Technical Credit: Industry Views on Benefits and Barriers

Alessio Bucaioni
Mälardalen University
Västerås, Sweden
alessio.bucaioni@mdu.se

Ian Gorton
Northeastern University
Boston, USA
i.gorton@northeastern.edu

Patrizio Pelliccione
Gran Sasso Science Institute
L'Aquila, Italy
patrizio.pelliccione@gssi.it

Abstract

Technical credit has recently been proposed as a concept encompassing system elements and engineering practices that reduce development friction and enable sustainable software evolution. While promising, it remains an emerging idea with limited consensus on its definition and practical implications.

This paper reports an empirical study investigating how practitioners perceive and operationalize technical credit, and what challenges they foresee in its adoption. We conducted an online survey with 31 experienced software professionals, complemented by six semi-structured interviews for deeper insights.

Our findings show the concept strongly resonates with industry practitioners. We highlight concrete scenarios and artifacts where technical credit is seen as impactful, as well as opportunities for embedding it into tools, processes, and architectural decision-making.

Keywords

Technical credit, technical debt

ACM Reference Format:

Alessio Bucaioni, Ian Gorton, and Patrizio Pelliccione. 2026. Technical Credit: Industry Views on Benefits and Barriers. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEIP '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3786583.3786862

1 Introduction

Practitioners in software-intensive industries constantly navigate the tension between rapid product delivery and building systems that remain adaptable and cost-effective over time. Tight deadlines, evolving requirements, and competitive pressures often push teams toward short-term decisions that accelerate delivery today but compromise maintainability tomorrow. The metaphor of Technical Debt (TD) has become the dominant way to describe these negative long-term consequences [9], a language for the liabilities created by expedient trade-offs.

In practice, teams also make deliberate investments that pay dividends for many years: modular architectures, automation frameworks, and refined APIs that reduce future development friction. An automotive platform team, for example, may invest in building a modular architecture to enable entire subsystem replacement without impacting the rest of the codebase. This choice may increase initial delivery time, yet later allows rapid compliance updates, product variants, and feature integration, saving millions downstream in engineering costs. These long-term wins are rarely identified, tracked, or communicated with the same visibility as TD.

Recently, Technical Credit (TC) has been proposed as a complementary concept: "the benefits that result from making strategic design decisions that require a higher initial investment, but offer highly advantageous long-term effects for the evolution of a system" [10]. Where TD describes the drag on future development, TC acts as a counterbalance, capturing the enablers of sustainable evolution. Elements of this idea are not entirely new. As Weiss and Parnas observed decades ago[1], a key role of the architect is to anticipate future system changes and design accordingly. What TC promises are instruments for recognising, valuing, and communicating such strategic foresight. From this perspective, TC is a practical counterpart to TD that could inform prioritisation, justify investment, and guide decision-making. However, despite its intuitive appeal, TC remains ill-defined. There is little empirical evidence on how industry perceives it, in which contexts it is most applicable, and what barriers may impede its adoption.

To address this gap, we aim to answer the following Research Questions (RQs):

- RQ1: *How do practitioners understand and interpret the concept of TC?* We examine whether practitioners see TC as a useful complement to TD, and whether their interpretation is consistent with the definition. The question also addresses how TC resonates with their current mental models of software quality, maintainability, and long-term architectural thinking.
- RQ2: *In which software engineering contexts and scenarios do practitioners perceive TC as valuable or applicable?* We explore the relevance and utility of TC in software engineering contexts and scenarios. This question aims to ground the TC concept in everyday engineering practices and workflows.
- RQ3: *What are the perceived benefits and challenges of adopting TC?* We identify benefits practitioners associate with TC, such as improved prioritization, architectural foresight, or cross-team communication, and the challenges they foresee operationalizing it. These may include measurement difficulties, resistance to change, or overlap with existing quality models.
- RQ4: *How willing are practitioners to operationalize TC in their projects?* We assess how practitioners may introduce TC concepts into their organizations. We explore how they rank TC's priority relative to existing practices, and whether they see a path toward integrating TC into tools, processes, or decision frameworks.

To address these RQs, we gathered industrial perspectives through a survey of 31 experienced practitioners across multiple sectors, followed by six in-depth interviews to explore key themes.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
ICSE-SEIP '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2426-8/26/04
https://doi.org/10.1145/3786583.3786862

[1] David M. Weiss, David Lorge Parnas, Definition and Documentation of Software Architectures, https://www.academia.edu/download/41648293/Definition_and_Documentation_of_Software20160127-1781-1yhbfru.pdf

The findings suggest strong interest from industry, reveal scenarios where TC could be most impactful, and surface actionable considerations for integrating TC into tools, processes, and architectural decision-making.

2 Study Design

Given the novelty of TC and our aim to investigate real-world practice, we adopted a mixed-method research design. This combines a structured online survey with semi-structured follow-up interviews. This approach is consistent with established empirical software engineering guidelines for studying emerging phenomena in industrial contexts [18, 22]. The survey provides breadth, capturing trends and patterns across a diverse participant pool. The interviews add depth, enabling richer contextualization and interpretation of practitioner viewpoints. Together, these methods offer a holistic picture of how TC is perceived in industrial software engineering settings. We provide a public replication package[2] including the survey instrument, anonymized responses, interview transcripts, and corresponding codings.

Population. The targeted population consists of practitioners involved in the development and evolution of software-intensive systems. Following recommendations in [14], to ensure that respondents had relevant and substantial experience, we selected participants from our extended professional network (purpose-sampling strategy [17]). The survey was administered using Google Forms. Invitations were sent via email to potential participants and participation was voluntary and anonymous.

All participants were informed of the purpose of the study, data handling procedures, and their rights under the General Data Protection Regulation (GDPR). The design and administration of the survey adhered to the ACM Publications Policy on Research Involving Human Participants and Subjects [1], ensuring full compliance with ethical and privacy standards. Several survey participants volunteered their email address, indicating their willingness to engage in further discussions.

Survey. Table 1 summarizes the structure of our survey, the questions, and how they relate to our RQs. The survey has been designed following practices for empirical software engineering research [8, 18, 22]. Before launching the survey, we conducted a pilot with five experienced practitioners from our professional network [14]. The pilot helped refine questions, identify inconsistencies, and detect mistakes in the survey structure. As a result, we removed one question that duplicated the intent of the current Q15, focusing on refining the definition of TC. We included optional open-ended fields for respondents to elaborate on their selections or highlight perspectives not anticipated during survey design.

Interviews. To complement the survey findings we conducted semi-structured interviews with a subset of respondents. Of the 12 interview participants who provided their emails for follow-up, we scheduled 6 interviews. These participants had diverse roles and backgrounds, including architects, software engineers, and technical managers, with a minimum of six years professional experience.

