable impact. This data-oriented perspective supports more objective decision-making and helps align TD management with broader organizational goals.

- **Productivity metrics**, such as team velocity and cycle time, are reported by Aldaeej[M1], Chicote [M11], and Cico [M12] as useful to measure the impact of TD on efficiency.
- **Quality metrics**, presented by Aldaeej[M1], Aldaeej and Seaman [M9], [M2], Besker et al. [M3], Gauthier et al. [M4], Klotins et al. [M5], Njima et al. [M6], Ramasubbu and Kemerer [M7], and Cico et al. [M14], assess defect density, complexity, and reliability.

Corrective Approaches Corrective approaches focus on addressing existing technical debt (TD) by actively improving the quality and maintainability of software systems. Rather than preventing new debt, these practices aim to remediate accumulated issues through structured interventions, ensuring long-term stability and scalability. By implementing techniques such as refactoring, startups can systematically reduce the negative effects of TD while preserving functionality and supporting continuous evolution.

- **Refactoring**, advocated by Aldaeej[M1], Besker et al. [M3], Aldaeej and Seaman [M9], Bajohr et al. [M10], Chicote [M11], Cico [M12], Apa et al. [M13], and Cico et al. [M14], restructures code to improve maintainability without altering behavior.

Strategic Approaches Strategic approaches address TD from a higher-level perspective, linking remediation efforts to long-term business and architectural goals. Rather than focusing solely on isolated fixes, these practices emphasize deliberate, organization-wide actions that shape the evolution of the software system and its underlying structure. By incorporating planned interventions such as design moves, startups can reduce TD in a way that supports scalability, adaptability, and sustained innovation.

- **Design moves**, described by Aldaeej[M1] and Besker et al. [M3], include architectural adjustments, refactoring, and documentation improvements aimed at broader TD reduction.

Process-based Approaches Process-based approaches provide a structured framework for managing TD in startups by embedding preventive and quality-focused practices into everyday workflows. Rather than treating TD remediation as an ad-hoc activity, these approaches integrate change management and quality control into the software evolution process, establishing consistent controls and monitoring mechanisms. This enables startups to maintain agility while systematically reducing the introduction and impact of new debt.

- **Change management**, presented by Aldaeej[M1], Besker et al. [M3], Aldaeej and Seaman [M9], Chicote [M11], and Cico et al. [M14], prevents new debt during software evolution by establishing controls.
- **Quality control**, reported by Aldaeej[M1], Besker et al. [M3], Chicote [M11], Cico [M12], Apa et al. [M13], and Cico et al. [M14], ensures continuous monitoring of attributes to preserve software quality.

Visual Approaches Visual approaches offer startups an effective way to make the often abstract concept of technical debt (TD) more concrete and understandable. By translating hidden or complex issues into visible artifacts, these practices enhance team communication, promote shared awareness, and support collaborative decision-making. This increased transparency helps align technical and non-technical stakeholders around TD management and fosters a proactive culture of improvement.

- **Visual Thinking**, introduced by Chicote [M11], uses physical representations (e.g., adhesive tape) to make TD tangible to team members, improving communication and awareness.

Table 6 presents a diverse set of tools used in the management of technical debt (TD) in startups, supporting both technical analysis and the control of TD-related processes and tasks.

Code Analysis Tools Among these, SonarQube and ConDebt stand out for their wide adoption in code analysis and TD estimation. Both perform static code analysis to detect issues such as duplication, excessive complexity, and violations of best practices, contributing to continuous code quality monitoring. Aldaeej[M1] highlights that these tools provide objective metrics that help teams prioritize refactoring efforts and reduce long-term maintenance costs, particularly valuable during periods of rapid growth.

Project and Task Management Tools In the area of project management, tools such as Jira, Trello, and Team Foundation Server (TFS) are frequently used to manage tasks, track bugs, and map technical risks, while also facilitating team communication. Aldaeej[M1], Klotins et al. [M5] describe how Jira offers fine-grained control over TD-related tasks through specific backlogs that integrate with agile practices. Trello, by contrast, is better suited for less formal environments, offering a visual, card-based interface that supports collaborative prioritization of tasks, as noted by Klotins et al. [M5].

