Socio-Technical Factors Computing Technical Debt: A Systematic Literature Review and Meta-Analysis

Registered Report

David Botton Ferenc Varga Léandre Gerday

Group Linus Torvalds, INFO B302 - IDS (2024 - 2025)
Faculty of Computer Science, University of Namur, Belgium

1 Introduction

Technical debt is a term coined in 1993 by Cunningham [3]. It describes decisions and shortcuts taken that incur future maintenance costs. Cunningham originally defined it as: "*Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt*" [3]. Aimed at software programming, research increasingly uncovers organizational and human factors that contribute to technical debt [2, 6, 13, 21]. For instance, Besker et al. empirically demonstrate how technical debt directly impacts developers' productivity, revealing that "*developers waste, on average, 23% of their time due to TD (Technical Debt) and that developers are frequently forced to introduce new TD*" [2]. This finding underlines the cascading nature of technical debt, where unresolved issues accumulate and birth further debt throughout organizational processes.

Tamburri et al. introduced the concept of social debt as "*analogous to technical debt in many ways: it represents the state of software development organisations as the result of 'accumulated' decisions. In the case of social debt, decisions are about people and their interactions*" [21]. They define social debt as "*the additional cost occurring when strained social and organisational interactions get in the way of smooth software development and operation*" [21]. Their research demonstrates that "*social debt is strongly correlated with technical debt and both forces should be reckoned with together during the software process*" [21], suggesting that effectively managing technical debt requires addressing both technical and social dimensions of software development.

This systematic review investigates socio-technical factors and technical debt using three extraction criteria: (1) socio-technical factors explicitly measured [1,18,19]; (2) methodological approaches that establish these relationships [9, 23]; and (3) published results with effect sizes [10, 17]. We classify relationships as direct or indirect and categorize them temporally as debt-creating, debt-sustaining, or debt-reducing factors [7, 12]. Our objectives are to identify socio-technical factors associated with technical debt; analyze methodological approaches and potential researcher biases [17]; and synthesize an integrated framework connecting social structures to technical outcomes [20, 22].

Current technical debt research remains fragmented [5, 16]. Li et al. provided a mapping study of technical debt concepts [11], but few studies systematically extract data on relationship mechanisms between socio-technical factors and debt outcomes. Most critically, Shepperd et al. demonstrated that researcher group explained 31% of experimental variance in defect prediction, dramatically outweighing technical factors (1.3%). As they concluded, "*Surprisingly we find that the choice of classifier has little impact upon performance (1.3%) and in contrast the major (31%) explanatory factor is the researcher group. It matters more who does the work than what is done*" [17]. This raises profound questions about similar biases affecting technical debt studies.

Our research differentiates itself through: structured data extraction protocols specifically designed for socio-technical relationships [14]; meta-analytical techniques to quantify both factor effects and researcher biases [4, 17]; and focus on causal mechanisms connecting social structures to technical outcomes [15, 21]. This approach follows PRISMA guidelines [15] integrated with systematic review methodology and improved with Shepperd's bias detection techniques [17].

2 Background

This section establishes the foundational concepts necessary to understand socio-technical factors in technical debt and our systematic review approach. We first trace the evolution of technical debt and its expansion into socio-technical dimensions, then ground our analysis in socio-technical systems theory, outline our systematic review methodology and conclude with an examination of researcher bias.

2.1 Technical Debt Concept and Evolution

"Although the Technical Debt metaphor was proposed two decades ago, it has only received significant attention from researchers in the past few years... it has been gradually extended to software architecture, detailed design and even documentation, requirements and testing." [11] Li et al. also identified ten distinct types of technical debt, demonstrating its pervasiveness across software systems. Ernst et al. [6] revealed a critical disconnect between research and practice; while *"using code quality analysis techniques to understand technical debt has been the dominant focus in research and by tool vendors,"* their study found that *"architectural decisions are the most important source of technical debt."*. This insight directly motivates our socio-technical perspective on technical debt.

2.2 Socio-Technical Systems Theory

Socio-technical systems theory recognizes software development as a human activity conducted within organizational contexts. Tamburri et al. [20] introduced "social debt" as the hidden costs arising from suboptimal organizational structures. "*Social debt is analogous to technical debt in many ways: it represents the state of software development organisations as the result of 'accumulated' decisions. In the case of social debt, decisions are about people and their interactions*" [20]. Besker et al. [2] demonstrated how team communication patterns and organizational constraints create debt undetectable through code analysis alone. "*To date, there is limited research available that attempts to quantify empirically how TD negatively affects software development productivity. The existing literature [...] states that TD becomes a constant drain on software productivity*" [2]. This theoretical framework provides our basis for categorizing relationships between social structures and technical outcomes in technical debt manifestation.