The interviews helped triangulate results, clarify ambiguous responses, and gather deeper insights from practitioners on how the concept of TC can be applied in real-world settings. Each online interview lasted approximately 30 minutes and followed a semi-structured interview protocol. This allowed us to explore aspects of the survey responses in greater depth while remaining open to emergent themes. Key discussion areas included individual interpretations of TC, perceived advantages and limitations, and organizational readiness for adoption.

With participant's informed approval, we automatically transcribed the interviews. Transcriptions were then anonymized to remove any identifying information, including names, company references, or project-specific details. The study complied with GDPR and followed the ACM Publications Policy on Research Involving Human Participants and Subjects [1]. Formal ethics approval was not sought, as the minimal-risk, voluntary nature of the study did not require review at our institution. Informed consent, data protection, and anonymization procedures were strictly followed. Fully anonymized interview transcripts are provided in the replication package. Sharing anonymized qualitative data is permitted under institutional guidelines and consistent with participant consent. No identifiable or sensitive personal data is included, and all data were handled confidentially and used solely for research purposes.

Table 1: Summary of Survey Questions on Technical Credit. Abbreviations of question types: OC = Open-ended, MC = Multiple choice (single answer), MS = Multiple selection (choose all that apply), LS = Likert scale (1 to 5), YN = Yes/No question.

RQ	ID	Question (truncated)	Type
—	Q1	Job Title	OC
—	Q2	Years of Experience in Software Engineering	MC
—	Q3	Primary Industry Sector	MC
—	Q4	Primary Role in Software Development	MC
—	Q5	Team Size You Typically Work With	MC
—	Q6	Type of Development Projects you work In	MC
—	Q7	Familiarity with Technical Debt (TD)	LS
—	Q8	Importance of Managing TD in Your Role	LS
—	Q9	What aspects of TD do you find most difficult? (choose 3)	MS
—	Q10	Frequency of encountering TD in your projects	LS
—	Q11	Value of quantifying TD for your work	LS
—	Q12	Additional challenges related to TD	OC
RQ1/4	Q13	Value in TC as a "credit score"	LS
RQ1	Q14	Agreement with TC definition	LS
RQ1	Q15	Suggestions for refining the definition of TC	OC
RQ1/4	Q16	Importance of industry-wide TC standards	LS
RQ2	Q17	When is TC most critical?	MC
RQ2	Q18	Scenario: Abstraction layer for platform changes	LS
RQ2	Q19	Scenario: Configuration for client customization	LS
RQ2	Q20	Scenario: Configurable performance metrics	LS
RQ2	Q21	Scenario: Documenting ADRs	LS
RQ2	Q22	Scenario: CI/CD pipeline with automated tests	LS
RQ2	Q23	Scenario: Compliance-ready healthcare system	LS
RQ2	Q24	Scenario: Modular e-commerce platform	LS
RQ2	Q25	Scenario: Product line development	LS
RQ2	Q26	Other scenarios representing TC	OC
RQ3	Q27	Benefits of TC	MS
RQ3	Q28	Challenges in operationalizing/measuring TC	MS
RQ4	Q29	Priority for adopting TC in your organization	LS
RQ1	Q30	TC vs TD in importance for long-term success	MC
—	Q31	Willingness to join 30-min interview	YN
—	Q32	Contact information (if yes)	OC

[2]https://github.com/AlessioBucaioni/icseseip2025/tree/main

Data Analysis and Synthesis. Our study employed both quantitative and qualitative data collection instruments, including closed-ended survey items (e.g., Likert-scale, multiple-choice), open-ended questions, and interviews. Accordingly, we applied a combination of descriptive statistics and qualitative data analysis techniques to interpret the results.

For survey questions, we used descriptive statistics to summarize the responses. For ordinal data we report the mode, median, and mean to capture perceptions. For nominal and categorical questions, we computed response distributions and visualized these through frequency plots and tables to highlight dominant patterns. For the open-ended survey responses and interview transcripts, we conducted a qualitative analysis based on grounded theory methods [19]. This approach is particularly suited for exploring how practitioners interpret and engage with new or emerging concepts such as TC.

Using an inductive analysis strategy, we applied a two-step coding process: (i) open coding identified recurring concepts across participants' responses; and (ii) axial coding related these concepts, organized them into higher-level categories (e.g., benefits, operational challenges), and examined their interdependencies. This procedure was applied consistently across both open-ended survey questions and the six semi-structured interviews.

Coding was performed independently by two authors. Initial discrepancies were discussed and resolved collaboratively, ensuring inter-coder agreement and analytical consistency. We continued coding until we reached thematic saturation, where no substantially new categories or themes emerged. The results offer many valuable insights, As examples, we directly cite representative quotations to support interpretation and preserve the voice of participants. This integration of coded insights and illustrative quotes improves transparency and grounding.

Threats to Validity. Following established guidelines [18, 22], the threats to validity of our findings are categorized into construct, internal, external, and conclusion validity.

Construct validity. We based the questionnaire design on prior survey research practices in software engineering [8, 18, 22], we reviewed the questions iteratively to reduce unintentional bias in question phrasing or expected outcomes, and conducted a pilot study with five experienced practitioners. This process allowed us to refine the wording of questions and remove one duplicate.

To minimize participant misinterpretation, questions were framed using domain-relevant terminology, with optional open-ended responses to clarify intent or expand on predefined answers. Participants could still respond differently simply due to the awareness of being studied - a phenomenon known as evaluation apprehension. To address this, the invitation email explained the general purpose of the study and avoided language that could bias responses. The follow-up interviews also helped mitigate this effect by allowing participants to verbalize their understanding of TC in their own words.

These conversations helped identify cases where interpretations diverged and to clarify ambiguous survey responses. However, interviews also introduce new construct-related risks, including interviewer bias and differential probing, which we mitigated by using a semi-structured interview protocol and taking detailed, neutral notes during each session. The questionnaire was also fully anonymous, with contact details collected only voluntarily, reducing the likelihood of social desirability bias.

Internal validity. We mitigated internal threats in two ways. First, we ensured that the instrument was as clear and direct as possible, avoiding abstract or compound questions that require interpretation. Second, instrumentation threats where flaws in the instrument affect the results were addressed through the pilot and validation process mentioned above. Maturation effects, such as participant fatigue or disengagement due to a lengthy survey, were considered during the pilot phase. The average completion time was around 10 minutes, and the final survey was structured to balance multiple choice, Likert-scale, and optional comment fields to maintain engagement.

The fact that all 31 participants completed the survey suggests that this threat was minimal. We conducted interviews within a few weeks of survey completion, reducing the likelihood that participants' views changed over time. However, because the interviews were conducted by the researchers themselves, researcher expectations and confirmation bias cannot be completely ruled out.

External validity. Our sampling approach was purposive, selecting participants from our professional network to ensure relevant expertise. While this limits statistical generalizability, it aligns with our study goal: understanding the perceptions of experienced practitioners toward TC. The sample included a diverse set of roles (e.g., architects, engineers, managers), sectors (e.g., IT, automotive, finance), and experience levels, improving the ecological validity of our findings.