Table 5. Methods and Techniques for Managing TD in Startups

Category	Method/Tech	Description	ID Paper
Analysis	Architecture Analysis	Evaluates the software design to identify architectural deficiencies such as excessive coupling, low cohesion, or design violations.	[M12] , [M1], [M3], [M9], [M10]
Analysis	Code Analysis	Uses static and dynamic tools to detect issues such as duplication, complexity, and style violations.	[M14] , [M1], [M2], [M3], [M9], [M10],[M12], [M13]
Analysis	Defect Classification	Categorizes defects (e.g., code, design, documentation) to facilitate prioritization.	[M3] , [M1]
Analysis	Documentation Analysis	Assesses documentation completeness and quality to identify gaps that affect maintainability.	[M3], [M1]
Analysis	Test Analysis	Assesses test coverage and effectiveness to identify risks related to insufficient testing.	[M3], [M1]
Automation	Test Automation	Automates testing activities to improve reliability and reduce TD.	[M13]
Corrective	Refactoring	Restructures code to improve maintainability and quality without changing its behavior.	[M14], [M1], [M3], [M9], [M10], [M11], [M12], [M13]
Management	Technical Debt Backlog Management	Organizes and prioritizes TD-related items for future iterations.	[M13]
Metrics	Productivity Metrics	Uses indicators like team velocity and cycle time to measure the impact of TD on development efficiency.	[M12], [M1], [M11]
Metrics	Quality Metrics	Includes defect counts, failure rates, and code complexity to assess software quality.	[M14], [M1], [M2], [M3], [M4] ,[M5], [M6], [M7], [M9]
Process-based	Change Management	Implements practices to control software changes and prevent the introduction of new TD.	[M14] , [M1], [M3], [M9], [M11]
Quality	Quality Control	Applies quality assurance practices to monitor and improve software quality, minimizing new TD.	[M14], [M1], [M3], [M11], [M12], [M13]
Risk-based	Business Risk Identification	Identifies and prioritizes TD that poses significant business risks.	[M1], [M3]
Risk-based	Cost-Benefit Analysis	Evaluates costs and benefits of addressing TD to support prioritization decisions.	[M1], [M3], [M2]
Risk-based	Risk Analysis	Identifies and evaluates potential TD-related risks and their business impact.	[M9]
Standards	Coding Standards	Defines and enforces consistent coding practices to reduce the risk of accumulating TD.	[M13]
Strategic	Design Moves	Strategic interventions such as architectural changes, refactoring, and documentation improvements.	[M3] , [M1]
Visual	Visual Thinking / Physical Representation	Uses visual artifacts (e.g., tape) to make TD tangible and visible to the team.	[M11]

Note: Summary of methods and techniques for TD management in software startups, sorted alphabetically by category.

Visual and Collaborative Tools Chicote [M11] introduces the Tape Board, a visual technique based on Visual Thinking, in which pieces of adhesive tape represent units of TD and are displayed on physical boards accessible to the entire team. This tangible representation fosters shared awareness and enables more transparent and participatory debt management. Complementing this approach, Scrum Velocity Charts allow teams to correlate productivity with the presence of TD, providing visual indicators of its impact and supporting more informed decision-making.

Integrated Use of Tools Taken together, the coordinated use of these tools enables startups to address TD across multiple dimensions — from automated detection and analysis to visual monitoring and strategic planning of remediation activities. In environments characterized by tight deadlines, scarce resources, and the need for rapid innovation, such tools are crucial for maintaining software quality and business scalability over time, as emphasized by Aldaeej[M1], Klotins et al. [M5].

3.3 Q3. What are the challenges and opportunities associated with managing TD?

Managing TD in startups is a multifaceted endeavor, shaped by the fast-paced, resource-constrained, and highly adaptive nature of these organizations. Table 7 provides an overview of the main challenges that hinder effective TD management, as well as the opportunities that emerge when it is approached strategically and systematically.

Challenges One of the most critical challenges is the lack of standardized definitions and metrics for TD. The absence of consensus on what constitutes TD, along with the scarcity of clear and widely accepted measurement criteria, hinders its identification and complicates comparisons across projects and organizational contexts. Besker et al. [M3], Gauthier et al. [M4], Klotins et al. [M5], Ramasubbu and Kemerer [M7], and as well as Aldaeej[M1], emphasize that this ambiguity generates uncertainty in decision-making, particularly when assessing the actual impact of TD on productivity, quality, and product evolution.