2.3 Systematic Review Methodology

Our systematic review methodology integrates Kitchenham's software engineering guidelines [8] with PRISMA reporting standards [15]. Kitchenham established protocols emphasizing transparent search strategies, explicit inclusion/exclusion criteria and rigorous quality assessment, noting that *"Quality relates to the extent to which the study minimises bias and maximises internal and external validity"* [8]. PRISMA extends this framework with specific reporting standards for systematic reviews and meta-analyses. *"The aim of the PRISMA Statement is to help authors improve the reporting of systematic reviews and meta-analyses. We have focused on randomized trials, but PRISMA can also be used as a basis for reporting systematic reviews of other types of research, particularly evaluations of interventions."* [15]. This methodological triangulation enables both qualitative synthesis of contextual factors and quantitative analysis of effect sizes.

2.4 Researcher Bias in Empirical Software Engineering

The presence of researcher bias represents a significant concern that has received insufficient attention in software engineering research. Several studies have demonstrated how researcher expectations can significantly affect experimental design, analysis procedures and interpretation of results. Bias can manifest through various mechanisms, including selective reporting of positive results, confirmation bias in hypothesis testing and unconscious manipulation of parameters until desired outcomes emerge. Unlike medical research, where double-blind protocols are standard, computational experiments in software engineering rarely implement blinding techniques. The concerning implication is that published findings may reflect researcher expectations rather than objective technical reality. As Shepperd noted, *"If a result is highly dependent on who does the experiment, then we need to question if we're really making progress in the field."* [17]. This skepticism is reinforced by observations that groups tend to report favorable results for techniques they themselves developed. By applying meta-analytical techniques to technical debt research, we can distinguish robust findings from potential artifacts of researcher bias.

3 Research questions

This study aims to address the following primary research question: **What socio-technical factors influence the creation, sustenance and reduction of technical debt in software development organizations?** To systematically explore this complex domain, we have formulated four sub-research questions:

SRQ1: What socio-technical factors have been empirically linked to technical debt in existing literature? This question aims to systematically identify and classify relationship mechanisms between organizational/social structures and technical debt outcomes. By extracting explicit socio-technical factors from studies like Tamburri et al. [21] on social debt and Besker et al. [2] on developer productivity loss, we can create a comprehensive taxonomy of factors that organizations should monitor.

SRQ2: How do methodological approaches in technical debt research impact reported relationships between socio-technical factors and debt outcomes? Building on Shepperd et al.'s [17] findings that researcher group explains 31% of experimental variance in defect prediction, this question examines how research methodology influences reported correlations between social structures and technical debt. This will help establish confidence levels for different relationship claims and identify methodological best practices.

SRQ3: What causal mechanisms connect specific social structures to technical debt outcomes? This question investigates the underlying mechanisms through which social factors like team composition, communication patterns and organizational decisions translate into technical debt. Drawing from Ernst et al. [6], who identified architectural decisions as critical sources of technical debt, we will analyze how social structures influence these architectural decisions.

SRQ4: How can measurement frameworks quantify the socio-technical dimensions of technical debt? Following Li et al.'s [11] systematic mapping of technical debt management, this question explores metrics and measurement approaches that can capture the socio-technical aspects of debt. This will address the current gap where "few studies systematically extract data on relationship mechanisms between socio-technical factors and debt outcomes."

4 Evaluation

Our evaluation follows PRISMA guidelines with Kitchenham's software engineering review methodology as a three-phase approach: search and selection, data extraction and synthesis. For our search strategy, seven digital libraries are examined using a comprehensive search string combining technical debt terminology with socio-technical concepts. We will supplement this with backward snowballing and examination of key conferences (ICSE, FSE, ESEM, ICSME) and journals (TSE, JSS, IST, EMSE) between 2005-2025. We'll consider empirical studies that examine socio-technical aspects and technical debt, and we'll not include secondary studies, technical studies and grey literature. Two reviewers will independently screen studies with a third resolving conflicts, and Cohen's kappa for reliability will be calculated. Data harvesting will collect generic metadata on publication details, research context as well as socio-technical factors using Tamburri's framework, technical debt categories using Li's taxonomy, relationship characterization examining direct/indirect effects and methodological approaches with published outcomes. Each study will be quality assessed using a 12-point instrument addressing research design, data collection, sample representation, analysis rigor and researcher reflexivity. Studies scoring below 6 will be excluded. Our mixed-methods synthesis will follow Shepperd's approach to analyze researcher influence. Qualitative synthesis performed using thematic analysis and framework synthesis, while quantitative meta-analysis will utilize a random-effects model. We will estimate heterogeneity by I^2 statistic and test publication bias using funnel plots and Egger's test. The result will be a model that combines social structures and technical outcomes, with relationship and temporal processes specified. We will minimize validity threats through the utilization of comprehensive search strategies, rigorous selection criteria, dual extraction and sensitivity analyses. The framework will provide practitioners with actionable knowledge on which organizational interventions most reduce technical debt. Visualization of the interrelations between technical debt consequences and social structures will highlight the strength of evidence and intervention areas. This new paradigm extends beyond the current tool-centric research to incorporate the organizational and human aspects of technical debt management.