A possible threat is opinion bias, where participants with particularly strong views may be more likely to respond. However, the distribution of responses (including neutral or undecided positions) suggests that this was not a dominant effect in our data. The interview subset consisted of six survey volunteers. While this enabled us to delve deeper into the survey responses and validate key findings, the sample may reflect self-selection bias, as participants with particular interested or investment in the topic may have been more likely to volunteer.

Conclusion validity. Given our sample size of 31 responses, we do not conduct statistical hypothesis testing. Instead we focus on descriptive analysis and qualitative interpretation. This limits the statistical generalizability, but is appropriate for an exploratory study centred on perception and adoption readiness. For the open-ended survey questions and interviews, we applied a two-phase qualitative coding process as described earlier.

To enhance transparency and preserve the richness of practitioner insights, we also include direct quotes to illustrate key discussion points. This combination of coding and illustrative quoting balances methodological rigour with interpretability. As with any qualitative analysis however, the coding process involves researcher interpretation. While we took care to remain neutral and consistent across coders, some degree of subjectivity is inevitable.

3 Survey Results

Table 2 presents the demographic and professional characteristics of the 31 survey participants. Each participant is assigned an ID (Ri), and those who participated in follow-up interviews are also assigned an Interviewee ID (I). 38.7% of participants identified as

Table 2: Demographic and professional characteristics of survey participants. Abbreviations: L/M = Leader/Manager, SE = Software Engineer, Arch = Architect, S/RE = Safety/Reliability Engineer, Exp. = Experience (years), Int. ID = Interviewee ID.

ID	Role	Exp.	Industry	Size	Projects	Int. ID
R1	L/M	> 10	Automotive	> 50	Both	
R2	SE	6–10	Automotive	< 10	Greenfield	
R3	L/M	> 10	Varied	< 10	Both	
R4	L/M	> 10	IT	10–50	Both	
R5	Arch	> 10	Government	10–50	Both	
R6	L/M	> 10	Automotive	> 50	Both	
R7	Arch	> 10	Automotive	> 50	Both	
R8	L/M	> 10	Other	10–50	Both	I3
R9	Arch	> 10	Varied	> 50	Both	
R10	SE	> 10	IT	> 50	Both	
R11	SE	> 10	Varied	< 10	Both	
R12	S/RE	6–10	Other	> 50	Both	
R13	Arch	> 10	Other	> 50	Both	I5
R14	L/M	6–10	IT	10–50	Both	I4
R15	Arch	> 10	IT	> 50	Evol./Maint.	I2
R16	SE	6–10	IT	10–50	Both	
R17	SE	> 10	Varied	10–50	Both	
R18	SE	2–5	IT	10–50	Both	
R19	SE	> 10	Finance	< 10	Both	
R20	L/M	6–10	IT	> 50	Both	
R21	Arch	> 10	Finance	> 50	Both	
R22	S/RE	< 2	Automotive	10–50	Evol./Maint.	
R23	Arch	> 10	Automotive	< 10	Both	
R24	SE	2–5	Other	< 10	Both	
R25	L/M	> 10	IT	< 10	Evol./Maint.	
R26	Arch	> 10	IT	< 10	Both	
R27	SE	> 10	Automotive	> 50	Both	
R28	SE	> 10	Government	10–50	Both	
R29	Arch	> 10	IT	< 10	Both	I1
R30	SE	> 10	IT	> 50	Both	
R31	SE	> 10	Automotive	10–50	Both	I6

software engineers, 29.0% as architects, 25.8% as leaders or managers. A smaller proportion (6.5%) reported working specifically in safety or reliability engineering roles. 74.2% of participants reported significant industry experience, having more than 10 years of professional background in software development or architecture. 16.1% had between 6 and 10 years of experience, while a smaller segment had between 2 and 5 years (6.5%) or less than 2 years (3.2%).

Respondents came from a variety of sectors. 35.5% worked in IT, 22.6% in automotive, and a mix of government, finance, and other sectors (each 6.5%). Participants categorized as varied (12.9%) reported experience across multiple domains, while another (12.9%) included niche sectors not otherwise specified.

Participants reported working in teams of varying sizes. 38.7% worked in large teams (>50 people), followed by medium-sized teams (10 to 50 people, 32.3%) and small teams (<10 people, 29.0%). 87.1% of participants reported involvement in both greenfield and maintenance/evolution projects, suggesting broad experience across the software lifecycle. Only a small number worked exclusively on either greenfield (3.2%) or maintenance (9.7%) projects.

3.1 How Do Practitioners Perceive the Concept of TC? (RQ1)

To explore how practitioners perceive the concept of TC we examined their responses to the four specific questions targeted at this RQ (Table 1). These are summarised in Tables 3 and 4. Participants were asked if they saw value in operationalizing TC as a credit score that quantifies long-term, often-overlooked software advantages. Responses leaned positively, with a median and mode of 4 on a 5-point Likert scale. The mean score was 3.55, suggesting a moderately positive outlook. Specifically, 58% rated the concept with a value of 4 or 5, indicating that most practitioners perceive such a scoring system as potentially useful.

Table 3: Summary of Responses for RQ1: Perception of TC (1=Low, 5=High)

Question	Likert Value	Count	%	Median	Mode	Mean
Q13—Value in TC as a "credit score	1	1	3.23%	4	4	3.55
	2	6	19.35%			
	3	6	19.35%			
	4	11	35.48%			
	5	7	22.58%			
Q14—Agreement with TC definition	1	3	9.68%	4	4	3.77
	2	1	3.23%			
	3	4	12.90%			
	4	15	48.39%			
	5	8	25.81%			
Q16—Importance of industry-wide TC standards	1	3	9.68%	4	4	3.52
	2	3	9.68%			
	3	8	25.81%			
	4	9	29.03%			
	5	8	25.81%			

We also asked participants whether our definition of TC resonated with their understanding. The results were favourable, (median=4 mode=4, mean =3.77), with 74% selecting 4 or 5, indicating the majority of respondents found the definition clear, relevant, and consistent with their mental model of software sustainability and beneficial architectural choices. To better understand how the definition could be improved, we asked participants for suggestions. Ten respondents provided input. Analysis using open and axial coding resulted in five categories: Clarification of Scope (4), Communication Improvements (2), Measurement & Applicability (1), Conceptual overlap & Redundancy (1), and Expansion With New Lenses (1).

The most common theme was clarifying the scope of TC. For instance, R6 asked if TC should also *"cover supporting systems and infrastructure, not only design"*, while R8 wondered whether it includes *"implementation, early automation of modules, [and] investment in documentation"*. R10 pointed to communication challenges: *"The stakeholder justification needs fleshing out"*. R26 stressed the need for practical applicability: *"It needs to be defined as something that can be measured by the teams themselves"*. Finally, R14 offered an expanded version of the definition, framing TC as a strategic asset that *"empowers the solution provider to proactively steward the product-building trust through technical leadership, aligning agile delivery with evolving business goals"*. When asked how important it

might be to develop industry-wide standards or frameworks for TC, participants gave slightly more varied responses (median=mode=4, mean=3.52). 55% of participants selected 4 or 5, suggesting a general consensus on the value of formalizing TC at an industry level, even though some respondents were uncertain or sceptical.