Understanding the long-term consequences of TD also presents difficulties, as its effects often manifest indirectly—such as reduced productivity, increased maintenance effort, or user dissatisfaction—making it harder to quantify the real cost of not addressing debt promptly. Klotins et al. [M5], and Besker et al. [M3] highlight these difficulties.

Prioritizing TD remediation is another recurring obstacle. It requires strategic decisions about which debts to address first—decisions frequently constrained by limited time, scarce resources, and gaps in technical expertise. Aldaeej and Seaman [M2], Gauthier et al. [M4], and together with Aldaeej[M1], show that the constant trade-off between speed and quality often leads startups to postpone essential improvements, resulting in debt accumulation that compromises product sustainability.

Communication breakdowns further exacerbate the problem. Explaining the relevance of TD to non-technical stakeholders, such as managers or investors, is often difficult due to technical jargon and the lack of clear visualizations of risk. Besker et al. [M3], and Aldaeej and Seaman [M2] emphasize how this gap may delay or prevent the support needed for corrective actions and continuous improvement initiatives.

The speed-driven culture prevalent in startups intensifies these issues. The pressure to release features rapidly frequently outweighs concerns about technical quality, fostering environments where TD accumulates unchecked. Besker et al. [M3] argue that shifting this mindset toward a more strategic, long-term vision demands organizational maturity and leadership committed to sound engineering practices, as also reinforced by Gauthier et al. [M4] and Santos et al. [M8].

Another limiting factor is the lack of standardized tools and methods, which makes TD management fragmented and heavily reliant on individual developer or team experience. Klotins et al. [M5], Njima et al. [M6], and Aldaeej[M1] report that this hampers the establishment of consistent and replicable processes for TD control.

Opportunities Despite these challenges, effective TD management presents several opportunities. One of the most significant is the improvement of software quality. When TD is addressed systematically, the resulting software tends to be more robust, scalable, and maintainable. Gauthier et al. [M4], and Besker et al. [M3] highlight that this enables startups to respond more quickly to market demands without compromising technical integrity.

Another notable benefit is the increase in team productivity. By reducing rework, technical ambiguity, and code complexity, developers can focus on higher-value tasks aligned with business goals. Aldaeej[M1], Besker et al. [M3] emphasize these productivity gains.

Effective TD management also contributes to risk reduction. Minimizing technical failures improves the startup's reputation, reliability, and customer retention—critical factors in early-stage ventures, as shown by Aldaeej and Seaman [M2].

Furthermore, a structured approach fosters alignment between technical and business areas. It enhances communication and facilitates decision-making across developers, managers, and stakeholders. Besker et al. [M3], and Aldaeej[M1] emphasize that this alignment promotes shared priorities and ensures that TD remediation efforts support broader strategic goals.

Startups that successfully manage their TD also gain the ability to deliver new features faster, maintaining agility without compromising quality. Njima et al. [M6] show that this responsiveness is essential for competitiveness and continuous innovation.

Lastly, the inherently dynamic and experimental nature of startups creates a favorable environment for adopting innovative approaches to TD management, such as visual thinking, visual tracking boards, or value-based decision models. Gauthier et al. [M4], Ramasubbu and Kemerer [M7] reinforce this, while Besker et al. [M3], Gauthier et al. [M4], and Santos et al. [M8] argue that startups' organizational flexibility allows new practices and tools to be tested and refined rapidly, positioning them as natural laboratories for advancing software engineering practices.

Table 6. Tools for Managing TD in Startups

Tool	Description	ID Paper
AnaConDebt	Tool designed to estimate and analyze technical debt through static analysis and quantitative indicators, supporting refactoring decisions.	[M1]
Jira	Project management platform that enables tracking of TD-related tasks, bugs, and risks through dedicated backlogs integrated with agile practices.	[M5], [M1]
Scrum Velocity Charts	Visual indicators that relate team productivity to TD accumulation, helping teams assess performance degradation and TD impact.	[M11]
SonarQube	Static code analysis tool used to detect code smells, complexity, duplication, and other indicators of technical debt, supporting continuous quality monitoring.	[M1]
Tape Board	Visual thinking technique where adhesive tape represents units of TD and is displayed on a shared board to enhance awareness and team engagement.	[M11]
Team Foundation Server (TFS)	Integrated development environment supporting task management, version control, and traceability of technical issues related to TD.	[M1]
Trello	Lightweight visual management tool used to organize TD tasks and promote collaborative prioritization, particularly suited to less formal environments.	[M5]

Note: Inventory of technical tools and visual aids used to assist Technical Debt management in startups, sorted alphabetically.