References

[1] Nicolli S.R. Alves, Thiago S. Mendes, Manoel G. De Mendonça, Rodrigo O. Spínola, Forrest Shull, and Carolyn Seaman. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70:100–121, February 2016.

[2] Terese Besker, Antonio Martini, and Jan Bosch. Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. *Journal of Systems and Software*, 156:41–61, October 2019.

[3] Ward Cunningham. The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, April 1993.

[4] Bill Curtis, Jay Sappidi, and Alexandra Szynkarski. Estimating the Principal of an Application's Technical Debt. *IEEE Software*, 29(6):34–42, November 2012.

[5] André B. De Carvalho, Aurora Pozo, and Silvia Regina Vergilio. A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software*, 83(5):868–882, May 2010.

[6] Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L. Nord, and Ian Gorton. Measure it? Manage it? Ignore it? software practitioners and technical debt. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 50–60, Bergamo Italy, August 2015. ACM.

[7] Rick Kazman, Yuanfang Cai, Ran Mo, Qiong Feng, Lu Xiao, Serge Haziyev, Volodymyr Fedak, and Andriy Shapochka. A Case Study in Locating the Architectural Roots of Technical Debt. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, pages 179–188, Florence, Italy, May 2015. IEEE.

[8] Barbara Kitchenham and Stuart M. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University and University of Durham, January 2007.

[9] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. Technical Debt: From Metaphor to Theory and Practice. *IEEE Software*, 29(6):18–21, November 2012.

[10] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, 34(4):485–496, July 2008.

[11] Zengyang Li, Paris Avgeriou, and Peng Liang. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220, March 2015.

[12] Antonio Martini, Terese Besker, and Jan Bosch. Technical Debt tracking: Current state of practice. *Science of Computer Programming*, 163:42–61, October 2018.

[13] Antonio Martini and Jan Bosch. The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*, pages 1–10, Montreal, QC, Canada, May 2015. IEEE.

[14] Antonio Martini and Jan Bosch. Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 422–429, Madeira, Portugal, August 2015. IEEE.

[15] D. Moher, A. Liberati, J. Tetzlaff, D. G Altman, and for the PRISMA Group. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *BMJ*, 339(jul21 1):b2535–b2535, July 2009.

[16] Carolyn Seaman and Yuepu Guo. Measuring and Monitoring Technical Debt. In *Advances in Computers*, volume 82, pages 25–46. Elsevier, 2011.

[17] Martin Shepperd, David Bowes, and Tracy Hall. Researcher Bias: The Use of Machine Learning in Software Defect Prediction. *IEEE Transactions on Software Engineering*, 40(6):603–616, June 2014.

[18] Forrest Shull, Davide Falessi, Carolyn Seaman, Madeline Diep, and Lucas Layman. Technical Debt: Showing the Way for Better Transfer of Empirical Results. In Jürgen Münch and Klaus Schmid, editors, *Perspectives on the Future of Software Engineering*,

pages 179–190. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[19] Clauirton Siebra A. Siebra, Graziela S. Tonin, Fabio Q.B. Silva, Rebeka G. Oliveira, Antonio L.O.C. Junior, Regina C.G. Miranda, and Andre L.M. Santos. Managing technical debt in practice: An industrial report. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 247–250, Lund Sweden, September 2012. ACM.

[20] Damian A. Tamburri, Philippe Kruchten, Patricia Lago, and Hans Van Vliet. What is social debt in software engineering? In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 93–96, San Francisco, CA, USA, May 2013. IEEE.

[21] Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans Van Vliet. Social debt in software engineering: Insights from industry. *Journal of Internet Services and Applications*, 6(1):10, December 2015.

[22] Damian A. Tamburri, Patricia Lago, and Hans Van Vliet. Organizational social structures for software engineering. *ACM Computing Surveys*, 46(1):1–35, October 2013.

[23] Edith Tom, Aybüke Aurum, and Richard Vidgen. An exploration of technical debt. *Journal of Systems and Software*, 86(6):1498–1516, June 2013.