Table 4: Summary of Responses for RQ1: Perception of TC

Q30—TC vs TD in importance for long-term success	Count	%
Both are equally important	10	32.26%
TC is more important	8	25.81%
TD is more important	6	19.35%
I'm not sure	7	22.58%

To understand how TC is positioned relative to the established concept of TD, participants were asked to compare the importance for long-term software success (Table 4). 32.3% considered TC and TD equally important, 25.8% stated that TC is more important, and 19.4% considered TD more important. Notably, 22.6% were unsure, which likely reflects the novelty of the TC concept. These results indicate that TC is seen by many as complementary to TD, and not a competing or secondary concern.

Summary RQ1: Responses suggest practitioners are open to the concept of TC and find the proposed definition clear and relevant. While opinions on standardization and relative importance vary, the responses indicate early-stage conceptual acceptance and perceived value. These results lay the groundwork for further refinement, operationalization, and community discussion.

3.2 Where do practitioners perceive TC as valuable or applicable? (RQ2)

To understand when TC is perceived as most relevant, we asked participants to reflect on both general development stages and specific engineering practices (Table 5 and Table 6). Regarding the overall development lifecycle (Table 5), participants were evenly split between those who believe TC is equally important across all stages (41.94%) and those who emphasized the early phases of software development, such as architecture and design (also 41.94%). 9.68% highlighted the late stages (maintenance and scaling) or mid-lifecycle activities (feature implementation, 6.45%).

Table 5: Summary of Responses for RQ2: Where to Apply TC

Q17—When is TC most critical?	Count	%
Equally important across all stages	13	41.94%
Early stages (architecture and design)	13	41.94%
Late stages (maintenance and scaling)	3	9.68%
Mid-development (feature implementation)	2	6.45%

When evaluating concrete practices, respondents valued architectural strategies that future-proof systems and reduce long-term costs (Table 6). For instance, the use of abstraction layers to decouple systems from underlying platforms (e.g., middleware, databases) received particularly high support (median=mode=5, mean=4.23). 71.0% of participants rated this scenario 4 or 5, showing strong alignment with TC principles. Similarly, Architectural Decision Records (ADRs) to capture design rationales scored highly (median=mode=5, mean=4.13), reinforcing that TC is linked with long-term knowledge preservation and design traceability.

Table 6: Summary of Responses for RQ2: Where to Apply TC (1=Low, 5=High)

Question	Likert Value	Count	%	Median	Mode	Mean
Q18—Abstraction layer for platform changes	1	1	3.23%	5	5	4.23
	2	1	3.23%			
	3	5	16.13%			
	4	7	22.58%			
	5	17	54.84%			
Q19—Configuration parameters for customization	1	2	6.45%	4	4	3.52
	2	3	9.68%			
	3	9	29.03%			
	4	10	32.26%			
	5	7	22.58%			
Q20—Configurable metrics for SLA monitoring	1	3	9.68%	4	4	3.37
	2	5	16.13%			
	3	7	22.58%			
	4	11	35.48%			
	5	4	12.90%			
Q21—Documenting ADRs	1	1	3.23%	5	5	4.13
	2	2	6.45%			
	3	5	16.13%			
	4	7	22.58%			
	5	16	51.61%			
Q22—CI/CD pipeline with automated testing	1	4	12.90%	4	5	3.84
	2	1	3.23%			
	3	7	22.58%			
	4	14	45.16%			
	5	5	16.13%			
Q23—Compliance-ready healthcare system	1	2	6.45%	4	4	3.55
	2	7	22.58%			
	3	3	9.68%			
	4	10	32.26%			
	5	9	29.03%			
Q24—Modular e-commerce platform	1	1	3.23%	4	4	3.87
	2	2	6.45%			
	3	6	19.35%			
	4	11	35.48%			
	5	10	32.26%			
Q25—Product line development strategy	1	3	9.68%	3	3	3.52
	2	10	32.26%			
	3	8	25.81%			
	4	6	19.35%			
	5	2	6.45%			

Other scenarios received positive assessments. CI/CD pipelines with automated testing were widely recognized as valuable investments for enabling scalable and stable evolution (median 4, mode 5, mean 3.84), as was the use of modular architectures to integrate with third-party services such as payment providers (median=mode=4, mean=3.87). These practices highlight how TC is associated with architectural flexibility, automation, and readiness for change. Similarly, designing healthcare systems to comply with evolving regulations without requiring major redesigns was positively rated (median=mode=4, mean=3.55).

The use of configuration parameters for customer-specific customization, avoiding code changes, also scored well (median=mode=4, mean=3.52), suggesting that TC is not limited to architecture; instead, it extends to implementation strategies that enhance system adaptability. Emitting performance metrics for monitoring Service Level Agreements (SLAs), while still seen as valuable, received mixed ratings (median=mode=4, mean=3.37), possibly due to differing perceptions of how observability practices relate to strategic credit.

The most mixed views emerged when assessing the TC value of product line strategies, where systems are developed as part of a shared family rather than stand-alone projects. Although 14 participants rated this scenario as 4 or 5, the ratings ranged widely (median=mode=3, mean=3.52). This suggests that while some practitioners recognize its long-term benefits, others may see it as context-dependent or less directly connected to individual engineering decisions. 11 participants provided open-ended responses describing additional scenarios where TC can be applicable. Analysis through open and axial coding to resulted in six categories: Strategic Considerations (4), Architectural Flexibility (2), Automation and Tool Support (2), Reuse and Standardization (1), Internal Quality Practices (1), and Knowledge Transfer (1).

The most frequent category, Strategic Considerations, included reflections on how TC plays out in different organizational contexts. R5 noted that TC can span *"enterprise/common capabilities that can be used across a number of systems"*. R20 emphasized the situational nature of TC: *"If we have a stable product [...] then knowing the current TD and the possible variants to move forward, (along) with their cost and TC gain would be of value"*. In terms of Architectural Flexibility, R29 explained that *"trying to keep the architecture open and flexible and integrating with existing standards and formats is a good TC practice"*. Similarly, Automation and Tool Support included ideas such as (R6) *"use of machine-readable product descriptions as an enabler for CI/CD and automatic synthesis, assessment, documentation"*.

Less frequent categories included Reuse and Standardization, exemplified by references to centralizing architectural decisions across systems; Internal Quality Practices, which emphasized the importance of design choices that prioritize simplicity and quality; and Knowledge Transfer, reflected in scenarios like feeding research back into engineering practice.

Summary RQ2: The finding indicate TC is linked with long-term knowledge preservation, design traceability, architectural flexibility, automation, and readiness for change. Practices that promote future-proofing, architectural documentation, automation, and modularity were strongly associated with TC. The results highlight that TC is most often perceived as valuable during early architectural decisions, but a significant share of practitioners view it as relevant throughout the system lifecycle.