Table 7. Challenges and Opportunities of TD Management

Challenges and Opportunities	ID Paper
Challenges	
Absence of Standardized Tools and Methods	[M14], [M6] , [M13]
Balancing Speed and Quality in Delivery	[M13], [M9], [M10],[M12]
Decision-Making in Resource-Constrained Environments	[M11] , [M9]
Difficulty in Prioritizing Technical Debt	[M14], [M1], [M6]
Lack of a Debt Management Culture	[M12] , [M9]
Lack of Awareness and Shared Perception of TD	[M12]
Lack of Communication and Visibility	[M11]
Lack of Communication with Non-Technical Stakeholders	[M6] , [M1], [M2]
Limited Understanding of Long-Term TD Impact	[M13], [M2], [M6]
Managing TD Across Different Startup Growth Stages	[M10] , [M9]
Organizational Culture Focused on Fast Releases	[M6], [M2]
Unclear Definition and Measurement of TD	[M14] , [M2], [M6], [M9], [M13]
Opportunities	
Adoption of Innovative Practices in Startups	[M14], [M2], [M6], [M9],[M12], [M13]
Cultural and Process Transformation	[M14], [M10]
Enhanced Software Quality	[M2], [M6], [M9]
Improved Communication Among Team Members and Stakeholders	[M6] , [M2]
Increased Team Productivity	[M6], [M2]
Long-Term Benefits from Structured TD Management	[M6], [M9]
Proactive Technical Debt Management	[M14], [M12]
Reduction of Business Risk	[M6] , [M1]
Strategic Use of TD	[M10]

Note: Categorization of common challenges and potential opportunities related to Technical Debt management in software startups.

4 Discussion

This section presents the findings of the systematic mapping on TD management in startups. It interprets the results in light of the specific characteristics of startup environments, drawing connections with prior literature, and explores broader implications for both research and practice. The discussion is organized into three parts: an interpretation of the main findings, an identification of potential threats to the study's validity, and a reflection on the implications of these results for future research and practical applications.

4.1 Interpretation of Results

The study systematically mapped the literature on TD management in startups, highlighting the significant impact of TD on productivity, software quality, and long-term sustainability. Startups, driven by the need for rapid innovation and expansion, often prioritize delivery speed over technical quality, resulting in TD accumulation that hinders the maintenance and evolution of their products over time.

A total of fourteen studies published between 2017 and 2024 were analyzed, addressing approaches, methods, and tools for managing TD. One of the key findings is the growing adoption of automated tools, such as SonarQube and AnaConDebt, to detect and quantify TD. These tools are widely used for static code analysis, enabling teams to identify code duplication, complexity, and violations of coding standards. However, their effectiveness remains limited when it comes to identifying architectural issues or human and organizational factors, as noted by Klotins et al. [M5].

The results suggest that effective TD management requires a holistic approach, one that goes beyond technical detection and incorporates strategic decision-making, cultural adaptation, and communication practices. Studies such as Besker et al. [M3] show how the pressure for rapid delivery leads startups to postpone essential refactoring and maintenance activities, increasing the risk of long-term degradation.

Lean-based practices have been widely adopted to manage TD in agile environments. For example, Klotins et al. [M5] report how startups, often constrained by limited resources, make deliberate trade-offs that delay technical improvements. Similarly, Ramasubbu and Kemerer [M7] emphasize the importance of integrating TD management into quality assurance processes, arguing that TD is not solely a technical concern but also a matter of business strategy and organizational behavior.

Visual and collaborative practices also emerged as effective facilitators of TD awareness and management. The Visual Thinking technique proposed by Chicote [M11], which uses adhesive tape to represent units of TD on physical boards, exemplifies how abstract concepts can be made tangible and visible to the entire team. This method enhances communication between technical and non-technical stakeholders, increasing engagement and transparency. Similarly, tools such as Scrum velocity charts help correlate productivity drops with TD accumulation, supporting more informed prioritization decisions.

The review also reveals that startups that integrate TD management into continuous quality processes tend to achieve better long-term outcomes. As observed by Ramasubbu and Kemerer [M7], the combination of automated tools, continuous refactoring, and effective communication contributes to reducing the negative impacts of TD on scalability and maintainability.

Several studies reinforce the idea that the startup context, marked by urgency, scarcity of resources, and constant change, leads to suboptimal technical decisions and deferred maintenance.

Cultural factors were also emphasized. Santos et al. [M8] observed in their study of Brazilian startups that the emphasis on fast releases often impedes structured TD management. Balancing innovation with technical sustainability remains a significant challenge and a key differentiator between startups and more established organizations.

Moreover, the findings indicate a need for more empirical studies connecting TD management practices with measurable business outcomes. Tailored methodologies that reflect the specific constraints and dynamics of startups may lead to more effective and context-sensitive solutions.

Ultimately, this study expands the traditional view of TD management by situating it within the unique and dynamic environment of startups. While TD is often framed as a purely technical issue to be solved via refactoring, the evidence suggests that a broader perspective is needed, one that integrates organizational culture, business strategy, and human factors. This reinforces the call for holistic frameworks to support sustainable growth in early-stage software ventures.

4.2 Threats to Validity

We carefully considered the threats to the validity of this study and analyzed them according to four categories, as described by Wohlin et al. (2000): construct validity, internal validity, external validity, and conclusion validity.

With regard to conclusion validity, which concerns the extent to which the conclusions drawn from the data are reliable and replicable, the main threats include subjectivity in study selection and in the extraction and synthesis of data. To mitigate these risks, we explicitly documented the entire research protocol—comprising research questions, search string, inclusion and exclusion criteria, and selected databases—to enhance transparency and replicability. We conducted the selection process in multiple phases with clear criteria applied at each stage. Data extraction was performed using a standardized form (derived from the StArt tool and implemented in Microsoft Excel), initially completed by the lead author and subsequently reviewed by three other researchers to resolve ambiguities and enhance reliability. The data synthesis was descriptive and grounded directly in the findings of the primary studies.

Internal validity refers to the rigor of the review process, ensuring that findings accurately reflect the primary literature rather than researcher bias. Threats here include the possibility of study selection bias, since relevant papers could have been missed despite the comprehensive search string, and the risk of misclassification during the screening process. Data extraction bias is also possible, as interpretation may be subjective even when standardized forms are employed. We

mitigated these risks by conducting searches in three major databases (Scopus, IEEE Xplore, and ACM DL), complemented with backward and forward snowballing to reduce omissions. We systematically applied the inclusion and exclusion criteria, and multiple researchers participated in reviewing the data extraction to reduce bias and ensure accuracy.

Construct validity relates to how well the theoretical concepts were operationalized in the study, particularly regarding the adequacy of the search string and the categorization of data. Threats in this dimension involve the risk that the keywords used in the search string did not capture all terminological variations for technical debt or startup contexts, as well as the potential subjectivity in classifying methods, techniques, challenges, and opportunities. We mitigated these threats by iteratively refining the search string based on common terms in the literature and expanding it to include synonyms. We also tailored the data extraction form to capture elements aligned with the research questions, and categorization decisions were discussed among all authors to ensure consistency. The entire process was guided by the systematic mapping protocol proposed by Kitchenham (2007), which supports reliable construct operationalization.

Finally, external validity concerns the generalizability of the study's findings beyond the analyzed set of papers. Possible threats include publication bias, since studies with negative or inconclusive results may be underrepresented; limitations in search scope, given that the review focused on English-language articles published between 2017 and 2024 in selected databases, possibly excluding studies in other languages or venues; and the fact that findings are tailored to software startups, which may limit applicability to other domains or more mature organizations. We mitigated these risks by selecting a review period (2017–2024) that reflects recent developments in TD management, acknowledging the language restriction while noting that English remains predominant in relevant publications, and explicitly contextualizing the findings within software startups, recognizing the limitations of broader applicability.