3.3 Perceived benefits and challenges of adopting TC (RQ3)

Participants identified a range of benefits and challenges in the adoption of TC (Table 7). Regarding benefits (Q27), the most popular advantage was encouraging investment in long-term, strategic architectural design decisions (77.42%). This suggests practitioners recognize TC as a useful lens for promoting sustainability-oriented thinking in architectural work. Similarly, promoting engineering practices (e.g., DevOps) to enhance system sustainability and evolution was chosen by 58.06% of respondents, further underlining the association of TC with long-term quality and maintainability.

Table 7: Summary of Responses for RQ3: Benefits and Challenges of TC

Q27—Benefits of TC	Count	%
Encouraging investment in long-term, strategic architectural design decisions	24	77.42
Promote engineering practices (e.g., DevOps) that enhance system sustainability and evolution	18	58.06
Enhancing software architecture design	17	54.84
Facilitating communication of technical design to non-technical stakeholders or management	14	45.16
Acting as a countermeasure to balance TD	13	41.94
Providing a common means to assess the value of a software system	10	32.26
Other	3	9.68
Q28—Challenges in operationalizing/measuring TC	**Count**	**%**
Potentially high upfront investment with unclear ROI	27	74.19
Resistance from teams prioritizing short-term deliverables	22	54.84
Difficulty in defining widely agreed on standards, metrics and benchmarks	16	51.61
Difficulty in communicating TC to stakeholders	9	29.03
Other	8	25.81

Additional benefits include TC's role in enhancing software architecture design (54.84%) and in facilitating communication of technical design to non-technical stakeholders or management (45.16%). These responses point to TC's potential as both a technical and communication tool. 41.94% also viewed TC as a countermeasure to balance TD, indicating a conceptual pairing where TC represents positive long-term value, balancing the cost-oriented framing of TD. In contrast, participants foresee several challenges in operationalizing and measuring TC (Q28). The most common was the potentially high upfront investment with unclear return on investment (ROI), selected by 27 participants (74.19%). This highlights a key adoption barrier: even if TC promises long-term gains, it may be de-prioritized due to short-term pressures. Resistance from teams prioritizing short-term deliverables was also reported as a challenge by 22 participants (54.84%). Additionally, 51.61% pointed to the difficulty in defining widely agreed-on standards, metrics, and benchmarks, reinforcing concerns raised earlier about the ambiguity of the TC concept.

Summary RQ3: TC is viewed as a promising concept with strategic and communications benefits. Its practical adoption depends on addressing challenges such as definitional vagueness, difficulty in quantifying ROI, and competing organizational priorities.

3.4 Willingness to operationalize TC (RQ4)

To evaluate practitioners' willingness to operationalize TC, we asked three survey questions. The first two, concerning the perceived value of TC as a quantifiable score (Q4) and the importance of developing industry-wide standards (Q6), were discussed above in the RQ1 section. The majority of responses expressed agreement (median and mode of 4 for both), suggesting that the concept holds promise and is seen as relevant for future practice.

Table 8: Summary of Responses for RQ4: Willingness to adopt TC (1=Low, 5=High)

Question	Likert Value	Count	%	Median	Mode	Mean
Q13—Value in TC as a "credit score"	1	1	3.23%	4	4	3.55
	2	6	19.35%			
	3	6	19.35%			
	4	11	35.48%			
	5	7	22.58%			
Q16—Importance of industry-wide TC standards	1	3	9.68%	4	4	3.52
	2	3	9.68%			
	3	8	25.81%			
	4	9	29.03%			
	5	8	25.81%			
Q31—Priority of TC adoption in your organization	1	3	9.68%	3	4	3.2
	2	5	16.13%			
	3	8	41.94%			
	4	13	25.81%			
	5	2	6.45%			

Q31 focused on how participants would prioritize the adoption of TC within their organizations (Table 8). The most frequently selected value was 4 (13 participants), with the median=3 and mean=3.19, indicating a moderate level of prioritization. Only 22.6% chose a high-priority score (4 or 5), 35.4% selected a low-priority score (1 or 2), and 42% opted for the neutral value of 3. This distribution suggests that, although the conceptual appeal of TC is understood, its concrete implementation is not yet a clear organizational priority for most respondents.

Summary RQ4: Practitioners generally view TC as a promising concept, recognizing its value and supporting industry-wide standardization efforts, However, their willingness to prioritize its adoption within their own organizations is more cautious.

4 Interviews results

We conducted interviews with six respondents (I1–I6), whose profiles are reported in Table 2. They include architects, managers, and engineers from IT, automotive, and other sectors, working in teams with fewer than 10 to over 50 members, and involved in both greenfield and maintenance projects. All reported moderate to high familiarity with the concept of TD. Each interview was transcribed and independently analysed using open and axial coding. The consolidated axial coding categories and new insights into the overarching categories are reported in Table 9. For each category, we summarize the core idea, list the associated axial codes and interviewees (for traceability), and highlight new perspectives introduced during the interviews.

TC as Strategic Design Discipline. This category concerns how TC is associated with structured architectural decisions to support system scalability and long term evolution. In this framing, TC is a consequence of early design practices to anticipate future change. The concept was informed by several axial subcategories, including strategic cost-benefit, decoupling, model-based reasoning, and a proactive approach to software design. These themes were observed in interviews with I1, I4, and I5. I5 described how TC was created using formal modelling tools. Simulink and Enterprise Architect were used to generate system artifacts and validate design behaviour before implementation. These tools supported early verification, consistency checking, and documentation, which were framed as mechanisms to reduce future costs.

I1 characterized TC as linked to a deeper understanding of system goals and implementation details. In this account, TC required decisions that generalized system functionality in anticipation of future reuse or evolution. I4 discussed TC as a planning activity during system design when long-term requirements were not fully known. They indicated that future extensibility was designed into the architecture despite the absence of explicit requirements. Common across these accounts was the view that TC involves structural investments made prior to their explicit need. I5 stated that *"if you can use your formal methods expertise... then you've actually got a fighting chance to scale linearly,"*. This suggests TC is associated with managing complexity through appropriate engineering mechanisms.

TC as Economic Investment and Business Case. This category captures how TC is framed as an economic investment, planned and justified through business-oriented reasoning. Rather than a purely technical concern, TC is discussed as an activity that aligns with budget practices, client engagement strategies, and cost-avoidance planning. The notion was articulated by I1, I4, I5, and I6. I4 described allocating part of a project's margin to TC-related improvements with a goal of reinforcing long-term relationships with clients.