By addressing these threats and their respective mitigation strategies, we provide a transparent account of our methodological rigor and the reliability of our results.

4.3 Implications for Research and Practice

This study highlights several directions for future research and practical improvements in TD management within startup environments.

First, there is a pressing need for more quantitative and longitudinal studies that assess the impact of TD over time. Such studies would enable a deeper understanding of how TD influences key aspects such as productivity, software quality, and cost, particularly in the dynamic context of startups.

While the reviewed literature highlights the potential of automated tools for identifying and managing TD, there remains a lack of empirical validation of their effectiveness. Future research should evaluate these tools in real-world scenarios to determine their practical value and guide their adoption.

Another significant gap is the absence of standardization in definitions, taxonomies, and metrics related to TD. This heterogeneity hampers the comparison of findings across studies and limits the development of a unified body of knowledge. Establishing consistent conceptual frameworks and measurement criteria is essential to advancing both research and practice.

Most of the analyzed studies rely on qualitative methods, such as interviews and case studies, which, although valuable for exploring context-specific insights, are subject to interpretation biases. The field would benefit from a greater number of robust quantitative investigations to strengthen the generalizability and replicability of findings.

Moreover, the rapid evolution of development tools suggests that some of the solutions discussed in the literature may be outdated or underexplored. Continuous assessment of new technologies and approaches, such as AI-assisted code review, automated refactoring, and technical debt dashboards, is necessary to keep TD management practices aligned with current industry needs.

For practitioners, the findings underscore the importance of integrating TD management into strategic planning, cultivating a culture of continuous improvement, and enhancing communication between technical and non-technical stakeholders. Startups that adopt such practices are more likely to strike a balance between agility and long-term sustainability.

5 Conclusion

The authors highlight the importance of understanding and managing this phenomenon to ensure the sustainability and quality of software products developed by emerging companies. It reveals that although Technical Debt (TD) may offer short-term benefits by accelerating development and product launch, its long-term negative impacts can significantly compromise productivity, quality, and business viability.

Startups, pressured by the need to innovate rapidly and compete in dynamic markets, often accumulate technical debt as a result of suboptimal technical decisions made in favour of speed. While this behavior may initially seem advantageous, it leads to a build-up of problems that manifest as increased maintenance effort, higher costs, reduced product quality, and challenges in software evolution. Therefore, managing TD is a critical aspect that must be addressed strategically and integrated throughout the product life cycle.

The authors identified various approaches through the systematic mapping, methods, and tools that have been used to manage TD, ranging from code and architecture analysis to more comprehensive techniques such as the use of quantitative metrics and management frameworks. However, the research also highlights several limitations, including the lack of standardization in management practices and the absence of robust empirical evidence validating the effectiveness of existing approaches. These limitations underscore the need for more quantitative, long-term studies and the development of new automated tools that can integrate technical, human, and organizational aspects.

Moreover, the authors point out tha effective TD management requires a holistic approach that considers not only technical issues but also cultural and organizational factors.

Effective communication between technical and managerial teams, continuous education on the impacts of TD, and the implementation of agile practices are essential elements for mitigating associated risks. Integrating such practices can help startups balance the need for rapid innovation with maintaining product quality.

Managing TD in startups is a complex yet essential challenge for the sustainable success of these companies. Adopting well-structured mitigation strategies supported by empirical evidence can significantly enhance startups' ability to grow and compete in an ever-evolving environment.

The authors suggest that future research should focus on developing and empirically validating context-aware frameworks tailored to the specific constraints of startups. This includes creating tools and models that integrate technical, organizational, and human factors, as well as conducting longitudinal and quantitative studies to measure the long-term impact of TD on software quality, team productivity, and business sustainability. Additionally, studies that explore how startups can balance rapid innovation with sustainable technical practices, particularly during different stages of growth, would contribute valuable insights to both academia and industry.

Mapped Studies

[M1] Abdullah Aldaeej. "Towards Effective Technical Debt Decision Making in Software Startups". In: vol. 44. 2020, p. 22. doi: 10.1145/3356773.3356793. url: https://doi.org/10.1145/3356773.3356793.

[M2] Abdullah Aldaeej and Carolyn Seaman. "The Negative Implications of Technical Debt on Software Startups: What They Are and When They Surface". In: Proceedings of the 5th International Workshop on Software-Intensive Business: Towards Sustainable Software Business. 2023, pp. 23–30. isbn: 9781450393027. doi: 10.1145/3524614.3528629. url: https://doi.org/10.1145/3524614.3528629.

[M3] T. Besker, A. Martinia, and R. et al Lokuge. "Embracing Technical Debt, from a Startup Company Perspective". In: Proceedings 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2018, pp. 10–11. isbn: 9781538628010. doi: 10.1109/ICSME.2018.0000051.

[M4] Francois Gauthier et al. "Trade-Offs in Managing Risk and Technical Debt in Industrial Research Labs: An Experience Report". In: Proceedings of the 3rd International Conference on Technical Debt (2020). doi: 10.1145/3387906.3388623.

[M5] Eriks Klotins et al. "Exploration of Technical Debt in Start-Ups". In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (2018). doi: 10.1145/3183519.3183539.

[M6] Njima, M., Demeyer, S. (2019). Value-Based Technical Debt Management: An Exploratory Case Study in Start-ups and Scale-ups. In Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems (IWSiB 2019) (pp. 54–59). Association for Computing Machinery. DOI: 10.1145/3340481.3342739.

[M7] Narayan Ramasubbu and Chris F. Kemerer. "Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests". In: IEEE Transactions on Software Engineering 45.3 (2019), pp. 285–300. doi: 10.1109/TSE.2017.2774832.

[M8] Adauto Santos et al. "Technical Debt in Brazilian Software Startups: Perceptions of Professionals in Paran´a". In: Proceedings of the XXII Brazilian Symposium on Software Quality. 2023, pp. 120–127. doi: 10.1145/3629479.3629482

[M9] Abdullah Aldaeej and Carolyn Seaman. "A rule-based decision model to support technical debt decisions: A multiple case study of web and mobile app startups". In: Information and Software Technology 175 (2024), p. 107542. doi: 10.1016/j.infsof.2024.107542. url: https://doi.org/10.1016/j.infsof.2024.107542.23

[M10] Stefan Bajohr et al. "Startups Transitioning from Early to Growth Phase: A Pilot Study of Technical Debt Perception". In: Proceedings of the 11th International Conference on Software Business (ICSOB). Vol. 389. Lecture Notes in Business Information Processing. Springer, 2020, pp. 45–59. doi: 10.1007/978-3-030-67295-9 4.

[M11] Marcos Chicote. "Startups and Technical Debt: Managing Technical Debt with Visual Thinking". In: 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart). Buenos Aires, Argentina: IEEE, 2017, pp. 10–14. doi: 10.1109/SoftStart.2017.6.

[M12] Orges Cico. "Technical Debt Trade-Off: Experiences from Software Startups Becoming Grownups". In: Proceedings of the International Conference on Software Business (IC-SOB). Vol. 370. Lecture Notes in Business Information Processing. Trondheim, Norway: Springer, 2019, pp. 413–418. doi: 10.1007/978-3-030-33742-1 34.

[M13] Cecilia Apa et al. "The Perception and Management of Technical Debt in Software Startups". In: Software Startups. Vol. 370. Lecture Notes in Business Information Processing. Springer, 2020, pp. 52–67. doi: 10.1007/978-3-030-35983-6 4.

[M14] Orges Cico et al. "Toward a Technical Debt Relationship with the Pivoting of Growth Phase Startups". In: Product-Focused Software Process Improvement. Ed. by Luca Ardito et al. Vol. 13126. Lecture Notes in Computer Science. Springer, 2021, pp. 265–280. doi: 10.1007/978-3-030-91452-3 18.

References

Albuquerque, D., Guimarães, E., Tonin, G., Rodriguez, P., Perkusich, M., Almeida, H., Perkusich, A., and Chagas, F. (2022). Managing technical debt using intelligent techniques - a systematic mapping study. *IEEE Transactions on Software Engineering*, XX(X):1–19.

Berg, S., Nguyen-Duc, A., and Abrahamsson, P. (2018). Software startup engineering: A systematic mapping study. *Journal of Systems and Software*, 144:255–274.