These improvements were not explicitly requested but targeted to support future opportunities. TC was sometimes funded as part of pre-sales or internal investment. I6 emphasized the difficulty of communicating TC to non-technical stakeholders, noting that metrics or standard frameworks could make such conversations easier. I5 framed TC as a means to reduce long-term maintenance costs by investing in tooling and process infrastructure. These investments were tied to avoiding recurring defects and effort duplication. I4 noted that *"if we don't do it, it's costing us money"*, reflecting how TC was a preventive measure to avoid major rework. Across these accounts, TC was treated as a cost with deferred payoff, requiring justification to be accepted in delivery organizations.

TC as Counter-Narrative to Technical Debt. This category describes how TC is framed in relation to TD, not only as its conceptual opposite, but also as a complementary, coexisting phenomenon. Several interviewees positioned TC as a positive narrative to balance the predominantly negative discourse surrounding TD. The theme was raised by I1, I2, I4, and I5. Some noted that the boundary between TC and TD is not always clear. I2 stated that *"something can be both technical credit and technical debt at the same time,"* pointing to architectural decisions that provide flexibility, but also

Table 9: Interviews' Thematic Categories with Axial Subcategories, Interviewee Traceability, and Emergent Insights

Category	Core Idea	Axial Subcategory	Interviewee	New Perspective
TC as Strategic Design Discipline	TC reflects deliberate architectural foresight, often operationalized through modeling and tooling.	Strategic cost-benefit framing	I1, I5	TC can be embedded in formal modeling tools (e.g., Simulink, EA) to simulate behaviors, detect flaws, scale sustainably, and quantify ROI.
		Decoupling as core tactic	I5	
		Model-based reasoning	I5	
		Proactive engineering philosophy	I1, I4	
TC as Economic Investment & Business Case	TC is treated like ROI: planned, budgeted, and tracked.	TC framed as economic investment	I4, I6	TC can be budgeted into project margins or sales costs like any business investment—directly tied to retention or upselling.
		Business case necessity	I6, I5	
		TC to grow client accounts	I4	
		Rewrite avoidance	I4	
TC as Counter Narrative to TD	TC provides a constructive, forward-looking narrative to complement TD.	TC TD fluidity	I2, I3	TC can coexist with or even become TD, depending on outcomes—highlighting the ambiguity of design foresight.
		Conceptual counterweight	I2, I1	
		Reactive vs. proactive framing	I5, I4	
		Positive framing of investment	I1, I4	
TC as Communication & Translation Tool	TC enables engineers to convey complex decisions in business terms.	Architect as translator	I3, I6	TC requires storytelling to make invisible benefits legible to stakeholders—especially non-technical managers.
		Cost-risk narratives	I5, I3	
		Stakeholder bridge	I4, I5	
		Communication challenge	I6, I3	
TC as Context Sensitive & Culturally Embedded	TC's meaning and applicability vary across industries, methods, and cultures.	Agile extremism	I1, I6	TC is often ignored in agile-dominant cultures and thrives in hybrid models where ideation and delivery coexist.
		Context-shaping TC	I2, I3	
		Hybrid agile (Scrumban)	I4	
		Historical learning	I3, I4	
TC as Invisible When Successful	TC's success leads to invisibility; systems work and aren't questioned.	Success lacks visibility	I5	TC is paradoxically penalized for working well—because crises never occur, recognition is lost.
		Benefits unrecognized	I6, I3	
		No feedback loop	I3, I6	
TC as Qualitative Judgment over Metrics	TC is hard to quantify and better supported through practitioner experience.	Skepticism of metrics	I2, I5	Qualitative judgment (e.g., use cases, context, interviews) may be more actionable than abstract KPIs.
		Practitioner heuristics	I1, I3	
		Narrative framing	I6, I3	

introduce fragility or complexity. This suggests that the same design outcome may be interpreted differently depending on how it performs over time.

I1 and I4 emphasized that TC is most meaningful when contrasted with debt. They described TC as the deliberate investment that helps avoid the rework associated with TD. They also acknowledged that deliberate investments do not guarantee long-term benefit, especially if the investment is based on incorrect assumptions about future needs. TC should be an instrument that helps in making decisions and in monitoring and measuring ROI during the entire development. I5 discussed TC in terms of proactive planning, arguing that it helps minimize reactive work caused by accumulating debt. TC was still not positioned as a clear binary opposite to TD. Rather, I5 indicated that many decisions carry both debt and credit potential, depending on context and outcome. This category reflects a framing where TC serves as a useful counterpoint to TD, but is not always inherently distinct from it. Instead, the two are often interwoven in practice and retrospectively evaluated based on how systems evolve.

TC as Communication and Translation Tool. This category focuses on TC as a communication vehicle between technical teams and business stakeholders. Interviewees described TC as a means to make long-term technical investments more understandable and defensible in non-technical contexts. The theme was discussed by I3, I4, I5, and I6. I3 and I6 noted TC can help translate architectural intentions into terms that resonate with decision-makers. Rather than presenting internal improvements as abstract benefits, TC offers a way to link them to business outcomes such as flexibility and reduced risk and cost.

I5 highlighted the use of trade-off narratives as part of this translation process. In their account, architectural decisions were often discussed in terms of future cost scenarios and optionality. By articulating TC in terms of what it enables or mitigates, teams could argue for its value even in the absence of immediate payoff.

I4 described how an investment in advanced modelling approaches improved communications with management. Simulink and MATLAB were used to generate a key component. The generated code had zero defects, facilitating lower cost downstream development. When the cost of the licenses for these tools was questioned by management, the ROI from the generated component and the lower costs it afforded could be used as justification. In this example, TC concerns making the investment and its associated benefits visible and explainable, thus improving communication.

I4 emphasized the importance of tailoring these narratives to the client's perspective, particularly when stakeholders lacked technical understanding. In one case, the team used TC to justify extensibility features that were not required at the time but were positioned as enablers for future product integration. This category reflects the role of TC in bridging communication gaps. It allows long-term technical reasoning to be expressed in business-relevant terms and facilitates alignment across organizational boundaries.

TC as Context-Sensitive and Culturally Embedded. This category captures how the interpretation and application of TC varies depending on organizational, cultural, and methodological context. Rather than a universally defined construct, TC was described as

contingent on factors such as process models, team maturity, and organizational history. The theme were raised by I1, I2, I3, and I4. I1 expressed concern about how extreme adherence to agile practices can discourage upfront investments, framing such environments as biased toward short-term delivery over architectural foresight. I4, on the other hand, described working in hybrid process models, i.e., combining agile elements with structured planning phases, which provided space for TC-related activities such as proactive feature design or architectural extensibility.

I3 noted that an organization's past failures with maintainability or platform complexity often shaped how future investments were perceived. In these cases, TC became more visible and acceptable after prior difficulties exposed the costs of not planning ahead. I2 similarly remarked that *"quality is in the eye of the client"*, suggesting that the recognition of credit-like decisions depends on external expectations and pressures.

Across these interviews, TC was framed as something to be negotiated within local conditions. The willingness to pursue or justify TC varied depending on delivery models, stakeholder expectations, and past experiences with technical debt.

TC as Invisible When Successful. This category addresses how TC often remains unnoticed or undervalued precisely when it is effective. Interviewees described a recurring difficulty in attributing value to technical decisions that prevent problems rather than solve visible ones. This theme was articulated by I3, I5, and I6. I5 noted many TC-related actions, such as investing in reusable components or modular design, go unacknowledged because they prevent issues that might otherwise arise. As a result, teams may receive more recognition for fixing failures than for preventing them.

I6 reflected on how the invisibility of successful technical choices hinders their justification in future projects. Without visible payoffs, investments in TC are often viewed as optional or excessive, especially under time pressure. I3 pointed out that the absence of a feedback loop means TC rarely generates signals to reinforce its perceived value. Once a system functions reliably, the design decisions that enabled that outcome can fade from attention. I2 noted, *"it's very hard to prove that you're doing the right thing just because it didn't break"*, emphasizing a lack of visibility for preventative technical actions and the difficulty of attributing neasurable value to them. The interviews suggested that TC is salient only when it is lacking, typically after maintainability or extensibility issues surface. When successful, it tends to blend into the background of normal system behaviour, making its benefits difficult to trace or quantify.

TC as Qualitative Judgment over Metrics. This category focuses on the challenges of measuring TC and the reliance on qualitative judgment in its assessment and justification. Interviewees expressed scepticism about the feasibility of quantifying TC through metrics alone, and emphasized the role of experience, and contextual reasoning. The theme was described by I1, I2, I3, I5, and I6. I1 stated that while metrics support communication, TC often depends on reasoning that cannot be captured numerically. Instead, decisions are framed through use cases, architectural scenarios, and future constraints. I3 also questioned the practicality of metric-based approaches, observing that most team decisions are guided by implicit trade-offs and local knowledge rather than formal measurement.

I6 shared an example involving an internal tool that approximated code health but acknowledged that such metrics were limited in scope and interpretation. I5 argued that understanding TC requires reflecting on project history and potential alternative outcomes, which resist codification. Across interviews, there was general agreement that TC is more effectively reasoned about through structured judgment than through standardized metrics. This category indicates that practitioners currently approach TC as a context-sensitive construct, relying on interpretive framing rather than calculable values.

5 Related Work

To the best of our knowledge, a limited number of peer-reviewed papers explore the concept of TC. While largely absent from the software engineering literature, TC has been described in systems engineering by Berenbach [4]. This defines TC as extra effort put into designing and building systems in anticipation of future benefits from emergent properties, a term used in systems engineering for unknown future requirements. In our previous work [10], we explored this concept in the context of software engineering, proposing a research agenda to investigate its implications.

The concept of TD, introduced by Cunningham [9], has become foundational in explaining how expedient decisions can accumulate long-term cost. Over time, TD has evolved into a framework encompassing different types of debt (e.g., design, architecture, code), with associated detection tools, management strategies, and economic models [2]. Studies such as by Li et al. [16] identify recurring challenges such as lack of visibility, prioritization difficulties, and misalignment between technical and business stakeholders. These themes remain highly relevant in our findings, but they are inverted in the case of TC, where the concern shifts from reducing harm to strategically enabling long-term benefits.

Recent tooling efforts such as CodeScene illustrate how debt-oriented thinking is evolving. CodeScene's Code Health metric [21] empirically correlates low code health with increased defects and longer cycle times. The tool aggregates over 25 structural and process indicators (e.g., complexity, duplication, developer congestion) to identify "hotspots" and guide remediation efforts. While Code Health aligns conceptually with TC, valuing maintainability and forward-looking design, it is inherently reactive, operating only on existing codebases. This provides retrospective insights to identify and mitigate problematic areas post-implementation. By contrast, TC emphasizes proactive decisions, often made before code exists. Our study underscores this as a cognitive and organizational investment, capturing foresight that may never manifest in code symptoms but significantly impacts system evolvability.

A recent study [6], leverages CodeScene's Code Health metric to investigate business-level impacts of code quality. Analyzing 79 proprietary systems, the study finds non-linear returns: improvements from high to very high maintainability yield disproportionately large gains in defect reduction and development speed. This underscores that pursuing high code quality offers significant business leverage. These findings move beyond TD by quantifying its positive complement and reinforcing the emergent notion of TC as valuable design capital, not just avoidance of debt.

Multiple authors have approached software engineering and architecture through an economic lens. Kazman et al.'s Cost-Benefit Analysis Method (CBAM) [12] and Boehm's Value-Based Software

Engineering [5] both frame design choices as investment decisions, emphasizing trade-offs between cost and long-term return. These approaches highlight how certain design strategies can yield option-like flexibility under uncertainty, a perspective that closely aligns with how practitioners in our study conceptualize TC.

The idea of sustainable software architecture has grown in prominence, emphasizing resilience, adaptability, and long-term system health [3]. Concepts such as "architecture health" and patterns for modifiability and testability [3] provide the technical substrate on which TC can be built. Our data shows that practitioners implicitly refer to TC when describing actions that "future-proof" systems, including abstraction layers, configuration flexibility, and modular architectures. In particular, the literature on architectural tactics aligns with the tangible manifestations of TC reported by respondents, such as decoupling, reusable modules, and platform independence. These help mitigate future change costs.

Another theme in our findings is the role of tooling and modelling environments. These tools allow architects to simulate, validate, and generate artifacts with fewer defects and reduced maintenance cost. This aligns with prior work in model-driven engineering (MDE) [13] and DevOps automation, both of which advocate up-front investment in system understanding and operational readiness. While these tools are typically positioned as enablers of productivity and consistency, our analysis shows that they can be leveraged specifically to accumulate technical credit, creating capabilities that greatly enhance system evolution.

Architectural Decision Records (ADRs) are gaining traction as a means of improving traceability and long-term system understanding [11]. Our participants describe these practices as ways to preserve strategic intent, which is crucial when advocating for TC in organizations with high turnover or shifting priorities. However, both our study and the broader literature reflect persistent organizational barriers: a focus on short-term delivery, limited stakeholder understanding, and insufficient incentive structures to prioritize long-term design benefits [15].

6 Final Discussion and future work

We discuss several issues arising from our results below:

Upfront Investment. It is clear from the survey and interview analyses that technical credit does not necessarily imply additional upfront investment. Responses emphasized that design decisions are made based on requirements, problem analysis, and perceived benefits. Some decisions may involve financial or time investments, while others simply reflect trade-offs to prioritize certain system qualities and functionalities over others. As with any investment, it is important to track these decisions and evaluate whether they yield long-term value. Investment tracking could provide valuable input for downstream project planing and retrospectives, and contribute to collective team knowledge. From this perspective, TC is pervasive across the entire development lifecycle (see Section 3.2).

TC in Practice. One issue, reflected in the responses to RQ4 (willingness to operationalise TC), is that while the conceptual appeal of TC is widely recognised, its concrete implementation needs elevating to become an organization priority. Participants recognize and intuitively enact practices that create TC, but these commonly remain invisible. A challenge therefore for the research community is to create awareness, methods, and tools for explicitly recognizing and valuing TC in projects.