Cunningham, W. (1992). The wycash portfolio management system. In *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum), OOPSLA '92*, New York. ACM, ACM.

Das, D., Maruf, A. A., Lambaria, N., Abdelfattah, A. S., et al. (2021). Technical debt resulting from architectural degradation and code smells: a systematic mapping study. *ACM SIGAPP Applied Computing Review*, 21(4):1–16.

Gandomani, T. J., Zulzalil, H., and Bahsoon, R. (2024). Empowering software startups with agile methods and practices: A design science research. *Software: Practice and Experience*, 55(2):220–242.

Gomes, F. ; Dos Santos, E. . F. S. . M. M. . M. T. . S. R. (2022). Investigating the point of view of project management practitioners on technical debt - a preliminary study on stack exchange.

Guerino, G., Martinelli, S., Choma, J., Leal, G. C., Balancieri, R., and Zaina, L. (2024). Investigating ux work in software startups: A survey about attitudes, methods, and key challenges. *Journal of the Brazilian Computer Society*, 30(1):621–638.

Guerino, G. C., Martinelli, S., Choma, J., Leal, G. C. L., Balancieri, R., and Zaina, L. (2023). Perceptions about usefulness and attitudes toward ux work: A survey with software startup brazilian professionals. In *Proceedings of the 22nd Brazilian Symposium on Human Factors in Computing Systems (IHC 2023)*, pages 120–127, Maceió, AL, Brazil. Association for Computing Machinery.

Hamed, A. S. E. and et al. (2023). A proposed technical debt management approach applied on software projects in egypt. *Journal of Systems and Software*, 171:158–176.

Hernandes, E., Zamboni, A., Fabbri, S., and di thommazo, A. (2012). Using gqm and tam to evaluate start – a tool that supports systematic review. *CLEI Electronic Journal*, 15.

Kitchenham, B. Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. In *Technical Report EBSE-2007-01*, Keele University. ACM, School of Computer Science and Mathematics.

Klimczyk, P. and Madeyski, L. (2020). Technical debt aware estimations in software engineering: A systematic mapping study. *e-Informatica Software Engineering Journal*, 14(1):61–76.

Klot, T., Martini, A., and Bosch, J. (2019). Technical debt triage in backlog management. In *Proceedings of the Second International Conference on Technical Debt*, pages 13–22.

Li, Z., Avgeriou, P., and Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 110:1–20.

Murillo, M. I., López, G., Spínola, R., Guzmán, J., Rios, N., and Pacheco, A. (2023). Identification and management of technical debt: A systematic mapping study update. *Journal of Software Engineering Research and Development*, 11(8):1–21.

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., and Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. *Information and Software Technology*, 56(10):1200–1218.

Ribeiro, L. F., de F. Farias, M. A., Mendonça, M., and Spínola, R. O. (2016). Decision criteria for the payment of technical debt in software projects: A systematic mapping study. In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016)*, pages 572–579, Salvador, Bahia, Brazil. SCITEPRESS.

Saraiva, D., Neto, J. G., Kulesza, U., Freitas, G., Reboucas, R., and Coelho, R. (2021). Technical debt tools: A systematic mapping study. *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021)*, 2:88–98.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. The Kluwer International Series in Software Engineering. Springer.

A Quality Assessment of Primary Studies

Table 8. Quality Assessment Scores of the Selected Studies

Study ID	QA1	QA2	QA3	QA4	QA5	Total Score
[M1]	1	1	1	1	1	5.0
[M2]	1	0.5	1	1	0.5	4.0
[M3]	1	0.5	0.5	1	0.5	3.5
[M4]	1	1	1	1	1	5.0
[M5]	1	1	1	1	0.5	4.5
[M6]	1	1	1	1	0.5	4.5
[M7]	1	0.5	1	1	1	4.5
[M8]	1	1	1	1	0.5	4.5
[M9]	1	0.5	0.5	1	0.5	3.5
[M10]	1	1	0.5	1	0.5	4.0
[M11]	1	0.5	1	1	1	4.5
[M12]	1	0.5	0.5	1	0.5	3.5
[M13]	1	1	1	1	0.5	4.5
[M14]	1	0.5	1	1	0.5	4.0