Enhance Organizational Knowledge and Recognition. TC can be viewed as an instrument to support decision-making by monitoring, measuring, documenting, and evaluating the effects of specific choices. As a consequence, TC can also serve as a record of successful, impactful decisions. This can provide traceability and a shared understanding that informs future choices, and makes them accessible to non-technical stakeholders. For example, as illustrated in the case of the Simulink and MATLAB code generation module (Section 4), explicitly recognized TC would represent the return on investment (ROI) of the architectural decisions. This explicit recognition would make the value of the efforts, namely the TC, visible and measurable to directors and managers.

Counterbalance to TD. TC complements TD by providing a positive counterbalance. TD captures the liabilities of past compromises, TC represents long-term value created by strategic design choices. The two naturally co-exist. Interestingly, development tools are moving beyond TD recognition and capturing aspects of TC. For example, CodeScene[3], mentioned by an interviewee, incorporates Code Health metrics designed to make the business impact of quality code visible. Further, Törnqvist et al. [20], conducted a study of 39 proprietary production codebases, showing that high-quality code delivers clear business benefits: 15 times fewer defects, twice the development speed, and substantially more predictable issue resolution times. Similarly, Borg et al. [7] propose a value-creation model demonstrating that improvements in maintainability yield increasing, not diminishing, returns. Crucially, this work underscores the need to better communicate these benefits at the business and executive level, and aligns with the objectives of TC.

Definition (Technical Credit). We conclude by offering a refined definition of technical credit from [10], extending it with insights gained from this study: Technical credit is the measurable benefit arising from design or infrastructure decisions which, through deliberate investment or balanced trade-offs, create enduring advantages for system evolution. TC makes the value of such decisions visible and understandable across stakeholders, supporting trust, alignment with business goals, and informed decision-making.

Future Work. Future work should focus on translating the concept of TC into practices for industry. This includes lightweight methods and tools to make TC visible in day-to-day development, embedding TC into processes such as retrospectives and ADRs, and conducting longitudinal studies to demonstrate TC's impact on business outcomes. A further priority is understanding how TC can be effectively communicated across managerial stakeholders, ensuring its value is recognized beyond engineering teams.

Acknowledgments

This work is supported by the Swedish Agency for Innovation Systems through the project "Secure: Developing Predictable and Secure IoT for Autonomous Systems" (2023-01899) 'Advancing AI-Driven Software Architecture: A Collaborative Exchange between MDU and SMU" (2024-02068), and by the Key Digital Technologies Joint Undertaking through the project "MATISSE: Model-based engineering of digital twins for early verification and validation of industrial systems" (101140216).

[3] https://codescene.com/product/code-health

References

[1] Association for Computing Machinery. 2024. ACM Publications Policy on Research Involving Human Participants and Subjects. https://www.acm.org/publications/policies/research-involving-human-participants-and-subjects. Accessed: 2025-07-04.

[2] Paris Avgeriou, Rainer Koschke, Harald B. R. Lee, Ann R. B. Jansen, Iman Keivanloo, and Tom Maibaum. 2016. Managing Technical Debt in Software Engineering. In Dagstuhl Reports, Vol. 6. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 1–19. doi:10.4230/DagRep.6.7.1

[3] Len Bass, Paul Clements, and Rick Kazman. 2021. Software architecture in practice. Addison-Wesley Professional.

[4] Brian Berenbach. 2014. On Technical Credit. Procedia Computer Science 28 (2014), 505–512. doi:10.1016/j.procs.2014.03.062 2014 Conference on Systems Engineering Research.

[5] Barry Boehm. 2003. Value-based software engineering: reinventing. ACM SIGSOFT Software Engineering Notes 28, 2 (2003), 3.

[6] Markus Borg, Ilyana Pruvost, Enys Mones, and Adam Tornhill. 2024. Increasing, not diminishing: Investigating the returns of highly maintainable code. (2024), 21–30.

[7] Markus Borg, Ilyana Pruvost, Enys Mones, and Adam Tornhill. 2024. Increasing, not Diminishing: Investigating the Returns of Highly Maintainable Code. In Proceedings of the 7th ACM/IEEE International Conference on Technical Debt. Association for Computing Machinery, New York, NY, USA, 21–30. doi:10.1145/3644384.3644471

[8] Victor R Basili1 Gianluigi Caldiera and H Dieter Rombach. 1994. The goal question metric approach. Encyclopedia of software engineering (1994), 528–532.

[9] Ward Cunningham. 1992. The WyCash Portfolio Management System. In OOPSLA '92 Experience Report. Available at: http://c2.com/doc/oopsla92.html.

[10] Ian Gorton, Alessio Bucaioni, and Patrizio Pelliccione. 2025. Technical Credit. Commun. ACM 68, 2 (2025), 30–33.

[11] Neil B Harrison, Paris Avgeriou, and Uwe Zdun. 2007. Using patterns to capture architectural decisions. IEEE software 24, 4 (2007), 38–45.

[12] Rick Kazman, Jayatirtha Asundi, and Mark Klein. 2002. Making Architecture Design Decisions: An Economic Approach. Technical Report Carnegie Mellon University, Software Engineering Institute's Digital Library CMU/SEI-2002-TR-035. https://doi.org/10.1184/R1/6575189.v1 Accessed: 2025-Sep-23.

[13] Stuart Kent. 2002. Model driven engineering. In International conference on integrated formal methods. Springer, 286–298.

[14] Barbara A Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research part 2: designing a survey. ACM SIGSOFT Software Engineering Notes 27, 1 (2002), 18–20.

[15] Valentina Lenarduzzi, Terese Besker, Davide Taibi, Antonio Martini, and Francesca Arcelli Fontana. 2021. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools. Journal of Systems and Software 171 (2021), 110827.

[16] Zengyang Li, Paris Avgeriou, and Peng Liang. 2015. A systematic mapping study on technical debt and its management. Journal of systems and software 101 (2015), 193–220.

[17] Michael Quinn Patton. 2014. Qualitative research & evaluation methods: Integrating theory and practice. Sage publications, Fourth edition, ISBN 9781412972123.

[18] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14 (2009), 131–164.

[19] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded theory in software engineering research: a critical review and guidelines. In Proceedings of the 38th International conference on software engineering. 120–131.

[20] Adam Tornhill and Markus Borg. 2022. Code red: the business impact of code quality - a quantitative study of 39 proprietary production codebases. In Proceedings of the International Conference on Technical Debt (Pittsburgh, Pennsylvania) (TechDebt '22). Association for Computing Machinery, New York, NY, USA, 11–20. doi:10.1145/3524843.3528091

[21] Adam Tornhill and Markus Borg. 2022. Code red: the business impact of code quality-a quantitative study of 39 proprietary production codebases. (2022), 11–20.

[22] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. Experimentation in software engineering. Vol. 236. Springer.